Alexander Gelbukh (Ed.)

Computational Linguistics and Intelligent Text Processing

LNCS 3406

6th International Conference, CICLing 2005
Mexico City, Mexico, February 2005
Proceedings

Springer

Lecture Notes in Computer Science 3406

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

Alexander Gelbukh (Ed.)

Computational Linguistics and Intelligent Text Processing

6th International Conference, CICLing 2005
Mexico City, Mexico, February 13-19, 2005
Proceedings

Springer

Volume Editor

Alexander Gelbukh
National Polytechnic Institute (IPN)
Center for Computing Research (CIC)
Col. Zacatenco, CP 07738, D.F., Mexico
E-mail: gelbukh@gelbukh.com

Library of Congress Control Number: 2005920311

CR Subject Classification (1998): H.3, I.2.7, I.7, I.2, F.4.3

ISSN 0302-9743
ISBN 3-540-24523-5 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11384427 06/3142 5 4 3 2 1 0

Preface

CICLing 2005 (www.CICLing.org) was the 6th Annual Conference on Intelligent Text Processing and Computational Linguistics. It was intended to provide a balanced view of the cutting-edge developments in both the theoretical foundations of computational linguistics and the practice of natural-language text processing with its numerous applications. A feature of CICLing conferences is their wide scope that covers nearly all areas of computational linguistics and all aspects of natural language processing applications.

This year we were honored by the presence of our keynote speakers *Christian Boitet* (CLIPS-IMAG, Grenoble), *Kevin Knight* (ISI), *Daniel Marcu* (ISI), and *Ellen Riloff* (University of Utah), who delivered excellent extended lectures and organized vivid discussions and encouraging tutorials; their invited papers are published in this volume.

Of 151 submissions received, 88 were selected for presentation; 53 as full papers and 35 as short papers, by exactly 200 authors from 26 countries: USA (15 papers); Mexico (12); China (9.5); Spain (7.5); South Korea (5.5); Singapore (5); Germany (4.8); Japan (4); UK (3.5); France (3.3); India (3); Italy (3); Czech Republic (2.5); Romania (2.3); Brazil, Canada, Greece, Ireland, Israel, the Netherlands, Norway, Portugal, Sweden, Switzerland (1 each); Hong Kong (0.5); and Russia (0.5) including the invited papers. Internationally co-authored papers are counted in equal fractions.

Of the accepted papers, the Program Committee selected the following three papers for the Best Paper Award:

1st place: Finding Instance Names and Alternative Glosses on the Web: WordNet Reloaded, by *Marius Paşca*;
2nd place: Unsupervised Evaluation of Parser Robustness, by *Johnny Bigert, Jonas Sjöbergh, Ola Knutsson,* and *Magnus Sahlgren*;
3rd place: Learning Information Extraction Rules for Protein Annotation from Unannotated Corpora, by *Jee-Hyub Kim* and *Melanie Hilario*.

The authors of these papers were also given extended time for their presentations. In addition, the Best Presentation Award and Best Poster Award winners were selected through voting by the participants of the conference.

Besides its high scientific level, one of the success factors of CICLing conferences is their excellent cultural program. CICLing 2005 was held in Mexico, a wonderful country very rich in culture, history, and nature. The participants of the conference—in their souls active explorers of the world—had a chance to see the solemn 2000-years-old pyramids of the legendary Teotihuacanas, a monarch butterfly wintering site where the old pines are covered with millions of butterflies as if they were leaves, a great cave with 85-meter halls and a river flowing from it, Aztec warriors dancing in the street in their colorful plumages, and the largest anthropological museum in the world; see photos at www.CICLing.org.

Very special thanks go to Ted Pedersen and Rada Mihalcea for their invaluable support in the reviewing process and in the preparation of the conference.

December 2004 Alexander Gelbukh

Organization Committee

Conference Chair

Alexander Gelbukh (CIC-IPN, Mexico)

Program Committee

Boitet, Christian (CLIPS-IMAG, France)
Bolshakov, Igor (CIC-IPN, Mexico)
Bontcheva, Kalina (U. Sheffield, UK)
Calzolari, Nicoletta (ILC-CNR, Italy)
Campbell, Nick (SCOPE, Japan)
Carroll, John (U. Sussex, UK)
Cristea, Dan (U. Iasi, Romania)
Gelbukh, Alexander (**Chair**; CIC-IPN, Mexico)
Hallett, Cătălina (U. Brighton, UK)
Harada, Yasunari (Waseda U. Japan)
Hovy, Eduard (ISI of U. Southern California, USA)
Kharrat, Alma (Microsoft Research, USA)
Kilgarriff, Adam (Lexical Computing Ltd., UK)
Kittredge, Richard (CoGenTex Inc., USA)
Kübler, Sandra (U. Tübingen, Germany)
López López, Aurelio (INAOE, Mexico)
Maegard, Bente (Centre for Language Technology, Denmark)
Martín-Vide, Carlos (U. Rovira i Virgili, Spain)
Mel'čuk, Igor (U. Montreal, Canada)
Metais, Elisabeth (U. Versailles, France)
Mihalcea, Rada (U. North Texas, USA)
Mitkov, Ruslan (U. Wolverhampton, UK)
Murata, Masaki (KARC-CRL, Japan)
Nevzorova, Olga (Kazan State U., Russia)
Nirenburg, Sergei (U. Maryland, USA)
Palomar, Manuel (U. Alicante, Spain)
Pedersen, Ted (U. Minnesota Duluth, USA)
Pekar, Viktor (U. Wolverhampton, UK)
Piperidis, Stelios (Inst. for Language and Speech Processing, Greece)
Pustejovsky, James (Brandeis U., USA)
Ren, Fuji (U. Tokushima, Japan)
Sag, Ivan (Stanford U., USA)
Sharoff, Serge (U. Leeds, UK)
Sidorov, Grigori (CIC-IPN, Mexico)
Sun, Maosong (Tsinghua U., China)
Tait, John (U. Sunderland, UK)
Trujillo, Arturo (Canon Research Centre Europe, UK)
T'sou Ka-yin, Benjamin (City U. Hong Kong, Hong Kong)
Van Guilder, Linda (MITRE Corp., USA)
Verspoor, Karin (Los Alamos National Laboratory, USA)
Vilares Ferro, Manuel (U. Vigo, Spain)
Wilks, Yorick (U. Sheffield, UK)

Best Paper Award Selection Working Group

Gelbukh, Alexander (**coordinator**)
Hovy, Eduard
Mihalcea, Rada
Pedersen, Ted
Wiks, Yorick

Additional Reviewers

Babych, Bogdan (U. Leeds, UK)
Bel Enguix, Gemma (Rovira i Virgili U., Spain)
Darriba Bilbao, Victor Manuel (U. Vigo, Spain)
Dediu, Adrian-Horia (U. Politehnica, Romania and Rovira i Virgili U., Spain)
Doi, Kouichi (Nara Inst. Science and Technology, Japan)
Fattah, Mohamed Abdel (U. Tokushima, Japan)
Iordachioaia, Gianina (U. Tübingen, Germany)
Korelsky, Tanya (CoGenTex Inc., USA)
Li, Yaoyong (U. Sheffield, UK)
Montes y Gómez, Manuel (INAOE, Mexico)
Montoyo Guijarro, Andrés (U. Alicante, Spain)
Nagy, Benedek (U. Debrecen, Hungary and Rovira i Virgili U., Spain)
Nechitaylov, Yury (St. Petersburg State U., Russia and Rovira i Virgili U., Spain)
Ribadas Peña, Francisco Jose (U. Vigo, Spain)
Solorio, Thamar (INAOE, Mexico)
Suárez, Armando (U. Alicante, Spain)
Vilares Ferro, Jesus (U. La Coruña, Spain)
Wang, Xiaojie (Beijing U. Posts and Telecommunications, China)
Wunsch, Holger (U. Tübingen, Germany)

Organizing Committee

Arreola Ceja, Arturo
Ayala, Juan Antonio
Calvo Castro, Hiram
Cu Tinoco, José Ángel
García Araoz, Ignacio
Gelbukh, Alexander (**chair**)
Muñoz Porras, Valentina
Haro Martínez, Martín
Sandoval Reyes, Alejandro
Torres Frausto, Raquel

Organization, Website, and Contact

The conference was organized by the Natural Language and Text Processing Laboratory of the Center for Computing Research (CIC, www.cic.ipn.mx) of the National Polytechnic Institute (IPN), Mexico City, Mexico. Website: www.CICLing.org. Contact: gelbukh@CICLing.org; also gelbukh@gelbukh.com; see www.gelbukh.com.

Table of Contents

Computational Linguistics Research

Computational Linguistics Formalisms

Invited Paper

An Overview of Probabilistic Tree Transducers for Natural Language Processing
Kevin Knight, Jonathan Graehl 1

A Modular Account of Information Structure in Extensible Dependency Grammar
Ralph Debusmann, Oana Postolache, Maarika Traat 25

Modelling Grammatical and Lexical Knowledge: A Declarative Approach
Palmira Marrafa 37

Constructing a Parser for Latin
Cornelis H.A. Koster 48

Parsing Korean Case Phenomena in a Type-Feature Structure Grammar
Jong-Bok Kim, Jaehyung Yang 60

A Computational Model of the Spanish Clitic System
Luis A. Pineda, Ivan V. Meza 73

A Parallel Approach to Syllabification
Anca Dinu, Liviu P. Dinu 83

Semantics and Discourse

Invited Paper

Towards Developing Probabilistic Generative Models for Reasoning with Natural Language Representations
Daniel Marcu, Ana-Maria Popescu 88

Putting Pieces Together: Combining FrameNet, VerbNet and WordNet for Robust Semantic Parsing
Lei Shi, Rada Mihalcea 100

Assigning Function Tags with a Simple Model
Vasile Rus, Kirtan Desai 112

Finding Discourse Relations in Student Essays
Rohana Mahmud, Allan Ramsay 116

Parsing and Syntactic Disambiguation

Regional Versus Global Finite-State Error Repair
Manuel Vilares, Juan Otero, Jorge Graña 120

Lexicalized Beam Thresholding Parsing with Prior and Boundary Estimates
Deyi Xiong, Qun Liu, Shouxun Lin 132

Best Paper Award (2nd Place)

Unsupervised Evaluation of Parser Robustness
Johnny Bigert, Jonas Sjöbergh, Ola Knutsson, Magnus Sahlgren 142

Mutual Information Independence Model Using Kernel Density Estimation for Segmenting and Labeling Sequential Data
Guodong Zhou, Lingpeng Yang, Jian Su, Donghong Ji 155

Applying Conditional Random Fields to Chinese Shallow Parsing
Yongmei Tan, Tianshun Yao, Qing Chen, Jingbo Zhu 167

Distributional Thesaurus Versus WordNet: A Comparison of Backoff Techniques for Unsupervised PP Attachment
Hiram Calvo, Alexander Gelbukh, Adam Kilgarriff 177

Morphology

Automatic Recognition of Czech Derivational Prefixes
Alfonso Medina Urrea, Jaroslava Hlaváčová 189

Korma 2003: Newly Improved Korean Morpheme Analysis Module for Reducing Terminological and Spacing Errors in Document Analysis
Ho-cheol Choi, Sang-yong Han 198

Word Extraction Based on Semantic Constraints in Chinese Word-Formation
Maosong Sun, Shengfen Luo, Benjamin K T'sou 202

Using Directed Graph Based BDMM Algorithm for Chinese Word Segmentation
Yaodong Chen, Ting Wang, Huowang Chen 214

Anaphora and Coreference

Entity-Based Noun Phrase Coreference Resolution
Xiaofeng Yang, Jian Su, Lingpeng Yang 218

The Right Frontier Constraint as Conditional
Claudia Sassen, Peter Kühnlein 222

Word Sense Disambiguation

Name Discrimination by Clustering Similar Contexts
Ted Pedersen, Amruta Purandare, Anagha Kulkarni 226

Word Sense Disambiguation by Semi-supervised Learning
Zheng-Yu Niu, Donghong Ji, Chew-Lim Tan, Lingpeng Yang 238

Crossing Parallel Corpora and Multilingual Lexical Databases for WSD
Alfio Massimiliano Gliozzo, Marcello Ranieri, Carlo Strapparava..... 242

A Mapping Between Classifiers and Training Conditions for WSD
Aarón Pancardo-Rodríguez, Manuel Montes-y-Gómez,
Luis Villaseñor-Pineda, Paolo Rosso..... 246

Multiwords and Word Sense Disambiguation
Victoria Arranz, Jordi Atserias, Mauro Castillo 250

Context Expansion with Global Keywords for a Conceptual Density-Based WSD
Davide Buscaldi, Paolo Rosso, Manuel Montes-y-Gómez 263

Two Web-Based Approaches for Noun Sense Disambiguation
Paolo Rosso, Manuel Montes-y-Gómez, Davide Buscaldi,
Aarón Pancardo-Rodríguez, Luis Villaseñor Pineda..... 267

Lexical Resources

Best Paper Award (1st Place)

Finding Instance Names and Alternative Glosses on the Web: WordNet Reloaded
Marius Paşca 280

Automatic Synonym Acquisition Based on Matching of Definition Sentences in Multiple Dictionaries
Masaki Murata, Toshiyuki Kanamaru, Hitoshi Isahara 293

Enriching WordNet with Derivational Subnets
Karel Pala, Radek Sedláček 305

Customisable Semantic Analysis of Texts
Vivi Nastase, Stan Szpakowicz 312

ITOLDU, a Web Service to Pool Technical Lexical Terms in a Learning Environment and Contribute to Multilingual Lexical Databases
Valérie Bellynck, Christian Boitet, John Kenwright 324

Building a Situation-Based Language Knowledge Base
Qiang Zhou, Zushun Chen 333

Unsupervised Learning of P NP P Word Combinations
Sofía N. Galicia-Haro, Alexander Gelbukh 337

Natural Language Generation

Evaluating Evaluation Methods for Generation in the Presence of Variation
Amanda Stent, Matthew Marge, Mohit Singhai 341

Reconciling Parameterization, Configurability and Optimality in Natural Language Generation via Multiparadigm Programming
Jorge Marques Pelizzoni, Maria das Graças Volpe Nunes 352

Machine Translation

Invited Paper

Message Automata for Messages with Variants, and Methods for Their Translation
Christian Boitet 357

The UNL Initiative: An Overview
Igor Boguslavsky, Jesús Cardeñosa, Carolina Gallardo, Luis Iraola 377

Interactive Resolution of Intrinsic and Translational Ambiguity in a Machine Translation System
Igor M. Boguslavsky, Leonid L. Iomdin, Alexander V. Lazursky, Leonid G. Mityushin, Victor G. Sizov, Leonid G. Kreydlin, Alexander S. Berdichevsky 388

Chinese-Japanese Clause Alignment
Xiaojie Wang, Fuji Ren 400

Direct Combination of Spelling and Pronunciation Information for Robust Back-Transliteration
Slaven Bilac, Hozumi Tanaka 413

Speech and Natural Language Interfaces

A Prosodic Diphone Database for Korean Text-to-Speech Synthesis System
Kyuchul Yoon 425

On a Pitch Detection Method Using Noise Reduction
Jongkuk Kim, Ki Young Lee, Myung Jin Bae 429

Toward Acoustic Models for Languages with Limited Linguistic Resources
Luis Villaseñor-Pineda, Viet Bac Le, Manuel Montes-y-Gómez, Manuel Pérez-Coutiño 433

A Study on Pitch Detection in Time-Frequency Hybrid Domain
Wangrae Jo, Jongkuk Kim, Myung Jin Bae 437

VoiceUNL: A Semantic Representation of Emotions Within Universal Networking Language Formalism Based on a Dialogue Corpus Analysis
Mutsuko Tomokiyo, Gérard Chollet 441

Combining Multiple Statistical Classifiers to Improve the Accuracy of Task Classification
Wei-Lin Wu, Ru-Zhan Lu, Feng Gao, Yan Yuan 452

Language Documentation

A Finite State Network for Phonetic Text Processing
Edward John Garrett 463

Language Documentation: The Nahuatl Grammar
Mike Maxwell, Jonathan D. Amith 474

Intelligent Text Processing Applications

Information Extraction

Invited Paper

Creating Subjective and Objective Sentence Classifiers from Unannotated Texts
Janyce Wiebe, Ellen Riloff..... 486

Instance Pruning by Filtering Uninformative Words: An Information Extraction Case Study
Alfio Massimiliano Gliozzo, Claudio Giuliano, Raffaella Rinaldi 498

Incremental Information Extraction Using Tree-Based Context Representations
Christian Siefkes 510

Best Paper Award (3rd Place)

Learning Information Extraction Rules for Protein Annotation from Unannotated Corpora
Jee-Hyub Kim, Melanie Hilario 522

Transformation-Based Information Extraction Using Learned Meta-rules
Un Yong Nahm 535

A Machine Learning Approach to Information Extraction
Alberto Téllez-Valero, Manuel Montes-y-Gómez, Luis Villaseñor-Pineda 539

Automatic Time Expression Labeling for English and Chinese Text
Kadri Hacioglu, Ying Chen, Benjamin Douglas 548

Integrating Natural Language Techniques in OO-Method
Isabel Díaz, Lidia Moreno, Inmaculada Fuentes, Oscar Pastor 560

Information Retrieval

Document Re-ordering Based on Key Terms in Top Retrieved Documents
Yang Lingpeng, Ji Donghong, Nie Yu, Guodong Zhou 572

Merging Case Relations into VSM to Improve Information Retrieval Precision
Hongtao Wang, Maosong Sun, Shaoming Liu 584

Evaluating Document-to-Document Relevance Based on Document Language Model: Modeling, Implementation and Performance Evaluation
Ge Yu, Xiaoguang Li, Yubin Bao, Daling Wang 593

Retrieval Efficiency of Normalized Query Expansion
Sofia Stamou, Dimitris Christodoulakis 604

Selecting Interesting Articles Using Their Similarity Based Only on Positive Examples
Jiří Hroza, Jan Žižka 608

Question Answering

Question Classification in Spanish and Portuguese
Thamar Solorio, Manuel Pérez-Coutiño, Manuel Montes-y-Gómez, Luis Villaseñor-Pineda, Aurelio López-López 612

Learning the Query Generation Patterns
Marcin Skowron, Kenji Araki 620

Exploiting Question Concepts for Query Expansion
Hae-Jung Kim, Ki-Dong Bu, Junghyun Kim, Sang-Jo Lee 624

Experiment on Combining Sources of Evidence for Passage Retrieval
Alexander Gelbukh, NamO Kang, Sang-yong Han 628

Summarization

Summarisation Through Discourse Structure
Dan Cristea, Oana Postolache, Ionuț Pistol 632

LexTrim: A Lexical Cohesion Based Approach to Parse-and-Trim Style Headline Generation
Ruichao Wang, Nicola Stokes, William Doran, Eamonn Newman, John Dunnion, Joe Carthy 645

Generating Headline Summary from a Document Set
Kamal Sarkar, Sivaji Bandyopadhyay 649

Extractive Summarization Based on Word Information and Sentence Position
Carlos Méndez Cruz, Alfonso Medina Urrea 653

Automatic Extraction and Learning of Keyphrases from Scientific Articles
Yaakov HaCohen-Kerner, Zuriel Gross, Asaf Masa 657

Automatic Annotation of Corpora for Text Summarisation: A Comparative Study
Constantin Orăsan 670

Text Classification, Categorization, and Clustering

Techniques for Improving the Performance of Naive Bayes for Text Classification
Karl-Michael Schneider 682

Efficient Modeling of Analogy
Lars G. Johnsen, Christer Johansson 694

A Supervised Clustering Method for Text Classification
Umarani Pappuswamy, Dumisizwe Bhembe, Pamela W. Jordan, Kurt VanLehn 704

Unsupervised Text Classification Using Kohonen's Self Organizing Network
Nirmalya Chowdhury, Diganta Saha 715

Enhancement of DTP Feature Selection Method for Text Categorization
Edgar Moyotl-Hernández, Héctor Jiménez-Salazar 719

FASiL Adaptive Email Categorization System
Yunqing Xia, Angelo Dalli, Yorick Wilks, Louise Guthrie 723

ESPClust: An Effective Skew Prevention Method for Model-Based Document Clustering
Xiaoguang Li, Ge Yu, Daling Wang, Yubin Bao 735

A Method of Rapid Prototyping of Evolving Ontologies
Pavel Makagonov, Alejandro Ruiz Figueroa 746

Named Entity Recognition

Resolution of Data Sparseness in Named Entity Recognition Using Hierarchical Features and Feature Relaxation Principle
Guodong Zhou, Jian Su, Lingpeng Yang 750

Learning Named Entity Recognition in Portuguese from Spanish
Thamar Solorio, Aurelio López-López 762

A Simple Rule-Based Approach to Organization Name Recognition in Chinese Text
Houfeng Wang, Wuguang Shi 769

Language Identification

Disentangling from Babylonian Confusion – Unsupervised Language Identification
Chris Biemann, Sven Teresniak 773

On the Syllabic Similarities of Romance Languages
Anca Dinu, Liviu P. Dinu 785

Automatic Language Identification Using Multivariate Analysis
Vinosh Babu James, Baskaran Sankaran 789

Spelling and Style Checking

Design and Development of a System for the Detection of Agreement Errors in Basque
Arantza Díaz de Ilarraza, Koldo Gojenola, Maite Oronoz 793

An Experiment in Detection and Correction of Malapropisms Through the Web
Igor A. Bolshakov 803

A Paragraph Boundary Detection System
Dmitriy Genzel 816

Author Index..... 827

An Overview of Probabilistic Tree Transducers for Natural Language Processing

Kevin Knight and Jonathan Graehl

Information Sciences Institute (ISI) and Computer Science Department,
University of Southern California
{knight, graehl}@isi.edu

Abstract. Probabilistic finite-state string transducers (FSTs) are extremely popular in natural language processing, due to powerful generic methods for applying, composing, and learning them. Unfortunately, FSTs are not a good fit for much of the current work on probabilistic modeling for machine translation, summarization, paraphrasing, and language modeling. These methods operate directly on trees, rather than strings. We show that tree acceptors and tree transducers subsume most of this work, and we discuss algorithms for realizing the same benefits found in probabilistic string transduction.

1 Strings

Many natural language problems have been successfully attacked with finite-state machines. It has been possible to break down very complex problems, both conceptually and literally, into cascades of simpler probabilistic **finite-state transducers** (FSTs). These transducers are bidirectional, and they can be trained on sample input/output string data. By adding a probabilistic **finite-state acceptor** (FSAs) language model to one end of the cascade, we can implement probabilistic **noisy-channel** models.[1] Figure 1 shows a cascade of FSAs and FSTs for the problem of transliterating names and technical terms across languages with different sounds and writing systems [1].

The finite-state framework is popular because it offers powerful, generic operations for statistical reasoning and learning. There are standard algorithms for:

- **intersection** of FSAs
- **forward application** of strings and FSAs through FSTs
- **backward application** of strings and FSAs through FSTs
- **composition** of FSTs
- **k-best** path extraction
- supervised and unsupervised **training** of FST transition probabilities from data

[1] In the noisy-channel framework, we look for the output string that maximizes P(output | input), which is equivalent (by Bayes Rule) to maximizing P(output) · P(input | output). The first term of the product is often captured by a probabilistic FSA, the second term by a probabilistic FST (or a cascade of them).

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 1–24, 2005.

Fig. 1. A cascade of probabilistic finite-state machines for English/Japanese transliteration [1]. At the bottom is an optically-scanned Japanese katakana string. The finite-state machines allow us to compute the most probable English translation (in this case, "Masters Tournament") by reasoning about how words and phonemes are transformed in the transliteration process

Even better, these generic operations are already implemented and packaged in research software toolkits such as AT&T's FSM toolkit [2], Xerox's finite-state calculus [3, 4], van Noord's FSA Utilities [5], the RWTH toolkit [6], and USC/ISI's Carmel [7].

Indeed, Knight & Al-Onaizan [8] describe how to use generic finite-state tools to implement the statistical machine translation models of [9]. Their scheme is shown in Figure 2, and [8] gives constructions for the transducers. Likewise, Kumar & Byrne [10] do this job for the phrase-based translation model of [11].

2 Trees

Despite the attractive computational properties of finite-state tools, no one is particularly fond of their representational power for natural language. They are not adequate for long-distance reordering of symbols needed for machine translation, and they cannot implement the trees, graphs, and variable bindings that have proven useful for describing natural language processes. So over the past several years, researchers have been developing probabilistic **tree-based** models for

- machine translation (e.g., [13, 14, 12, 15, 16, 17])
- summarization (e.g., [18])

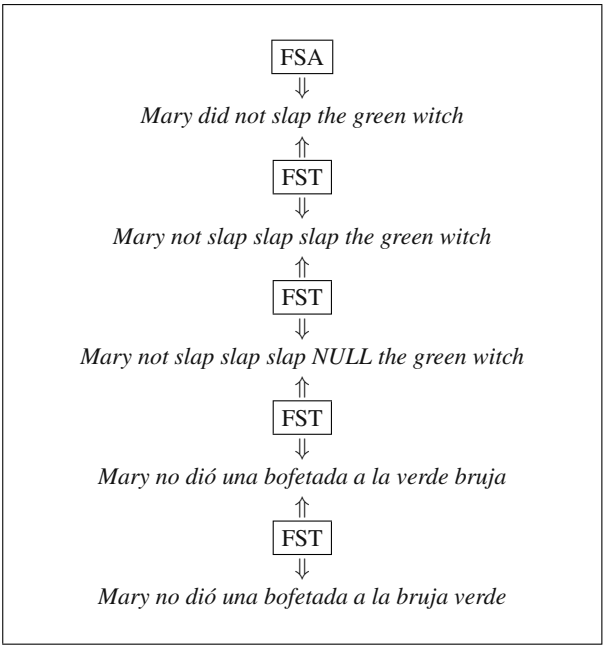


Fig. 2. The statistical machine translation model of [9] implemented with a cascade of standard finite-state transducers [8]. In this model, an observed Spanish sentence is "decoded" back into English through several layers of word substitution, insertion, deletion, and permutation

- paraphrasing (e.g., [19])
- natural language generation (e.g., [20, 21, 22])
- question answering (e.g., [23]), and
- language modeling (e.g., [24])

An example of such a tree-based model is shown in Figure 3. Unlike in the previous figures, the probabilistic decisions here are sensitive to syntactic structure.

We have found that most of the current syntax-based NLP models can be neatly captured by probabilistic **tree automata**. Rounds [25] and Thatcher [26] independently introduced top-down **tree transducers** as a generalization of FSTs. Rounds was motivated by problems in natural language:

> "Recent developments in the theory of automata have pointed to an extension of the domain of definition of automata from strings to trees ... parts of mathematical linguistics can be formalized easily in a tree-automaton setting ... We investigate decision problems and closure properties ... results should clarify the nature of syntax-directed translations and transformational grammars." [25]

Top-down tree transducers [25, 26] are often called **R-transducers**. (R stands for "**R**oot-to-frontier"). An R-transducer walks down an input tree, transforming it and processing branches independently in parallel. It consists of a set of simple transformation **productions** (or **rules**) like this:

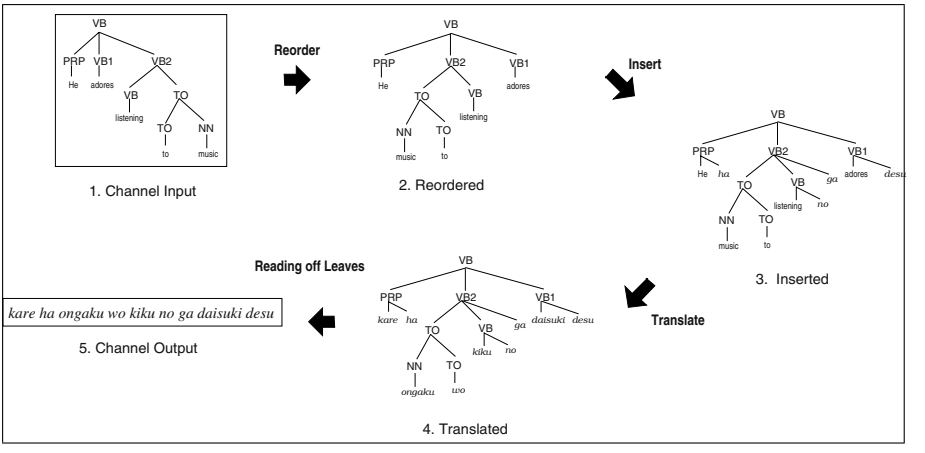
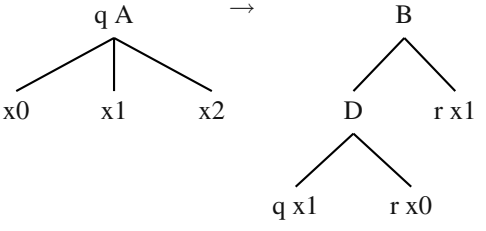


Fig. 3. A syntax-based machine translation model [12]



This production means:

> *When in state q, facing an input tree with root symbol A and three children* **about which we know nothing**, *replace it with a subtree rooted at B with two children, the left child being a D with two children of its own. To compute D's children, recursively process A's middle child (with state q) and A's left child (with state r). To compute B's right child, recursively process A's middle child (with state r).*

Here, "recursively" means to take the subtree rooted at A's child and look for productions that might in turn match it. R productions are limited to left-hand sides that match only on <state, symbol> pairs, while the right-hand side may have any height. Productions are often written textually, e.g.:

$$q\ A(x0, x1, x2) \rightarrow B(D(q\ x1, r\ x0), r\ x1)$$

If probabilities are attached to individual productions, then the tree transducer becomes a probabilistic tree transducer.

Figure 4 shows a probabilistic R-transducer based on a machine translation model similar to [12]. This model probabilistically reorders siblings (conditioned on the parent's and siblings' syntactic categories), inserts Japanese function words, and translates English words into Japanese, all in one top-down pass. It defines a conditional probability distribution P(J | E) over all English and Japanese tree pairs. Figure 5 shows an

```
/* translate */

1. q.s S(x0, x1)          →0.9 S(q.np x0, q.vp x1)
2. q.s S(x0, x1)          →0.1 S(q.vp x1, q.np x0)
3. q.np x                 →0.1 r.np x
4. q.np x                 →0.8 NP(r.np x, i x)
5. q.np x                 →0.1 NP(i x, r.np x)
6. q.pro PRO(x0)          →1.0 PRO(q x0)
7. q.nn NN(x0)            →1.0 NN(q x0)
8. q.vp x                 →0.8 r.vp x
9. q.vp x                 →0.1 S(r.vp x, i x)
10. q.vp x                →0.1 S(i x, r.vp x)
11. q.vbz x               →0.4 r.vbz x
12. q.vbz x               →0.5 VP(r.vbz x, i x)
13. q.vbz x               →0.1 VP(i x, r.vbz x)
14. q.sbar x              →0.3 r.sbar x
15. q.sbar x              →0.6 SBAR(r.sbar x, i x)
16. q.sbar x              →0.1 SBAR(i x, r.sbar x)
17. q.vbg VBG(x0)         →1.0 VP(VB(q x0))
18. q.pp PP(x0, x1)       →1.0 NP(q.np x1, q.p x0)
19. q.p P(x0)             →1.0 PN(q x0)
20. q he                  →1.0 kare
21. q enjoys              →0.1 daisuki
22. q listening           →0.2 kiku
23. q to                  →0.1 o
24. q to                  →0.7 ni
25. q music               →0.8 ongaku
26. r.vp VP(x0, x1)       →0.9 S(q.vbz x0, q.np x1)
27. r.vp VP(x0, x1)       →0.1 S(q.np x1, q.vbz x0)
28. r.sbar SBAR(x0, x1)   →0.1 S(q.vbg x0, q.pp x1)
29. r.sbar SBAR(x0, x1)   →0.9 S(q.pp x1, q.vbg x0)
30. r.np NP(x0)           →0.1 q.pro x0
31. r.np NP(x0)           →0.8 q.nn x0
32. r.np NP(x0)           →0.1 q.sbar x0
33. r.vbz VBZ(x0)         →0.7 VB(q x0)

/* insert */

34. i NP(x0)              →0.3 PN(wa)
35. i NP(x0)              →0.3 PN(ga)
36. i NP(x0)              →0.2 PN(o)
37. i NP(x0)              →0.1 PN(ni)
38. i SBAR(x0, x1)        →0.7 PS(no)
39. i VBZ(x0)             →0.2 PV(desu)
```

Fig. 4. A probabilistic tree transducer

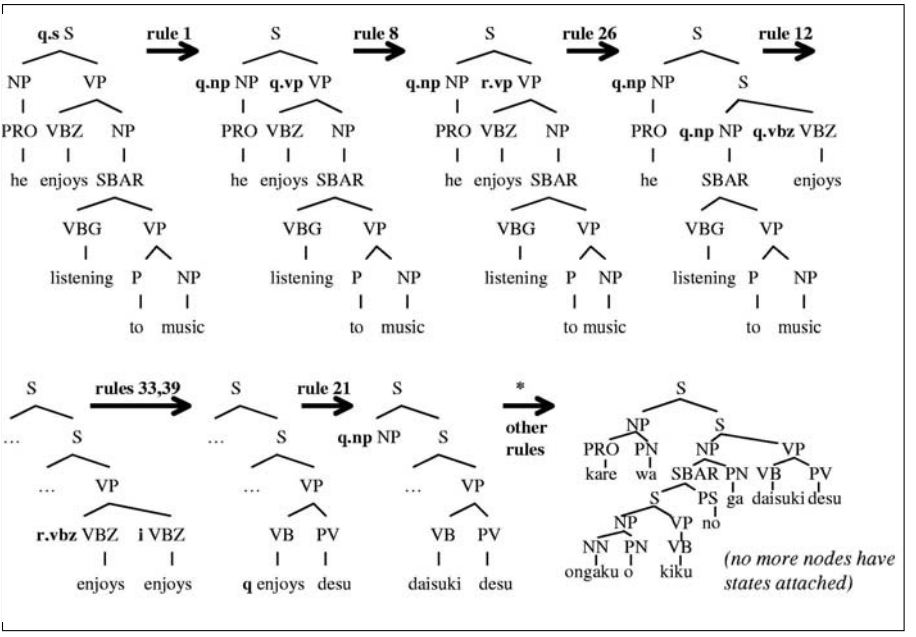


Fig. 5. An English (input) tree being transformed into a Japanese (output) tree by the tree transducer in the previous figure. Because the transducer is non-deterministic, many other output trees are also possible

R-derivation from one English input tree (with start state q.s) to one possible Japanese output.

Figure 6 shows the corresponding **derivation tree**, which is a record of the rules used in the derivation shown in Figure 4. There are several things worth noting:

- Some rules (like Rule 4) may appear multiple times in the same derivation tree.
- Both the input and output trees can be recovered from the derivation tree, though this takes some work.
- There are usually many derivation trees connecting the same input tree (in this case, English) with the same output tree (in this case, Japanese), i.e., multiple ways of "getting from here to there."

The fact that we can use tree transducers to capture tree-based models proposed in the natural language literature is significant, because extensive work goes into each of these models, for both training and decoding. These are one-off solutions that take a long time to build, and they are difficult to modify. By casting translation models as R-transducers, Graehl & Knight [27] discuss how to add productions for linguistic phenomena not captured well by previous syntax-based translation models, including non-constituent phrase translation, lexicalized reordering, and long-distance wh-movement (Figure 7). Having generic, implemented R operations would allow researchers to focus on modeling rather than coding and specialized algorithm development.

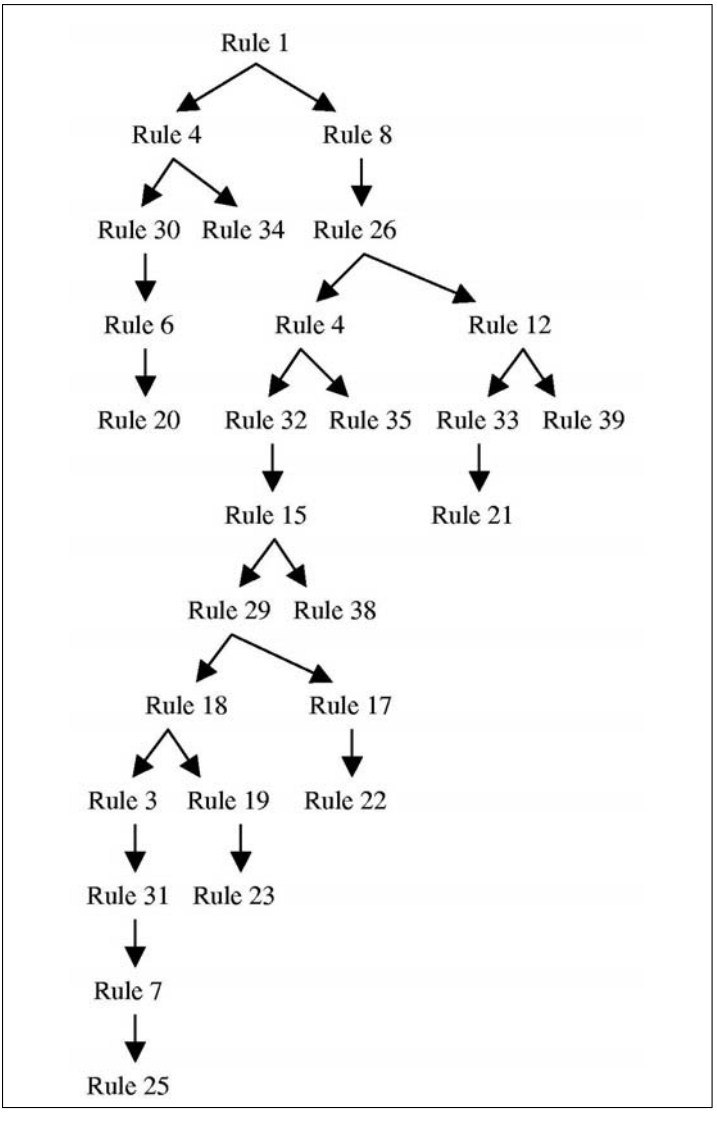


Fig. 6. A derivation tree, or record of transducer rules used to transform a particular input tree into a particular output tree. With careful work, the input and output trees can be recovered from the derivation tree

The purpose of this paper is to explore whether the benefits of finite-state string automata can be enjoyed when we work with trees—i.e., what are the implications of trying to reason with tree models like Figure 4, in transducer cascades like those shown in Figures 1 and 2? We survey the extensive literature on tree automata from the focused viewpoint of large-scale statistical modeling/training/decoding for natural language.

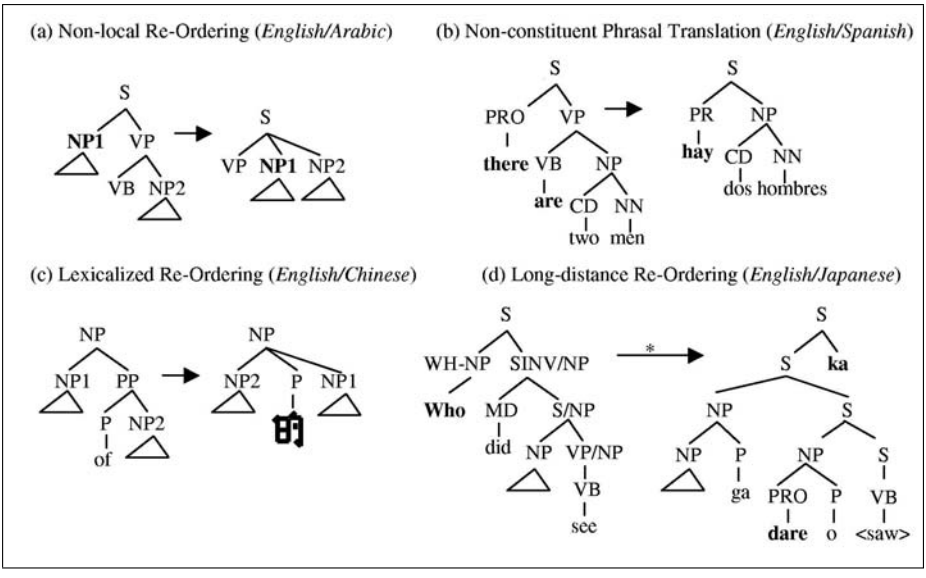


Fig. 7. Tree transducer operations for capturing translation patterns

In this paper, we give many examples of tree transducers and discuss their consequences, but we will not give formal mathematical definitions and proofs (e.g., "... a tree transducer is a 7-tuple ..."). For readers wishing to dig further into the tree automata literature, we highly recommend the excellent surveys of Gécseg & Steinby [28] and Comon et al [29].

Because tree-automata training is already discussed in [27, 30], this paper concentrates on *intersection*, *composition*, forward and backward *application*, and *search*.

3 Intersection

In finite-state string systems, FSAs represent sets of (weighted) strings, such as English sentences. FSAs can capture only regular languages (which we refer to here as **regular string languages** (RSLs) to distinguish them from tree languages). Other ways to capture RSLs include regular grammars and regular expressions. The typical operation on FSAs in a noisy-channel cascade is the intersection of channel-produced candidate strings with an FSA language model. It is very convenient to use the same English language model across different applications that need to map input into English (e.g., translation).

When working with trees, an analog of the RSL is the **regular tree language**, which is a (possibly infinite) set of trees. A probabilistic RTL assigns a P(tree) to every tree. An RTL is usually specified by a **regular tree grammar** (RTG), an example of which is shown in Figure 8. Alternatively, there exists an RTL recognition device [31] that crawls over an input, R-style, to accept or reject it. This device is the analog of an FSA.

Σ = {S, NP, VP, PP, PREP, DET, N,
V, run, the, of,
sons, daughters}
N = {qnp, qpp, qdet, qn, qprep}
S = q
P = {q $\rightarrow^{1.0}$ S(qnp, VP(VB(run))),
qnp $\rightarrow^{0.6}$ NP(qdet, qn),
qnp $\rightarrow^{0.4}$ NP(qnp, qpp),
qpp $\rightarrow^{1.0}$ PP(qprep, np),
qdet $\rightarrow^{1.0}$ DET(the),
qprep $\rightarrow^{1.0}$ PREP(of),
qn $\rightarrow^{0.5}$ N(sons),
qn $\rightarrow^{0.5}$ N(daughters)}

Sample accepted trees:

S(NP(DET(the), N(sons)),
VP(VB(run)))

(*prob = 0.3*)

S(NP(NP(DET(the), N(sons)),
PP(PREP(of),
NP(DT(the),
N(daughters)))),
VP(VB(run)))

(*prob = 0.036*)

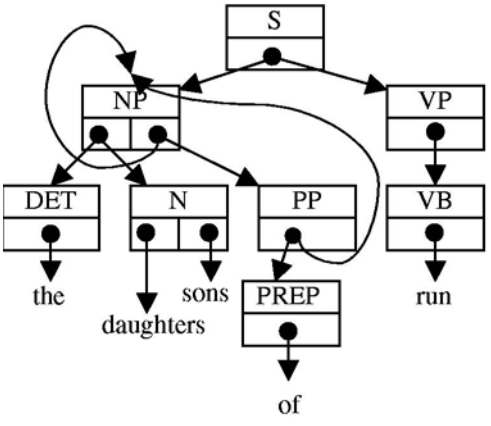


Fig. 8. A sample probabilistic regular tree grammar (RTG). This RTG accepts/generates an infinite number of trees, whose probabilities sum to one

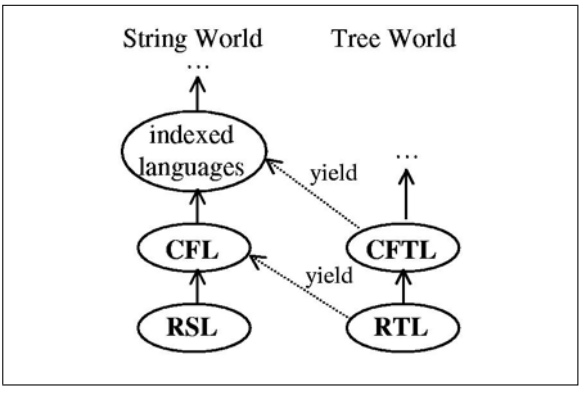


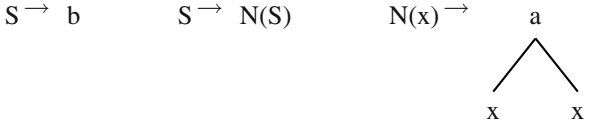
Fig. 9. String language classes and tree language classes. These classes include regular string languages (RSL), regular tree languages (RTL), context-free (string) languages (CFL), context-free tree languages (CFTL), and indexed (string) languages

In contrast with FSAs, non-deterministic top-down RTL recognizers are strictly more powerful than deterministic ones.

When working with trees, an analog of the RSL is the **regular tree language**, which is a (possibly infinite) set of trees. A probabilistic RTL assigns a P(tree) to every tree. An RTL is usually specified by a **regular tree grammar** (RTG), an example of which is shown in Figure 8. Alternatively, there exists an RTL recognition device [31] that crawls over an input, R-style, to accept or reject it. This device is the analog of an FSA. In contrast with FSAs, non-deterministic top-down RTL recognizers are strictly more powerful than deterministic ones. There exists a standard hierarchy of tree language classes, summarized in Figure 9. Some sets of trees are not RTL. An example is shown in Figure 10—left and right subtrees are always of equal depth, but there is no way to guarantee this for arbitrary depth, as a regular tree grammar must produce the subtrees independently.

The connection between string and tree languages (Figure 9) was observed by Rounds [25]—if we collect the leaf sequences (**yields**) of the trees in an RTL, we are guaranteed to get a string set that is a **context-free language** (CFL). Moreover, every CFL is the yield language of some RTL, and the derivation trees of a CFG form an RTL. However, an RTL might not be the set of derivation trees from any CFG (unless re-labeling is done).

The tree language in Figure 10 is not RTL, but it is a context-free tree language (CFTL). CFTG allows the left-hand side of a production to have variables—the CFTG for Figure 10 is:



The yield language of this non-RTL is $\{b^{2^n} : n \geq 0\}$, which is not a CFL. We do not pursue CFTG further.

How do the properties of tree languages stack up against string languages? Figure 11 summarizes. The good news is that RTLs have all of the good properties of RSLs. In particular, RTLs are closed under intersection, so there exists an algorithm to intersect two RTGs, which allows probabilistic RTG language models to be intersected with possibly infinite sets of candidate trees coming through the noisy channel. RTGs are therefore a suitable substitute for FSAs in such probabilistic cascades.

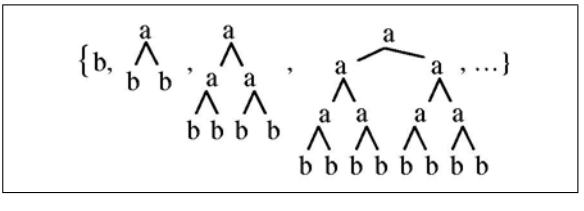


Fig. 10. A tree language that is not a regular tree language (RTL)

	Strings		Trees
	RSL (FSA, regexp)	CFL (PDA, CFG)	RTL (RTA, RTG)
closed under union	YES ([32] p. 59)	YES ([32] p. 131)	YES ([28] p. 72)
closed under intersection	YES ([32] p. 59)	NO ([32] p. 134)	YES ([28] p. 72)
closed under complement	YES ([32] p. 59)	NO ([32] p. 135)	YES ([28] p. 73)
membership testing	O(n) ([32] p. 281)	$O(n^3)$ ([32] p. 140)	O(n) ([28] p. 110)
emptiness decidable	YES ([32] p. 64)	YES ([32] p. 137)	YES ([28] p. 110)
equality decidable	YES ([32] p. 64)	NO ([32] p. 203)	YES ([28] p. 110)

Fig. 11. Properties of string and tree language classes

It is also interesting to look at how RTGs can capture probabilistic syntax models proposed in the natural language literature. RTGs easily implement **probabilistic CFG** (PCFG) models initially proposed for tasks like natural language parsing[2] and language modeling.[3] These models include parameters like P(NP → PRO | NP). However, PCFG models do not perform very well on parsing tasks. One useful extension is that of Johnson [33], which further conditions a node's expansion on the node's parent, e.g., P(NP → PRO | NP, parent=S). This captures phenomena such as pronouns appearing in subject position more frequently than in object position, and it leads to better performance. Normally, Johnson's extension is implemented as a straight PCFG with new node types, e.g., P(NP^S → PRO^{NP} | NP^S). This is unfortunate, since we are interested in getting out trees with labels like NP and PRO, not NP^S and PRO^S. An RTG elegantly captures the same phenomena without changing the node labels:

{qstart → S(qnp.s, qvp.s)
 qvp.s → VP(qv.vp, qnp.vp)
 qnp.s $\rightarrow^{0.3}$ NP(qpro.np)
 qnp.s $\rightarrow^{0.7}$ NP(qdet.np, qn.np)
 qnp.vp $\rightarrow^{0.05}$ NP(qpro.np)
 qnp.vp $\rightarrow^{0.95}$ NP(qdet.np, qn.np)
 qpro.np $\rightarrow^{0.5}$ PRO(he)
 qpro.np $\rightarrow^{0.5}$ PRO(him)
 ...}

Here we can contrast the 0.3 value (pronouns in subject position) with the 0.05 value (pronouns in object position).

A drawback to the above model is that the decision to generate "he" or "him" is not conditioned on the subject/object position in the sentence. This can also be addressed in an RTG—the more context is useful, the more states can be introduced.

[2] The parsing task is frequently stated as selecting from among the parse trees of a input sentence the one with highest P(tree).

[3] The language modeling task is to assign a high P(tree) only to grammatical, sensible English trees.

High-performing models of parsing and language modeling [34, 24] are actually lexicalized, with probabilistic dependencies between head words and their modifiers, e.g., P(S=sang → NP=boy VP=sang | S=sang). Because they are trained on sparse data, these models include extensive backoff schemes. RTGs can implement lexicalized models with further use of states, though it is open whether good backoff schemes can be encoded by compact RTGs.

Another interesting feature of high-performing syntax models is that they use a **markov** grammar rather than a finite list of rewrite rules. This allows them to recognize or build an infinite number of parent/children combinations, e.g., S → NP VP PP*. Formal machinery for this was introduced by Thatcher [35], in his **extended context-free grammars** (ECFG). ECFGs allow regular expressions on the right-hand side of productions, and they maintain good properties of CFGs. Any string set represented by an ECFG can also be captured by a CFG, though (of course) the derivation tree set may not be captured.

While standard PCFG grammars are parameterized like this

$$\mathrm{P}_{rule}(\mathrm{S} \rightarrow \mathrm{ADV\ NP\ VP\ PP\ PP} \mid \mathrm{S}),$$

markov grammars [34, 24] are parameterized somewhat like this:

$$\begin{aligned}
&\mathrm{P}_{head}(\mathrm{VP} \mid \mathrm{S}) \cdot \\
&\mathrm{P}_{left}(\mathrm{NP} \mid \mathrm{VP, S}) \cdot \\
&\mathrm{P}_{left}(\mathrm{ADV} \mid \mathrm{VP, S}) \cdot \\
&\mathrm{P}_{left}(\mathrm{STOP} \mid \mathrm{VP, S}) \cdot \\
&\mathrm{P}_{right}(\mathrm{PP} \mid \mathrm{VP, S}) \cdot \\
&\mathrm{P}_{right}(\mathrm{PP} \mid \mathrm{VP, S}) \cdot \\
&\mathrm{P}_{right}(\mathrm{STOP} \mid \mathrm{VP, S})
\end{aligned}$$

We can capture this exactly by replacing the right-hand side of each ECFG production with a probabilistic FSA, with transitions weighted by the parameter values above, resulting in a **probabilistic ECFG**. The extended CFG notion can then be carried over to make an **extended probabilistic RTG**, which is a good starting point for capturing state-of-the-art syntax models.

4 Tree Transducer Hierarchy

Before turning to tree transducer composition, we first cover the rich class hierarchy of tree transducers. Some of these classes are shown in Figure 12. (This figure was synthesized from many sources in the tree automata literature). Standard acronyms are made of these letters:

- **R**: Top-down transducer, introduced before.
- **F**: Bottom-up transducer ("**F**rontier-to-root"), with similar rules, but transforming the leaves of the input tree first, and working its way up.
- **L**: **Linear** transducer, which prohibits copying subtrees. Rule 4 in Figure 4 is example of a copying production, so this whole transducer is R but not RL.

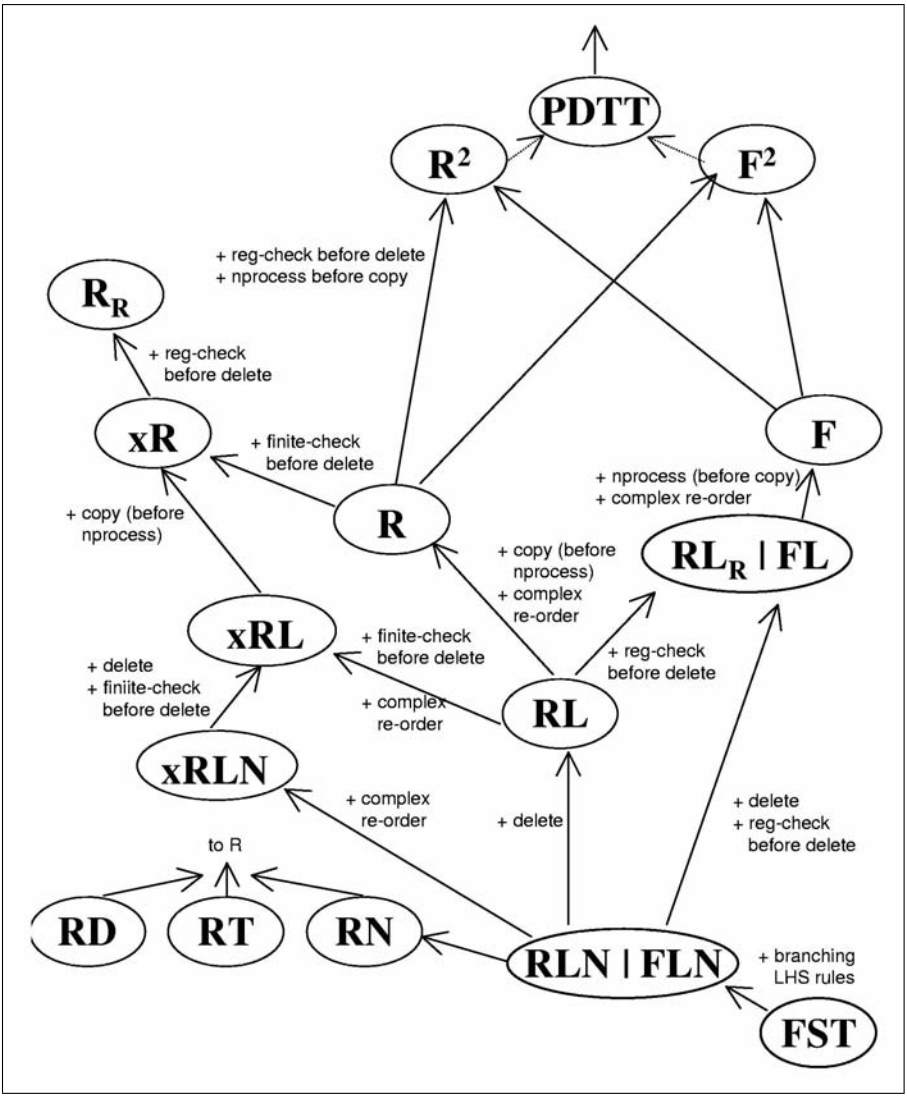


Fig. 12. Some tree transducer classes. Upward arrows indicate increasing levels of transduction power (higher classes can capture more kinds of transductions). Arrows are labeled with a rough description of the power that is added by moving to the higher class. R^2 represents the transduction capability of two R-transducers chained together

– **N**: **Non-deleting** transducer, which requires that every left-hand-side variable also appear on the right-hand side. A deleting R-transducer can simply delete a subtree (without inspecting it). The transducer in Figure 4 is the deleting kind, because of rules 34-39. It would also be deleting if it included a rule for dropping English determiners, e.g., q NP(x0, x1) → q x1.

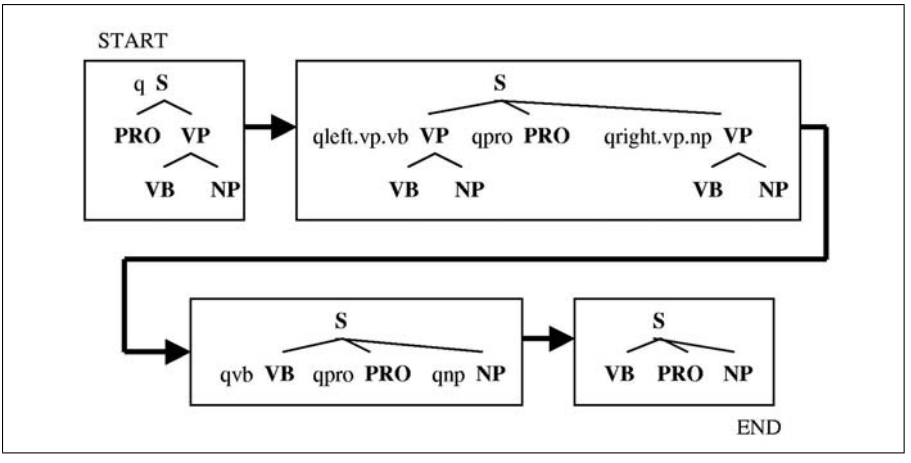
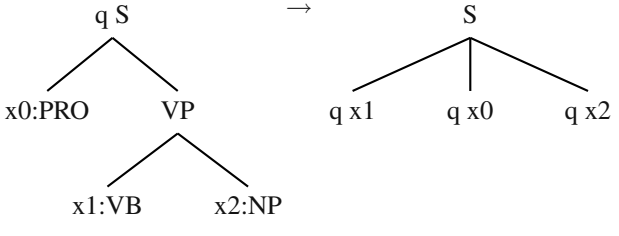


Fig. 13. Complex re-ordering with an R-transducer

- **D**: **Deterministic** transducer, with a maximum of one production per <state, symbol> pair.
- **T**: **Total** transducer, with a minimum of one production per <state, symbol> pair.
- **PDTT**: Push-down tree transducer, the transducer analog of CFTG [36].
- subscript$_R$: **Regular-lookahead** transducer, which can check to see if an input subtree is tree-regular, i.e., whether it belongs to a specified RTL. Productions only fire when their lookahead conditions are met.

We also introduce the prefix **x** for transducers with **extended left-hand-side** productions[4] that can look finitely deeply into the input, performing tests or grabbing deeper material. xR-transducers are easier to write; for example, we can have a production like



which moves a subject pronoun in between the verb and direct object, as happens in machine translation between certain language pairs.

Actually, this case can be captured with R, through the use of states and copying, as demonstrated with these productions:

[4] *Extended* left-hand-side productions are not related to *extended* context-free grammars—the notions unfortunately overwork the same adjective.

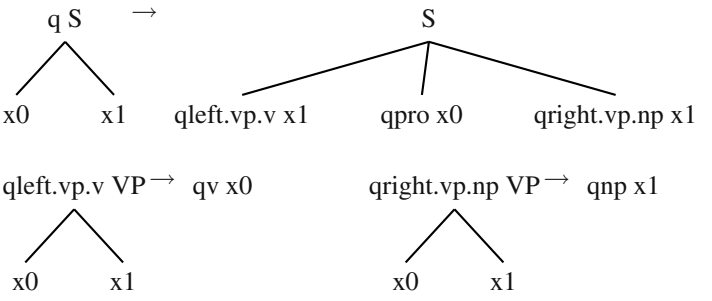


Figure 13 shows how the transformation is carried out. This kind of rule programming is cumbersome, however, and the single xR production above is preferred.

Because of their good fit with natural language applications, extended left-hand-side productions were briefly touched on already in Section 4 of Rounds' paper [25], though not defined. xR cannot quite be simulated by R, because xR has the power to check the root label of a subtree before deleting that subtree, e.g.:

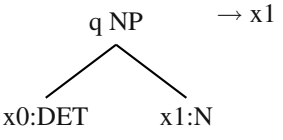


Figure 12 shows that R and F are incomparable. R can delete subtrees without looking at them, while F cannot. F can non-deterministically modify a tree bottom-up, then copy the result, while R has to make the copies first, before modifying them. Since they are modified independently and in parallel, R cannot guarantee that copies are modified in the same way.

We can also see that non-copying RL and FL transducers have reduced power. Prohibiting both deletion and copying removes the differences between R and F, so that RLN = FLN. Regular lookahead adds power to R [37].

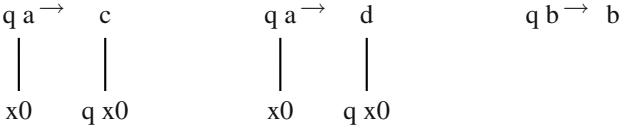
The string-processing FST appears in Figure 12 at the lower right. If we write strings vertically instead of horizontally, then the FST is just an RLN transducer with its left-hand-side productions limited to one child each. Standard FSTs are non-deleting. FSTs can also have transitions with both epsilon input and epsilon output. We show the tree analog of this in Rule 3 of Figure 4; we note that proofs in the tree automata literature do not generally work through epsilon cases.

5 Composition

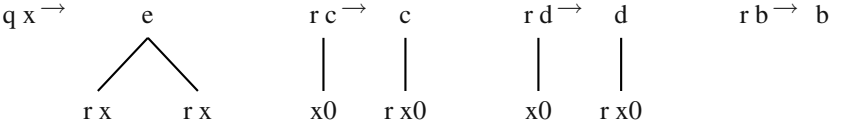
Now we turn to composition. For the string case, FSTs are closed under composition, which means that a long FST cascade can always be composed offline into a single FST before use.

By contrast, R is not closed under composition, as demonstrated by Rounds [25]. The proof is as follows. We set up one R-transducer to non-deterministically modify a

monadic (non-branching) input tree composed of some number of a's followed by a b; this transducer changes some of the a's to c's and other a's to d's:

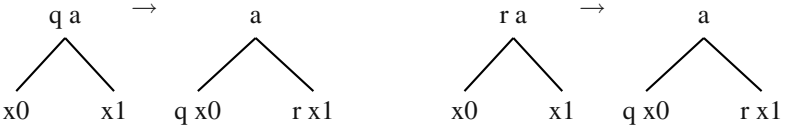


We then set up another R-transducer that simply places two copies of its input under a new root symbol e:



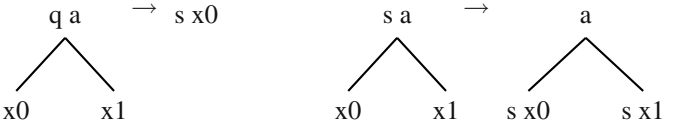
Each of these two transformations is R, but no single R-transducer can do both jobs at once, i.e., non-deterministically modify the input and make a copy of the result. The copy has to be made first, in which case both copies would be processed independently and the branches would diverge. The fact that R is not closed under composition is denoted in Figure 12 by the fact that R^2 (the class of transformations that can be captured with a sequence of two R-transducers) properly contains R.

RL is also not closed under composition [38]. To show this, we set up one RL transducer to pass along its input tree untouched, but only in case the right branch passes a certain test (otherwise it rejects the whole input):



q b → b q f → f r f → f

We set up another RL transducer to simply delete the right branch of its input:



s b → b s f → f

No transducer can do both jobs at once, because no R or RL transducer can check a subtree first, then delete it.

Better news is that RLN (= FLN) is closed under composition, and it is a natural class for many of the probabilistic tree transformations we find in the natural language literature. If we are fond of deleting, then FL is also closed under composition. RTD is also closed, though total deterministic transformations are not of much use in statistical

modeling. There are also various other useful compositions and decompositions [38], such as the fact that RL composed with RLN is always RL. These decompositions can help us analyze transducer cascades with mixed transducer types.

We note in passing that none of the tree transducers are closed under intersection, but string FSTs fare no better in this regard (Figure 14). We also note that all the listed transducers are as efficiently trainable as FSTs, given sample tree pairs [27].

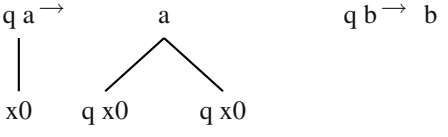
6 Forward and Backward Application

Next we turn to transducer application, e.g., what happens if we send an input tree through an R-transducer? What form will the output take?

First, the string case: if we feed an input FSA to an FST, the output (**image**) is always an FSA. Likewise, if we present an observed output FSA, we can ask the FST what inputs might have produced any of those strings (**inverse image**), and this is also always an FSA. In practice, the situation is even simpler—we turn input FSAs into identity FSTs, then use FST composition in place of application.

For trees, we can also create an identity tree transducer out of any RTG, but general composition may not be possible, as described in the previous section. However, straight application is a different story—known results are summarized in the remainder of Figure 14.

The bad news is that R does not preserve regularity. For example, consider the transducer



	Strings	Trees				
	FST	R	RL	RLN (= FLN)	F	FL
closed under composition	YES	NO [28] p. 162	NO [28] p. 158	YES from FL	NO [28], p. 162	YES [28], p. 158
closed under intersection	NO	NO	NO	NO	NO	NO
efficiently trainable	YES [39]	YES [27]	YES from R	YES from R	YES	YES
image of tree is:	RSL [28], p. 52	RTL	RTL [28], p. 175	RTL from R	not RTL	RTL [28], p. 174
inverse image of tree is:	RSL [28], p. 52	RTL [28], p. 162	RTL from R	RTL from R	RTL [28], p. 162	RTL from F
image of RTL is:	RSL [28], p. 52	not RTL [28], p. 179	RTL [28], p. 175	RTL from R	not RTL [28], p. 179	RTL [28], p. 174
inverse image of RTL is:	RSL [28], p. 52	RTL [28], p. 162	RTL from R	RTL from R	RTL [28], p. 162	RTL from F

Fig. 14. Properties of string and tree transducers

If we supply this transducer with an input RTL consisting of all the monadic trees a*b, then we get the non-RTL output tree set shown back in Figure 10. The same holds for F. In fact, the situation is worse for F, because even a single input tree can produce non-RTL output.

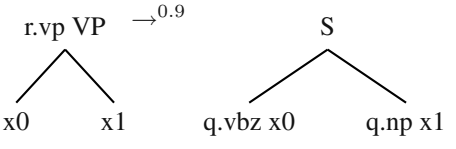
The good news is that sending a single tree (or by extension a finite RTL) through R guarantees RTL output. Moreover, the inverse image of an RTL through any of R, RL, RLN, F, or FL is still RTL. This is relevant to our noisy-channel cascade, in which the observed tree is passed backwards through the transducer cascade until it reaches the language model. This will work even for a cascade of R-transducers, despite the fact that it is not possible to compose those transducers off-line in the general case.

Getting a correct inverse image is tricky when the transducer is a deleting one (e.g., R or RL). A fully-connected RTL, capable of generating any tree in the language, must be inserted at every delete-point going backwards.[5] Copying also complicates inverse imaging, but it can be handled with RTL intersection.

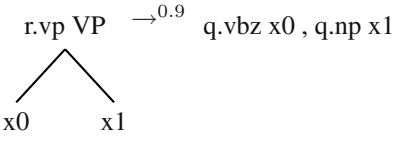
7 Tree-to-String Transducers

Figure 15 shows properties of **tree-to-string** transducers, denoted with suffix **s**. Tree-to-string transducers were introduced by Aho and Ullman [40] for compilers. Several recent machine translation models are also tree-to-string [12, 30].

We can easily turn the R-transducer in Figure 4 into an Rs-transducer by removing the internal structure on the right-hand-sides of productions, yielding a comma-separated list of leaves. For example, instead of rule 26:



we use:



The sample input English tree of Figure 5 would then probabilistically transduce into a flat Japanese string, e.g.:

kare , wa , ongaku , o , kiku , no , ga , daisuki , desu

In contrast with R, the empty string is allowed in the right-hand side of Rs productions.

[5] In practice, we may think twice about using delete-before-check to delete an English determiner, e.g., q NP(x0 x1) → x1. In decoding from a foreign language without determiners, the inverse image will contain every imaginable English subtree vying to fill that determiner slot.

	Tree-to-String		
	Rs	RLs	RLNs
closed under composition	n.a.	n.a.	n.a.
image of tree is:	CFL	CFL	CFL
inverse image of string is:	RTL	RTL	RTL
image of RTL is:	not CFL	CFL	CFL
inverse image of RSL is:	RTL	RTL	RTL

Fig. 15. Properties of tree-to-string transducers. These properties derive from the imaging behaviors of R, RL, and RLN, respectively

The inverse image of a string through Rs is guaranteed to be an RTL, as is the inverse image of a whole FSA. This result is similar to results that permit parsing of lattices by CFGs [41]. This means we can put an Rs transducer at the end of a tree transducer cascade, accepting observed inputs in string form (as we usually observe natural language inputs). Or if we have noisy input, as from speech or optical character recognition, then we can accept a whole FSA.

We may also want to use tree-to-string transducers in the forward direction, perhaps passing the result onto a string-based language model. In this case, we note that if the input to an RLs-transducer is an RTL, then the output is guaranteed to be a CFL. However, if we use an Rs-transducer, then the output may be non-CFL, as in the non-context-free yield language of the tree set in Figure 10.

8 Search

To solve problems with string automata, we often send an observed string backwards through a noisy-channel cascade of FSTs and FSAs—this ultimately transforms the string into a probabilistic FSA which encodes and ranks all possible "answer strings." We can then extract the best answer from this FSA with Dijkstra's algorithm [42], and we can extract the k-best paths with the fast algorithm of Eppstein [43], as is done in Carmel [7]. Mohri and Riley [44] also describe how to extract the k-best distinct strings from a weighted automaton.

The situation is similar with tree transducers. Once we manage to send our observed tree (or string, in the case of Rs) back through a noisy-channel cascade of transducers/acceptors, we wind up with a probabilistic RTG of answer trees. We next want to extract the best trees from that RTG. In general, this RTG may represent an infinite number of trees; related problems have been studied for finite forests in [45, 46].

Knuth [47] has considered how to extract the highest-probability derivation tree from a probabilistic CFG (what he calls the **grammar problem**).[6] Here, we adapt his algorithm for RTG extraction. Knuth gives an explicit quadratic algorithm and points to an improvement using priority queues, which we work out in Algorithm 1. This is

[6] Knuth's setting is actually more general—he considers weighted CFGs and various weight-combination functions.

Algorithm 1 Adaptation of Knuth's algorithm to extract the best tree from an RTG.

Input: **RTG** with n nonterminals N, and e productions indexed by $1 \leq i \leq e$: with rhs nonterminals T_i and lhs h_i, with a rule cost function of the form $c_i(\alpha : T_i \rightarrow \mathbb{R}) = w_i + \sum_{x \in T_i} \#_i(x)\alpha(x)$, where $\#_i(x)$ is the number of occurrences of nonterminal t in production i, and $w_i = \log P(i|h_i)$.

Output: For all $Y \in N$, $\pi[Y] = i$ is the index of the production that builds the cheapest tree possible from nonterminal $h_i = Y$ (with $\pi[T_i]$ recursively giving the cheapest trees for the child nonterminals), and $\mu[Y]$ is cost of that tree. $\pi[Y] = 0$ if there are no complete derivations from Y. Time complexity is $O(n \lg n + t)$ where (t is the total size of the input) if a Fibonacci heap is used, or $O(e \lg n + t)$ if a binary heap is used.

begin
- **for** $y \in N$ **do**
  - $\mu[y] \longleftarrow \infty$
  - $\pi[y] \longleftarrow 0$
  - $Adj[y] \longleftarrow \{\}$
- $\mu[\omega] \longleftarrow 0$
- $\pi[\omega] \longleftarrow 0$
- $Adj[\omega] \longleftarrow \{\}$
- $P \longleftarrow$ ***HEAP-CREATE***()
- ***HEAP-INSERT***$(P, \omega, 0)$
- **for** $1 \leq i \leq e, T_i = \{x_1, \ldots, x_k\}$ **do**
  - $w[i] \longleftarrow w_i$
  - **if** $k = 0$ **then**
    - `/* fictitious sink nonterminal in rhs: ω */`
    - $k \longleftarrow 1, x_1 \longleftarrow \omega$
  - $r[i] \longleftarrow k$ `/* r[i] is the number of rhs nonterminals remaining before production i's cost is known. */`
  - **for** $1 \leq j \leq k$ **do** $Adj[x_j] \longleftarrow Adj[x_j] \cup \{i\}$
- **while** $P \neq \emptyset$ **do**
  - $y \longleftarrow$ ***HEAP-EXTRACT-MIN***(P)
  - **for** $i \in Adj[y]$ **do**
    - **if** $w[i] < \mu[h_i]$ **then**
      - $w[i] \longleftarrow w[i] + \#_i(y)\mu[y]$
      - $r[i] \longleftarrow r[i] - 1$
      - **if** $r[i] = 0$ **then**
        - **if** $w[i] < \mu[h_i]$ **then**
          - **if** $\mu[h_i] = \infty$ **then** ***HEAP-INSERT***$(P, h_i, w[i])$
          - **else** $HEAP - DECREASE - KEY(P, h_i, w[i])$, $\pi[h_i] \longleftarrow i$,
          - $\mu[h_i] \longleftarrow w[i]$

end

a generalization of Dijkstra's algorithm, greedily finalizing the cost of one RTG nonterminal at a time and remembering the lowest-cost production used to expand it. Each production potentially reduces the cost of its left-hand-side exactly once, when the last nonterminal in its right-hand-side is finalized.

To extract the k-best trees, it is useful to view RTG extraction as a hypergraph shortest path problem [48, 49]. It appears that no very efficient k-best hyperpath algorithms exist yet.

9 Conclusion

We have investigated whether the benefits of finite-state string automata (elegant, generic operations in support of probabilistic reasoning for natural language processing) can be carried over into modeling with trees. We have distilled relevant theory and algorithms from the extensive literature on tree automata, and our conclusion is positive.

Many open problems now exist in finding efficient algorithms to support what is theoretically possible, and in implementing these algorithms in software toolkits. Here we list some of the problems of interest:

1. What is the most efficient algorithm for selecting the k-best trees from a probabilistic regular tree grammar (RTG)?
2. How can efficient integrated search be carried out, so that all tree acceptors and transducers in a cascade can simultaneously participate in the best-tree search?
3. What search heuristics (beaming, thresholding, etc.) are necessary for efficient application of tree transducers to large-scale natural language problems?
4. What is the most efficient algorithm for composing probabilistic linear, non-deleting (RLN) tree transducers?
5. What is the most efficient algorithm for intersecting probabilistic RTGs?
6. What are the most efficient algorithms for forward and backward application of tree/tree and tree/string transducers?
7. For large tree transducers, what data structures, indexing strategies, and caching techniques will support efficient algorithms?
8. What is the linguistically most appropriate tree transducer class for machine translation? For summarization? Which classes best handle the most common linguistic constructions, and which classes best handle the most difficult ones?
9. Can compact RTGs encode high-performing tree-based language models with appropriate backoff strategies, in the same way that FSA tools can implement n-gram models?
10. What are the theoretical and computational properties of extended left-hand-side transducers ($\mathbf{x}$)? E.g., is xRLN closed under composition?
11. Where do synchronous grammars [50, 17] and tree cloning [15] fit into the tree transducer hierarchy?
12. As many syntactic and semantic theories generate acyclic graphs rather than trees, can graph transducers adequately capture the desired transformations?
13. Are there tree transducers that can move unbounded material over unbounded distances, while maintaining efficient computational properties?
14. In analogy with extended context-free grammars [35], are there types of tree transducers that can process tree sets which are not limited to a finite set of rewrites (e.g., S $\rightarrow$ NP VP PP*)?

15. Can we build tree-transducer models for machine translation that: (1) efficiently train on large amounts of human translation data, (2) accurately model that data by assigning it higher probability than other models, and (3) when combined with search algorithms, yield grammatical and accurate translations?
16. Can we build useful, generic tree-transducer toolkits, and what sorts of programming interfaces will be most effective?

Acknowledgements

This work was supported by NSF grant IIS-0428020 and by DARPA contract N66001-00-1-9814.

References

1. Knight, K., Graehl, J.: Machine transliteration. Computational Linguistics **24** (1998)
2. Mohri, M., Pereira, F., Riley, M.: The design principles of a weighted finite-state transducer library. Theor. Comput. Sci. **231** (2000)
3. Kaplan, R., Kay, M.: Regular models of phonological rule systems. Computational Linguistics **20** (1994)
4. Karttunen, L., Gaal, T., Kempe, A.: Xerox finite-state tool. Technical report, Xerox Research Centre Europe (1997)
5. van Noord, G., Gerdemann, D.: An extendible regular expression compiler for finite-state approaches in natural language processing. In Boldt, O., Juergensen, H., eds.: 4th International Workshop on Implementing Automata, Springer Lecture Notes in Computer Science (2000)
6. Kanthak, S., Ney, H.: Fsa: An efficient and flexible C++ toolkit for finite state automata using on-demand computation. In: Proc. ACL. (2004)
7. Graehl, J.: Carmel finite-state toolkit. `http://www.isi.edu/licensed-sw/carmel/` (1997)
8. Knight, K., Al-Onaizan, Y.: Translation with finite-state devices. In Farwell, D., Gerber, L., Hovy, E., eds.: Proceedings of the 3rd Conference of the Association for Machine Translation in the Americas on Machine Translation and the Information Soup (AMTA-98), Berlin, Springer (1998) 421–437
9. Brown, P., Della Pietra, S., Della Pietra, V., Mercer, R.: The mathematics of statistical machine translation: Parameter estimation. Computational Linguistics **19** (1993) 263–311
10. Kumar, S., Byrne, W.: A weighted finite state transducer implementation of the alignment template model for statistical machine translation. In: Proc. NAACL. (2003)
11. Och, F., Tillmann, C., Ney, H.: Improved alignment models for statistical machine translation. In: Proc. ACL. (1999)
12. Yamada, K., Knight, K.: A syntax-based statistical translation model. In: Proc. ACL. (2001) 523–530
13. Wu, D.: Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. Computational Linguistics **23** (1997) 377–404
14. Alshawi, H., Bangalore, S., Douglas, S.: Learning dependency translation models as collections of finite state head transducers. Computational Linguistics **26** (2000) 45–60
15. Gildea, D.: Loosely tree-based alignment for machine translation. In: Proc. ACL, Sapporo, Japan (2003)

16. Eisner, J.: Learning non-isomorphic tree mappings for machine translation. In: Proc. ACL (companion volume). (2003)
17. Melamed, I.D.: Multitext grammars and synchronous parsers. In: Proc. NAACL. (2003)
18. Knight, K., Marcu, D.: Summarization beyond sentence extraction: A probabilistic approach to sentence compression. Artificial Intelligence **139** (2002)
19. Pang, B., Knight, K., Marcu, D.: Syntax-based alignment of multiple translations extracting paraphrases and generating new sentences. In: Proc. NAACL. (2003)
20. Langkilde, I., Knight, K.: Generation that exploits corpus-based statistical knowledge. In: Proc. ACL. (1998)
21. Bangalore, S., Rambow, O.: Exploiting a probabilistic hierarchical model for generation. In: International Conference on Computational Linguistics (COLING 2000), Saarbrucken, Germany (2000)
22. Corston-Oliver, S., Gamon, M., Ringger, E.K., Moore, R.: An overview of Amalgam: A machine-learned generation module. In: Proceedings of the International Natural Language Generation Conference, New York, USA (2002) 33–40
23. Echihabi, A., Marcu, D.: A noisy-channel approach to question answering. In: Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics, Sapporo, Japan (2003)
24. Charniak, E.: Immediate-head parsing for language models. In: Proc. ACL. (2001)
25. Rounds, W.C.: Mappings and grammars on trees. Mathematical Systems Theory **4** (1970) 257—287
26. Thatcher, J.W.: Generalized^2 sequential machine maps. J. Comput. System Sci. **4** (1970) 339–367
27. Graehl, J., Knight, K.: Training tree transducers. In: Proc. NAACL. (2004)
28. Gécseg, F., Steinby, M.: Tree Automata. Akadémiai Kiadó, Budapest (1984)
29. Comon, H., Dauchet, M., Gilleron, R., Jacquemard, F., Lugiez, D., Tison, S., Tommasi, M.: Tree automata techniques and applications. Available on `www.grappa.univ-lille3.fr/tata` (1997) release October, 1st 2002.
30. Galley, M., Hopkins, M., Knight, K., Marcu, D.: What's in a translation rule? In: NAACL, Boston, MA (2004)
31. Doner, J.: Tree acceptors and some of their applications. Journal of Computer and System Sciences **4** (1970) 406–451
32. Hopcroft, J., Ullman, J.: Introduction to Automata Theory, Languages, and Computation. Addison-Wesley Series in Computer Science. Addison-Wesley, London (1979)
33. Johnson, M.: PCFG models of linguistic tree representations. Computational Linguistics **24** (1998) 613–632
34. Collins, M.: Three generative, lexicalised models for statistical parsing. In: Proc. ACL. (1997)
35. Thatcher, J.: Characterizing derivation trees of context-free grammars through a generalization of finite automata theory. J. Comput. Syst. Sci. **1** (1967) 317–322
36. Yamasaki, K., Sodeshima, Y.: Fundamental properties of pushdown tree transducers (PDTT) — a top-down case. IEICE Trans. Inf. and Syst. **E76-D** (1993)
37. Engelfriet, J.: Top-down tree transducers with regular look-ahead. Math. Systems Theory **10** (1977) 289–303
38. Engelfriet, J.: Bottom-up and top-down tree transformations — a comparison. Math. Systems Theory **9** (1975) 198–231
39. Baum, L.E., Eagon, J.A.: An inequality with application to statistical estimation for probabilistic functions of Markov processes and to a model for ecology. Bulletin of the American Mathematical Society **73** (1967) 360–363
40. Aho, A.V., Ullman, J.D.: Translations of a context-free grammar. Information and Control **19** (1971) 439–475

41. van Noord, G.: The intersection of finite state automata and definite clause grammars. In: Proc. ACL. (1995)
42. Dijkstra, E.W.: A note on two problems in connexion with graphs. Numerische Mathematik **1** (1959) 269–271
43. Eppstein, D.: Finding the k shortest paths. SIAM Journal on Computing **28** (1999) 652–673
44. Mohri, M., Riley, M.: An efficient algorithm for the n-best-strings problem. In: Proc. ICSLP. (2002)
45. Langkilde, I.: Forest-based statistical sentence generation. In: Proc. NAACL. (2000)
46. Nederhof, M.J., Satta, G.: Parsing non-recursive CFGs. In: Proc. ACL. (2002)
47. Knuth, D.: A generalization of Dijkstra's algorithm. Info. Proc. Letters **6** (1977)
48. Klein, D., Manning, C.: Parsing and hypergraphs. In: International Workshop on Parsing Technologies. (2001)
49. Nederhof, M.J.: Weighted deductive parsing and Knuth's algorithm. Computational Linguistics **29** (2003)
50. Shieber, S.M., Schabes, Y.: Synchronous tree-adjoining grammars. In: Proceedings of the 13th International Conference on Computational Linguistics. Volume 3., Helsinki, Finland (1990) 253–258

A Modular Account of Information Structure in Extensible Dependency Grammar

Ralph Debusmann[1], Oana Postolache[2], and Maarika Traat[3]

[1] Programming Systems Lab, Saarland University, Saarbrücken, Germany
rade@ps.uni-sb.de
[2] Computational Linguistics, Saarland University, Saarbrücken, Germany
Al. I. Cuza University, Iaşi, Romania
oana@coli.uni-sb.de
[3] Informatics, University of Edinburgh, Scotland
M.Traat@sms.ed.ac.uk

Abstract. We introduce a modular, dependency-based formalization of Information Structure (IS) based on Steedman's prosodic account [1, 2]. We state it in terms of Extensible Dependency Grammar (XDG) [3], introducing two new dimensions modeling 1) prosodic structure, and 2) theme/rheme and focus/background partitionings. The approach goes without a non-standard syntactic notion of constituency and can be straightforwardly extended to model interactions between IS and other dimensions such as word order.

1 Introduction

Information Structure (IS) is the way in which people organize their utterances. Usually, in an utterance there is a part that links the content to the context, and another that advances the discourse by adding or modifying some information. IS is an important factor in determining the felicity of an utterance in a given context. Among the many applications where IS is of crucial importance are content-to-speech systems (CTS), where IS helps to improve the quality of the speech output [4], and machine translation (MT), where IS improves target word order, especially that of free word order languages [5].

In this paper we present a modular, dependency-based account of IS based on Steedman's prosodic account of IS for Combinatory Categorial Grammar (CCG) [1, 2]. Similarly to Steedman, we establish a bi-directional correspondence between IS and prosodic structure, i.e. when the IS is known, we can determine the prosodic structure (e.g. in CTS systems), and when we have the prosodic information, we can extract the IS (e.g. to augment dialog transcripts).

We state our approach in terms of Extensible Dependency Grammar (XDG) [3], which allows us to take a modular perspective on IS. We distinguish three notions of constituency: syntactic, prosodic, and information structural, which are related, but not identical. Thus, differently from Steedman, we can decouple

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 25–36, 2005.

syntax from information structure, and do not assume non-standard syntactic constituents. By this, we can monotonically add IS to existing XDG grammars. Moreover, our technique is prepared to straightforwardly state constraints on the interplay of IS, prosody and word order, as required for free word order languages such as Czech. This would bring XDG closer to Functional Generative Description (FGD) [6], Kruijff's Dependency Grammar Logic (DGL) [7], Kruijff's and Baldridge's generalized CCG approach [8], and Kruijff's and Duchier's approach using Topological Dependency Grammar (TDG) [9]. The latter account, although stated in a similar framework, is quite different from ours: it concentrates less on modularity, and more on the interaction of different aspects (prosody, word order etc.) in the realization of IS.

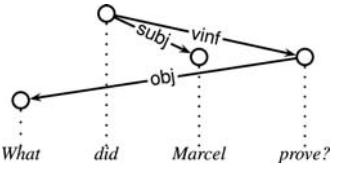
The paper is organized into five sections. Section 2 gives an overview of the XDG grammar formalism. Section 3 is a short introduction to the existing IS approaches; we concentrate on Steedman's prosodic account and his two levels of IS: theme/rheme and focus/background. In section 4, we integrate IS into XDG, introducing two new dimensions: one to model the prosodic structure of the sentence and one to describe the theme/rheme and focus/background distinctions. In section 5, we conclude and outline avenues for future research.

2 Extensible Dependency Grammar

In this section we introduce Extensible Dependency Grammar (XDG) [3]. XDG is a grammar formalism based on dependency grammar [10] and a generalization of Topological Dependency Grammar (TDG) [11]. XDG is all about *modularity*, striving to transplant ideas from software engineering into the context of grammar formalisms. Modularity ensures both *re-usability* and *compositionality*: XDG grammars are consequently composed from layers of simple, re-usable modules. This yields new possibilities for grammar engineering and cross-linguistic modeling.

2.1 Dependency Grammar

Dependency grammar models the syntax of a natural language in terms of relations between words, which correspond 1:1 to nodes. Dependency relations involve *heads* and *dependents*. For example, in the dependency analysis displayed below, in (1), the finite verb *did* is the head of the dependent *Marcel* and *prove* is the head of *what*. Dependency relations are often further specified: in (1), *Marcel* is a subj-dependent of *did*, i.e. the subject, and *what* is the object of *prove*.



(1)

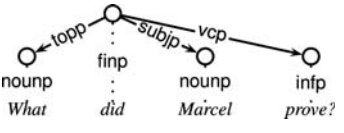
2.2 Multiple Dimensions

Dependency grammar was originally concerned with surface syntax only [10]. However, nothing stops the general concept of dependency grammar—stating relations between words—from being transferred to other linguistic areas, including morphology, deep syntax and semantics. In this generalized sense, pioneered by the Prague and Russian Schools [6, 12], a dependency analysis consists of *multiple* dependency graphs, one for each linguistic dimension. XDG also adopts this idea.

The components of a multi-dimensional dependency analysis are not independent. For instance, semantic arguments can only be realized by appropriate syntactic functions (e.g. agents by subjects). [6, 12] use functional mappings between architecturally adjacent dimensions (e.g. surface and deep syntax). XDG goes beyond that: each dimension can be made to interact with *any* other by bi-directional, relational constraints.

2.3 Word Order

XDG allow splitting up surface syntax into the dimensions *Immediate Dominance* (ID) and *Linear Precedence* (LP). This is essential for the successful treatment of complex word order phenomena in [11]. The ID dimension is solely devoted to syntactic function: with word order factored out, an ID analysis is an *unordered* tree as in (1) above. Word order is taken care of in the LP dimension. LP analyses are *ordered* trees, flatter than the corresponding ID trees. We display an example LP analysis in (2) below:



(2)

Here the finite verb *did* is the head of three dependents: *what*, *Marcel* and *prove*. *What* is a topp-dependent, i.e. it is in topicalized position. Similarly, *Marcel* is in subject position, and *prove* in verbal complement position. In the LP analysis each node carries a node label. This is used to order heads with respect to their dependents. In (2), *did* has node label finp. A well-formed LP analysis must be correctly ordered according to a global order on the set of labels, e.g.:

$$\mathsf{topp} \prec \mathsf{finp} \prec \mathsf{subjp} \prec \mathsf{vcp} \tag{3}$$

Here, we state that topicalized words must precede finite verbs, subjects and verbal complements.

2.4 Semantics

XDG allows us to go far beyond surface syntax. In [3], the authors introduce the dimensions of *Predicate-Argument structure* and *Scope structure* to represent semantics. Because of the relational nature of XDG, this syntax-semantics

interface is automatically bi-directional: syntax can disambiguate semantics and vice-versa, e.g. semantic attachment preferences can resolve modifier attachments in syntax. However, as semantics does not concern us in this paper, we omit further mention of it.

2.5 Principles

The well-formedness conditions of an XDG analysis are specified by *principles* from an extensible *principle library*. Each principle has a declarative semantics and can be parametrized. The *tree principle*, for example, constrains an analysis to be a tree and is parametrized by dimension. Thus, the same principle can be used to constrain the analyses on the ID and LP dimensions to be trees. The *valency principle*, in addition, is *lexicalized*, and constrains the incoming and outgoing edges of each node. On ID, for instance, a finite verb such as *did* requires a subject (outgoing edges), and only nouns *can* be subjects (incoming edges).

The principles so far were *one-dimensional principles*, constraining only one dimension. To constrain the relation between multiple dimensions, XDG offers two means: 1) the lexicon, and 2) multi-dimensional principles. Firstly, the lexicon assigns to each word a set of lexical entries simultaneously constraining all dimensions. Secondly, the principle library includes *multi-dimensional principles*, parametrized by multiple dimensions, which directly constrain their relation.

2.6 Lexicon

XDG grammars rely heavily on the lexicon. To ease the creation of the lexicon and the statement of linguistic generalizations, XDG provides facilities in the spirit of Candito's metagrammar [13], extended in [14]. Basically, the XDG *metagrammar* is an abstract language to describe the lexicon, which is automatically compiled out to yield the lexicon itself.

2.7 Parsing and Generation

XDG parsing and generation is done by the constraint-based XDG solver. Given that XDG solving is NP-complete in the worst case, handcrafted grammars have yielded good performance in the average case. This makes XDG already interesting for the exploration of new linguistic theories such as the one presented here. A comprehensive grammar development toolkit including the XDG solver is freely available and easy to install and use [15]. So far, the XDG solver cannot yet parse induced grammars (e.g. from the Prague Dependency Treebank) competitively [16], but research is underway to improve its performance.

3 Information Structure

In this section, we introduce the concept of Information Structure (IS), illustrate it with examples, and briefly touch upon selected issues related to it. We devote

specific attention to Steedman's [1, 2] prosodic account of information structure, which we have chosen as the basis for our realization of IS in XDG.

3.1 Information Structure Basics

By Information Structure (IS) we mean the way people organize the content they want to communicate in an utterance. There are usually several ways for the same propositional content to be presented. An alternative name for the same concept is Information Packaging, introduced by Chafe [17]. He illustrated its meaning as follows:

> [The phenomena at issue here] have to do primarily with how the message is sent and only secondarily with the message itself, just as the packaging of toothpaste can affect sales in partial independence of the quality of the toothpaste inside.

IS is typically realized by a combination of various means, depending on the typology of the language. In languages with relatively fixed word order, such as English, prosody is often a prominent factor. Free word order languages are more likely to realize IS by word order variation, whereas other languages, such as Japanese, realize IS by morphology (e.g. the special topic marker *-wa*).

Different names have been used for the sub-divisions in IS: topic and focus, theme and rheme, ground and focus, relatum and attributum, to name just a few. What all these divisions have in common, with minor differences, is that they distinguish a part of an utterance that links it to the previous discourse, and another part that is a novel contribution. For a more extensive overview of different approaches to IS, see [18] and [7]. We use the terms *theme* and *rheme* as introduced by the Prague circle of linguists (note that our use of these terms differs from the use by Halliday [19]). Theme is the part that relates the utterance to the previous discourse, while rheme adds or modifies some information about the theme.

As hinted at in Chafe's definition, IS does not affect the propositional content of an utterance. What it does affect, is the contextual felicity of the utterance. For example, while (4)a is a suitable answer for the question in (4), (4)b is not acceptable in the given context:

What did Marcel prove?
a. $[\text{Marcel proved}]_{th}\ [\text{COMPLETENESS.}]_{rh}$ (4)
b. $*\ [\text{MARCEL}]_{rh}\ [\text{proved completeness.}]_{th}$

The words in small capitals in (4) carry the main prosodic accent of the sentence. Assuming that this accent marks the rheme, we can see why only (4)a, but not (4)b is an appropriate answer to the question: *completeness* is the new information asked for, not *Marcel*.

3.2 Prosodic Account of Information Structure

Steedman [1, 2] divides IS into theme and rheme. In his approach, the IS division follows prosodic phrasing. Both theme and rheme can be further divided into

background and focus. The focused material in the theme and rheme are the words that carry pitch accents, while the unaccented words are the background. The most common kind of theme is the so-called "un-marked" theme, where no words carry a pitch accent. Marked themes, as in (5), are used when one item stands in explicit contrast with another from the previous discourse.

I know that Marcel likes the man who wrote the musical.
But who does he ADMIRE?

Marcel	ADMIRES	the woman who	DIRECTED	the musical.	(5)
background	*focus*	*background*	*focus*	*background*	
theme		*rheme*			

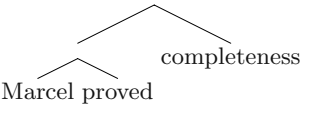
Steedman claims that there is a specific set of pitch accents in English that can accompany the theme, and another that can accompany the rheme, the most common theme pitch accent being L+H* and the most common rheme pitch accent being H*.[1]

Boundary tones delimit prosodic phrases. There are various boundary tones, the most frequently occurring being a low boundary — LL% — and a rising boundary — LH%. There is a tendency for LH% to occur at the end of an intonational phrase containing the theme pitch accent L+H*, and for LL% to occur after the rheme pitch accent H*.

According to the prosodic phrasing, Combinatory Categorial Grammar (CCG) [1] provides different parses for the same string of words, giving rise to different interpretations with respect to the information structure:

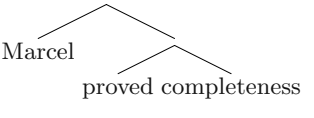
I know what Mark proved, but what did Marcel prove?

MARCEL proved		COMPLETENESS.		(6)
L+H*	LH%	H*	LL%	



I know what Mark did, but what did Marcel do?

MARCEL	proved	COMPLETENESS.		(7)
L+H*	LH%	H*	LL%	



While pitch accents are seen as properties of the words that carry them, boundary tones are seen as individual lexical entries and independent phrasal constituents.

[1] The intonational notation used is due to Pierrehumbert [20]. According to her, intonational phrases are made up of the following components: pitch accent(s), phrasal tone and boundary tone. In Steedman's [1], [2] representation the last two have been joined together under the name '*boundary tone*'. L stands for low pitch, and H for high pitch.

4 Adding Information Structure to XDG

In this section, we present a way of modeling information structure within the XDG formalism. We follow Steedman's approach [1, 2], sketched in section 3.2, which we adapt to XDG by introducing two new dimensions: *Prosodic Structure* (PS) and *Information Structure* (IS). While Steedman views only pitch accents as properties of words, and treats boundary tones as separate lexical items, we treat both pitch accents and boundary tones as properties of words.

4.1 PS Dimension

An analysis on the PS dimension is a tree whose shape is determined by edges representing boundary tones and pitch accents. The root of the tree corresponds to the punctuation mark at the end of the sentence. The daughters of the root are the words carrying boundary tones. Thus, the outgoing edges of the root may be labeled with LL% (low boundary tone), LH% (high boundary tone), H*_LL% (falling pitch accent and low boundary tone) and L+H*_LH% (rising pitch accent and high boundary tone). For simplicity, we consider only the two most frequent types of boundary tones (LL% and LH%), the two most frequent types of pitch accents (H* and L+H*) and their combinations.

Boundary tones delimit non-overlapping, contiguous prosodic constituents. Each word that has a boundary tone attached to it is the head of a prosodic constituent and has the node label b (for *boundary*). To its left it can have accented (may be labeled with H* or L+H*) or non-accented daughters (na), both having the node label nb (for *non-boundary*).

We constrain the PS dimension by the following one-dimensional principles: 1) tree, 2) valency, and 3) order. The tree principle constrains PS analyses to be trees, and the valency principle lexically restricts the incoming and outgoing edges of each node. The order principle serves three purposes: a) it restricts the node labels of each node, b) it requires PS constituents to be projective, i.e. non-overlapping, and c) the order of the daughters of each node must be compatible with the following global order, stating that boundary tones follow everything else:[2]

$$\{\mathsf{L{+}H^*},\ \mathsf{H^*},\ \mathsf{LH\%},\ \mathsf{LL\%},\ \mathsf{L{+}H^*\_LH\%},\ \mathsf{H^*\_LL\%},\ \mathsf{na},\ \mathsf{nb}\} \prec \{\mathsf{b}\} \tag{8}$$

The restrictions on the incoming and outgoing edges (valency: in and out features) and on the node labels (order: on feature) of each node are stipulated in the lexicon. As an example, we show the lexical class *pa_bo* for words carrying both an H* pitch accent and an LL% boundary tone:

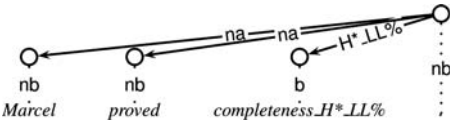
$$pa\_bo ::= \left[PS : \left[\begin{array}{r} \mathsf{in} : \{\mathsf{H^*\_LL\%?}\} \\ \mathsf{out} : \{\mathsf{H^*}*, \mathsf{na}*\} \\ \mathsf{on} : \{\mathsf{b}\} \end{array} \right] \right] \tag{9}$$

[2] This global order actually orders sets of labels instead of just labels, contrary to the total order given in (3) above. The order of labels within these sets is unrestricted.

Words inheriting from this class can only have an incoming edge H*_LL%, and can have an arbitrary number of outgoing edges to accented words, labeled H*, or non-accented ones, labeled na. Their node label is b.

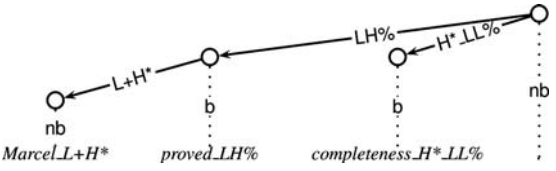
For illustration, we display some example PS trees below, corresponding respectively to (4), (6) and (7) from section 3:

What did Marcel prove?
Marcel proved COMPLETENESS.
H* LL%



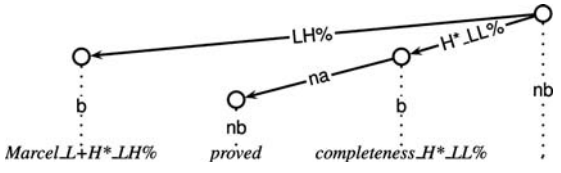
(10)

I know what Mark proved, but what did Marcel prove?
MARCEL proved COMPLETENESS.
L+H* LH% H* LL%



(11)

I know what Mark did, but what did Marcel do?
MARCEL proved COMPLETENESS.
L+H* LH% H* LL%



(12)

4.2 IS Dimension

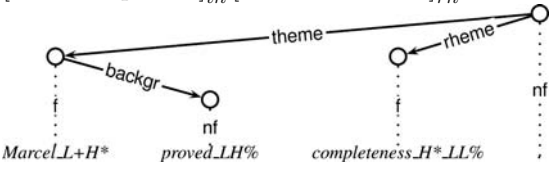
Using XDG's modular methodology, we can simplify our account by first specifying the IS dimension independently from prosodic structure (PS dimension). Only later we will regulate the interplay of the two using the lexicon, and a multi-dimensional principle.

An IS analysis is again a tree whose root corresponds to the punctuation mark. The daughters of the root are the words carrying a pitch accent (instead of those carrying a boundary tone as in the PS), which we call *foci* following Steedman. Their incoming edge label is either theme or rheme, and their node label is f (for *focus*). We require each sentence to have at least one rheme (but cf. *all theme utterances* in [1]), while the theme is optional.

Foci are the heads of non-overlapping, contiguous information structural constituents (i.e. themes or rhemes). Their daughters are the words constituting the

background (edge label backgr). These have node label nf (for *non-focus*). Contrary to boundary tones on the PS, which have to be positioned rightmost in PS constituents, the position of foci within IS constituents is unconstrained. Here is an example IS analysis (cf. (11) in section 4.1):

I know what Mark proved, but what did Marcel prove?
[MARCEL proved]$_{th}$ [COMPLETENESS.]$_{rh}$



(13)

We constrain the IS dimension by re-using the following one-dimensional principles: 1) tree, 2) valency, and 3) order. In the IS dimension, the purpose of the order principle is twofold: a) it restricts the node labels of each node, and b) it requires IS constituents to be non-overlapping. It does not, however, prescribe an order on the set of labels.

As an example, we show the lexical class *rf* for the foci of rhemes:

$$rf ::= \left[IS: \left[\begin{array}{r} \mathsf{in}: \{\mathsf{rheme?}\} \\ \mathsf{out}: \{\mathsf{backgr}*\} \\ \mathsf{on}: \{\mathsf{f}\} \end{array} \right] \right] \tag{14}$$

Words which inherit from this class have the node label f. They can only have an incoming edge labeled rheme, whilst the number of outgoing edges, which are labeled backgr, is arbitrary.

The dimensions of IS and PS are certainly not independent. We constrain their relationship by two means: 1) the lexicon, and 2) a multi-dimensional principle. Firstly, we constrain the lexicon such that nodes with incoming edges L+H* and L+H*_LH% in the PS must have the incoming edge theme in the IS, and those with incoming edges H* and H*_LL% in the PS must have the incoming edge rheme in the IS. Secondly, we use a multi-dimensional principle called the *island principle*, which states that IS constituents must always either coincide with a corresponding PS constituent, or be subparts of it. In other words, IS constituents cannot cross the prosodic constituent boundaries. This principle generalizes over the two cases of marked themes (where the IS constituents coincide with the PS constituents), and unmarked themes (where the IS constituents are subparts of the PS constituents).

As an example for the lexicon constraining the relation between PS and IS, we show the lexical class *rheme_pa_bo*, resulting from the combination of the classes *pa_bo* ((9) above) and *rf* ((14) above):

$$rheme\_pa\_bo = \left[\begin{array}{ll} PS: & \left[\begin{array}{r} \mathsf{in}: \{\mathsf{H*\_LL\%?}\} \\ \mathsf{out}: \{\mathsf{H}*, \mathsf{na}*\} \\ \mathsf{on}: \{\mathsf{b}\} \end{array} \right] \\ IS: & \left[\begin{array}{r} \mathsf{in}: \{\mathsf{rheme?}\} \\ \mathsf{out}: \{\mathsf{backgr}*\} \\ \mathsf{on}: \{\mathsf{f}\} \end{array} \right] \end{array} \right] \tag{15}$$

Words inheriting from this class have the pitch accent H* and the boundary tone LL% (incoming edge label H*_LL%) on the PS. Since these tones accompany only rhemes, they must consequently have incoming edge label rheme in the IS.

So far, we have not dealt with the issue of unmarked themes, which contain no pitch accents and consequently no foci. Here, the IS can be ambiguous, while the PS is unambiguous. Consider (16) which could be an answer to any of the questions (17), (18) and (19):

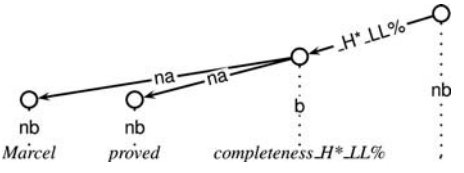
Marcel proved completeness_H*_LL%. (16)

What did Marcel prove? (17)

What did Marcel do? (18)

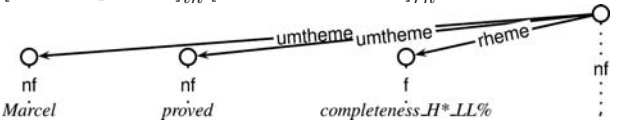
What's new? (19)

The PS of (16), displayed in (20), is unambiguous. Given this PS, however, the IS is ambiguous. The three alternative analyses (21), (22) and (23) correspond respectively to questions (17), (18) and (19). In these analyses, we make each word in the unmarked theme form a singleton IS constituent (including only itself), and having the incoming edge label umtheme (for *un-marked theme*).



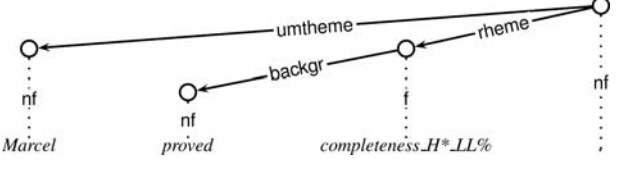
(20)

What did Marcel prove?
[Marcel proved]$_{th}$ [COMPLETENESS.]$_{rh}$



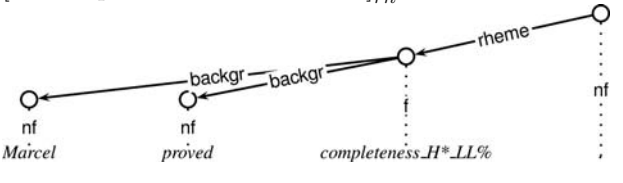
(21)

What did Marcel do?
[Marcel]$_{th}$ [proved COMPLETENESS.]$_{rh}$



(22)

What's new?
[Marcel proved COMPLETENESS]$_{rh}$



(23)

5 Conclusions

We presented a new, modular and dependency-based formalization of IS couched in the framework of Extensible Dependency Grammar (XDG) [3]. As a starting point, we chose Steedman's prosodic account of IS [1, 2]. Our reformulation of his ideas in XDG resulted in a different perspective on the interplay of IS and syntax, decoupling the two to a much higher degree. Thus, we did not introduce non-standard syntactic constituents and could add IS monotonically to any existing XDG grammar.

The approach presented here is not just theoretical: we have already implemented an English grammar using the XDG Development Kit (XDK) [15], which reflects precisely the account given in this paper.

The most interesting avenue for future research is the interplay of IS with other linguistic areas such as word order. In English, IS is mostly realized by prosody, but the picture changes for free word order languages such as Czech, where word order is another prominent factor [7]. Our XDG-based account is perfectly prepared to accommodate such interactions. It allows for the straightforward statement of constraints that relate the IS dimension to the dimension of word order, for example, that topicalized material must be the theme (e.g. in certain dialects of English).

Acknowledgements

We thank Ciprian Gerstenberger and Stefan Thater for their valuable contribution during the IGK summer school project in which this new approach has been developed. We also thank Timothy Jones for his helpful advice when proofreading the pre-final versions of the paper.

References

1. Steedman, M.: The Syntactic Process. MIT Press (2000)
2. Steedman, M.: Information Structure and the Syntax-Phonology Interface. Linguistic Inquiry **31** (2000) 649–689
3. Debusmann, R., Duchier, D., Koller, A., Kuhlmann, M., Smolka, G., Thater, S.: A Relational Syntax-Semantics Interface Based on Dependency Grammar. In: Proceedings of COLING 2004, Geneva/SUI (2004)
4. Prevost, S., Steedman, M.: Information Based Intonation Synthesis. In: Proceedings of the ARPA Workshop on Human Language Technology, Princeton/USA (1994)
5. Stys, M., Zemke, S.: Incorporating Discourse Aspects in English-Polish MT: Towards Robust Implementation. In: Recent Advances in NLP, Velingrad/BUL (1995)
6. Sgall, P., Hajicova, E., Panevova, J.: The Meaning of the Sentence in its Semantic and Pragmatic Aspects. D. Reidel, Dordrecht/NL (1986)
7. Kruijff, G.J.M.: A Categorial-Modal Architecture of Informativity. PhD thesis, Charles University, Prague/CZ (2001)

8. Kruijff, G.J.M., Baldridge, J.: Generalizing Dimensionality in Combinatory Categorial Grammar. In: Proceedings of COLING 2004, Geneva/SUI (2004)
9. Kruijff, G.J.M., Duchier, D.: Information Structure in Topological Dependency Grammar. In: Proceedings of EACL 2003, Budapest/HUN (2003)
10. Tesnière, L.: Eléments de Syntaxe Structurale. Klincksiek, Paris/FRA (1959)
11. Duchier, D., Debusmann, R.: Topological Dependency Trees: A Constraint-based Account of Linear Precedence. In: Proceedings of ACL 2001, Toulouse/FRA (2001)
12. Mel'čuk, I.: Dependency Syntax: Theory and Practice. State Univ. Press of New York, Albany/USA (1988)
13. Candito, M.H.: A Principle-based Hierarchical Representation of LTAG. In: Proceedings of COLING 1996, Kopenhagen/DK (1996)
14. Crabbé, B., Duchier, D.: Metagrammar Redux. In: Proceedings of the International Workshop on Constraint Solving and Language Processing, Roskilde/DK (2004)
15. Debusmann, R., Duchier, D.: XDG Development Kit (2004) http://www.mozart-oz.org/mogul/info/debusmann/xdk.html.
16. Bojar, O.: Problems of Inducing Large Coverage Constraint-Based Dependency Grammar. In: Proceedings of the International Workshop on Constraint Solving and Language Processing, Roskilde/DK (2004)
17. Chafe, W.L.: Givenness, Contrastiveness, Definiteness, Subjects, Topic, and Point of View. In Li, C., ed.: Subject and Topic. Academic Press, New York/USA (1976) 25–55
18. Vallduví, E., Engdahl, E.: The Linguistic Realisation of Information Packaging. Journal of Linguistics **34** (1996) 459–519
19. Halliday, M.A.K.: Notes on Transitivity and Theme in English, Part II. Journal of Linguistics **3** (1967) 199–244
20. Pierrehumbert, J.: The Phonetics and Phonology of English Intonation. PhD thesis, Massachusetts Institute of Technology, Bloomington/USA (1980)

Modelling Grammatical and Lexical Knowledge: A Declarative Approach

Palmira Marrafa

University of Lisbon, Faculty of Arts
Department of Linguistics and CLG-Computation of Lexical
and Grammatical Knowledge Research Group (Centre of Linguistics)
Palmira.Marrafa@netcabo.pt

Abstract. This paper depicts the fundamentals of a computational grammar able to provide adequate representations for Portuguese simple sentences with several kinds of ambiguities. Besides the description of the architecture of the system proposed and the way it works, the paper focuses on the discussion of the nature of the specifications to encode in order to get a high level of precision. From a linguistic point of view, an endocentric phrase structure approach is adopted. The information is encoded in a DCG-like formalism, implemented in PROLOG.

1 Introduction

Modelling grammatical knowledge entails the specification of a large set of intertwined syntactic and semantic properties of linguistic expressions, which are highly structured and exhibit local and long distance dependencies ruled by several types of constraints.

In view of the complexity of the information to encode, the development of grammars that are suitable enough both for precision and coverage represents a great challenge.

As well-known, precision and coverage are conflicting requirements of natural language modelling, since a more precise grammar tends to be a more constrained grammar while constraints tend to reduce coverage (see [9] for a brief discussion of this trade-off).

Without neglecting coverage, this work is particularly concerned with precision, an essential requirement both for Theoretical Computational Linguistics central aims and for a wide range of applications.

Accordingly, the fragment of grammar presented here is able to rule out ill-formed expressions and inappropriate interpretations and to assign at least one representation to each well-formed expression for the constructions at issue. It covers the basic structure of simple sentences with several types of predication relations.

Such sentences frequently involve syntactic ambiguity, a major problem for computational natural language analysis.

Despite the complexity of the phenomena involved, the grammar has a suitable level of parsimony, since grammatical rules make appeal to the lexical entries which contain fine grained specifications of the syntactic and semantic restrictions imposed by the lexical units.

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 37 – 47, 2005.

The paper is organized as follows: Section 2 depicts the general architecture of the computational analysis system, the organization of the different modules it integrates and the way they interact; Section 3 is concerned with the analysis of the empirical data at stake and the kind of representations to be provided; Section 4 presents some results accounting for the descriptive adequacy of the system; finally, Section 5 presents some conclusion remarks.

2 System Architecture

Roughly, the system presented is a Definite Clause Grammar (DCG), implemented in PROLOG.

Similarly to what happens in most language technologies, the linguistic specifications and the computational procedures integrate different components. Let us observe Fig. 1, which provides a scheme of the overall system and the way the different components interact:

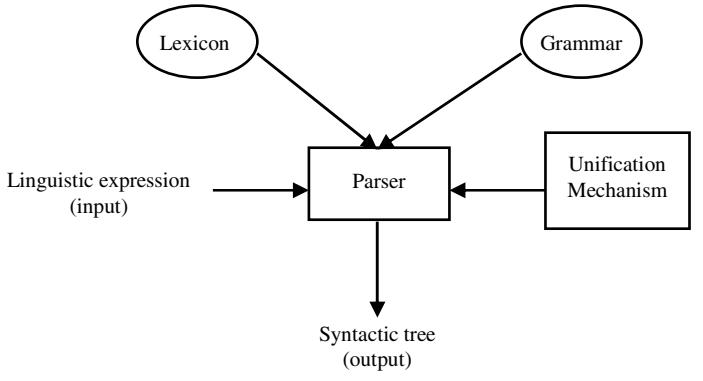


Fig. 1. General System Architecture

The lexical specifications and the grammatical rules are encoded in separated modules for the sake of an easier control.

The formalism used to encode the information is a DGC-like formalism with slight modifications. The main change concerns the fact that the right side of the rules is a list, in order to allow its interpretation by a bottom-up parser.

The option for a bottom-up parsing strategy has to do with the fact that several rules involve recursivity, namely the rules regarding modifiers, secondary predicates and complex predicates, as discussed in Section 3.

One of the components of the system is an unification mechanism, which deals with feature structures. It is worth to note that unification is a fundamental ingredient of the most used formalisms in Computational Linguistics because, among other reasons, it permits the easy encoding of information independently of any specific processing algorithm (on this matter, cf., for instance, [12]). The unification mechanism has a crucial role in this work, since it operates like a well-formedness filter, as illustrated in the next subsections.

2.1 Grammar Module

The syntactic configurations are defined in the Grammar module, which is basically a DCG, with slight modifications in order to allow its interpretation by a bottom-up parser. As referred to before, the main change concerns the fact that the right side of the rules is a list. The rules have the following format:

grammatical symbol ---> [any sequence of grammatical
symbols and control predicates].

Grammatical symbols are represented as follows:

<symbol designation> (<syntactic tree>, <feature structure>).

As mentioned, each lexical entry includes a feature structure that specifies the relevant properties of the corresponding lexical unit.

The control predicates used — *extract* and *unify* — guarantee the observation of the restrictions specified in the lexical entries. In other words, the control predicates guarantee that any linguistic expression whose syntactic structure is not consistent with the properties specified for the lexical items it integrates is ruled out. Both predicates are three-place predicates that make appeal to the feature structures included in the lexical entries: *extract(T,A,V)*; *unify(A,B,C)*.

Informally, *extract* is satisfied *if and only if T* is a feature that includes the attribute *A* and *A* has the value *V*; *unify* establishes that *A* and *B* unify in *C if and only if* for any attribute γ common to *A* and *B*, γ has the same value in *A* and *B*.

Let us examine two simple illustrative examples:

```
v1(v1(V,NP),T1)--->
   [v(V,T1),
    extract(T1,scat,[np]),
    np(NP,T2)].
```
(1)

```
np(np(Det,N),T)--->
   [det(Det,T1),
    n(N,T2),
    unify(T1,T2,T)].
```
(2)

In (1), the predicate *extract* guarantees that only verbs whose feature structure (*T1*) specifies the value *np* for the attribute *scat* enter in this rule. Therefore, it avoids the undesirable analysis of any *V NP* sequence as *V1*. It applies, for instance, to *joga este jogo* ("plays this game") but not to *joga esta manhã* ("plays this morning").

Through *unify*, the rule in (2) guarantees agreement between a noun and its determiner at the noun phrase level. The information specified in the feature structure of the latter (*T1*) and the information specified in the feature structure of the former (*T2*) have to unify. Consequently, anomalous expressions like *os gato* ('the_plural cat') are straightforwardly ruled out.

Since the syntactic structures are determined to a large extent by the properties of the lexical heads, specified in the Lexicon, the Grammar includes a relatively short number of rules. This way, the suitable economy of the system is preserved.

2.2 Lexicon Module

The descriptive adequacy of the representations provided by the system crucially depends on the information encoded in this module.

As rendered evident in Section 3, to capture the information to be included in the lexical entries is not a trivial task. A very fine grained syntactic and semantic characterization of the lexical units is required. Besides, the specifications for a given item have to be concerted with the specifications for many others.

Contrarily, encoding the information is relatively easy. Both syntactic and semantic properties are encoded by means of feature structures. Each property corresponds to an *attribute: value* pair. Let us exemplify with the abbreviated entries for *o* ('the_masc') and *dar* ("to give"):

```
det(det(o),[…,numb:sing,gend:masc,…])--->[o].        (3)
```

```
v(v(dar),[…,scat:[np,pp],…])--->[dar].               (4)
```

The determiner *o* is specified for the attributes *numb* (number) and *gend* (gender), whose values are *sing* (singular) and *masc* (masculine), respectively. The verb *dar* is specified for the attribute *scat* (subcategorization) whose value is the list *[np, pp]*.

Semantic properties can also be encoded in a similar way.

3 Linguistic Specifications

From a linear order point of view, the expressions treated so far in this project have the following skeleton: *N V N A*.

Despite the apparent simplicity of this sequence, it can correspond to several very distinct structures. Depending on several factors – in particular, the semantic properties of the elements involved –, the adjectival constituent can be interpreted as: (i) a secondary predicate oriented to the subject – (5a); a secondary predicate oriented to the object – (5b); a modifier within the NP object – (5c); part of a complex predicate – (5d). Let us observe some corresponding illustrative examples:

a. Ele dança a valsa descalço.
"He dances the waltz barefoot"

b. Ele viu a Maria furiosa. (5)
'He saw the Maria furious'
"He saw Maria furious"

c. Ele prefere o café brasileiro.
'He prefers the coffee Brazilian'
"He prefers Brazilian coffee"

d. Ele põe a Maria alegre.
'He makes the Maria happy'
"He makes Maria happy"

The syntactic and semantic properties of the structure corresponding to the different types of predications are discussed in the next sub-sections. The analysis is informed by

the assumption that syntactic expressions are endocentric. In other words, syntactic expressions are considered to be projections of lexical heads (so far, functional heads are not taken into account). The X-bar convention is adopted to represent syntactic configurations.

3.1 Secondary Predication

As illustrated above, secondary predication involves an argument of the primary predicate – the external argument in (5a); the internal argument in (5b) – and a non-verbal predicate (for the sake of simplicity of the explanation only APs are considered in this paper, but PPs and AdvPs can also be secondary predicates) which expresses an atomic event (a state, in other words) that occurs in the same temporal interval in which occurs the primary event. These circumstances justify the secondary predicates co-occurrence restrictions imposed by the head of the primary predicate, illustrated below:

a. Ele dançou descalço.
"He danced barefoot"
b. *Ele dançou arrependido.
"He danced regretful"
c. Ele chegou arrependido.
"He arrived regretful"

(6)

a. Ele convidou a Maria bêbeda
'He invited the Maria drunk'
"He invited Maria$_i$ drunk$_i$"
b. *Ele convidou a Maria indecisa.
'He invited the Maria undecided'
"He invited Maria$_i$ undecided$_i$"
c. Ele viu a Maria indecisa.
'He saw the Maria undecided'
"He saw Maria$_i$ undecided$_i$"

(7)

Marrafa [6] argues, along with the basic lines of Marrafa [3], that secondary predicates, lacking independent participants and time information, are a kind of "parasites" of primary predicates, as synthesized in (8), where *J* a temporal interval, *t1…ti…tn* sub-intervals of *J*, *e1…ei...en* the sub-events of a primary event that occur in *t1…ti…tn*, a secondary (atomic) event, *x* a participant in *e1…ei...en* and in ε, p^k the set of properties assigned to *x* through *e1…ei...en* in *t1…ti…tn*, p^m the set of properties assigned to *x* through ε in *t1…ti…tn*.

In other terms, p^m, the set of properties associated to the event denoted by a secondary predicate (referred to above as secondary event), applies to a participant of the event denoted by a primary predicate in all the temporal sub-intervals in which its sub-events occur. That is, p^m and p^k, the set of properties associated to *e1…ei...en*, apply to the same participant in the same temporal intervals. Consequently, p^m has to be compatible with p^k. And the system has "to know" this.

$$\left[\begin{array}{l} J \\ e_1(x)^{pk}{}_{t1}\ldots ei(x)^{pk}{}_{ti}\ldots(en(x)^{pk}{}_{tn} \\ \\ \varepsilon(x)^{pm}{}_{t1\cdots\, ti\cdots tn} \\ \\ \end{array}\right] \tag{8}$$

Let us now re-examine the example (5c), here renumbered (9a), in comparison with (9b):

a. Ele prefere o café brasileiro.
'He prefers the coffee Brazilian'
"He prefers Brazilian coffee / *He prefers the coffee Brazilian"
b. Ele prefere o café frio.
'He prefers the coffee cold'
"He prefers the coffee cold / He prefers the cold coffee" (9)

As we can observe, (9b), but not (9a), is ambiguous between an interpretation where the adjectival constituent is a modifier of *café* ("coffee") and an interpretation where it is a secondary predicate oriented to the object. In (9a) the interpretation corresponding to *He prefers the coffee Brazilian* is excluded (in coherence with the ungrammaticality of the English expression). More precisely, in this case, the interpretation corresponding to the secondary predication is not available.

It seems obvious that the contrast above derives from the semantic properties of the adjectival constituents involved. Concretely, considering the dichotomy accidental properties *vs*. permanent or inherent properties (this distinction goes back to Milsark [7], [8] and Carlson [1]), the property denoted by *brasileiro* ("Brazilian") belongs to the latter class and the property denoted by *frio* ("cold") to the former one.

It is apparent from the data that secondary predication is only compatible with the expression of accidental properties. This restriction takes place also when the secondary predicate is oriented to the subject, as exemplified below:

a. Ele partiu feliz.
"He left happy"
b. *Ele partiu alto.
"He left tall" (10)

However, the characterization of the adjectives on the basis of this dichotomy is not straightforward, since adjectives can be ambiguous in relation to those properties, as it is the case of *triste* ("sad") in the examples below:

a. Ele encontrou a rapariga triste.
"He$_i$ met the girl sad$_i$"/ "He met the sad girl"/ "He met the girl$_i$ sad$_i$"
b. Ele leu o livro triste.
"He$_i$ read the book sad$_i$"/ "He read the sad book"/ "*He read the book$_i$ sad$_i$" (11)

Both sentences are ambiguous, but (11a) has one interpretation more than (11b). In the former sentence *triste* can be secondary predicate of *Ele* ("He") and both modifier and secondary predicate of *rapariga* ("girl"), while in the latter one the interpretation of secondary predicate oriented to the object is not available.

Despite their complexity, all the restrictions have to be encoded in order to avoid both over- and under-generation of representations.

Regarding the syntax of these constructions, the co-occurrence restrictions imposed by the verb to the secondary predicates suggest that they are not excluded of the maximal projection of *V*.

Nevertheless, the restrictions imposed to the predicates oriented to the object are stronger than those imposed to the predicates oriented to the subject. Moreover, the order *predicate oriented to the object* < *predicate oriented to the subject* is obligatory, as rendered evident by the contrast below:

a. Ele bebeu o café frio triste.
'He drank the coffee cold sad'
b. *Ele bebeu o café triste frio.
'He drank the coffee sad cold' (12)

On the basis on these facts, the predicates oriented to the object are represented in adjunction to *V1* (the numeric notation is used for the sake of coherence with the notation used in the modelling formalism) and the predicates oriented to the subject in adjunction to *V2* (*VP*), as shown in (13):

$$[\ldots[_{V2}[_{V2}[_{V1}[_{V1}[_{V}]\ldots][_{pred\_object}]]][_{pred\_subject}]] \quad (13)$$

It is worthwhile to note that this representation also satisfies the subject-predicate reciprocal m-command constraint extensively argued for by Marrafa [3] and further related work, but the discussion of this issue is not within the goals of this paper.

3.2 Complex Predicates

Concerning complex predicates, this paper focuses on lexical-conceptual structure deficitary verbs and follows mainly Marrafa's [4] and [5] proposals and previous related work.

In order to clarify the concept of lexical-conceptual structure deficitary verb let us start by examining the following example:

Ele pintou a parede de amarelo.
"He painted the wall yellow" (14)

The situation described in (13) entails that *a parede* ("the wall") became *amarela* ("yellow") as a result of painting. This means that the verb denotes an event with a definite endpoint (cf. Wechsler [13], among others). In other terms, the verb denotes a transition event (in the sense of Pustejovsky [10], [11]), which is structured as stated below:

$[_T [_P\ e_1 ...e_n]\ e_m]$
T, Transition; P, Process; e, atomic event; $e_m > e_n$; $e_m \neq e_1$ (15)

Accordingly, the sentence in (14) has the following lexical-conceptual structure (Pustejovsky's LCS'):

LCS' { [[act(ele,parede)&~pintada_de_amarelo(parede)],
[pintada_de_amarelo(parede)]]
"{ [[act(he,wall)&~painted_yellow(wall)],
[painted_yellow(wall)]]" (16)

As it becomes evident, the verb plus the resultative expression, *de amarelo* ("yellow"), form a lexical-conceptual unit, that is, a complex predicate, as extensively argued by Marrafa [3].

The absence of the resultative does not have any impact on the LCS', as we can observe:

a. Ele pintou a parede.
"He painted the wall"

b. LCS' { [[act(ele,parede)&~pintada(parede)],
[pintada(parede)]] (17)
"{ [[act(he,wall)&~painted(wall)],
[painted(wall)]]"

Let us now consider again the example in (5d) (here renumbered as (18a)):

a. Ele põe a Maria alegre.
'He makes the Maria happy' (18)
"He makes Maria happy"

The LCS' associated to it seems to be (18b) and not (18c).

b. LCS' { [[act(ele,Maria)&~feliz(Maria)],
[feliz(Maria)]]
"{ [[act(he,Maria)&~happy(Maria)],
[happy(Maria)]]"

c. LCS' { [[act(ele,Maria)&~tornada_feliz(Maria)],
[tornada_feliz(Maria)]]
"{ [[act(he,Maria)&~made_happy(Maria)],
[made_happy(Maria)]]"

This suggests that *Q* is instantiated just with the resultative. It is then expected that the absence of the resultative induces ungrammaticality, in coherence with the facts:

*Ele põe a Maria.
'He makes the Maria' (19)
"He makes Maria"

Along the same basic lines of Marrafa [3] and further work, verbs like *pôr* ("make") are argued here to be LCS' deficitary, in the sense that they do not include in their denotation the set of content properties of the final state of their LCS', as stated below:

Informal definition:

$\forall$v((verb(v), $\exists \mathcal{E}$, LCS'_of_v($\mathcal{E}$), $\exists$e, final_state(e), (20)
e $\subset \mathcal{E}$, $\exists$ π, set_of_semantic_features_of(π,e), $\pi = \varnothing$)
=> LCS'_deficitary(v))

Since that set is empty, the LCS' cannot hold an appropriate interpretation. A syntactic structure that projects an anomalous LCS' is, then, previewed to be ruled out (it does not satisfy the requirement of full interpretation).

In this case, the resultative fills the gap of the LCS' of the verb (cf. the contrast between (18a) and (19)).

Therefore, these facts show that the representation of the predicates at issue has to include information concerning the resultative expression.

Regarding the syntactic structure, the general internal structure of *V2* (*VP*) is largely inspired in Larson's [2] proposal. In what specifically concerns the complex predicate, it has to be represented as a syntactic unit to account for the data discussed above. That means that both elements have to be immediately dominated by the same node. Therefore, the non-verbal part of the predicate is represented in adjunction to *V*. To derive the canonical order, *V* moves to a higher position. This movement is captured by co-indexation between *V* and the *t*(race) that occupies its basic position. The internal structure of *V2* for complex predicates is, then, the following:

$[_{V2} \ldots [_{V1} V_i \ [_{V2} \ NP \ [_{V1} \ [_{V} [_{V} t_i] \ AP]]]]]$ (21)

A major problem that the computational analysis system has to deal with is the high level of ambiguity induced by certain polysemous verb forms with resultative interpretations, as it is the case of *deixar* ("to let"), referred to in the next section.

4 Results

The representations provided by the system for two ambiguous sentences are presented below. Constituents in a subject-predicate relation, a head-modifier relation or belonging to a complex predicate are marked in the syntactic tree by means of *co I*, *I* instantiated with the same value.

In spite of their superficial similarity, these two sentences have not the same level of ambiguity. In the case of (22) four interpretations are available, while in the case of (23) there are only two.

O Jorge deixou a rapariga triste.
'the Jorge left the girl sad'
"Jorge left the girl sad"
Interpretations: (i) deixou triste ≡ entristeceu ("made sad"): (22)
[*deixou triste*] complex predicate
(ii) [$_{NP}$a rapariga triste]: ap modifier
(iii) [$_{NP}$ a rapariga$_i$][triste$_i$]: ap pred-obj
(iv) [$_{NP}$ o Jorge$_i$][triste$_i$]: ap pred-subj

O Jorge leu o livro triste.
'the Jorge read the book sad'
"Jorge read the book sad" (23)
Interpretations: (i) [$_{NP}$o livro triste]: ap modifier
(ii) [$_{NP}$ o Jorge$_i$][triste$_i$]: ap pred-subj

([*deixou triste*] complex predicate)

```
f(f(np(det(o),n(jorge)),
    v2(v1(v(deixou)co_147,
    v2(np(det(a),n(rapariga)),
    v1(v(v(t)co_147,
    ap(a(triste)))))))));
```

([$_{NP}$a rapariga triste]: ap modifier)

```
f(f(np(det(o),n(jorge)),
    v2(v1(v(deixou),
    np(det(a),n(rapariga),ap(a(triste)))co_178)
                                        co_178)));
```

([$_{NP}$ a rapariga$_i$][triste$_i$]: ap pred-obj)

```
f(f(np(det(o),n(jorge)),
    v2(v1(v1(v(deixou),
    np(det(a),n(rapariga))co_178),
    ap(a(triste))co_178))));
```

([$_{NP}$ o Jorge$_i$][triste$_i$]: ap pred-subj)

```
f(f(np(det(o),n(jorge))co_78,
    v2(v2(v1(v(deixou),
    np(det(a),n(rapariga)))),
    ap(a(triste))co_78))).
```

([$_{NP}$o livro triste]: ap modifier)

```
f(f(np(det(o),n(jorge)),
    v2(v1(v(leu),
    np(det(o),n(livro),ap(a(triste)))co_174)
                                    co_174)));
```

([$_{NP}$ o Jorge$_i$][triste$_i$]: ap pred-subj)

```
f(f(np(det(o),n(jorge))co_75,
    v2(v2(v1(v(leu),
    np(det(o),n(livro)))),
    ap(a(triste))co_75))).
```

As we can observe, the representations provided are the adequate ones. All the licensed interpretations, and only the licensed interpretations, are assigned a representation.

5 Conclusions

Despite the intricacy of the syntactic and semantic restrictions of the constructions at stake, the system presented here is able to provide adequate representations, accounting for structural ambiguity.

The use of feature structures to specify lexical information allows to encode the information related to such restrictions in a very fine grained way.

In view of the richness of lexical entries, interfacing lexical descriptions with grammar rules allows for a relatively parsimonious grammar.

The modular and declarative formulation adopted greatly facilitates the extension of the grammar to another kind of structures, as well as porting it from one formalism to another.

References

1. Carlson, G.: Reference to Kinds in English. PhD dissertation, University of Massachusetts-Amherst (1977).
2. Larson, R.: On the Double Object Construction. Linguistic Inquiry, 19, (1988) 335-391.
3. Marrafa, P.: Computação de Ambiguidades Sintácticas: evidências em favor dos modelos baseados em conhecimento linguístico. IN COGNITO (forthcoming).
4. Marrafa, P.: Extending WordNets to Implicit Information. Proceedings of LREC 2004, International Conference on Language Resources and Evaluation. Lisboa, Portugal, (2004) 1135-1138.
5. Marrafa, P.: The representation of Telic Complex Predicates in Wordnets: the case of lexical-conceptual structure deficitary verbs. Proceedings of Convergences03, International Conference on the Convergence of Knowledge, Culture, Language and Information Technologies. Library of Alexandria, Alexandria, Egypt (2003).
6. Marrafa, P.: Predicação Secundária e Predicados Complexos em Português: Análise e Modelização. PhD dissertation. University of Lisbon (1993).
7. Milsark, G.: Existential Sentences in English. PhD dissertation, MIT (1974).
8. Milsark, G.: Toward an Explanation of Certain Pecularities of the Existencial Construction in English. Linguistic Analysis, 3, (1977)1-29.
9. Pereira, F.: Sentence Modeling and Parsing. In G. Varile & A. Zampolli (eds) Survey of the State of the Art in Human Language Technology, 130-140. National Science Foundation and European Comission (1995).
10. Pustejovsky, J.: The Syntax of Event Structure. Cognition, 41, (1991) 47-81.
11. Pustejovsky, J.: The Generative Lexicon. Cambridge, MIT Press, Mass. (1995).
12. Uszkoreit, H. & Zaenen, A.: Language Analysis and Understanding. In G. Varile & A. Zampolli (eds) Survey of the State of the Art in Human Language Technology. National Science Foundation and European Commission (1995) 130-140.
13. Wechsler, S. An Analysis of English Resultatives Under the Event-Argument Homomorphism Model of Telicity. Proceedings of the 3rd Workshop on Text Structure. University of Texas (2000).

Constructing a Parser for Latin

C.H.A. Koster

Computing Science Institute,
University of Nijmegen,
The Netherlands
kees@cs.kun.nl

Abstract. We describe the construction of a grammar and lexicon for Latin in the AGFL formalism, in particular the generation of the lexicon by means of transduction and the description of the syntax using the Free Word Order operator. From these two components, an efficient Top-Down chart parser is generated automatically. We measure the lexical and syntactical coverage of the parser and describe how to increase it.

The morphological generation technique described here is applicable to many highly-inflected languages. Since the Free Word Order operator described can cope with the extremely free word order in Latin, it may well be used for the description of free-word-order phenomena in modern languages.

1 Introduction

Why would anybody in his right mind construct a formal grammar of Latin? Although there exist some active speakers of the language and according to some its best poetry was produced in the nineteenth century, the language is as dead as a doornail. A large corpus of latin texts is extant, but there is no expectation of any important additions. Most texts have been translated into many other languages in which they can be enjoyed without the drudge of learning Latin. Commercial application of an Information Retrieval system for Latin is inconceivable. Furthermore there already exists an overwhelming number of learned grammars and dictionaries for it, to which any formal grammar would add nothing new.

It was for the Latin language, and earlier for Greek, that the science of linguistics as we know it was developed. Everyday concepts and terminology of Latin still pervade western linguistic thinking. The understanding of the structure of Latin provides a framework in which not only its linguistic relatives, but also utterly unrelated languages could be analysed, modeled and described. The Latin language has a number of properties (detailed and rich morphology, very free word order) which together with its quite regular structure make it an interesting object for formal description. Practically, it is the mother of the Romance languages and the aunt of many other languages (including English) which do have practical and even commercial value. Lastly, describing it with the aid of modern grammar and parsing technology may be of therapeutic value for one who has been forcefed on it for a number of years in high school.

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 48–59, 2005.

1.1 About Grammars

Two utterly differents kind of grammars can be distinguished:

1. **Grammar**$_1$: to real linguists, a grammar is a thick book, in which every aspect of the structure of a certain language is described in an informal but highly rigorous fashion.
2. **Grammar**$_2$: to computer scientists and computer linguists a grammar is a description of the morphosyntax of a language in a formalism such that (given also a lexicon) a parser can be constructed from it by automatic means.

Of course there are incredibly many variations on these concepts, but the basic fact is that any linguists happily working with the one kind of grammar has very little patience for the other kind.

In this paper we try to reconciliate the two, deriving a practically functioning grammar$_2$ (a *formal* grammar) from the knowledge contained in a grammar$_1$. We base ourselves on [Redde Rationem] and [Latijnse Leergang].

1.2 About Latin

The Latin language presents a challenge to its description in current syntactic formalisms and automatic parsing methods, due to the fact that it is definitely not a Context-Free language:

- it has a rich morphology and agreement rules, governed by (classical) features like Number, Person, Gender, Case, Time, Voice and Tense.
- it displays a close approximation to Free Word Order, in that nearly all constituents of a phrase can be permuted without affecting the meaning of the phrase.

In fact, the family of two-level grammars was conceived for the formal description of such rich morphological and agreement rules (although mostly motivated by the description of programming languages with their typing and identification rules, rather than natural languages).

In this note, we shall make use of AGFL [Koster, 1991], which is a member of the family of two-level grammars.

1.3 About AGFL

Affix Grammars over a Finite Lattice (AGFL) are a form of two-level grammars in which the features take on as values any subset of a given finite set. The typical examples of such features are completely classical: an affix `NUMBER` distinguishing between singular and plural, and another affix `PERSON` distinguishing between the first, second and third person may be defined in AGFL by the affix rules

```
NUMBER :: sing | plur.
PERSON :: first | secnd | third.
```

An affix may take on a set-value (any non-empty subset of its domain) rather than a specific single value, indicating partial knowledge, e.g. `PERSON = {first | third}` indicates the knowledge that the Person-feature does not have the value `second`.

Affixes are used as parameters to the nonterminal-rules of the grammar to indicate agreement, e.g.

```
sentence:
   subject(NUMBER,PERSON), verb(NUMBER,PERSON), object.
```

where the *consistent substitution rule* ensures that different occurrences of the same affix must have the same value (enforcing agreement in Person and Number between subject and verb). The notation of AGFL should be similar enough to PROLOG with DCG's (which is indeed a related formalism) to allow a computer linguist to comprehend the following examples without further explanation.

An AGFL grammar describes the constituency structure of a language, with the commas in a syntax rule indicating sequential order and the semicolons indicating alteration. However, there is also a proviso for Free Word Order (FWO): members separated by ampersands my occur in any order. The development of a grammar for Latin provided a nice opportunity to exercise (and debug) this facility, which is intended for the compact description of FWO languages.

AGFL also allows a rule to describe a (compositional) *transduction*: every alternative may indicate how its translation is to be composed out of the translation of its members (where a terminal symbol is translated to itself). We'll use this transduction instead of parse trees to show the results of parsing, but also use it to construct our lexicon.

2 Constructing a Latin Lexicon

Although some lists of Latin words are freely available on the internet, there is no machine-readable lexicon to be found, and we had to develop one from scratch.

In principle, the word(forms) of a language together with their parts of speech (POS) may simply be enumerated in the grammar by rules like

```
LEXV(ind,prm,sg,perfct,act): "amavi".
```

but there would be very many of such rules, and the efficiency of recognizing the enumerated wordforms would be terrible. A more principled approach would be to enumerate a list of stems and lists of infixes and suffixes according to their conjugation, combining them by rules like

```
VERB(ind,prm,sg,perfct,act):
  V_STEM(CONJ,act),
    V_INFIX(CONJ,perfct),
      V_SUFFIX(perfct,prm,sg,act).
```

It is quite possible to enumerate the stems and fixes in the lexicon, so that they do not clotter up the grammar and are recognized by means of lexicon lookup rather than by exhaustive trial. Lexicon entries (in the notation of AGFL) look like

```
"ama-"  V_STEM(a_conj,act)
"-v-"   V_INFIX(a_conj,perfct)
"-i"    V_SUFFIX(perfct,prm,sg,act)
```

While this approach is feasible, it takes a lot of effort describing the morphology of Latin in great detail (copying all this wisdom from a grammar$_1$). Especially the irregular or partly regular words will lead to many fine distinctions. It therefore makes sense to generate those irregular forms once-and-for-all and put the complete wordforms into the lexicon:

```
"sum"   TOBE(ind,prm,sg,praes,act)
"es"    TOBE(ind,sec,sg,praes,act)
"est"   TOBE(ind,trt,sg,praes,act)
"sumus" TOBE(ind,prm,pl,praes,act)
"estis" TOBE(ind,sec,pl,praes,act)
"sunt"  TOBE(ind,trt,pl,praes,act)
```

However this idea points to another approach which is more uniform and simple: to generate also *all* regular wordforms once-and-for-all and put them into the lexicon. Rather than recognizing a wordform from its parts the parser will then recognize a wordform as a whole, by means of whole-word lexicon lookup. A `generative` solution, rather than an *analytic* one.

An objection to this approach may be seen in its efficiency – since very many wordforms may come form one stem, we would have a very large lexicon. Indeed a simple verb like contestare will generate 168 wordforms, including the declinated forms of participles and even some adverbs. The lexicon system of AGFL (using compacted trie structures) is however highly efficient, both in time (faster than parsing the parts) and space (the lexicon takes about the same space as the list of all words contained in it), so that the efficiency is actually better than in the previous solution.

Another objection might be the severe overgeneration expected from putting all forms in the lexicon, without verifying whether they are attested in any text. However, the same objection applies to the corresponding analytical approach which would recognise precisely the same wordforms. Word forms which do not "exist" will not be used, it is as simple as that. But it should be noted that, in order to reach sufficient coverage of the lexicon in spite of limited effort, certain rules will have to be included in the grammar in order to "guess" the POS of out-of-vocabulary words; thus the overgeneration can be seen as a positive contribution to robustness! Anyway, the overgeneration can if needed be avoided by "filtering" the lexicon through a wordlist obtained from a large corpus.

This then is the approach we have taken: to generate the lexicon for the large open classes (noun, verb, adjective and adverb) from a very classical resource:

word lists and stem times, as contained in any standard grammar$_1$C, with a separate treatment for irregular words (that can be generated in the same way, correcting the irregularities by hand).

2.1 Metarules for the Lexicon

The metarules defining the affixes used in the lexicon with their domains are the following:

```
CASUS:: nom | voc | gen | dat | acc | abl | loc.
```

The six cases of Latin.

```
NUM:: sg | pl.
GENUS:: fem | masc | ntr.
PERS:: prm | sec | trt.
```

Now come the affixes detailing the POS of verbs:

```
MODUS:: ind | con | imp | inf | part | pperf | gerund.
TEMPUS:: praes | imprf | futur | perfct | pqperf | futex.
VGENUS:: act | pas.
```

The following affix pertains to adjectives and adverbs, which are also generated from verbs:

```
GRADUS:: pos | comp | super.
```

2.2 Nouns

The noun entries are generated from a list of entries, one per line, for different declinations, like:

```
a poeta masc
a plaga fem
b poculum
b populus
c portio portionis fem
c plebs plebis masc
e facies masc
i aer aeris masc
ie hostis hostis masc
```

From these list entries, lexicon entries are generated of type

```
LEXS(NUM,GENUS,CASUS)
```

by means of a grammar describing the transduction from a list entry to all corresponding lexicon entries. The following rule for the a-declination is typical:

```
declinatio prima:
   "a", radix, "a", ",genus." /
     "\"",radix,"a\"\tLEXS(sg,",genus,",nom|voc|abl)\n",
     "\"",radix,"ae\"\tLEXS(sg,",genus,",gen|dat)\n",
     "\"",radix,"am\"\tLEXS(sg,",genus,",acc)\n",
     "\"",radix,"ae\"\tLEXS(pl,",genus,",nom|voc)\n",
     "\"",radix,"arum\"\tLEXS(pl,",genus,",gen)\n",
     "\"",radix,"is\"\tLEXS(pl,",genus,",dat|abl)\n",
     "\"",radix,"as\"\tLEXS(pl,",genus,",acc)\n".

genus: "masc"; "fem"; "ntr".

radix:$MATCH(".*-").
```

Notice that the root of the word is matched by a Regular Expression. For the word puella (feminine) the transducer generates

```
"puella"        LEXS(sg,fem,nom|voc|abl)
"puellae"       LEXS(sg,fem,gen|dat)
"puellam"       LEXS(sg,fem,acc)
"puellae"       LEXS(pl,fem,nom|voc)
"puellarum"     LEXS(pl,fem,gen)
"puellis"       LEXS(pl,fem,dat|abl)
"puellas"       LEXS(pl,fem,acc)
```

The list of regular nouns is 1429 entries long, generating 36798 lexicon entries, which are extended with 28 irregular forms.

2.3 Other Categories

Verbs also come in different conjugations; as in traditional grammars, a verb is specified by giving its infinitive, perfectum and participium perfectum or futurum, of which the latter two may be missing (dash):

```
a obscurare obscuravi obscuratus
c abhorrere abhorrui -
e abstinere abstinui abstentus
io abicere abieci abiectus
i adoriri - adoriturus
```

Note that deponentia are indicated by a passive infinitive. The output of the transduction process is a list of lexicon entries of the following types:

```
LEXV(MODUS,PERS,NUM,TEMPUS,VGENUS)
LEXV(inf,TEMPUS,VGENUS)
LEXV(MODUS,NUM,GENUS,CASUS)
LEXA(GRADUS,NUM,GENUS,CASUS),
LEXX(GRADUS)
```

(V stands for verb, A for adjective and X for adverb). Some examples of each (for the verb amare):

```
"amo"   LEXV(ind,prm,sg,praes,act)
"amabatis"  LEXV(ind,sec,pl,imprf,act)
"amasti"    LEXV(ind,sec,sg,perfct,act)
"amare" LEXV(inf,praes,act)
"amari" LEXV(inf,praes,pas)
"amando"    LEXV(gerund,sg,masc|ntr,dat|abl)
"amantes"   LEXV(part,pl,masc|fem,nom|voc|acc)
"amaturi"   LEXA(pos,pl,masc,nom|voc)
"amabilis"  LEXA(pos,sg,GENUS,nom|gen|voc)
"amabiliter"    LEXX(pos)
"amatius"   LEXX(comp)
```

Verbs are much more productive than nouns, and there are about as many verbs as nouns, so that it is no wonder that they generate most of the lexicon entries: The 1242 entries in the verb list generate 179404 lexicon entries, to which 4339 irregular verb forms are added.

Finally, there is a list of adjectives of three different declinations: The 624 different adjective entries generate a total of 71718 lexicon entries, to which 122 irregular forms have been added by hand.

The remaining (closed) lexical categories include numerals, adverbia and various kinds of pronomina. Of these lexical types there are a total of 1606 lexicon entries.

2.4 Achieving Coverage

The first word lists were created from the lists and examples given in the two textbooks. Then a grammar `sweep.gra` was developed for analysing the coverage of the lexicon. It transduces every word of the input text to a copy marked with a rough lexical category, according to the schema:

```
sententia:
   known word / marker, known word;
   looks like name / !
   any word / "?:", any word.
```

using the following markers for known words:

`V:` verb form, form of esse
`N:` noun form
`A:` adjective
`Q:` quantity
`X:` adverbium
`P:` pre- or postposition
`D:` determiner, demonstrative or pronoun

Unknown words are marked by `?:`, potential names (unknown words starting with a capital letter) are skipped.

A text corpus (St. Augustine's Confessiones) was cut into words and swept. A frequency list was built from the marked words, using standard UNIX utilities

```
cat corpus| tr -cs "[:alpha:]" "[\n*]"| sweep -tP1| grep ':' >words
sort +1 words | uniq -c | sort -nr > freq
```

Then some days were spent analysing the unrecognized wordforms, from the highest frequency downwards, then extending the word lists and bootstrapping, until I got bored. At that point the lexical coverage (number of known words in the corpus divided by the total number of words) was 92%. Applying the same lexicon to the Vulgate translation of the Psalms gave a coverage of 87%.

It is striking to see the low level of lexical ambiguity of Latin compared to English, once a few cases of systematic overlap (e.g. participia and adjectiva) are resolved. The general strategy is to remove all nouns and adjectives from the word lists that can be generated from a verb.

3 Constructing the Grammar

We shall describe the grammar of Latin in a Bottom-Up manner, which was also roughly the order in which it was constructed and tested.

The lexical interface on which it rests has already been described in the previous section. The basic approach is

- describe the noun phrase as a noun generalized by projection, and similarly the adjective phrase as a generalized adjective and the verb phrase as a generalized verb form (which is optional, implying TOBE) together with its complements
- for every composed phrase enumerate the possible orders of its constituents
- then describe the way in which sentences can be glued together into longer sentences.

This is a general strategy which works reasonably well for many languages; but of course it encounters many problems and complications.

3.1 The NP

We describe the derivation of the Noun Phrase syntax in some detail. The noun phrase has as its kernel an N (standing for nomen), which may have several realizations, among which a lexical noun is preferred. An adjective phrase (AP) or a quantity is also accepted.

```
N(NUM,GENUS,CASUS):
    LEXS(NUM,GENUS,CASUS);
    $PENALTY,robust nomen(NUM,GENUS,CASUS);
    $PENALTY,AP(pos,NUM,GENUS,CASUS);
    $PENALTY(2),LEXQ(NUM,GENUS,CASUS).
```

In order to achieve some lexical robustness, there are rules for guessing the type of out-of-vocabulary words. These are only invoked for words which have no lexicon entry.

```
Nbar(CASUS),Nbar(NUM,GENUS,CASUS):
   N(NUM,GENUS,CASUS);
   $PENALTY, PRDET(NUM,GENUS,CASUS);
   PRPER(PERS,NUM,CASUS);
   $PENALTY, PRDEM(NUM,GENUS,CASUS);
   AP(pos,NUM,GENUS,CASUS), Nbar(NUM,GENUS,CASUS);
   PRPOS(NUM,GENUS,CASUS), Nbar(NUM,GENUS,CASUS);
   N(NUM,GENUS,CASUS), AP(pos,NUM,GENUS,CASUS);
   N(NUM,GENUS,CASUS), PRPOS(NUM,GENUS,CASUS).
```

Besides an N also certain pronouns are accepted, and an AP or possessive pronoun is admitted as a modifier. Note that these modifiers may precede or follow the N. The description in the last four lines is not as symmetric and general as we would like, it would be better to have

```
   Nbar(NUM,GENUS,CASUS), AP(pos,NUM,GENUS,CASUS);
   Nbar(NUM,GENUS,CASUS), PRPOS(NUM,GENUS,CASUS).
```

but the AGFL system presently does not allow left-recursion.

At this level also the clitics `-que` and `-ve` are introduced:

```
NBAR(pl,GENUS,CASUS):
   N(NUM,GENUS,CASUS), N(NUM1,GENUS,CASUS),clitic.
```

One level higher, the numerals are introduced, as well as the relative sentence, the genitive adject and explicative interjections:

```
 Nbarbar(CASUS), Nbarbar(NUM,GENUS,CASUS):
    numerus(NUM,GENUS,CASUS),Nbar(NUM,GENUS,CASUS);
    Nbar(NUM,GENUS,CASUS),explicatio(CASUS);
    Nbar(NUM,GENUS,CASUS),[Nbar(gen)];
    Nbar(NUM,GENUS,CASUS),[comma],relsent(NUM,GENUS,CASUS).
```

The latter two are both exemplified in the sentence laudare te vult homo, aliqua portio creaturae tuae.

```
explicatio(CASUS):
    $PENALTY,[comma],NP(CASUS),[comma].
```

The numerals are either spelled as words from the lexicon or they are in the form of roman numerals in capital letters:

```
numerus(NUM,GENUS,CASUS):
  LEXQ(NUM,GENUS,CASUS);
  $SKIP("[MDCLXVI][MDCLXVI]*").
```

Finally, the NP may be composed of one or more Nbarbars. The formulation given here is defective, because NUM and GENUS should be influenced by the number and gender of elements.

```
NP(CASUS), NP(NUM,GENUS,CASUS):
    Nbarbar(NUM,GENUS,CASUS);
    Nbarbar(NUM,GENUS,CASUS),conj(co),NP(NUM,GENUS,CASUS).
```

The adjective phrase, again, is a generalized adjective. A participium or determinative pronoun may also serve as an adjective.

An adverb may precede as well as follow an adjective, but the former is preferred, to prevent spurious ambiguity of an adverb *between* two adjectives, and also because that feels intuitively right.

3.2 Verb Phrases

The verbal part of a sentence is very simple, since latin has no separate auxiliary verbs.

```
V(MODUS,PERS,NUM,VGENUS):
  LEXV(MODUS,PERS,NUM,TEMPUS,VGENUS);
  X,V(MODUS,PERS,NUM,VGENUS).
```

Some other verbal constructions:

```
tobe(MODUS,PERS,NUM,TEMPUS,VGENUS):
  [adject], TOBE(MODUS,PERS,NUM,TEMPUS,VGENUS).

infinitivus(VGENUS):
   LEXV(inf,TEMPUS,VGENUS), [object],[adject],
     (conj(co),LEXV(inf,TEMPUS,VGENUS);
      LEXV(inf,TEMPUS,VGENUS), clitic; ).


participium(NUM,GENUS,CASUS):
   [X],LEXV(part|pperf|gerund,NUM,GENUS,CASUS), [object].
```

Notice that no subcategorization information is available for the verbs, which causes needless overgeneration.

3.3 Sentence Structure

We adopt a very simple discourse structure: a sentence is composed of simple sentences glued together by certain separators. Three kinds of simple sentences are distinguished, statements, questions and commands, of which the latter are still poorly developed.

We distinguish between statements with an explicit main verb, and those with an (explicit or implicit) form of esse and a predicate.

```
statement:
   SVOC phrase;
   SxP phrase.
```

By an SVOC phrase we mean a phrase maximally containing Subject, Verb, Object and Complements in some order, the main verb being obligatory. This can be expressed in AGFL using the FWO operator as

```
[subject(PERS,NUM)] & V(MODUS,PERS,NUM,act) & [object] & [adject]
```

From this, topicalized versions (e.g. preposing an adjective from the subject or object) can be constructed.

The predicative sentences have many optional elements, including a predicate, but a form of esse must be present:

```
[subject(PERS,NUM,GENUS)]&TOBE(MODUS,PERS,NUM,TEMPUS,act)&
    [predicate(PERS,NUM,GENUS,nom)]&[adject]
```

The analysis of the discourse structure has to be refined on the basis of corpus study. The grammar is still incomplete, the wordlist contain errors and lacunae. It is hoped that some latin scholars will nevertheless find a use for the parser, and will extend and improve this work.

4 Preliminary Results

The parser generated from the grammar and lexicon described in the previous sections was applied to two corpora

- the Confessiones of St. Augustine (downloaded from [Augustinus Confessiones]), beautiful and well-polished latin
- Julius Caesar's accounts of the Gallic, Hispanic, African and Alexandrian wars, which consist of more rough-hewn political/military prose.

The lexicon was derived from the first corpus alone.

4.1 Coverage

In order to measure the coverage of the parser (number of words covered by the preferred analysis, divided by the total number of words in the text) we put at the root of the grammar a simple transduction, accepting either a sentence, which was then enclosed between square brackets, or an NP, which was enclosed between round brackets. Any word not covered by these was looked up in the lexicon. If it occurred in the lexicon with any category it was marked as SKIPped, and otherwise as UNKNown.

In the following table, we indicate the number of words in the text covered by the preferred analysis, the words from the lexicon not covered by the analysis, and the words not occurring in the lexicon.

corpus	number of words	covered	skipped	unknown
Augustinus	78784	65264 (82.8%)	8763 (11.1%)	4757 (6.0%)
Caesar	49961	37634 (75.3%)	7990 (16.0%)	4337 (8.7%)

4.2 Speed

We also measured the CPU time needed to obtain the preferred analysis for each segment on a 700Mhz INTEL PC on the two corpora.

corpus	total time	words parsed / second
Augustinus	3 min 51 sec	341
Caesar	2 min 1 sec	413

Since processors four times as fast are easy to find, this may also well be the fastest Latin parser ever constructed! Most of the time is actually spent in lexicalization (lexicon lookup, robust recognition of proper names).

5 Conclusions

The Latin grammar and lexicon were developed by one person in a few lost weeks between Christmas and the Spring term. This was of course only possible by the availability of good tools - the AGFL formalism, a specialized parser generator for Natural Language parsing, UNIX tools for text transformations - and a good schooling in classical latin. Of course quite a lot of improvement and error correction is still needed.

However, the generative technique employed to produce the lexicon is suitable for any language with a complicated morphosyntactic structure. The trie-based lexicon system of AGFL has no problems with millions of wordforms, so that even highly inflected languages lend themselves to generative lexicon production and maintenance. The AGFL formalism, which was developed for describing English turned out to be very suitable for the compact description of Latin, including its free-word-order aspects.

The parser, latin grammar and lexicon described here are freely available from the [AGFL website].

References

[Koster, 1991] Cornelis H.A. Koster, Affix Grammars for Natural Languages. In: H. Alblas and B. Melichar (Eds.), *Attribute Grammars, applications and systems.* SLNCS 545, Heidelberg, 1991, pag. 469-484.

[linguistics textbook] `http://www.u-grenoble3.fr/lebarbe/Linguistic_Lexicon/`

[Redde Rationem] A.G. de Man et al (1979), *redde rationem - recapulationes*, Wolters - Noordhoff.

[Latijnse Leergang] S.F.G. Rotteveel Mansfeld en R. Waleson (1970), /em Latijnse leergang, tweede druk, Wolters - Noordhoff.

[Augustinus Confessiones] `http://www.thelatinlibrary.com/august.html`

[another latin parser] `http://www.levity.com/alchemy/latin/latintrans.html`

[AGFL website] `http:www.cs.kun.nl/agfl`

Parsing Korean Case Phenomena in a Type-Feature Structure Grammar

Jong-Bok Kim[1] and Jaehyung Yang[2]

[1] School of English, Kyung Hee University, Seoul, Korea 130-701
[2] School of Computer Engineering, Kangnam University, Kyunggi, 449-702, Korea

Abstract. For a free-word order language such as Korean, case marking remains a central topic in generative grammar analyses for several reasons. Case plays a central role in argument licensing, in the signalling of grammatical functions, and has the potential to mark properties of information structure. In addition, case marking presents a theoretical and computational test area for understanding the properties of the syntax-morphology interface of the language. This is why it is no exaggeration to say that parsing Korean starts from work on the case system of the language. This paper reports the case system of the Korean Phrase Structure Grammar (KPSG) developed as a Korean resource grammar for computational purposes and implemented in the Linguistic Knowledge Building (LKB) system. The grammar adopts the constraint-based mechanisms of feature unification and multiple inheritance type hierarchies as an extension of HPSG (Head-driven Phrase Structure Grammar) and is proved to be empirically and computationally sound and efficient.

1 Two Basic Issues

Nominal expressions in Korean can carry discourse as well as case markers that could indicate their grammatical, semantic, and discourse functions. One simple example would suffice to show the complexity of its case system:[1]

(1) sensayngnim-tul-pwuthe-ka ku chayk-ul sangca-ey-to neh-ess-ta
teacher-PL-SRC-NOM the book-ACC box-LOC-also put-PST-DECL
'To only teachers, such a letter is also delivered.'

As noted here, the NOM and ACC assigned by the verbal predicate indicate the syntactic functions (subject and object) of the nominals they are attached to. Cases like LOC express the semantic function of the NP 'box'. The delimiter marker *-to* 'also' can be attached to the semantic case marker, LOC. A further

[1] The abbreviations we use here are AGT (agentivity), ARG-ST (argument-structure), COMP (complementizer), COMPS (complements), Conj (conjunctive), NOM (Nominative), ACC (accusative), DAT (dative), HON (honorific), LEX (lexical), ORTH (orthography), SRC (source), PST (Past), Pl (plural), SYN (syntax), X-Delim (X-delimiter), Z-Delim (Z-delimiter), etc.

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 60–72, 2005.

complication arises from the co-occurrence of the grammatical case NOM with the semantic case SRC as in *sensayng-tul-pwuthe-ka*.[2]

In addition to such complexity, the language places tight ordering restrictions in the attachment of these discourse, semantic, and grammatical markers. The language also displays complicated case marking patterns, which we will see in due course. In addition to canonical case assignment patterns, it displays intriguing phenomena such as case stacking, case alternation, multiple nominative/accusative cases, case on adverbs and verbal elements, and so forth. In parsing Korean sentences, the prerequisites are thus (a) to build case-marked or noncase-marked elements properly and (b) constrain their occurrences in syntax in a systematic way.

1.1 Formation of Case-Marked Elements

Nominal expressions allow various particles (including case markers) to be attached but in strict ordering relationships, as exemplified in the traditional template in (2)a and one example in (2)b:

(2) a. N-base – (Hon) – (Pl) – (PostP) – (Conj) – (X-Delim) – (Z-Delim)
b. sensayng + (nim) + (tul) + (eykey) + (man) + (i)
teacher + Hon + Pl + Postp + only + NOM
'to the (honorable) teachers only'

As observed in (2)a, the GCASE markers such as NOM, ACC, and GEN can appear only in the final position, called Z-Delim(iter) position, whereas the SCASE markers (GOAL, LOC, etc) occupy the PostP position.[3] Even though we could adopt such a templatic mechanism to generate such nominals, it is more effective to generate nominals with a precisely defined type hierarchy system.[4] Another prevailing approach has been to take the nominal particles as independent syntactic head elements such as postpositions and determiners (e.g. Yoon 1995). Such a syntactic analysis in a sense does not reflect the inflectional properties, optionality, and tight ordering restrictions among these. We believe a more efficient way of approach is a lexicalist approach in which the particles are treated as suffixes attached to the nominal stems in the lexicon by a step-by-step process based on the hierarchy in (3).[5] The building process of nominal elements thus starts from the basic lexical elements of the type *nom-lxm* (nominal-lexeme), moving up to a higher type while any of these processes can be skipped and then directly be realized as (pumped up to) a *word* element in syntax. Thus the

[2] The KPSG grammar classifies case markers into two groups: GCASE and SCASE. The GCASE includes *nom* and *acc* as a subtype of *vcase* (cases assigned by a verbal element) and *gen* as a subtype of *ncase* (case assigned by a nominal element), whereas the SCASE includes *dat, goal, src, inst*, etc.

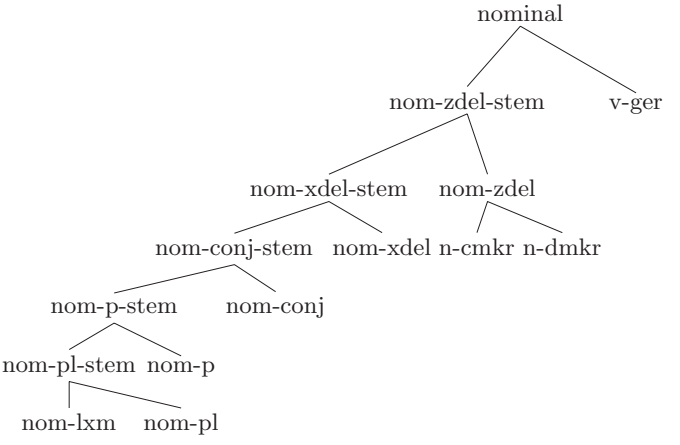
[3] X-Delim and Z-Delim are position names given to delimiters such as *-man* 'only' *-to* 'also', *-nun* 'topic', etc.

[4] The templatic approach suffers from issues of positing null elements for unrealized suffixes. See Kim (1998) for further discussion of problems in the templatic approach.

[5] The necessity of introducing each of these stems in the grammar could be easily supported by the fact that each of these stems appear in syntax.

attachment of the plural suffix to the *nom-lxm* will generate *nom-pl*, and that of a postposition suffix will produce a *nom-p* element.[6]

(3)



The constraints on each type place restrictions on the ordering relationship among nominal suffixes, as exemplified in (4):

(4) a. *nom-p* → [STEM *nom-pl-stem*]
b. *nom-zdel* → [STEM *nom-xdel-stem*]

These constraints mean that the type *nom-p* requires its STEM value to be a type of *nom-pl-stem*, and the type *nom-zdel* specifies its STEM value to be *nom-xdel-stem*. These constraints explain why (5)a is well-formed, but not (5)b:

(5) a. $[_{nom-p}\ [_{nom-pl}$ sensayngnim-tul]-eykey] 'teacher-PL-DAT'
b. *$[_{nom-p}\ [_{nom-zdel}$ sensayngnim-nun]-eykey] 'teacher-TOP-DAT'

The type *nom-pl* in (5)a is a subtype of *nom-pl-stem*, and this thus observes the constraint in (4)a. However, in (5)b, the type *nom-zdel* cannot serve as the STEM value of the postposition *-eykey* according to (4)a since it is not a subtype of *nom-pl-stem*.

This kind of type hierarchy system minimizes the burden of specifying what kind of STEM value is possible for each stem. For example, even though the case or discourse marking nominal *nom-zdel* requires its STEM value to be *nom-xdel-stem*, all of its subtypes could satisfy this constraint:[7]

(6) a. $[_{n-dmkr}\ [_{nom-lxm}$ sensayngnim]-un] 'teacher-TOP'
b. $[_{n-dmkr}\ [_{nom-p}$ sensangnim-eykey]-nun] 'teacher-DAT-TOP'
c. $[_{n-dmkr}\ [_{nom-xdel}$ sensayngnim-pwuthe]-ka] 'teacher-SRC-NOM'

[6] The grammar permits all the instances of type *nominal* to be realized as *n-word*. This in turn means any subtype of *nominal* can serve as a syntactic element in accordance of the type hierarchy in (2).

[7] The type *nom-zdel* has two subtypes *n-dmkr* and *n-cmkr* depending on the stem is with a discourse marker or with a case marker.

The type hierarchy system thus generates various options with no additional constraints. Once we assign more concrete information to each type in the process of building nominal expressions, we could have enriched lexical information as following for (6)b,c:

(7) a. $\begin{bmatrix} \text{ORTH } \langle \text{sensayngnim-eykey-nun} \rangle \\ \text{HEAD} \begin{bmatrix} \text{POS } \textit{noun} \\ \text{CASE} \begin{bmatrix} \text{GCASE } \textit{gcase} \\ \text{SCASE } \textit{dat} \end{bmatrix} \end{bmatrix} \end{bmatrix}$ b. $\begin{bmatrix} \text{ORTH } \langle \text{sensayngnim-pwuthe-ka} \rangle \\ \text{HEAD} \begin{bmatrix} \text{POS } \textit{noun} \\ \text{CASE} \begin{bmatrix} \text{GCASE } \textit{nom} \\ \text{SCASE } \textit{src} \end{bmatrix} \end{bmatrix} \end{bmatrix}$

1.2 Case Constraints in Syntax

Once we have the right generation of nominal elements with case information, the next issue is how argument-selecting heads and grammar rules contribute their case information to nominal elements. As noted by Bratt (1996), Yoo (2002), and others, phenomena such as case alternation illustrated in (8) make it hard to attribute to the case as lexical properties:

(8) a. John-i nokcha-ka/*lul coh-ta
John-NOM green.tea-NOM/*ACC like-DECL
'John is fond of green tea.'

b. John-i nokcha-lul/*ka coh-a hanta
John-NOM green.tea-ACC/*NOM like-COMP do
'John likes green tea.'

When the psych verb combines with the auxiliary verb and functions as a non-stative verb, its theme argument must be ACC.

The starting point of our analysis is to adopt the lexeme-based lexicon. The basic lexical entries we need to specify in the lexicon are just lexemes: all the verbal lexemes will minimally have the following information:

(9) $\textit{v-lxm} \rightarrow \begin{bmatrix} \text{HEAD} \mid \text{POS } \textit{verb} \\ \text{ARG-ST } \langle ..., [\text{GCASE } \textit{vcase}], ... \rangle \end{bmatrix}$

This means that any element in the ARG-ST gets the value *vcase* as its GCASE value: the *vcase* value can be either *nom* or *acc* in syntax.

The elements in the ARG-ST will be realized as SUBJ and COMPS in syntax in accordance with the Argument Realization Constraint (ARP) as represented in the following LKB description:[8]

```
v-word := word &
 [ SYN.VAL [ SUBJ < #first >, COMPS #rest ],
   ARGS < lexeme & [ SYN.HEAD.POS verb,
                     ARG-ST [ FIRST #first, REST #rest ] ] > ].
```

[8] The feature ARGS, standing for 'arguments', here is the daughter value(s) of the type *v-word*.

In the KPSG, it is thus at the valence level that the case value is sensitive rather than at the argument structure level. As an illustration of how this system works, let us consider one example. The lexical entry for the lexeme *ilk-* 'read' would be something like the following:

(10) $\begin{bmatrix} \textit{v-lxm} \\ \text{ORTH } \langle \text{ilk-} \rangle \\ \text{ARG-ST } \langle \text{NP}\left[\text{GCASE } \textit{vcase}\right], \text{NP}\left[\text{GCASE } \textit{vcase}\right] \rangle \end{bmatrix}$

Note here that the arguments of the lexeme do not maximally specify its GCASE value. By definition all the arguments of a lexical element get *vcase*. These arguments will be realized as SUBJ and COMPS in syntax:

(11) $\begin{bmatrix} \textit{v-word} \\ \text{ORTH } \langle \text{ilk-ess-ta 'read-PST-DECL'} \rangle \\ \text{SYN}\begin{bmatrix} \text{HEAD} \mid \text{POS } \textit{verb} \\ \text{VAL}\begin{bmatrix} \text{SUBJ } \langle \boxed{1} \rangle \\ \text{COMPS } \langle \boxed{2} \rangle \end{bmatrix} \end{bmatrix} \\ \text{ARG-ST } \langle \boxed{1}\text{NP}\left[\text{GCASE } \textit{vcase}\right], \boxed{2}\text{NP}\left[\text{GCASE } \textit{vcase}\right] \rangle \end{bmatrix}$

With this declarative verb *ilk-ess-ta* 'read-PST-DECL', the SUBJ element can be *nom* whereas the COMPS can be *acc*, but not the other grammatical case value as noted in (12):

(12) John-i/*ul chayk-ul/*i ilk-ess-ta
John-NOM/ACC book-ACC/NOM read-PST-DECL
'John read a book.'

Then, the question is which part of the grammar makes sure the SUBJ is *nom* whereas COMPS is *acc*. The determination of case value in the VAL is not by a lexical process but imposed by syntactic rules. That is, we assume that Korean X′ syntax includes at least the Head-Subject Rule encoded in the LKB as the following feature description:[9]

```
hd-subj-ph := ph-st &
 [ SYN.VAL [ SUBJ <>,
             COMPS #2 ],
   ARGS < #1 & [ SYN.HEAD [ CASE.GCASE nom, PRD - ] ],
          [ SYN.VAL [ SUBJ < #1 >,
                      COMPS #2 ] ] > ].
```

[9] One thing to note here is that *hd-subj-ph* makes no reference to the COMPS value, unlike English where the COMPS value should be empty. Placing no restrictions on the COMPS value allows us to combine the predicate and the subject first before the complement(s). Also, the grammar combines the head with one complement at a time. This system allows only binary structures. One strong advantage of this approach is that it enables us to capture sentence internal scrambling with no additional mechanism. See Kim and Yang (2004) for a similar analysis.

The rule simply says that when a head combines with the SUBJ, the SUBJ element is *nom*. As for the case value of a complement, it is a little bit more complicated since there are cases where the nonsubject argument gets NOM rather than ACC as in (8). In the language, nonagentive verbs like *coh-* assign NOM to their complements. Reflecting this type of case assignment and following Bratt (1996) and Yoo (2002), we adopt the head feature AGT (AGENTIVITY) and ramify the Head-Complement Rule into two as the following:[10]

(13) a. Head-Complement Rule A:

$$\left[\textit{hd-comp-ph}\right] \Rightarrow \boxed{1}\left[\text{CASE} \mid \text{GCASE } \textit{acc}\right], \mathbf{H}\left[\begin{array}{l}\text{HEAD} \mid \text{AGT } + \\ \text{COMPS} \left\langle ..., \boxed{1}, ... \right\rangle\end{array}\right]$$

b. Head-Complement Rule B:

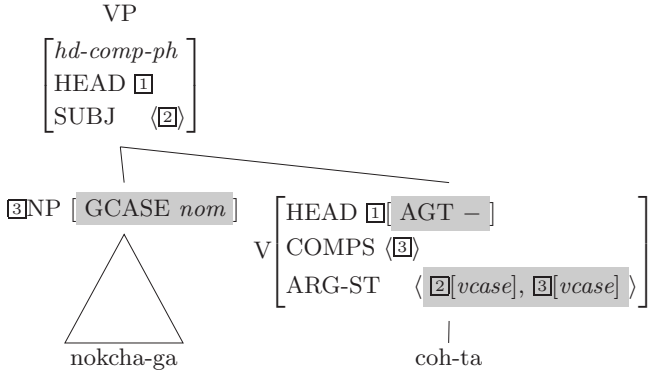
$$\left[\textit{hd-comp-ph}\right] \Rightarrow \boxed{1}\left[\text{CASE} \mid \text{GCASE } \textit{nom}\right], \mathbf{H}\left[\begin{array}{l}\text{HEAD} \mid \text{AGT } - \\ \text{COMPS} \left\langle ..., \boxed{1}, ... \right\rangle\end{array}\right]$$

Within this system, we then do not need to specify *nom* to the nonsubject complement of psych verbs, diverging from the traditional literature. Just like other verbs, the complement(s) of such psych verbs like *coh-ta* 'like-DECL' will bear just *vcase*, as a general constraint on verbal elements as represented in (14)a:

(14)
$$\left[\begin{array}{l}\text{HEAD}\left[\begin{array}{l}\text{POS } \textit{verb} \\ \text{AGT } -\end{array}\right] \\ \\ \text{ARG-ST} \left\langle \text{NP}\left[\text{GCASE } \textit{vcase}\right], \text{NP}\left[\text{GCASE } \textit{vcase}\right] \right\rangle\end{array}\right]$$

This lexical information would then project the following structure for (8):

(15)



[10] The positive value of the AGT (AGENTIVITY), similar to STATIVITY, is assigned to the verbs that have an external argument whereas the negative value is assigned to those with no external argument.

As noted here, the verb *coh-ta* 'like' bears the head feature [AGT −]. This means that the complement of this verb will get NOM even though in the ARG-ST its case value is *vcase*. This is guaranteed by the Head-Complement Rule B in (13).

2 Case in Auxiliary Constructions

2.1 Changing Case Value

One welcoming consequence of this analysis comes from the treatment of case alternation in auxiliary verbs as in (8)b. As noted previously, the psych verb *coh-ta* is [AGT −]. This allows its complement to get NOM. Then, why does the same theme argument in the auxiliary verb construction in (8)b get *acc* rather than *nom*? This is due to the agentive auxiliary verb *ha-n-ta* 'do-PRES-DECL', whose brief lexeme information is given in (16):[11]

(16)
$$\begin{bmatrix} \text{ORTH} \left\langle \text{ha- 'do'} \right\rangle \\ \text{HEAD} \begin{bmatrix} \text{AUX} + \\ \text{AGT} + \end{bmatrix} \\ \text{ARG-ST} \left\langle \boxed{1}\text{NP}, \begin{bmatrix} \text{LEX} + \\ \text{SUBJ} \quad \langle \boxed{1} \rangle \end{bmatrix} \right\rangle \end{bmatrix}$$

This lexical information tells us that the auxiliary verb selects one subject argument and a predicative lexical element whose subject is identical with its own subject. Such an auxiliary verb forms a complex predicate with a preceding verb in accordance with the following Head-Lex Rule:[12]

(17) Head-Lex Rule:
$$\begin{bmatrix} \textit{hd-lex-ph} \\ \text{COMPS L} \end{bmatrix} \rightarrow \boxed{1} \begin{bmatrix} \text{LEX} + \\ \text{COMPS L} \end{bmatrix}, \text{H} \begin{bmatrix} \text{AUX} + \\ \text{COMPS} \left\langle \boxed{1} \right\rangle \end{bmatrix}$$

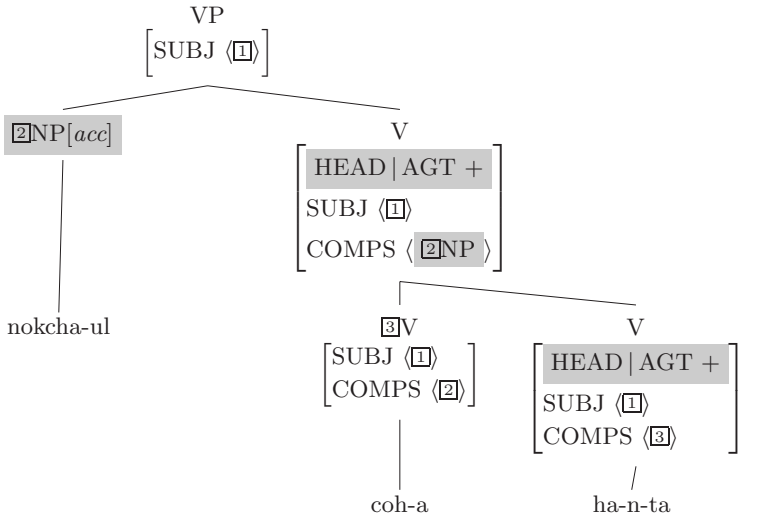
The rule specifies that the auxiliary head combines with a lexical complement, and that the COMPS value (L) of this lexical complement is passed up to the resulting mother.

Given these basic assumptions, the sentence (8)b would have the following structure:

[11] The HEAD feature LEX distinguishes a phrasal element from a lexical element. We take a complex predicate to be [LEX +] rather than [LEX −].

[12] See Bratt (1996) for concrete evidence to treat auxiliary verb constructions as complex predicates.

(18)



The psych verb lexeme *coh-* 'like' takes two arguments: one realized as subject (experiencer) and the other as a complement (theme). The auxiliary verb *ha-n-ta* 'do-PRES-DECL', selecting the main verb *coh-a* 'like-COMP' as well as the subject, forms a complex predicate with the verb. When the auxiliary combines with the main verb, the result inherits the main verb's COMPS value in accordance with the rule in (17). The complex predicate inherits the head feature [AGT +] from its head auxiliary verb. The Head-Complement Rule A requires the complement of this agentive complex predicate to be *acc*, rather than *nom*.

2.2 Free Case Alternation in Auxiliary Constructions

Although the cases discussed in the previous section allow only one case value, constructions with auxiliary verbs like *siph-* 'would-like' allow both:

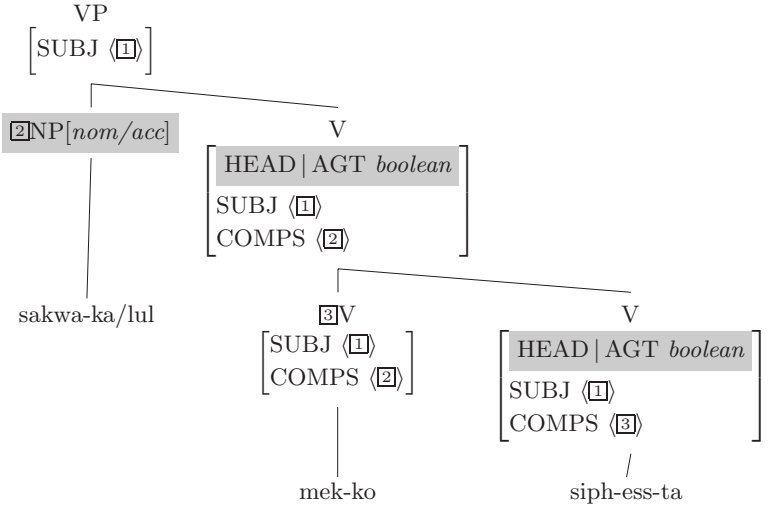
(19) John-i sakwa-ka/lul mek-ko siph-ess-ta
John-NOM apple-NOM/ACC eat-COMP would-like
'John would like to eat apples.'

The simple solution for such cases comes from the lexical information of the auxiliary *siph-ess-ta* 'would.like-PST-DECL':

(20)
$$\begin{bmatrix} \text{ORTH} \langle \text{siph-ess-ta 'like-PST-DECL'} \rangle \\ \text{HEAD} \begin{bmatrix} \text{AUX} + \\ \text{AGT } \textit{boolean} \end{bmatrix} \\ \text{VAL} \begin{bmatrix} \text{SUBJ} \langle \boxed{1} \rangle \\ \text{COMPS} \left\langle \begin{bmatrix} \text{LEX} + \\ \text{SUBJ} \quad \langle \boxed{1} \rangle \end{bmatrix} \right\rangle \end{bmatrix} \end{bmatrix}$$

Unlike agentive auxiliary verbs like *ha-*, this kind of auxiliary verb underspecifies its AGT value. This implies that its complement can either *nom* or *acc*, as represented in the following:

(21)



The feature value *boolean* can be either positive (+) or negative (−). This would then mean that the complement of the complex predicate can get either *nom* or *acc* as its case value in accordance with the Head-Complement Rule A and B in (13).

3 Further Merits of the Feature Unification

3.1 Two Nominative Cases

As noted in Yoon (1995), one tricky case pattern is the double occurrence of nominative markers:

(22) sensayngnim-kkeyse-man-i o-si-ess-ta
teacher-HON.NOM-only-NOM came
'Only the honorable teacher came.'

The marker *-kkeyse* here functions as a honorific subject marker and occupies the same morphological slot as the postposition marker. This marker cannot mark nominative objects or adjuncts: It marks only honorable nominative subjects. This implies that the stem produced by the attachment of *kkeyse* carries at least the following information:

(23)
$$\begin{bmatrix} \text{ORTH} \langle \text{sensayngnim-kkeyse 'teacher-HON.NOM'} \rangle \\ \text{HEAD} \begin{bmatrix} \text{POS noun} \\ \text{HON} + \\ \text{CASE} \mid \text{GCASE } \textit{nom} \end{bmatrix} \end{bmatrix}$$

The [GCASE *nom*] value accounts for why this stem can combine only with the nominative marker. If we attach an accusative marker there will be a clash between [GCASE *acc*] and [GCASE *nom*]. This is not a possible feature unification:

(24) $$*\begin{bmatrix}\text{ORTH } \langle\text{sayngkakha-kkeyse-man-ul 'teacher-HON.NOM-DEL-ACC'}\rangle \\ \text{HEAD}\begin{bmatrix}\text{POS noun} \\ \text{HON } + \\ \text{CASE}\begin{bmatrix}\text{GCASE } \textit{nom} \\ \text{GCASE } \textit{acc}\end{bmatrix}\end{bmatrix}\end{bmatrix}$$

3.2 Case Omission and Delimiters

Another welcoming consequence of the present analysis in which the unification and subsumption operations of feature structures play key roles in the KPSG comes from phenomena where case markers are not realized or replaced by delimiters. One main property of case markers is that they can be omitted or can be replaced by delimiters in proper context:

(25) haksayng-(tul) chayk-(to) ill-ess-e
student-PL book-even read
'Students even read a book.'

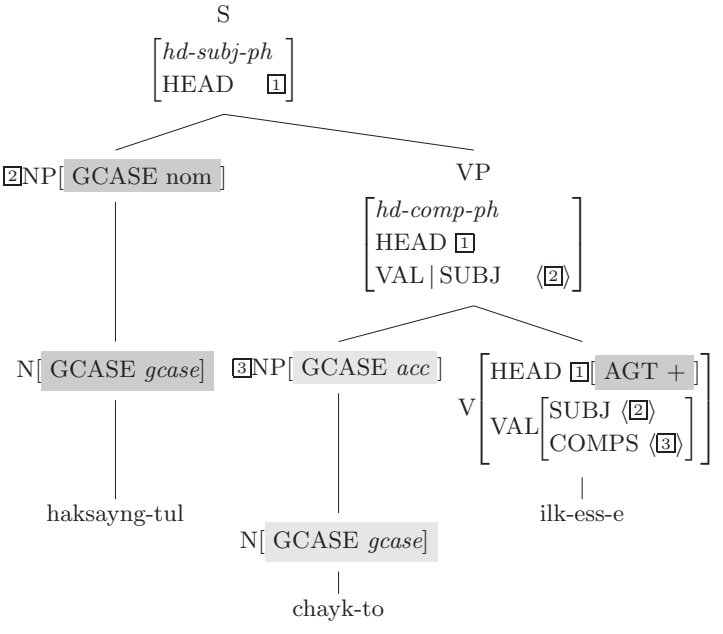
The basic lexical entries for the words in (25) would be something like the following:

(26) a. $$\begin{bmatrix}\text{ORTH } \langle\text{ilk-ess-e 'read-PST-DECL' }\rangle \\ \text{HEAD}\,|\,\text{AGT } + \\ \text{ARG-ST } \langle \text{NP}\boxed{\text{GCASE } \textit{vcase}}, \text{NP}\boxed{\text{GCASE } \textit{vcase}}\rangle\end{bmatrix}$$

b. $$\begin{bmatrix}\text{ORTH } \left\langle\text{haksayng-tul 'student-PL'}\right\rangle \\ \text{HEAD}\begin{bmatrix}\text{POS } \textit{noun} \\ \text{CASE}\left[\text{GCASE } \textit{gcase}\right]\end{bmatrix}\end{bmatrix}$$ c. $$\begin{bmatrix}\left\langle\text{chayk-to 'book-also'}\right\rangle \\ \text{HEAD}\begin{bmatrix}\text{POS } \textit{noun} \\ \text{CASE}\left[\text{GCASE } \textit{gcase}\right]\end{bmatrix}\end{bmatrix}$$

Note that the nouns here, projected to NPs, are not specified with any grammatical case value even though they may have semantic information coming from the delimiters. The present analysis generates the structure (27) to the sentence (25). As represented in the tree structure, since *gcase* is the supertype of *nom* and *acc*, there is no unification failure between the case information on the lexical element and the case requirement imposed by the Head-Subject and Head-Complement Rule. For example, in accordance with the Head-Complement Rule A, the complement of the agentive head must be *acc*, but the complement itself bears *gcase*. Since *gcase* is the supertype of *acc*, there is no feature clash. The case hierarchy, together with the feature unification and subsumption, thus allows us to capture no realization of the case markers in a straightforward manner.

(27)



3.3 Dative Cases, Case Stacking, and Alternation

Benefactive constructions such as the following have also been an important issue in case theory:

(28) John-i chayk-ul **Mary-(eykey)-(lul)** cwuessta
John-NOM book-ACC Mary-DAT-ACC gave
'John gave a book to Mary.'

Traditionally, it has been assumed that the dative here is assigned by the lexical predicate *cwuessta* 'gave'. Within a system that has no distinction between grammatical and semantic case, assigning *dat* to the benefactive argument is no surprise. However, our system, in which *dat* is a kind of semantic case, different from grammatical cases, calls upon no such lexical specification. The present system also assigns *vcase* to the benefactive argument in the lexicon:

(29) $\begin{bmatrix} \text{SYN} \mid \text{POS } \textit{verb} \\ \text{ARG-ST} \left\langle \text{NP}\left[\text{GCASE } \textit{vcase}\right], \text{NP}\left[\text{GCASE } \textit{vcase}\right], \text{NP}\left[\text{GCASE } \textit{vcase}\right] \right\rangle \end{bmatrix}$

The case value on the beneficiary is determined from the interactions with semantics. For example, we could posit constraints like (30) that associate a right semantic role to a right semantic case (cf. Bratt 1996, Choi 2003):

(30) $\textit{v-lxm} \rightarrow \begin{bmatrix} \text{ARG-ST } \langle ..., [\text{SCASE } \textit{dat}]_i, ... \rangle \\ \text{SEM}\begin{bmatrix} \textit{predication} \\ \text{GOAL i} \end{bmatrix} \end{bmatrix}$

Given such constraints, the lexical information in (29) could be expanded as following (cf. Choi 2003):

(31)
$$\begin{bmatrix} \text{SYN} \mid \text{HEAD} \begin{bmatrix} \text{POS } \textit{verb} \\ \text{AGT } + \end{bmatrix} \\ \text{ARG-ST} \left\langle \text{NP}_i, \text{NP}_j, \text{NP}_k \begin{bmatrix} \text{GCASE } \textit{vcase} \\ \text{SCASE } \textit{dat} \end{bmatrix} \right\rangle \\ \text{SEM} \begin{bmatrix} \text{RELATION } \textit{give} \\ \text{AGENT i} \\ \text{THEME j} \\ \text{GOAL k} \end{bmatrix} \end{bmatrix}$$

One immediate welcoming prediction of this analysis is the co-occurrence of this semantic case together with a grammatical case. As noted in (28), the benefactive argument can occur either with or without the accusative marker (cf. Choi 2003). Remember that the Head-Complement Rule A assigns *acc* to all the complements of an agentive verb. This would then allow us to assign *acc* to the benefactive argument in (30). The appearance of the semantic case is licensed by an independent semantic constraint such as (30). Though there exist more complicated cases that allow case alternation between *dat* and grammatical cases, and that require a more fine-grained theory of semantic roles (cf. Choi 2003), the present analysis could provide a firm base for such puzzling case alternation.

4 Testing the Feasibility of the System and Conclusion

The KPSG we have built within the typed-feature structure system and well-defined constraints, eventually aiming at working with real-world data, has been first implemented into the LKB.[13] In testing its performance and feasibility, we used the 231 (grammatical and ungrammatical) sentences from the literature and 292 sentences from the SERI Test Suites '97 (Sung and Jang 1997) designed to evaluate the performance of Korean syntactic parsers:

(32)

	# of Sentences	# of Words	# of Lexemes
SERI	292	1200	2679
Literature	231	1009	2168
Total	523	2209	4847

Of the 2209 words, the number of nominal elements is 1,342. These nominal elements include total 1,348 particles, which can be classified as follows:

[13] The space does not allow us to explicate the morphological and semantic system of the KPSG in Korean. As for morphology, we integrated MACH (Morphological Analyzer for Contemporary Hangul) developed by Shim and Yang (2002). This system segments words into sequences of morphemes with POS tags and morphological information.

(33)

	NOM	ACC	GEN	Delimiter	Semantic cases	Vocative	Total
Number	514	401	14	152	265	2	1,348

The system correctly generated all these 2209 words with or without case markings. The words generated from the lexicon which was built upon the type hierarchy with relevant constraints on each type are appropriately projected in the syntax under the defined case constraints. In terms of parsing sentences, the KPSG correctly parsed 274 sentences out of 292 SERI Test Suites and 223 out of 231 literature sentences, failing 26 sentences (497 out of 523 sentences). Failed sentences are related to the grammar that the current system has not yet written. For example, the SERI Test Suites include examples representing phenomena such as honorification, coordination, and left dislocation of subject. It is believed that once we have a finer-grained grammar for these phenomena, the KPSG will resolve these remaining sentences. Another promising indication of the test is that its mean parse (average number of parsed trees) for the parsed sentences marks 2.25, controlling spurious ambiguity at a minimum level.

As noted here, the test results provide clear evidence that the KPSG, built upon typed feature structure system, offers high performance and can be extended to large scale of data. Since the test sentences here include most of the main issues in analyzing the Korean language, we believe that further tests for designated corpus will surely achieve nearly the same result of high performance too.

References

Bratt, Elizabeth. 1996. *Argument composition and the lexicon: Lexical and periphrastic causatives in Korean.* Ph.D. dissertation, Stanford University.

Cho, Young-mee Yu and Peter Sells. 1995. A lexical account of inflectional suffixes in Korean. *Journal of East Asian Linguistics* 4, 119-74.

Choi, In-Cheol. 2003. *Case and Argument Structure in Korean and English.* Ph.D. Dissertation. Univ.of Texas at Austin.

Copestake, Ann. 2002. *Implementing Typed Feature Structure Grammars.* CSLI Publications.

Kim, Jong-Bok. 1998. Interface between Morphology and Syntax: A Constraint-Based and Lexicalist Approach. *Language and Information* 2: 177-233.

Kim, Jong-Bok and Jaehyung Yang. 2004. Projections from Morphology to Syntax in the Korean Resource Grammar: Implementing Typed Feature Structures. In *Lecture Notes in Computer Science* Vol.2945: 13-24. Springer-Verlag.

Shim, Kwangseob and Yang Jaehyung. 2002. MACH: A Supersonic Korean Morphological Analyzer. In *Proceedings of Coling-2002 International Conference*, pp.939-45, Taipei.

Sung, Won-Kyung and Myung-Gil Jang. 1997. SERI Test Suites '95. In *Proceedings of the Conference on Hanguel and Korean Language Information Processing.*

Yoo, Eun-Jung. 2002. Auxiliary Verbs and Structural Case Assignment in Korean. *Langauge Research* 38.4: 1009–1036.

Yoon, James Hye-Suk. 1995. Nominal, Verbal, and Cross-Categorial Affixation in Korean. *Journal of East Asian Linguistics*, 4: 325–356.

A Computational Model of the Spanish Clitic System

Luis A. Pineda and Ivan V. Meza

Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas (IIMAS)
Universidad Nacional Autónoma de México (UNAM)
{luis, ivanvladimir}@leibniz.iimas.unam.mx

Abstract. In this paper a computational model of the Spanish clitic system is presented. In this model clitic pronouns receive a dual analysis in which enclitics are considered inflexions while most proclitics are considered independent lexical units, hence proper clitics. The model covers the analysis of simple periphrases that, in addition to auxiliary and modals, have a single content verb (e.g. *puede comérselo*, *se lo ha querido comer*) and also the analysis of complex periphrases with more than one content verb (e.g. *le hubiera visto comérsela*, *se la hubiera visto comer*). The model introduces three operations on clictis: cancellation, composition and subsumption, and is formalized in Head-driven Phrase Structured Grammar; the standard machinery of this theory is extended with one combination scheme, the head-proclitic rule, and one principle, the clitic principle, that is satisfied by Spanish clitic sentences. A computational implementation of the theory with the Linguistic Knowledge Building (LKB) tool is also reported.

1 Introduction

Intuitively, a clitic is an unstressed particle that is attracted to a stressed word, its phonological host, and the resulting object is perceived as lexical unit[1]; unlike inflexions and derivations, that are assembled with their stems at the morpho-lexical level of linguistic representation, clitics are combined with their host at the syntactic level. According to Zwicky and Pullum (1983, pp- 503):

> "…word-clitic combinality is largely governed by SYNTACTIC considerations. The conditions governing the combinability of stems with affixes are of quite a different sort: they are MORPHOLOGICAL and/or LEXICAL in character, being concerned with the substructure of a finite set of words"

However, it is not always clear what is the linguistic level of representation for a given particle; in order to make this distinction Zwicky and Pullum (*ibid*.) advanced a number of criteria that we summarize as follows: (1) inflexions attach to words of specific syntactic categories while clitics do not exhibit this restriction, so clitics can attach to words of different categories and they often do so, (2) the combination host-clitic is very regular while inflexions show exceptions, (3) the meaning of clitic-host

[1] See, for instance, the introduction of Nevis (1991).

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 73 – 82, 2005.

combinations is the same as the meaning of expressions that show no such reduction (e.g. *she is gone* means the same as *she´s gone*) but inflexions do show idiosyncrasies, (4) cliticizised forms cannot be affected by syntactic operations, while affixed words can (e.g. no syntactic rule treats *I´ve* as a constituent[2]) and (5) clitics can attach to combinations already cliticisized, but inflexions cannot attach to already inflected words. Following these criteria Miller and Sag (1995) and also Abeillé *et al.* (1996) have classified French clitic pronouns as inflexions (*pronominal affixes* in Miller and Sag´s terminology) and Monachesi (1999) has adopted a similar criteria for Italian; however, the case for Spanish is not that clear: according to (1), and perhaps (2), clitic pronouns behave more like inflexions; according to (3) clitics present a dual behaviour, and according to the other three they behave more like clitics[3]. These criteria reflect a further implicit intuition about the architecture of the grammar and assume that the morpho-lexical and syntactic levels of representation are independent, and that the internal structure of units assembled in the former level (i.e. words) cannot be altered or broken down by syntactic operations. Consequently, if the combination takes place at the syntactic level, the resulting unit is a pseudo-word, or rather a clitic-host combination.

From this consideration, a common test to distinguish clitics from affixes is whether the particle can have a wider scope over coordination (point (4) in the list above): if the pronouns are inflexions assembled with the verb by a morphological operation, they cannot be factored out in coordination operations. However, in Spanish, *lo llevó y lo puso sobre la mesa* (he/she took it and put it on the table), for instance, can also be expressed as *lo llevó y puso sobre la mesa*, which is grammatical and has the same meaning. In other cases the grammaticality of the second form is marginal, as in *le gusta y quiere* (she likes him and loves him) and in others the construction is clearly ungrammatical as shown by *te vas o te quedas* (you go or you stay) versus **te vas o quedas*. The rule seems to be that when the pronoun substitutes the direct or indirect complement of a transitive verb, it can appear either next to their verbal host within a coordination or move out from this construction as a single realization; if the pronoun appears next to an intransitive verb, on the other hand, it cannot be moved out and has to be realized attached to its phonological host. In this latter case it behaves like an inflexion.

Further evidence about the realization of some proclitics as words is provided by interruptions and repairs in spontaneous speech; in our corpus, forms like *me…muéstrame otra vez los muebles* (to-me … show-me again the furniture) appear often (Villaseñor *et al*., 2001; Pineda *et al*., 2002); despite that words can be interrupted in inter-syllable positions, we have observed no cases in which the interruption splits off a stem from its inflexion. Accordingly, if the proclitic were an inflexion it could not be split off after lexical realization.

On the basis of these considerations, we propose a dual analysis for clitic constructions: on the one hand enclitics are considered inflexions, but proclitics that represent normal complements of verbs are considered independent lexical units, which combine with their phonological host in the syntax and are proper clitics; on

---

[2] Although this cannot be ruled out altogether if surface structure and intonation receive an incremental integrated analysis, as in Categorial Grammar (Steedmann, 1991).

[3] See also Klavans (1985).

the other hand, clitic pronouns that substitute complements with an idiosyncratic character (e.g complements to intransitive verbs, reflexive and pseudo-reflexive verbs, some ethical datives, and the adjectival phrases in attributives), either proclitics or enclitics, are considered inflexions.

2 The Basic Model

In the basic form of the phenomenon clitic pronouns substitute the direct and indirect object of verbs by accusative and dative pronouns that appear next to verb by its right or left side, forming the enclitic and proclitic constructions respectively. In simple clitic sentences there is only one verb of content, and the clitic pronouns substitute its arguments. Also, in non-periphrastic constructions the verb is both the cliticisized object and the phonological host. For instance, in *el padrino le sirve una copa al muchacho, y* ***éste se la da a la novia***[4] (the best man pours the glass to the boy, and he gives it to the bride[5]), the pronouns *se* and *la* substitute the direct and indirect objects of the verb *da*/gives (i.e. *una copa* (the glass) and *la novia* (the bride) respectively); also, the clitic *se* is a duplication of the explicit realization of the complement. The examples (1) illustrate the "standard" sentence of the previous example and a set of possible variations including clitic pronouns.

(1) a. *El padrino da* [*la copa*]$_i$ [*a la novia*]$_j$
The best man gives the glass$_i$ to the bride$_j$
b. *dala*$_i$ [*a la novia*]$_j$
c. *dale*$_j$ [*la copa*]$_i$
d. *dase*$_j$*la*$_i$
e. *dase*$_j$*la*$_i$ [*a la novia*]$_j$
f. *la*$_i$ *da* [*a la novia*]$_j$
g. *le*$_j$ *la*$_i$ *da* [*la copa*]$_i$
h. *se*$_j$ *la*$_i$ *da*
i. *se*$_j$ *la*$_i$ *da* [*a la novia*]$_i$

However, when clitics occur in periphrases, the phonological host can be an auxiliary or modal verb[6] different from the cliticisized one as in ***el post no lo he podido escribir por la mañana***[7] (I have not been able to write the post in the morning); we give two alternative realizations of this sentence in (2); although in (2.b) the cliticisized verb *escribir* (to write) is also the phonological host, in (2.c) the cliticisized verb and the phonological host (i.e. *haber*[8]) are different.

---

[4] The main examples in this paper were extracted from the internet, which we consider our corpus for the present paper. Other sentences sequences (1) to (4) are variants of the reference one that are acceptable for native speakers.

[5] http://omega.ilce.edu.mx:3000/sites/litinf/huasteca/html/sec_45.htm

[6] We adopt Gili Gaya's terminology and call modal verbs to intentional verbs appearing in periphrasis.

[7] http://blogs.ya.com/vivirsintabaco/

[8] In our model, auxiliary verbs are subject raising as they are not agentive, and their syntactic subject is the same as the subject of its complement, which is a verbal phrase; similarly, modals, like *querer* are subject control, as they also share their subject with their verbal phrase complements, although these latter forms are agentive (Pineda and Meza, 2004).

(2) a. *No he podido escribir* [*el post*]$_i$
Not have been-able to-write the post
I have not been able to write the post
b. *No he podido escribirlo*$_i$
c. *No lo he podido escribir*$_i$

For this reason we distinguish between the clitic host, the cliticisized verb, from the phonological host, and we say that in a well-formed clitic sentence the pronouns attached to the phonological host cancels the corresponding arguments of the clitic host. Following Miller and Sag (*ibid*.) and Monacheci (*ibid*.), we consider cliticizised verbs as valence reduced realizations of their basic forms, which require overt complements. We define cliticization as a lexical operation on the basic form of verb; this operation removes the cliticisized arguments from its complements list, and places them in a *clitic-lists* attribute which, in conjunction with the subject and complement attributes, defines the valence of verbs. Our approach has a lexical orientation and we postulate no movement, traces or empty categories, and non-local dependencies are captured through structure sharing, as commonly done in categorical and unification formal approaches to grammar. The model is framed in HPSG (Pollard and Sag, 1994; Sag and Wasow, 1999), and cancellation operations are defined through the standard combination principles of this theory (e.g. head-complement rule, head-specified rule, the GAP principle, etc.). For clictic cancellation to take place, the clitic host must be within the scope of the phonological host (e.g. *pudo verlo comersela* versus **la pudo verlo comerse*) as will be illustrated below.

Clitic pronouns sequences present a rigid and idiosyncratic order that poses a challenge to the analysis of the phenomenon. In our model we postulate that there is a clitic lexicon which codifies all clitic sequences that occur in a dialect, with the corresponding order and case information, and there is an entry in the clitic lexicon for each sequences of one, two or possible three pronouns; clitic pronouns have a default case (e.g. *lo* and *la* are accusative and *le* and *se* dative) but they can be used with a different case (e.g. *le* and *se* can be accusative given rise to the so-called *leísmo*) and we define an entry in the clitic lexicon for each sequence of pronouns with a different case assignment. This approach permits to analyze simple clitic sentences in terms of a single cancellation operation. We distinguish three cases: (a) simple lexical cancellation, (b) composite lexical cancellation and (c) syntactic cancellation. Simple lexical cancellation is defined in terms of a lexical rule that implements cliticization and performs the insertion of the pronouns in a single operation, permitting the analysis of (1b-1e) and (2b), for instance. Composite lexical cancellation is defined in terms of two lexical rules: one implements the cliticization operation on the clitic host, and the other performs lexical insertion on the phonological host if structure sharing between the clitic lists of both the clitic and phonological hosts is permitted (i.e. through the head-complement rule), as in (*la reina pudo haberlo visto y escuchado*/the Queen could have seen it and listened it). Finally, syntactic cancellation is analyzed in terms of the lexical rule that cliticisizes the host, and the head pro-clitic rule that combines an entry in the clitic lexicon with a verbal phrase if the structure of the clitic list attribute of the predicate corresponds with the structure of the sequence in the clitic lexicon (e.g 1f-1i and 2c); this rule captures the intuition that proclitics are proper clitics.

3 Complex Periphrasis

The model presented so far follows closely Monachesi´s analysis for Italian, with the exception of the use of the head-proclitic rule whose corresponding effect in Monachesi´s affixial approach is achieved through lexical rules; however, the analysis of the Spanish complex periphrases with more than one content verb motivates further our dual analysis. In *se lo oi decir en varios reportajes*[9] (I hear him to say it in several interviews) the subjects of the two content verbs are different (the speaker is the one who listens but a third party is the one who says it); in addition, the syntactic object of *oí* (hear) is shared with the subject of *decir* (to say) and the composite verbal phrase *oi decir* has a composite direct object "*se lo*". Examples (3) presents the "standard" non-cliticisized sentence and some of its clitized variations:

(3) a. *Oí* [*a el*]$_i$ *decir* [*el comentario*]$_j$
hear to him$_i$ to-say the comment$_j$
I hear him to say the comment
b. **Oilo$_i$ decirlo$_j$*
c. *Oyélo$_i$ decirlo$_j$*
d. **Oyélo$_i$lo$_j$ decir*
e. *Oyése$_i$lo$_j$ decir*
f. *Le$_i$ oi decirlo$_j$*
g. *Se$_i$ lo$_j$ oí decir*

In this sequence, the clitict *se lo* occurs as an enclitic in (3e) but as a proclitic in (3g). In this case, both of the pronouns are in the accusative (i.e. substitute direct objects) and *se* is used instead of *le* (with *leísmo*) or *lo*, as no sequence of two *l´*s pronouns is allowed in Spanish (e.g. 3.d). The sequence shows that two clitic hosts can compose their accusative clitizations if they are next to each other (i.e. accessible), and the result of this operation is composite clitic argument. We refer to this operation as clitic composition. This operation is implemented through lexical rules and structure sharing, and clitic sentences of this form are also analyzed in terms of single cancellation. The ungrammaticality of (3b) is due to an idiosyncratic lexical restriction of Spanish for the phonological host, as participles and finite forms (but imperatives) cannot have enclitics, while infinitive, imperatives and gerunds require enclitics always.

The composition operation illustrated in (3) "builds" a clitic word in which all constituting pronouns have a different referent; however, this is not always the case. In ***la vi comiéndose la mesa fría*** *con los ojos*[10] (I saw you/her eating the cold table with the eyes) the verb *comer* (to eat) has an idiosyncratic dative complement that co-refers with its subject, forming an ethical dative that marks that the subject of this action is also its beneficiary. We present some variations of this sentence in (4):

[9] http://www.carp.org.ar/eng/idolos.php3
[10] http://www.mundomatero.com/chistes/junio2000.html

(4) a. *Vi* [*a usted*]$_i$ *comiendo* [*la cena*]$_j$ [*por/para usted*]$_i$
see to you$_i$ eating the dinner$_j$ for you$_i$
I see you eating the dinner for you own sake
b. *Vi* [*a usted*]$_i$ *comiendose*$_i$ [*la cena*]$_j$
c. *Vi* [*a usted*]$_i$ *comiendose*$_i$*la*$_j$
d. **Víla*$_i$ *comiendose*$_i$*la*$_j$
e. *Vela*$_i$ *comiendose*$_i$*la*$_j$
f. **Vela*$_i$+*se*$_i$*la*$_j$ *comiendo*
g. *Vese*$_i$*la*$_j$ *comiendo* (i.e. *se*$_i$ = *la*$_i$+*se*$_i$)
h. *Se*$_i$ *la*$_j$ *vi comiendo*
i. *La*$_i$ *vi comiendose*$_i$*la*$_j$

Sentences (4a) does not really occur in the language and it is used only as an aid to illustrate the meaning of (4b) in which *comer* has already the dative reflexive *se* as enclitic; the clicitization of the direct objects of *vi* and *comiendo* gives rise to the composite predicate *vi comiendo,* with a composite direct object represented by *la*$_i$+*se*$_i$*la*$_j$. However, in this composition the object of *visto* co-refers with the dative *se* of *comiendo*, and the redundant form *la*$_i$+*se*$_i$ is reduced as *se*$_i$, with the dative case prevailing, and the remaining *se*$_i$*la*$_j$ form represents the whole of the composite clitic argument as shown (4g) and (4h) in the enclitic and proclitic forms respectively. We refer to the reduction of this argument, in which an accusative pronoun is subsumed by a co-indexed dative form, as clitic subsumption. If the co-indexed arguments have the same case, they can also be subsumed in a composition. The analysis of sentences with clitic subsumption is carried out with a single cancellation operation, and the ungrammaticality of (4d) is due to the lexical restriction on participles and finite forms for enclitics. (4f) shows, in addition, that two co-indexed pronouns cannot occur next to each other, and subsumption is obligatory, as shown in (4.g).

Next we illustrate the analysis of (4.h). The lexical entry of the word "*se la*" in the clitic lexicon is shown in Figure 1. This entry has a local *synsem* attribute with the attributes of category CAT and the restriction of the semantic content *CONT/RESTR*. Also the head value of this entry is *clitic*.

$$\left[\begin{array}{ll} se \quad la & \\ SS \mid LOC & \left[\begin{array}{ll} CAT & \left[\begin{array}{ll} HEAD & clitic \\ VAL \mid CL-LIST & \left\langle NP-dat_i, \quad NP-acc_j \right\rangle \end{array}\right] \\ CONT \mid RESTR & \left\{ [person-rel]_i, [female-rel]_j \right\} \end{array}\right] \end{array}\right]$$

Fig. 1. Clitic word

Now, we come to the cliticization of *comiendo* (the gerund of *comer*/to eat). The basic lexical entry for the verb *comer* is illustrated in Figure 2. The lexical rule that cliticisizes the verb is shown in Figure 3; in addition to including the direct object in the *CL-LIST*, this rule also adds an idiosyncratic extra complement in the *CL-LIST*, with a dative case (i.e. *se*), which is co-indexed with its subject, producing the reflexive connotation of the ethical dative.

$$\begin{bmatrix} tr-vrb & & & \\ SS \mid LOC & CAT & \begin{bmatrix} ORTH & comer \\ HEAD & verb \\ VAL \mid CL-LIST & \langle\ \rangle \\ ARG-ST & \langle NP_i, NP_j \rangle \end{bmatrix} \\ & CONT \mid RESTR & \left\{ \begin{bmatrix} show-rel & \\ ACT & i \\ UND & j \end{bmatrix} \right\} \end{bmatrix}$$

Fig. 2. Lexical entry for *comer*

$$\begin{bmatrix} ORTH & \#1 \\ HEAD & \#2 \,\&\, \begin{bmatrix} verb & \\ FORM & fin \end{bmatrix} \\ VAL \mid CL-LIST & \langle\ \rangle \\ ARG-ST & \langle \#3 \rangle \oplus \#a \end{bmatrix} \mapsto \begin{bmatrix} ORTH & \#1 \\ HEAD & \#2 \\ VAL \mid CL-LIST & \#a \\ ARG-ST & \langle \#3 \rangle \end{bmatrix}$$

Fig. 3. Cliticization rule for content verbs

Now we turn to the production of cliticisized *vi*. The basic form of object-control verbs is shown in Figure 4 and its cliticization rule in Figure 5. This rule removes the direct object from the complement list of the verb and includes it in its *CL-LIST* attribute; this argument is added on to the clitic list of its complement verb (e.g. *comiendo*), defining in this was a clitic composition. However, this clitic argument is co-indexed with the dative cliticisized argument of the second verb, and these two complements (of *vi* and *comer*) represent the same object and are subsumed into one.

$$\begin{bmatrix} ocv-lxm & & \\ SS \mid LOC \mid CAT \mid VAL & SUBJ & \langle [\ ] \rangle \\ & COMPS & \left\langle \#1 \,\&\, NP-acc_i, \begin{bmatrix} SUBJ & \#1_i \\ COMPS & \langle\ \rangle \end{bmatrix} \right\rangle \end{bmatrix}$$

Fig. 4. Lexical entry for *ocv-lxm*

The analysis of the final sentence is shown in Figure 6. The lexical entries for the verbs are produced by the lexical rules in Figure 3 and 5 out of the lexical entries in Figures 2 and 4 respectively; these combine to form the clitic composition *vi comiendo*, which in turn is combined with the clitic word "se_{DAT} la_{ACC}" through the Head-Proclitic rule that implements the syntactic cancellation scheme.

$$\begin{bmatrix} ocv-lxm & \\ ORTH & \#1 \\ HEAD & \#2 \\ ARG-ST & \langle \#3, []_i, \#4 \rangle \end{bmatrix} \mapsto \begin{bmatrix} ORTH & \#1 \\ HEAD & \#2 \\ VAL \mid CL-LIST & \langle NP-acc_i \rangle \oplus \#a \\ ARG-ST & \langle \#3, \#4 \,\&\, [CL-LIST \quad \#a] \rangle \end{bmatrix}$$

Fig. 5. Cliticization lexical rule for object-control verbs

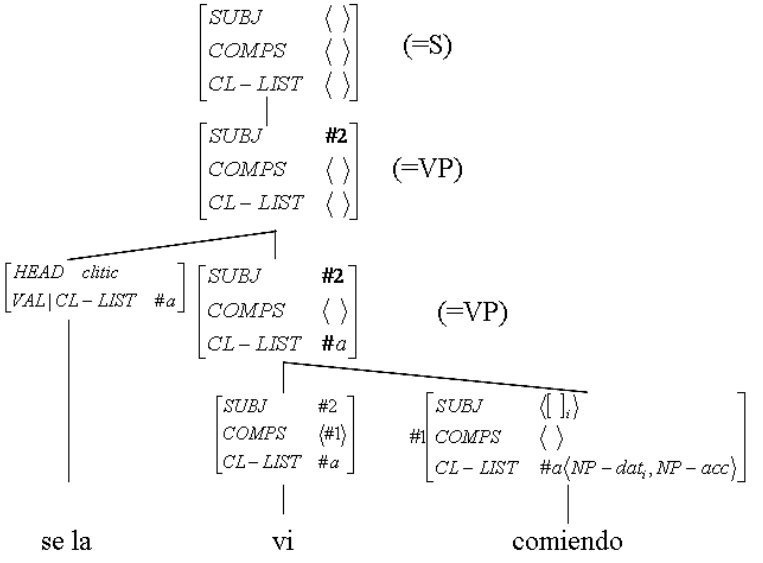


Fig. 6. Analysis of sentences with clitic subsumption

The reflexive connotation of the co-indexed pronouns can be better appreciated in (4i) *La*$_i$ *vi comiendose*$_i$*la*$_j$ where the cliticizations of both of the clitic hosts is not composed, and the direct object of *vi* appears as proclitic but the two complements of *comer* appear as enclitics; here, the two direct objects can be realized with the accusative *la* despite that they have different referents: the proclitic refers to the woman and the enclitic to the dinner; nevertheless, the proclitic is still co-indexed with the indirect object of *comer* represented by *se*, hence the reflexive interpretation. The analysis of this construction requires two cancellations: simple lexical cancellation by the right and syntactic cancellation by the left, but in both of these cases the clitic host is within the scope of its corresponding phonological host, and the two scopes do not overlap. We say that this kind of constructions has two independent clitic domains, and the sentence is analyzed in terms of one cancellation per independent clitic domain. More generally, the clitic host is within the scope of the phonological host if the former is within the clitic domain of the latter, and there is a binding path allowing the co-referring relation.

The composition and subsumption operations have an additional consequence: in coordinated structures, like *lo llevó y puso sobre la mesa*, the co-indexed cliticizations of both of the verbs are composed, and one argument is reduced by clitic subsumption

too, resulting in an composite clitic argument which is factored out as a proclitic to the whole coordination, and the analysis requires the head pro-clitic rule, as illustrated in Figure 7.

On the basis of these observations we propose the following clitic principle: Spanish clitic sentences can be analyzed in terms of one cancellation operation per independent clitic domain, and the clitic composition and subsumption operations. Or more simply: a cliticization either basic or produced through composition or subsumption must be within the scope of its phonological host.

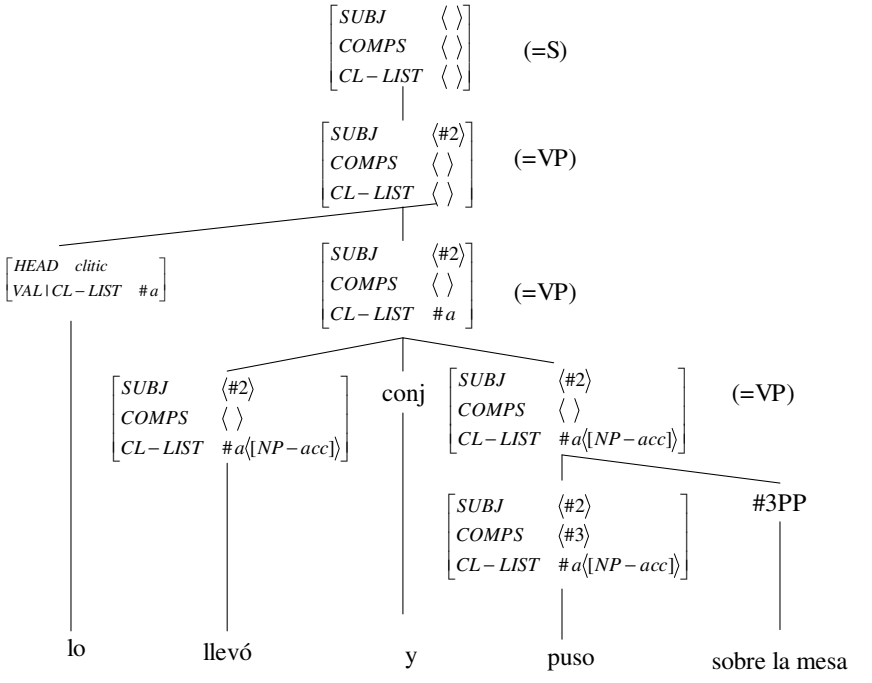


Fig. 7. Analysis of clitic coordinated sentence

4 Conclusions and Implementation

In this paper we have presented a theory for the analysis of Spanish clitic system with a dual character: proclitics that represent "normal complements" like direct and indirect objects of transitive verbs are independent lexical units and hence proper clitics, while enclitics are inflexions; other proclitics, representing extra complements (i.e. arguments extending the basic argument structure of the verb), whether these are proclitics or enclitics, are inflexions (e.g. *me voy*, *comerse*), and these attach to their hosts as lexical idiosyncrasies. In this theory the arguments of the cliticisized verb must be within the scope of the phonological host, and there is a single cancellation per independent clitic domain. Composite predicates with two content verbs can be formed by the clitic composition operation, and co-indexed arguments in compositions can be subsumed, producing composite predicates, as in complex periphrases and coordination. The theory postulates that the structure of clitic

sentences involves an underlying phenomenon of argument reduction which is a natural way to account for sentences involving complex predicates, built out of independent verbs. In reflexive sentences the subject is co-indexed with the direct or indirect object, and the reflexive relation holds as long as the second co-indexed argument is within the scope of the first; however, if the argument appears twice, due to structure sharing between constituents of composite predicates, the extra argument needs not to appear explicitly and it is reduced. The fact that this phenomena appears in unrelated constructions like complex periphrases and coordination provides further support and motivation for our analysis and theoretical machinery.

The theory has been formally developed in HPSG (Pineda and Meza, 2004) and the results are backed by its implementation in LKB (Copestake, 2002).

References

1. Abeillé, A, Godard, D., Miller, P. and Sag, Ivan. 1998. 'French Bounded Dependencies' in Luca Dini and Sergio Balari (eds.), Romance in HPSG, Standford: CSLI Publications.
2. Copestake, Ann. 2002. The LKB System, Stanford University, http://www-csli.stanford.edu/\symbol/~aac/lkb.html
3. Gili Gaya, Samuel. 1991. Curso Superior de Sintaxis Española. Biblograf, S. A., Barcelona.
4. Klavans, Judith L. 1985. The independence of syntax and phonology in cliticization. *Language* 61, 95-120.
5. Miller P. H. and Sag, Ivan. 1995. 'French Clitic Movement Without Clitics or Movement', *Natural Language and Linguistic Theory* 15, 573–639.
6. Monachesi, Paula. 1999. 'A Lexical Approach to Italian Clitization', Lecture Notes series No. 84, CSLI, Stanford, Cambridge University Press.
7. Nevis, J. A., Joseph, B. D., Wanner, D. and Zwicky, A. M. 1994. 'Clitics, A Comprehensive Bibliography 1892-1991'. Library and Information Sources in Linguistics, 22. John Benjamins Pub. Co., Amsterdam/Philadelphia.
8. Pineda, Luis, Massé, Antonio, Meza, Ivan, Salas, Miguel, Schwarz, Erik, Uraga, Esmeralda and Villaseñor, Luis. 2002. 'The Dime project', Proceedings of MICAI-2002, Lectures Notes in Artificial Intelligence 2313, pp.166–175.
9. Pineda, L. & Meza, I. Un modelo para la perífrasis española y el sistema de pronombres clíticos en *HSPG*, *Estudios de Lingüística Aplicada*, Num. 38, pp. 45-67, 2003.
10. Pineda, L. & Meza, I. The Spanish pronominal clitic system, Internal report, Department of Computer Science, IIMAS, UNAM, México, 2004 (48 pp).
11. Pollard, Carl and Sag, Ivan. 1994. Head-Driven Phrase Structure Grammar, CSLI, Stanford. The University of Chicago Press, Chicago & London.
12. Sag, Ivan and Wasow, Thomas. 1999. Syntactic Theory: A Formal Introduction, CSLI Publications, Stanford.
13. Steedman, Mark. 1991. Structure and Intonation, *Language* 67, 260–298.
14. Villaseñor, L., Massé, A. & Pineda, L. A. 2001. 'The DIME Corpus', Memorias 3º. Encuentro Internacional de Ciencias de la Computación ENC01, Tomo II, C. Zozaya, M. Mejía, P. Noriega y A. Sánchez (eds.), SMCC, Aguascalientes, Ags. México, Septiembre, 2001.
15. Zwicky, A, & Pullum, G. 1983. Clitization vs. Inflection: English N'T', *Language* 59, 502–513.

A Parallel Approach to Syllabification

Anca Dinu[1] and Liviu P. Dinu[2]

[1] University of Bucharest, Faculty of Foreign Languages,
5-7 Edgar Quinet, 70106, Bucharest, Romania
anca_radulescu@yahoo.com
[2] University of Bucharest, Faculty of Mathematics and Computer Science,
14 Academiei, 70109, Bucharest, Romania
ldinu@funinf.cs.unibuc.ro

Abstract. In this paper we propose a parallel manner of syllabification introducing some parallel extensions of insertion grammars. We use this grammars in an application to Romanian language syllabification.

1 Introduction

In formal language theory, most of the generative mechanisms investigated are based on the *rewriting* operation. Several other classes of mechanisms, whose main ingredient is the *adjoining* operation, were introduced along the time. The most important of them are *the contextual grammars* (Marcus, 1969), *the tree adjoining grammars* (TAG) (Joshi et al., 1975) and *the insertion grammars* (Galiukschov, 1981), all three of them introduced with linguistic motivations. Contextual grammars were introduced by Marcus (1969) and have their origin in the attempt to bridge the gap between the structuralism and generativism. The insertion grammars (or semi-contextual grammars) are somewhat intermediate between Chomsky context-sensitive grammars (where the non-terminal are rewritten according to specified contexts) and contextual grammars (where contexts are adjoined to specified strings associated with contexts).

In this paper we introduce some parallel extensions of insertion grammars and we use them to propose a parallel manner of word syllabification. Up to now, from our knowledge, most of the formal models of syllabification were treated in a sequential manner (Vennemann (1978), Koskenniemi (1983), Bird and Ellison (1994), Kaplan and Kay (1994), Muller (2002), Dinu (2003)).

This paper is structured as follows: in Section 2 we present *the insertion grammars* and introduce two new variants of them: *parallel insertion grammars* and *maximum parallel insertion grammars*. The syllabification of words, the definition of syllable and an application (Romanian words syllabification) of this approach of syllabification is given in Section 3.

2 Parallel Extensions of Insertion Grammars

For elementary notions of formal language theory, such as *alphabet, concatenation, language, free monoid, lengths of words,* etc. we refer to (Păun, 1997).

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 83–87, 2005.

The basic operation in insertion grammars is the adjoining of strings, as in contextual grammars, not rewriting, as in Chomsky grammars, but the operation is controlled by a context, as in context-sensitive grammars.

Definition 1 (Păun, 1997). *An insertion grammar is a triple $G = (V, A, P)$, where V is an alphabet, A is a finite language over V, and P is a finite set of triples of strings over V.*

The elements in A are called axioms and those in P are called insertion rules.

The meaning of a triple $(u, x, v) \in P$ is: x can be inserted in the context (u, v). Specifically, for $w, z \in V^*$ we write $w \Rightarrow z$ if $w = w_1uvw_2$, $z = w_1uxvw_2$, for $(u, x, v) \in P$ and $w_1, w_2 \in V^*$.

The language generated by G is defined by: $L(G) = \{z \in V^* \mid w \overset{*}{\Rightarrow} z, \text{ for } w \in A\}$.

Here we introduce two parallel extensions of insertion grammars.

Definition 2. *Let $G = (V, A, P)$ be an insertion grammar. We define the parallel derivation denoted $\Rightarrow_p$, by:*

$$w \Rightarrow_p z \text{ iff } w = w_1w_2 \dots w_r, \text{for some } r \geq 2,\ z = w_1x_1w_2x_2w_3 \dots x_{r-1}w_r \text{ and, for all } 1 \leq i \leq r-1, \text{ there is } (u_i, x_i, v_i) \in P \text{ and } \alpha_i, \beta_i \in V^* \text{ such that } w_ix_iw_{i+1} = \alpha_iu_ix_iv_i\beta_i \text{ and } w_i = \alpha_iu_i\ , w_{i+1} = v_i\beta_i.$$

Remark 1. For usual derivation $\Rightarrow$ we use one selector-pair, with no restriction; in parallel derivations the whole string is decomposed into selectors.

Definition 3. *For an insertion grammar $G = (V, A, P)$ we define the parallel derivation with maximum use of insertions (in short, we say maximum parallel derivation), denoted $\Rightarrow_{pM}$, by:*

$$w \Rightarrow_{pM} z \text{ iff } w = w_1w_2 \dots w_s,\ z = w_1x_1w_2x_2w_3 \dots x_{s-1}w_s,\ w \Rightarrow_p z \text{ and there is no } n > s \text{ such that } w = w'_1w'_2 \dots w'_n,\ z' = w'_1x'_1w'_2x'_2w'_3 \dots x'_{n-1}w'_n,\ w \Rightarrow_p z'.$$

Remark 2. The main difference between parallel derivation ($\Rightarrow_p$) and maximum parallel derivation ($\Rightarrow_{pM}$) with respect to an insertion grammar is that in the former we can insert any number of strings in a derivation step and in the later we insert the maximum possible number of strings in a derivation step.

For $\alpha \in \{p, pM\}$, we denote by $L_\alpha(G)$ the language generated by the grammar G in the mode α:

$$L_\alpha(G) = \{z \in V^* \mid w \overset{*}{\Rightarrow}_\alpha z, \text{ for some } w \in A\}.$$

The family of such languages is denoted by INS_α, $\alpha \in \{p, pM\}$.

We give here (without proofs) some results regarding the relations between INS_{pM} and Chomsky hierarchy.

Theorem 1. *INS_{pM} is incomparable to REG and CF, but not disjoint, where REG is the class of regular languages and CF is the class of context free languages.*

Theorem 2. *$INS_{pM} \subset CS$.*

3 On the Syllabification of Romanian Words via Parallel Insertion Grammars

In this section we use the insertion grammars and the maximum parallel insertion derivation to propose a parallel manner of syllabification of words.

Consider an insertion grammar $G = (V, A, P)$ and let $L_{pM}(G)$ be the language generated by G in parallel maximum mode. Set $w \Rightarrow_{pM} z$ a derivation in G, where $w = w_1 w_2 \dots w_s$ and $z = w_1 x_1 w_2 \dots x_{s-1} w_s$.

With respect to the above definitions, we define the syllables of w by:

$$Syl_{pM}(w) = \{w_1, w_2, \dots w_n\}.$$

Consider the Romanian alphabet *RO*={a, ă, â, b, c, d, e, f, g, h, i, î, j, k, l, m, n, o, p, q, r, s, ş, t, ţ, u, v, w, x, y, z} and its partition in vowels and consonants: *RO= Vow* ∪ *Con*, where *Vow*={a, â, ă, e, i, î, o, u, y} and *Con*={b, c, d, f, g, h, j, k, l, m, n, p, q, r, s, ş, t, ţ, v, w, x, z}.

Definition 4. *A word over RO is said to be regular if it contains no consecutive vowels.*

With respect to the above definitions, an insertion grammar for syllabification of Romanian regular words is $G_{syl} = (V_{syl}, A_{syl}, P_{syl})$, whose components are:

1. $V_{syl} = RO \cup \{\$\}$, where "$" is a new symbol that is not in *RO*; "$" is the *syllable boundary marker.*
2. A_{syl} is the set of the regular words over *RO* in Romanian language.
3. $P_{syl} = C_1 \cup C_2 \cup C_3 \cup C_4 \cup C_5 \cup C_6 \cup C_7 \cup C_8$ where:
   (a) $C_1 = \{(v_1, \$, cv_2) \mid v_1, v_2 \in Vow, c \in Con\}$
   (b) $C_2 = \{v_1, \$, c_1c_2v_2) \mid v_1, v_2 \in Vow, c_1c_2 \in$ {ch,gh} or $(c_1, c_2) \in$ {b, c, d, f, g, h, p, t} $\times \{l, r\}\}$
   (c) $C_3 = \{(v_1c_1, \$, c_2v_2) \mid v_1, v_2 \in Vow$ and c_1c_2 not as in the precedent case$\}$
   (d) $C_4 = \{v_1c_1, \$, c_2c_3v_2) \mid v_1, v_2 \in Vow,\ c_1c_2c_3 \notin$ {lpt, mpt, mpţ, ncş, nct, ncţ, ndv, rct, rtf, stm}$\}$;
   (e) $C_5 = \{v_1c_1c_2, \$, c_3v_2) \mid v_1, v_2 \in Vow,\ c_1c_2c_3 \in$ {lpt, mpt, mpţ, ncş, nct, ncţ, ndv, rct, rtf, stm}$\}$;
   (f) $C_6 = \{v_1c_1, \$, c_2c_3c_4v_2) \mid v_1, v_2 \in Vow,\ c_1 \in Con,\ c_2c_3c_4 \notin$ {gst, nbl}$\}$;
   (g) $C_7 = \{v_1c_1c_2, \$, c_3c_4v_2) \mid v_1, v_2 \in Vow,\ c_1 \in Con,\ c_2c_3c_4 \in$ {gst, nbl}$\}$;
   (h) $C_8 = \{v_1c_1c_2, \$, c_3c_4c_5v_2) \mid v_1, v_2 \in Vow,\ c_1c_2c_3c_4c_5 \in$ {ptspr, stscr}$\}$;

Example 1. Set the word *lingvistica*. We may have the following parallel derivations:

1. Some parallel derivations:
   $\underbrace{lin}_{w_1}\underbrace{gvisti}_{w_2}\underbrace{ca}_{w_3} \Rightarrow_p lin\$gvisti\$ca$, $\underbrace{lin}_{w_1}\underbrace{gvis}_{w_2}\underbrace{tica}_{w_3} \Rightarrow_p lin\$gvis\$tica$,
   $\underbrace{lingvis}_{w_1}\underbrace{ti}_{w_2}\underbrace{ca}_{w_3} \Rightarrow_p lingvis\$ti\$ca$, $\underbrace{lin}_{w_1}\underbrace{gvis}_{w_2}\underbrace{ti}_{w_3}\underbrace{ca}_{w_4} \Rightarrow_p lin\$gvis\$ti\$ca$,
   etc.
2. The parallel maximum derivation: $\underbrace{lin}_{w_1}\underbrace{gvis}_{w_2}\underbrace{ti}_{w_3}\underbrace{ca}_{w_4} \Rightarrow_{pM} lin\$gvis\$ti\$ca$.

Remark 3. For Romanian words, the only words which can have two different syllabifications are the words ending in "i" (e.g. ochi (noun) and o$chi (verb)) (Petrovici, 1934). If the final "i" is stressed, the rules $C_1 - C_8$ are applied ,or else the final "i" is considered as a consonant and then the same rules are applied.

Remark 4. In order to syllabicate a *non regular word*, we extracted a set of rules based on the context in which a sequence of 2-5 vowels appears. Thus, we notice that the same group of vowels has an identical syllabification if it has the same letters that precede and/or succeed it (Dinu, 2003). Once we have found a set of rules which characterize the behavior of a sequence of vowels, we use it to extend the grammar G_{syl}.

4 Conclusions

In this paper we have investigated the insertion grammars as generative models for syllabification. We introduced some constraints to the derivation relation, obtaining new classes of insertion languages: insertion languages with parallel derivation (INS_p) and insertion languages with maximum parallel derivation (INS_{pM}). Using the maximum parallel derivation we obtained an efficient method of word syllabification. We analyzed some of the relations between INS_{pM} and the Chomsky hierarchy.

References

1. Bird, S. and T. M. Ellison. One-level phonology: Autosegmental representations and rules as finite automata. *Computational Linguistics* 20, 55-90, 1994
2. Dinu, L.P., An approach to syllables via some extensions of Marcus contextual grammars. *Grammars*, 6 (1), 2003, 1-12.
3. Galiukschov, B.S. Semicontextual grammars (in Russian), *Mat. logica i mat. ling.*, Kalinin Univ. 38-50, 1981
4. *D.O.O.M.*. Ed. Acad., Bucureşti, 1982

5. Joshi, A.K., L.S. Levy, M. Takahashi. Tree adjoining grammars. *J. Computer System Sci.*, 19, 136-163, 1975
6. Kaplan, R.M. and M. Kay. Regular models of phonological rule systems. *Computational Linguistics*, 20(3), 331-379, 1994
7. Koskenniemi, K. *Two-level morphology: A general computational model for word-form recognition and production*. Doctoral dissertation, University of Helsinki, 1983
8. Marcus, S. Contextual grammars. *Rev Roum. Math. Pures Appl.* 14, 69-74, 1969
9. Müller, K. *Probabilistic Syllable Modeling Using Unsupervised and Supervised Learning Methods* PhD Thesis, Univ. of Stuttgart, Institute of Natural Language Processing, AIMS 2002, vol. 8, no.3, 2002
10. Petrovici, E. Le pseudo *i* final du roumain. *Bull. Linguistique*, 86-97, 1934
11. Păun, Gh. *Marcus Contextual Grammars*. Kluwer, 1997
12. Vennemann, T. Universal syllabic phonology. *Theoretical Linguistics* 5, 2-3, 175-215, 1978

Towards Developing Probabilistic Generative Models for Reasoning with Natural Language Representations

Daniel Marcu[1] and Ana-Maria Popescu[2]

[1] Information Sciences Institute and Department of Computer Science,
4676 Admiralty Way, Suite 1001, Marina del Rey, CA 90292
marcu@isi.edu,
http://www.isi.edu/~marcu
[2] Department of Computer Science, University of Washington,
Seattle, Washington 98105-2350, Box 352350
amp@cs.washington.edu,
http://www.cs.washington.edu/homes/amp

Probabilistic generative models have been applied successfully in a wide range of applications that range from speech recognition and part of speech tagging, to machine translation and information retrieval, but, traditionally, applications such as reasoning have been thought to fall outside the scope of the generative framework for both theoretical and practical reasons. Theoretically, it is difficult to imagine, for example, what a reasonable generative story for first-order logic inference might look like. Practically, even if we can conceive of such a story, it is unclear how one can obtain sufficient amounts of training data. In this paper, we discuss how by embracing a less restrictive notion of inference, one can build generative models of inference that can be trained on massive amounts of naturally occurring texts; and text-based deduction and abduction decoding algorithms.

1 Introduction

Reasoning is one of the most studied areas of Artificial Intelligence (Russell and Norvig, 1995). In introductory courses to Artificial Intelligence, we teach students the elegant language of first-order logic (FOL) and spend significant time explaining how they can turn inference rules into logical formulas. At the end of an AI course, good students are capable to take naturally occurring conditionals, such as that shown in (1), and turn them into well-formed first-order formulas, such as the formula shown in (2).

"If a company cuts production, its stock price will fall." (1)

$$\forall c\ \forall t1\ \forall p1\ (\text{company}(c) \wedge \text{cutProduction}(c,t1) \wedge \text{priceStock}(c,t1,p1) \rightarrow \exists\ t2,p2\ (t2 > t1\ \wedge \text{priceStock}(c,t2,p2) \wedge p2 < p1)) \quad (2)$$

The power of FOL and other formal languages comes from their well-defined semantics. Given a formula such as (2) and a set of assertions about the world (3), we can use Modus Ponens and mechanically derive new statements that are in true in all interpretation in which formulas (2) and (3) are true. For example, we can mechanically derive formula (4), which can be paraphrased as "The stock price of Crystal Lights will fall".

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 88 – 99, 2005.

$$\begin{array}{c} \text{company(CrystalLights)} \\ \text{cutProduction(CrystalLights,2004)} \\ \text{priceStock(CrystalLights,52)} \end{array} \quad (3)$$

$$\exists\ t2,p2\ (t2 > 2004\ \wedge \text{priceStock(CrystalLights,t2,p2)} \wedge p2 < 52)) \quad (4)$$

Despite their elegant syntax and semantics, FOL and other formal languages are of little use in large-scale natural language applications. To go from a natural language sentence such as "Crystal Lights recently announced that it cut production at 80% of its factories." to a set of assertions such as those in (3) is beyond the state of the art of current natural language processing techniques. And even if we can accomplish this feat, it is equally unlikely that we got lucky and encoded inference rule (2) in our knowledge base; after all, there are billions of billions of such rules that we would need to have created in order to produce a sufficiently comprehensive knowledge base.

In this paper, we propose an alternative approach to building open-domain, common-sense reasoning systems. Instead of trying to create large-scale formal knowledge bases that encode billions of assertions and inference rules and algorithms that map natural language statements into formal representations, we propose to develop algorithms that *operate directly* on natural language strings/sentences. In other words, *we propose to treat the inference problem as a translation problem*: given an input string such as "Chrystal Lights recently announced that it cut production at 80% of its factories.", we would like to *translate* it into "Crystal Lights's price stock will fall.", a likely consequent of the state of affairs described in the original sentence.

We start our discussion by reviewing the characteristics that mar the utility of first-order logic (and other formal languages) in text-based applications (Section 2). We also review briefly the core ideas specific to probabilistic noisy-channel modeling and discuss the impact of these models on a variety of applications (Section 3). We show how deductive and abductive reasoning with natural language strings can be couched in a noisy channel framework and present preliminary experiments aimed at building a string-based reasoning system (Section 4).

2 Difficulties Specific to Using First-Order Logic for Text-Based Inference

The Paraphrase Curse

Let us revisit the process of formalizing natural language statements that encode inferential knowledge. In the process of mapping if-then statements into FOL formulas, for example, what we end up doing is paraphrase natural language statements so that they fit better the constraints specific to FOL syntax. Note, for example, that the original if-then statement in (1) has been paraphrased into a logical formula (2) that can be glossed as shown below in (5) and (6)

If a company c cuts production at time t1 and the stock price at t1 is p1,
then there exists a time t2 that comes after t1, at which the price stock of c will be a value p2 that is smaller than the current value p1. (5)

For any three variables c, t1, and p1 for which the predicates company(c), cutProduction(c,t1) and priceStock(c,t1,p1) are true, there exist variables t2 and p2 such that predicates t2 > t1, priceStock(c,t2,p2), and p2 < p1 are true. (6)

It is arguable whether statements (1), (5) and (6) express the same meaning. For human consumption though, statement (1) is clearly preferred and easier to understand than (5) and (6).

The Choice of Predicate Names Curse

The choice of predicate names in equation (2) is arbitrary. An equally good FOL formulation of natural language statement (1) is the one in equation (7), which uses two predicates to express the meaning of the phrase "*stock price*". (The associated natural language rendering is given in (8)).

$$\forall c\ \forall t1\ \forall s\ \ \forall p1\ \ (\text{company}(c) \wedge \text{cutProduction}(c,t1) \wedge \text{stock}(c,s) \wedge \text{price}(s,t1,p1)\ \ \rightarrow \exists\ t2,p2\ (t2 > t1\ \ \wedge \text{price}(s,t2,p2) \wedge p2 < p1)) \quad (7)$$

If a company c cuts production at time t1, the company c has some stock s and the price at t1 of s is p1, then there exists a time t2 that comes after t1, at which the price of s will be a value p2 that is smaller than the value at time t1, p1. (8)

If we are to build inference systems that operate on textual representations, we need to have a consistent way of mapping natural language strings into logical statements. We also need to assign different predicates to the same strings, depending on context. Inferences that involve oil prices are different, for example, from inferences that involve house or computer prices. For unrestricted texts, this is currently beyond the state of the art.

The Syntax-Semantic Equivalence Trap

From a formal perspective, there is nothing in the language of first-order logic that makes equation (2) a better choice than equation (9).

$$\forall c\ \forall t1\ \forall p1\ (\text{elephant}(c) \wedge \text{cicling}(c,t1) \wedge \text{isi}(c,t1,p1)\ \ \rightarrow \exists\ t2,p2\ (t2 > t1\ \ \wedge \text{isi}(c,t2,p2) \wedge p2 < p1)) \quad (9)$$

This is because equations (2) and (9) are syntactically equivalent. In order to have meaning, they need to be associated with a semantic interpretation. It is this semantic interpretation that we give to the FOL predicates that makes the two formulas equivalent or not. If, for example, the formulas are interpreted in a world in which the predicate pairs (company, elephant), (cutProduction, cicling), and (priceStock,isi) are true for the same set of constants, then formulas (2) and (9) are equivalent. If the predicates are given a different semantic interpretation, then formulas (2) and (9) are not equivalent. It is a common error to assume that because we use predicate names that encode natural language knowledge and expectations, the inferences associated with these predicate names are going to be acceptable/reasonable.

The Low Expressive Power of First Order Logic

First order logic can express some of the facts of interest in natural language processing. It can express type facts – man(John); restaurant(CPK); property facts – rich(George); equality facts – numberOf(NP) = numberOf(VP); complex facts –

$(\forall x)(dog(x) \Rightarrow mammal(x))$; incomplete knowledge – loves(John, Mary) $\vee$ loves(John, Jane); $(\exists x)(restaurant(x) \wedge location(x, LosAngeles)$; terminological facts – $(\forall x)(man(x) \Rightarrow \neg\ woman(x))$; etc. However, FOL cannot deal with many types of quantifiers; non-existent entities; intensional readings; prototypical facts and defaults; probabilistic facts; vagueness; etc. Some extensions to FOL can deal with the issues above; yet, mapping naturally occurring sentences into formal representations remains a major challenge. The reader is strongly encouraged to try to write FOL formulas to express the meaning of the first few sentences in one of today's top New York Times stories – see text (10). It is the best exercise for understanding the limits of using FOL as a vehicle for natural language semantic interpretation. How many logical formulas do you think one needs in order to enable a complete "understanding", at the formal level, of text (10)?

Martian Robots, Taking Orders From a Manhattan Walk-Up

These days, when one of NASA's rovers drills a hole in a rock on Mars, the commands come from Lower Manhattan, from a second-floor office on Elizabeth Street, surrounded by dusted-off tenements. (10)

This is a street where pushcarts once fed and dressed Italians just off the boat. Now its old storefronts are of-the-moment restaurants and stores, and the only trace of the neighborhood's immigrant past is in its name - NoLIta, North of Little Italy. [New York Times; Nov 7, 2004]

The Semantic Interpretation Gap

In spite of significant progress in the natural language processing field, we are still far from being able to map arbitrary, natural occurring sentences into FOL. Not only it is unclear how we can represent in logic the subtleties present in natural language, but also, current semantic interpretation algorithms are too crude to handle naturally occurring sentences, such as those shown in (10).

3 Noisy-Channel Modeling—An Overview

In order to develop competitive noisy channel applications, one needs to

1. Have access or be able to collect significant amounts of training material in the form of (source, target) pairs.
2. Conceive of a stochastic generative process that explains how source data can be transformed/rewritten into target data.
3. Develop parameter estimation algorithms that are consistent with the generative process and use these algorithms in order to train model parameters from a representative set of examples.
4. Develop "decoding" algorithms that can recover the most likely source signal given a target signal/example.

This framework has been successfully applied to model problems as diverse as speech recognition (Jelinek, 1997), machine translation (Brown et al., 1993), summarization (Knight and Marcu, 2002), part of speech tagging (Church, 1988), and question answering (Echihabi and Marcu, 2003). In speech recognition, for example, starting from large collections of (natural language transcript; corresponding acoustic signal) tuples, one conceives stochastic processes that explain how natural language strings can be generated (the source model) and how they can then be mapped into acoustic signals (the channel model). Via training, one estimates then the parameters associated with both the source and channel models. When given as input an acoustic signal, a decoder is used to search for the string that maximizes a mixture of the source and channel model probabilities (See Figure 1 for a graphical depiction of this process).

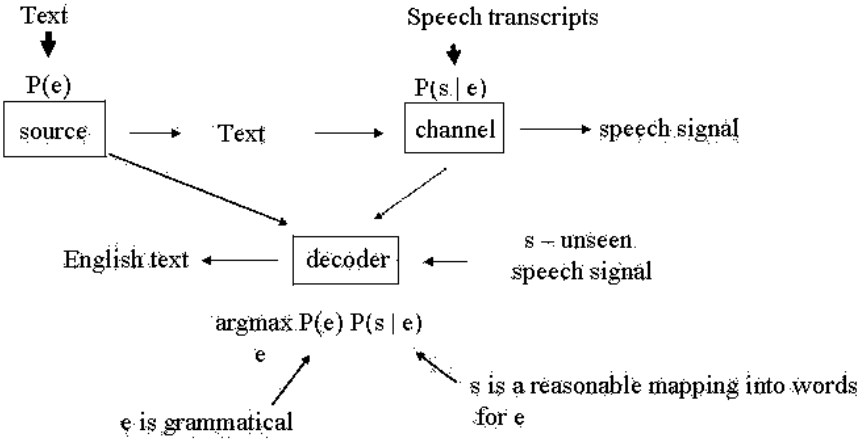


Fig. 1. Noisy-channel modeling in a speech recognition application

Most of the impressive progress in the field of speech recognition over the last 30 years is explained by significant advances in acoustic (noisy-channel) and source language modeling, improved parameter estimation and decoding algorithms, and continuous increase in the amounts of available training data. One of the defining characteristics of acoustic modeling (as well as the modeling employed in conjunction with tasks such as part of speech tagging and named entity recognition, for example) is that it is sequence-based: the modeling process explains how sequences can be stochastically rewritten into other sequences. However, noisy-channel applications should not be associated only with sequence-based transformations. Recently, noisy-channels that model mappings from bags of words to bags of words in information retrieval (Berger and Lafferty, 1999), strings to bags of words in headline generation (Banko et al., 2000) , tree to string mappings in machine translation (Yamada and Knight, 2001) and question answering (Echihabi and Marcu, 2003), and tree to tree mappings in summarization (Knight and Marcu, 2002) and machine translation (Gildea, 2004; Graehl and Knight, 2004) have started to emerge.

At the first sight, it is not clear at all what the connection between formal reasoning and noisy-channel modeling is; it seems that none of the requirements required to develop noisy-channel applications is met in the case of first-order reasoning:

1. For FOL problems, we don't have access to significant amounts of training material in the form of (premise, logical consequent) pairs.
2. It is not clear at all what the nature of the stochastic generative process that explains how FOL statements are mapped into other FOL statements is. In fact, it is not clear at all why we need a stochastic process. After all, Modus Ponens is deterministic!
3. Without a stochastic process to model, we cannot go about developing parameter estimation algorithms.
4. And similarly, without a stochastic process into place, there is no point to develop "decoding" algorithms that can map "source" FOL statements (premises) into "target" FOL statements (logical consequences).

4 Towards Reasoning with Natural Language Representations

Our long-term goal is to develop systems that can reason with natural language representations. For example, given a natural language sentence, such as "Alan Greenspan told reporters that he perceives several signs of economic strength", we would like to automatically mimic deductive reasoning and generate natural language sentences that depict possible effects, such as "Bond prices will fall." And given the sentence "Bond prices fell", we would like to mimic abductive reasoning and automatically suggest causes that led to this, such as "signs of economic strength". Clearly, such inferences do not have the well-formed semantics of FOL inferences; nevertheless, they are understandable and easy to interpret.

To reach our goal, we propose to abandon FOL working with first-order representations altogether. As in other noisy-channel approaches, we imagine that English Cause statements are generated by a stochastic process that assigns a probability P(C) to any string C. We also imagine that these Cause statements can be rewritten as Effect statements as they are corrupted by a noisy channel. We model the probability of generating an effect statement E from a cause statement C using the conditional P(E | C). If we are capable of obtaining massive amounts of <Cause; Effect> statements, then we can automatically train the parameters of both the source model P(C), and the channel model, P(E | C). Once these models are learned, we can provide our system with any unseen statement e and ask for its possible causes c. The system can generate these cause statements by looking for the strings that maximize the probability P(c) P(e | c). Figure 2 depicts graphically the main components of an abductive noisy-channel reasoning system. (A deductive noisy-channel reasoning system is a replica of the system in Figure 2, with Causes and Effects being swapped.)

In what follows, we discuss preliminary experiments aimed at addressing all four facets that pertain to developing noisy-channel-inspired reasoning systems that operate on naturally occurring strings.

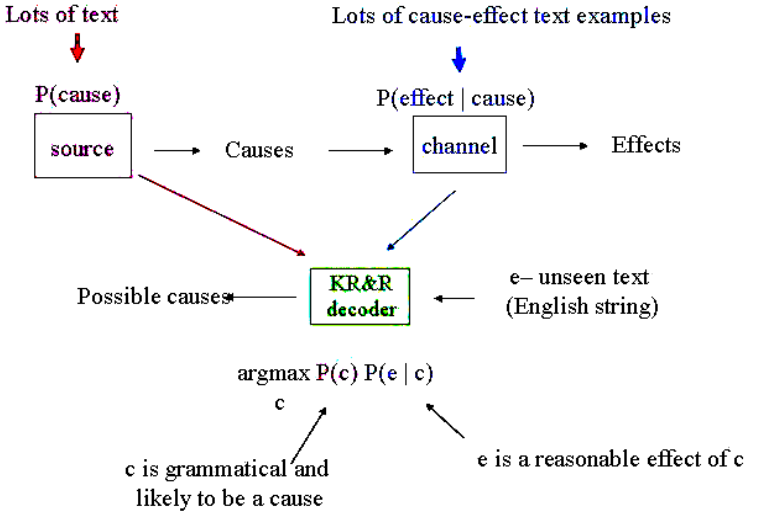


Fig. 2. The main components of a noisy-channel-based abductive reasoning system

1 Collect Significant Mounts of Training Material

One of our main assumptions is that much of the inferential knowledge that is employed by humans is explicitly marked linguistically in textual streams. For example, cue phrases such as "because", "due to", "if … then", and "this caused" often mark relations between cause and effect statements. Naturally, there are many other causal relations that are not marked explicitly in textual streams; and cue phrases do not always mark causal relations, i.e., they are noisy. However, by and large, we believe that discourse markers and other specific linguistic constructs provide a clean-enough source of inference examples. Because these markers are easy to identify and are used frequently in natural language texts, we expect that we can use them to collect large corpora of <Cause;Effect> statements.

To test this hypothesis, we mined for cause-effect patterns a corpus of 1 Billion words of English. We extracted from this corpus 730,648 examples that contain cause-effect statement pairs, such as those shown below in examples (11) and (12).

"I have been in management for more than 20 years, but I have been out of the work force for more than six months **because of** stress at work." (11)

"Saudi Arabia and other OPEC members will cut back production **because of** lower demand in Europe." (12)

Using automatic natural language processing tools that are publicly available or that we developed at ISI, we tagged all names/locations/organizations in these sentences (Bikel et al., 1999), syntactically parsed them (Lin, 1995), normalized the tenses and morphological variants, and applied a sentence compression algorithm in order to get rid of the words and syntactic constituents that are not directly relevant to the cause-effect statements we were interested to extract. Completely automatically, we thus extracted a corpus of 730,648 <Cause; Effect> statement pairs. For example, the sample sentences above yield the two <Cause; Effect> pairs shown in (13) and (14).

Effect: PERSON be out of work force for more than six months
Cause: stress at work (13)

Effect: ORGANIZATION member cut back production
Cause: low demand in LOCATION (14)

We believe that <Cause,Effect> pairs such as those shown in (13) and (14) provide an excellent resource for learning reasoning patterns that are employed frequently by humans.

2 Conceive of a Stochastic Generative Process That Explains How Cause Statements Can be Rewritten as Effect Statements

In order to model the rewriting process, as a proof of concept experiment, we use an existing noisy-channel model that we initially developed for statistical machine translation (Marcu and Wong, 2002). The model assumes that <Cause; Effect> statements are generated by the following stochastic process.

- Generate a set of concepts M.
- For each concept $m_i \in M$
  - Generate a cause-effect phrase pair, $<c_i, e_j>$ with probability $t(c_i,e_j)$
  - Order the phrases c_i, e_j according to a probability distribution d, which in machine translation usually captures reordering patterns specific to translating sentences from one language into another.

For example, the <Effect,Cause> in (14), may be generated by

- Generating two concepts m1 and m2;
- Generating from m1 the phrase pair (ORGANIZATION member; in LOCATION) and from m2 the phrase pair (cut back production; low demand)
- Swapping the order of the cause phrases so as to obtain the statements (ORGANIZATION member cut back production; low demand in LOCATION).

Clearly, this is a very weird model for how cause statements can be rewritten into effect statements through a sequence of stochastic steps. However, it is a model that is simple and that can automatically learn rewriting rules if they occur frequent enough in the training corpus.

3 Develop Parameter Estimation Algorithms That Are Consistent with the Generative Process and Use These Algorithms in Order to Train Model Parameters from a Representative Set of Examples

The main reason we chose the generative model above is because we have already implemented parameter estimation algorithms for it in our machine translation work (Marcu and Wong, 2002). When we train this model on the 730,648 <Cause; Effect> statement pairs we collected automatically, we learn many useful "translations". For example, we automatically learn that the most probable ways for people to die, i.e., the most probable ways to rewrite the effect string "PEOPLE die" into a cause string are those listed in Table 1 below. The most probable reasons for people dying as learned from natural occurring texts are listed below in decreasing order of probability, with the most probable cause being other PERSONs, storms, bombings, PEOPLE being lost, etc.

Table 1. Automatically mined causes of the statement "PEOPLE die"

– PERSON	– there be not	– superfund site
– Storm	– movie	– damage be extensive
– Bombing	– PERSON have	– hypertension over month
– PEOPLE be lost	– disaster	– old age illness other natural cause
– after work	– their product	– stiff regulation
– Coward	– inaction	– gross violation in planning
– drough-related illness	– PERSON live	– then there be not import restriction
– mine over past 14 years	– shoddy construction	– lack of health care insecurity
– crop failure shortage of potable water	– aggressor	– supervisor reject ordinance
– lack of soluble insulin hospital	– pesticide poisoning	– bad judgment
– chance be lot of those	– strike start	– poor water quality
– PEOPLE be stabbed	– there be vehicle PEOPLE bury under snow	– aggression atrocity hunger
– Heat	– recent outbreak of cholera in LOCATION	– approval be delayed
– Pollution	– hotel lack sprinkler smoke alarm	– PEOPLE trip over bags
– PERSON die	– Car exhaust particle	– PEOPLE lack necessary fund to pay
– continued doubt	– hotel lack important safety measure	– ruthless profit-hungry company
– there be vehicle	– many view thing	– emergency water
– PERSON be not there	– PEOPLE be exposed to air pollution	– excess alcohol consumption
– PERSON be admitted to hospital	– negligence at care facility	emergency
– withdrawal of advertising	– PEOPLE not live	
– payment be based on mortgage amount	– landslide	
– shortage of raw material	– restaurant food	
– PERSON become ill	– PEOPLE age	
– petty dispute over money	– PEOPLE not take off cap over mouthpiece	
– PEOPLE give up control chemistry metallurgy of their product	enough PEOPLE donate their organ	

As one can see, many of these learned causes are intuitively correct. Also, it is very likely that a human asked to come up with all these causes of death will fail to list many of them; yet, our system have discovered all these causes automatically.

4 Develop "Decoding" Algorithms That Can Recover the Most Likely Cause Given an Effect Statement

To complete our feasibility study, we also constructed a fully functional text-based reasoning system. Our reasoning system uses:

- A trigram language model to estimate P(c) and P(e).
- The channel model learned by the phrase-based learner described at the beginning of this section (Marcu and Wong, 2002).
- A statistical greedy decoder that was also developed in the context of our statistical machine translation work (Marcu and Wong, 2002).

We presented our decoder with test statements that *were not in the training corpus* and we asked it to translate those into causes and effect statements. When the decoder "translated" effect statements into cause statements it was doing abductive inferences using natural language strings. When it translated cause statements into effect statements the decoder was doing deductive inferences.

Below, in (15), we provide a few examples of abductive inferences that are automatically produced by our system (for convenience, we present here only the compressed statements used as input):

Input: *expectation of increased competition*
System output: *expiration of goods*

Input: *rate cut be discounted*
System output: *growth be slow*

Input: *rise in cash circulation*
System output: *paper price increase in LOCATION*

Input: *rising dollar make their car money cheap than domestic brand*
System output: *falling currency in car manufacturer consumer goods from LOCATION* (15)

Input: *PERSON agree to bad deal*
System output: *PERSON not want ORGANIZATION market*

Input: *investor sell stock*
System output: *inflation reduce tax*

Input: *PERSON not get president job*
System output: *PERSON not get PERSON in LOCATION*

Input: *PERCENT jump in rate*
System output: *increased competitiveness exchange*

The outputs that we produce clearly contain some useful nuggets. However, more research is needed to produce more grammatical outputs and to enable a higher level of generalization.

The noisy-channel approach performs well when there is significant redundancy in the training data. However, while there are frequently encountered causes and effects ("people die", "people resign", etc.), a large number of causes or effects occur only a

few times ("people are healthy", "people work the land", etc.). More research is needed to assess how related statements can be clustered in order to increase redundancy (for instance, "people are healthy" could be grouped with "good health", "person stays healthy", etc.). We could use lexical information (obtained from WordNet or learned from data) to identify such related statements and mitigate the sparse data problem. Another way to increase redundancy of cause-effect pairs is to identify and eliminate extraneous modifiers (for instance "LOCATION drought in DATE → people die" and "LOCATION drought → people die" are identical except for "in DATE").

Another major challenge in this area pertains to the interpretation of pronouns. Causal sentences have a high incidence of pronouns, as the same entities tend to be mentioned both in the cause and effect statements (e.g. "The mailman came back because *he* forgot the number"). It is likely that a simple pronoun resolution algorithm is likely to yield cleaner training sets; however, it is unclear though while the current state of the art pronoun resolution algorithms are sophisticated enough for the needs of our application.

5 Discussion

Traditionally, it was assumed that, starting from texts, one can perform useful inferences only to the extent to which natural language sentences are first mapped into formal representations that have well-defined syntax and semantics and deterministic inference algorithms. In this paper, we suggest that we may be capable to do intuitive deductive and abductive inferences by modeling the inference process within a noisy channel, probabilistic framework. More precisely, we show that we can map naturally occurring language sentences into grammatical, well-formed strings that depict likely causes or effects of the sentences given as input by exploiting noisy-channel models and decoders that were developed for statistical machine translation applications. We discuss methods for automatically acquiring arbitrary large training corpora and we show examples of an end-to-end, string-based reasoning system that performs abductions without using any formal language as intermediate representation.

The results we have obtained so far look promising, but are far from telling a complete story. We hypothesize that models and decoding algorithms that are developed specifically for reasoning will stand a better chance of making an impact on large-scale, generic reasoning applications that work with naturally occurring texts. We also expect that text-based reasoning algorithms will have a significant impact in future natural language processing applications.

References

1. Banko Michele, Vibhu Mittal and Michael Witbrock, 2000. *Headline Generation Based on Statistical Translation*. Proceedings of the Annual Meeting of the Association for Computational Linguistics.
2. Berger Adam and John Lafferty, 1999. *Information retrieval as statistical translation*. Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 222-229, Berkeley, California.

3. Bikel Dan and Richard Schwartz and Ralph Weischedel, 1999. An Algorithm that Learns What's in a Name. *Machine Learning*, 34 (1-3), pages 211-231.
4. Brown Peter F., Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics, 19(2):263--312.*
5. Church Kenneth W. 1988. *A stochastic parts program and noun phrase parser for unrestricted text*. In Proceedings of the Second Conference on Applied Natural Language Processing, Austin, TX.
6. Echihabi Abdessamad and Daniel Marcu, 2003. *A Noisy-Channel Approach to Question Answering*, Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics, July 7-12, Sapporo, Japan.
7. Gildea Dan, 2003. *Loosely Tree-Based Alignment for Machine Translation*, Proceedings of the 41th Annual Conference of the Association for Computational Linguistics, Sapporo, Japan, 2003.
8. Graehl Jonathan and Kevin Knight, 2004. *Training Tree Transducers*. Proceedings of the Human Language Technology and the North American Meeting of the Association for Computational Linguistics, Boston, MA.
9. Knight Kevin and Daniel Marcu. 2002. Summarization beyond sentence extraction: A probabilistic approach to sentence compression. *Artificial Intelligence 139(1): 91-107.*
10. Lin Dekang. 1995. MINIPAR. A Dependency-Based Parser. Available for download from http://www.cs.ualberta.ca/~lindek/minipar.htm.
11. Marcu Daniel and William Wong, *A Phrase-Based, Joint Probability Model for Statistical Machine Translation*, Proceedings of the Conference on Empirical Methods in Natural Language Processing, Philadelphia, PA, 2002.
12. Russell Stuart and Peter Norvig, 1995. *Artificial Intelligence. A Modern Approach*. Prentice Hall, New Jersey.
13. Yamada Kenji and Kevin Knight, *A Syntax-Based Statistical Translation Model*, Proceedings of the Conference of the Association for Computational Linguistics, 2001.

Putting Pieces Together: Combining FrameNet, VerbNet and WordNet for Robust Semantic Parsing

Lei Shi and Rada Mihalcea

University of North Texas,
Computer Science Department,
Denton, TX, 76203-1366
leishi@unt.edu,rada@cs.unt.edu

Abstract. This paper describes our work in integrating three different lexical resources: FrameNet, VerbNet, and WordNet, into a unified, richer knowledge-base, to the end of enabling more robust semantic parsing. The construction of each of these lexical resources has required many years of laborious human effort, and they all have their strengths and shortcomings. By linking them together, we build an improved resource in which (1) the coverage of FrameNet is extended, (2) the VerbNet lexicon is augmented with frame semantics, and (3) selectional restrictions are implemented using WordNet semantic classes. The synergistic exploitation of various lexical resources is crucial for many complex language processing applications, and we prove it once again effective in building a robust semantic parser.

1 Introduction

The goal of a semantic parser is to identify semantic relations between words in a text, resulting in structures that reflect various levels of semantic interpretation. Such structures can be used to improve the quality of natural language processing applications by taking into account the meaning of the text. Automatic techniques for semantic parsing have been successfully used in Information Extraction and Question Answering, and are currently evaluated in other applications such as Machine Translation and Text Summarization.

The process of semantic parsing typically implies a learning stage, where the semantic structures to be identified are acquired from an existing lexical resource, which explicitly identifies the range of possible semantic relations between words in a text. While there are several lexical resources suitable for semantic parsing, built with extensive human effort over years of work – including FrameNet [3], VerbNet [5], WordNet [7], or PropBank [4] – all previous approaches to semantic parsing have relied exclusively on only one of them, as there are no connections between these resources that would enable their exploitation in an unified way. However, each resource encodes a different kind of knowledge and has its own advantages, and thus combining them together can eventually result in a

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 100–111, 2005.

richer knowledge-base that can enable more accurate and more robust semantic parsing.

In this paper, we describe our work in integrating into a unified knowledge-base three different lexical resources: FrameNet [3] – a corpus-based frame resource; VerbNet [5] – a verb lexicon with explicit syntactic and semantic information based on Levin's verb classes; and WordNet [7] – a semantic network encoding a large number of concepts and semantic relations. We describe the procedures used to map the various semantic constituents identified by these lexical resources (e.g. frames, semantic roles, semantic classes), and evaluate these mappings against manually annotated data. We also shortly describe a robust rule-based semantic parser that relies on this unified knowledge-base to identify the semantic structure of any open text.

2 Knowledge Bases for Semantic Parsing

One major problem faced by semantic parsers is the fact that similar syntactic patterns may introduce different semantic interpretations, and similar meanings can be syntactically realized in many different ways. To deal with the large number of cases where the same syntactic relation introduces different semantic relations, we need knowledge about how to map syntax to semantics. To this end, we use two main types of knowledge – about words, and about relations between words. The first type of knowledge is drawn from WordNet – a large lexical database with rich information about words and concepts. We refer to this as *word-level knowledge*. The latter is derived from FrameNet – a resource that contains information about different situations, called *frames*, and from VerbNet – a verb lexicon based on Levin's verb classes which also provides selectional restrictions attached to semantic roles. We call this *sentence-level knowledge*.

In this section, we briefly describe each lexical resource, and show the type of information that the semantic parser extracts from these knowledge bases.

2.1 FrameNet

FrameNet [3] provides the knowledge needed to identify case frames and semantic roles. It is based on the theory of frame semantics, and defines a sentence level ontology. In frame semantics, a *frame* corresponds to a scenario that involves an *interaction* and its *participants*, in which participants play some kind of roles. A frame has a name, and we use this name to identify the semantic relation that groups together the semantic roles. In FrameNet, nouns, verbs and adjectives can be used to identify frames.

Each annotated sentence in FrameNet exemplifies a possible syntactic realization for the semantic roles associated with a frame for a given target word. By extracting the syntactic features and corresponding semantic roles from all annotated sentences in the FrameNet corpus, we are able to automatically build a large set of rules that encode the possible syntactic realizations of semantic frames. In our semantic parser, we use only verbs as target words for frame

identification. Currently, FrameNet defines about 3040 verbs attached to 320 different frames.

2.2 VerbNet

VerbNet is a verb lexicon compatible with WordNet, with explicitly stated syntactic and semantic information based on Levin's verb classification [6]. The fundamental assumption is that the syntactic frames of a verb as argument-taking elements are a direct reflection of the underlying semantics. VerbNet associates the semantics of a verb with its syntactic frames, and combines traditional lexical semantic information such as thematic roles and semantic predicates, with syntactic frames and selectional restrictions. It explicitly implements the close relation between syntax and semantics hypothesized by Levin.

Verb entries in the same VerbNet class share common syntactic frames, and thus they are believed to have the same syntactic behavior – an important property that can be used to extend the coverage of FrameNet. Shortly, by identifying the VerbNet verb class that corresponds to a FrameNet frame, we are able to parse sentences that include verbs not covered by FrameNet. This is done by exploiting a transitivity relation via VerbNet classes: verbs that belong to the same Levin class are likely to share the same FrameNet frame, and thus their frame semantics can be analyzed even if not explicitly defined in FrameNet.

2.3 WordNet

WordNet [7] is the resource used to identify shallow semantic features that can be attached to lexical units. WordNet covers the vast majority of nouns, verbs, adjectives and adverbs from the English language. The words in WordNet are organized in synonym sets, called *synsets*. Each synset represents a concept. WordNet 2.0 has a large network of 152,046 words, organized in 115,420 synsets. WordNet also includes an impressive number of semantic relations defined across concepts (more than 250,000 relations in WordNet 2.0), including hypernymy/hyponymy (ISA), meronymy/holonymy (HASA), antonymy, entailment, etc.

The information encoded in WordNet is used in several stages in the parsing process. For instance, attribute relations, adjective/adverb classifications, and others are semantic features extracted from WordNet and stored together with the words, so that they can be directly used by the semantic parser. The argument constraints encoded in VerbNet (e.g. *+animate*, *+concrete*) are mapped to WordNet semantic classes, to the end of providing selectional restrictions useful for improving the frame selection and role labeling process in the semantic parser. Moreover, the mapping between WordNet verb entries and FrameNet lexical units allows for an extension of the parser coverage, by assigning common frames to verbs that are related in meaning according to the WordNet semantic hierarchies.

3 Combining Resources

All these resources – FrameNet, VerbNet and WordNet – have their strengths and shortcomings. In this work, we aim to combine their strengths, and eliminate their shortcomings, by creating a unified knowledge-base that links them all together, allowing them to benefit from one another.

FrameNet provides a good generalization across predicates using frames and semantic roles. It also includes a fairly large corpus annotated for semantic structures, which provides empirical evidence for the syntactic realization of semantic frames. This corpus can be efficiently used to learn how to identify semantic relations starting with syntactic structures. However, FrameNet does not explicitly define selectional restrictions for semantic roles. Moreover, the construction of FrameNet required significant human effort, and thus its coverage and scalability are seriously limited.

VerbNet instead has better coverage, and defines syntactic-semantic relations in a more explicit way. VerbNet labels thematic roles, and provides selectional restrictions for the arguments of syntactic frames. On the down side, although the verb classes in VerbNet represent powerful generalizations for the syntactic behavior of verbs, most of the times the traditional abstract thematic roles are too generic to capture a scenario similar to that represented by a semantic frame.

Finally, perhaps one of the most useful properties of **WordNet** is its almost complete coverage of English verbs, and the rich information it encodes about semantic relations between verb senses (e.g. *ISA* relations, *entailment*, *antonymy*). However, the construction of the WordNet verb lexicon is primarily based on verb meanings, and does not encode syntactic or semantic verb behavior, such as predicate-argument structures (with the exception of a very small set of about 15 typical verb structures).

With the work reported in this paper, we aim to construct a unified framework that will exploit in a synergistic way the advantages of these three different resources, and will result in a richer resource suitable for robust semantic parsing. We augment the frame semantics with VerbNet verb classes by labeling FrameNet frames and semantic roles with VerbNet verb entries and corresponding arguments. We also extend the coverage of FrameNet verbs by exploiting both VerbNet verb classes and WordNet verb synonym and hyponym relations. Moreover, we identify explicit connections between semantic roles and semantic classes, by encoding selectional restrictions for semantic roles using the WordNet noun hierarchies.

Because of the lack of connectivity between these lexical resources, previous approaches to semantic parsing have relied exclusively on only one resource. For instance, Gildea and Jurafsky [2] proposed a statistical approach based on FrameNet data for annotation of semantic roles – which is however inherently limited to those verbs covered by FrameNet. Recent work on VerbNet semantic role labeling [10] led to an unsupervised system able to identify general thematic roles with respect to VerbNet, system that unfortunately cannot be extended to the more specific FrameNet roles, since the two resources are not connected.

Finally, there is no previous work that we are aware of that combines the analysis of semantic roles and semantic classes into one single system, as there is no resource available today that would encode explicit connections between these semantic entities.

In the following sections, we illustrate the algorithms we designed to connect these three lexical resources, and briefly describe a rule-based semantic parser that relies on this unified resource for robust semantic parsing.

4 Connecting VerbNet to WordNet: Defining Selectional Restrictions

Selectional restrictions – as part of the VerbNet-defined semantic knowledge – are used for both semantic role identification and syntax-semantics translation. Consider for example the sentence *I break the window* versus the sentence *The hammer breaks the window*. Although the participants in the interaction *break* have identical syntactic features in both sentences, they play however different semantic roles: *I* should be identified as playing an *agent* role, while *hammer* should play the role of an *instrument*. While this distinction cannot be made based on a difference in syntactic interpretation (since both sentences have identical parse trees), a correct semantic parse can still be achieved by realizing that the two participants *I* and *hammer* belong to different ontological categories (*I* refers to a person, and *hammer* refers to a tool), and thus they are likely to play different semantic roles.

Selectional restrictions are explicitly defined in VerbNet as constraints imposed on the arguments of syntactic frames. They are specified using generic terms such as *person*, *concrete* etc., which are attached to the constituents of a syntactic frame, to the end of providing ontological information about what can play the role of an argument. In order to generalize these selectional restrictions, and benefit from the semantic classes defined in WordNet, we map the semantic constraints identified by VerbNet to ontological categories in WordNet, which are defined as collections of entities subsumed by a given semantic concept. We say that an entity E belongs to an ontological category C if the noun E is a child node of C in the WordNet semantic hierarchy of nouns. For example, if we define the ontological category for the role *instrument* (VerbNet) as *instrumentality* (WordNet), then all hyponyms of *instrumentality* can play the role of *instrument*, while other nouns such as e.g. *boy*, which are not part of the *instrumentality* category, will be rejected.

Selectional restrictions are defined using a Disjunctive Normal Form (DNF) in the following format:

`[Onto(ID,P),Onto(ID,P),...],[Onto(ID,P),...],...`

where `Onto` is a noun and *ID* is its corresponding WordNet sense, which uniquely identifies `Onto` as a node in the semantic network. `P` can be set to p (positive) or n (negative), indicating whether a noun should belong to a given category or not. For example, the restriction:

```
[person(1,n),object(1,p)],[substance(1,p)]
```

indicates that the noun obeying this selectional restriction should belong to `object(sense #1)` in WordNet, but not `person(sense #1)`[1], or it should belong to `substance(sense #1)`.

The process of mapping VerbNet to WordNet is thus semi-automatic. We first manually link all semantic constraints defined in VerbNet (there are 36 such constraints) to one or more nodes in the WordNet semantic hierarchy. These mappings can be one-to-one, as in e.g. `+solid` → `solid(1,p)`, or one-to-many, as in e.g. `+time` → `[time_period(1,p),time_unit(1,p)]`. Next, any concepts subsumed by the ontological categories rooted by the nodes identified in WordNet are automatically mapped to the same VerbNet semantic constraints (note that this is an implicit mapping performed during the process of identifying selectional restrictions).

5 Connecting FrameNet to VerbNet

This second mapping problem can be divided into two main sub-tasks: (1) Mapping VerbNet verb entries to appropriate semantic frames in FrameNet; and (2) Linking arguments of VerbNet syntactic frames with corresponding FrameNet semantic roles. We have designed algorithms to automatically handle these tasks, and we evaluate their accuracy against manually annotated data.

5.1 Labeling VerbNet Verb Entries with Semantic Frames

In order to link a VerbNet verb entry to a FrameNet semantic frame, we need to identify corresponding verb meanings in these two lexical resources. [2]. VerbNet verb entries are already linked to WordNet, with a list of sense IDs being attached to each verb. In order to identify the corresponding FrameNet semantic frame for each such entry, we apply a similar annotation process to the FrameNet verb lexical units, and assign each of them with WordNet senses. To ensure maximum reliability, this annotation process was performed manually. Since WordNet sense distinctions are very fine-grained, a verb entry in FrameNet or VerbNet may be linked to multiple WordNet senses, with a set of senses being regarded as a coarse sense.

The first step of the mapping algorithm consists of dividing all VerbNet verb entries into two sets, depending on whether they have a direct match in FrameNet. The division of verbs in these two sets is performed by (1) identifying

[1] This exclusion has to be explicitly indicated, since `person(sense #1)` is a child node of `object(sense #1)` in WordNet.

[2] Note however that it is not always possible to identify a FrameNet frame for a VerbNet verb entry, since VerbNet and FrameNet cover different subsets of the English verbs. Currently, VerbNet defines 4159 verb entries, while FrameNet covers 3040 verb lexical units, with 2398 verb entries being defined in both resources.

identical word forms among VerbNet verb entries and FrameNet lexical units, and (2) ensuring that these verbs share a common meaning, by checking the intersection of their corresponding sense lists.

For the first verb set, consisting of VerbNet entries that have a direct counterpart in FrameNet, we label them with the corresponding frames by relying on their common WordNet senses, as described before. Note that multiple frames can be assigned to a single VerbNet verb entry. For example, the VerbNet sense numbers 1, 2, and 3 of *admonish* belong to the verb class *advise-37.9-1*, while in FrameNet sense numbers 1 and 2 of *admonish* are defined under the frame *Attempt_suasion* and sense number 3 under *Judgment_direct_address*. Hence the VerbNet entry *admonish(1,2,3)* will be labeled with both frames *Attempt_suasion* and *Judgment_direct_address*.

Algorithm 1: Map FrameNet frames to VerbNet verbs

```
for each LU in VerbNet
   if LU defined in FrameNet under the frame F
       frame(LU) = F;
   else
     if(synonym(LU) or hypernym(LU) defined in FrameNet)
         frame(LU) = frame(synonym(LU)) or frame(hypernym(LU));
     else
       if(some other verb in the same class defined in FrameNet)
           arrary fo[] = frames from verbs in the same class;
           fmax =  most frequently used frame in fo[];
           frame(LU) = fmax;
        else
           label with "no frame defined";
```

Fig. 1. Algorithm for mapping VerbNet verbs to FrameNet frames. LU = Lexical Unit. F = frame

For the second set, where no corresponding lexical unit can be identified in FrameNet, we have developed two algorithms that allow us to infer the correct frame. The first algorithm uses WordNet verb synonym and hyponym relations to identify frames for those verb entries for which FrameNet defines a verb with a similar meaning. The frame that a verb belongs to depends primarily on the meaning of the verb, and hence synonym and hyponym relations can be good indicators of their correct semantic frame. According to Miller [7], synonyms represent lexicalization for a common underlying concept, and therefore verbs in the same synset are likely to belong to the same semantic frame. Similarly, verbs connected by a hyponymy relation[3] share semantic components with their superordinate, and usually belong to a sub-frame or the same frame as the superordinate. For a verb entry of this category, the algorithm looks for its

[3] The hyponymy relation can be interpreted as *to V1 is to V2 in some manner*

synonym or hypernym in FrameNet and uses the frame for that verb entry. A total of 839 VerbNet verb entries were labeled with a FrameNet frame using WordNet synonym and hyponym relations.

The remaining verb entries for which we cannot find a direct correspondence in FrameNet, nor we can find a WordNet synonym and hyponym that would enable an indirect semantic correspondence, we identify the corresponding frame using a majority voting algorithm. Based on an empirical study reported by Baker and Ruppenhofer [1] that shows that Levin's verb classes often corresponds to FrameNet frames, we assume that verbs in the same class are likely to be in the same semantic frame. The majority voting algorithm can therefore infer the most likely frame by looking at other verbs in the same class. If other verbs have been already assigned with a frame using one of the previous algorithms, we can choose the most frequently used frame in the entire verb class as the candidate for this verb entry.

The pseudo-code of the algorithm used to assign FrameNet semantic frames to VerbNet verb entries is illustrated in Figure 1.

5.2 Labeling VerbNet Syntactic Frame Arguments with Semantic Roles

Once each VerbNet verb entry is mapped to a semantic frame, we can also identify a mapping between FrameNet semantic roles and VerbNet arguments. VerbNet verb classes are constructed based on syntactic frames, with verbs in the same class sharing a similar syntactic behavior. Thematic roles are provided for each argument in a syntactic frame, together with selectional restrictions. The major difference between the VerbNet thematic roles and the FrameNet semantic roles is that the thematic roles are generic and global with respect to language, while the semantic roles are local and specific only to their frame. The task we have is basically to identify for each VerbNet generic thematic role a FrameNet specific semantic role.

To this end, we use features that describe the syntactic realizations of these two types of roles as the basis for the mapping process. First, we extract such features from the syntactic description of the VerbNet arguments and the FrameNet corpus annotations. Next, we try to automatically find a correspondences between a VerbNet argument and a FrameNet semantic role by comparing such syntactic features against each other. The features we currently use include *Grammatical Function (GF)* (e.g. *subject*, *object*), *Phrase Type (PT)* (e.g. noun phrase *NP*, prepositional phrase *PP*), *Voice* (*active* or *passive*), *Head Word* of NP and PPs, and *Selectional Restriction (SR)* (defined for each argument in VerbNet). A correspondence is identified between an argument and a semantic role if their *GF*, *PT*, and *Voice* features are equivalent. If the phrase is a prepositional phrase, their prepositions should also agree. Moreover, the head word of the semantic role needs to meet the selectional restrictions of the argument.

For those verb entries that are defined in VerbNet, but not in FrameNet, we seek to identify the closest matching frame using the algorithm described in the previous section, and then use this indirect correspondence to label FrameNet

Algorithm 2: Map FrameNet semantic roles to VerbNet verb arguments

```
for each LU in VerbNet
    if (LU defined in frame F1)
        map_roles(LU,F1);
        if (some other verb(s) in the class has the same frame)
            for each argument of LU
               ra[] = all roles of the frame labeling the argument;
               role = most frequently used role in ra[];
        else
           if LU has a frame
              if some verb V in the same class has the same frame
                  use mapping of V for this verb;
              else
                  randomly choose a verb in the frame F2;
                  map_roles(LU,F2);
           else
              use original thematic roles;
```

map_role(LU,F)
Label arguments in LU with FrameNet semantic roles for frame F

```
for each argument in LU
    for each role in FrameNet
         if GF,PT,Voice agree and head word meets Selectional Restriction
             this role is used to label the argument;
         else
            if no appropriate role found
                use original VerbNet thematic role
```

Fig. 2. Algorithm for mapping VerbNet roles to FrameNet roles. LU = Lexical Unit. F = frame

roles to VerbNet arguments. Finally, for the verbs that do not belong to any frame available in FrameNet, their original thematic roles are used.

The pseudo-code of the algorithm for mapping between FrameNet and VerbNet semantic roles is shown in Figure 2.

5.3 Evaluation

The first mapping algorithm maps FrameNet frames to VerbNet verb entries, and returns a list of structures *(Verb, SenseID, Verb_Class, Frame)*, where *Verb* represents the word form of the verb, *SenseID* is the WordNet sense number denoting a particular meaning of the verb, *Verb_Class* is the VerbNet class it belongs to, and *Frame* is the FrameNet semantic frame. To evaluate this first algorithm, we checked the automatically produced mappings against a manually constructed list of such structures produced for a random subset of 400 verbs,

and found that 81.25% of the VerbNet entries were assigned with a correct FrameNet frame.

The second mapping algorithm identifies explicit connections between VerbNet arguments and FrameNet semantic roles, and returns a list of structures *(Verb, SenseID, Role_with_Syntax_List)*. The *Role_with_Syntax_List* field defines how arguments in the predicate-argument structure should be labeled with semantic roles based on their syntactic features. Basically, for each argument in the VerbNet predicate-argument structure, we identify a list of syntactic features (*GF*, *PT*, *Voice*, as described before), together with the corresponding selectional restriction and the FrameNet semantic role. We manually checked a randomly selected subset of 400 such mapped arguments, and found that 78.22% were labeled correctly with their corresponding FrameNet semantic role.

6 Semantic Parsing for Open Text

In this section, we briefly describe a rule-based semantic parser that relies on the unified knowledge-base that integrates FrameNet, VerbNet, and WordNet. A more detailed description of the parser can be found in [8, 9].

Similar in spirit with the syntactic parser – whose goal is to parse a valid natural language sentence into a parse tree indicating how the sentence can be syntactically decomposed into smaller syntactic constituents – the purpose of the semantic parser is to analyze the structure of sentence meaning. Sentence meaning is composed by *entities* and *interactions* between entities, where entities are assigned semantic *roles*, and entities and the interaction can be further modified by other *modifiers*. The meaning of a sentence is decomposed into smaller semantic units connected by various semantic relations by the principle of compositionality, and the parser represents the semantic structure – including semantic units as well as semantic relations, connecting them into a formal format.

For instance, the meaning of the sentence *I love to eat Mexican food, because it is spicy* can be decomposed into *I* and *to eat Mexican food* as participants in the interaction denoted by *love*; *because it is spicy* modifies the interaction *love* indicating *reason* of why this interaction takes place, and *spicy* modifies *it* indicating an attribute of *taste_property* for the entity *Mexican food*. Finally, *eat Mexican food* can be further decomposed as *eat* representing the interaction between *Mexican food* and a hidden participant *I*, with *Mexican* being a restrictive modifier for *food*. The semantic structure is recursive, and the complete semantic tree built by the semantic parser for this sample sentence is shown in Figure 3.

Our semantic parser – which we call SPOT (Semantic Parsing for Open Text) – is a rule-based parser that integrates both word and sentence level information, extracted from WordNet, VerbNet, and FrameNet. The semantic parsing process consists of four main steps: (1) A syntactic-semantic analyzer analyzes the syntactic structure, and uses lexical semantic knowledge to identify some semantic relations between constituents. It also prepares syntactic features for semantic role assignment in the next step. (2) The corresponding FrameNet

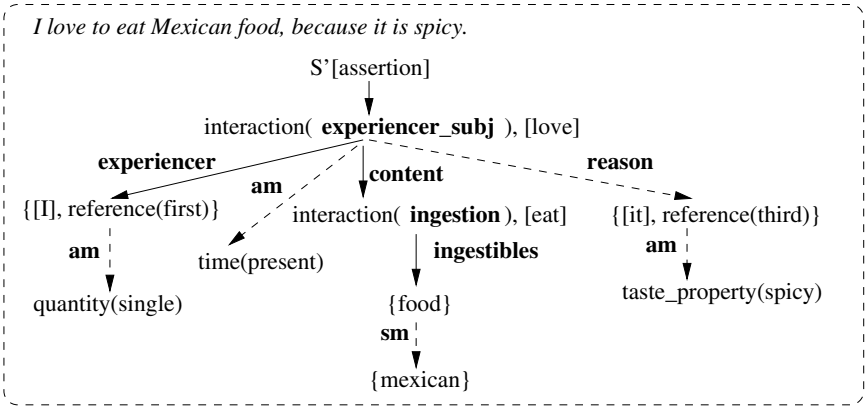


Fig. 3. Semantic parse tree for a sample sentence (am = attributive modifier, rm = referential modifier, sm = restrictive modifier)

frame is identified, either as a direct match, or indirectly via VerbNet and/or WordNet relations, using the mapping algorithms illustrated in the previous section. Selectional restrictions are also identified and added as features. (3) The role assigner labels semantic roles for identified participants, based on their syntactic features and selectional restrictions, as produced in the first two steps. (4) For those constituents not exemplified in FrameNet, we apply default rules to decide their default meaning.

Details on each of these steps, illustrative examples, and evaluation results are provided in [8].

By relying on knowledge extracted from several lexical resources, our semantic parser has several advantages as compared to previously developed parsers e.g. [2], [10]. First, it has significantly larger coverage, being able to parse any English sentence whose target verb is defined in either FrameNet, VerbNet, or WordNet. Second, the parser is able to identify additional semantic properties, such as attribute relations, adjective/adverb classifications, etc., which are explicitly encoded in WordNet. Finally, in addition to labeling the arguments of an interaction with semantic roles, the parser also identifies selectional restrictions linked to WordNet semantic classes, which makes the labeling process more general, and opens the doors to new text analysis applications such as the use of semantic parsing for word sense disambiguation, semantic information extraction, learning of semantic classes, and others.

7 Conclusions

Building accurate lexical resources requires extensive human effort. Each resource is usually intended to solve a particular type of problems, and may have strengths in some aspects and shortcomings in others. In this paper, we described our work in combining three lexical resources – FrameNet, VerbNet, and

WordNet – into a unified, richer knowledge-base[4]. By linking all these resources together, we built an improved resource in which (1) the coverage of FrameNet is extended, (2) the VerbNet lexicon is augmented with frame semantics, and (3) selectional restrictions are implemented using WordNet semantic classes. The synergistic exploitation of these lexical resources was found effective in building a robust semantic parser.

Acknowledgments

We thank Srinivasan Vaidyaraman for his help with the annotation and validation process. This work was partially supported by a National Science Foundation grant IIS-0336793.

References

1. Baker, C., and Ruppenhofer, J. Framenet's frames versus Levin's verb classes. In *Proceedings of the 28th Annual Meeting of the Berkeley Linguistics Society* (2002).
2. Gildea, D., and Jurafsky, D. Automatic labeling of semantic roles. In *Proceedings of the 38th Annual Conference of the Association for Computational Linguistics (ACL 2000)* (Hong Kong, October 2000), pp. 512–520.
3. Johnson, C., Fillmore, C., Petruck, M., Baker, C., Ellsworth, M., Ruppenhofer, J., and Wood, E. *FrameNet: Theory and Practice.* 2002. http://www.icsi.berkeley.edu/ framenet.
4. Kingsbury, P., Palmer, M., and Marcus, M. Adding semantic annotation to the Penn TreeBank. In *Proceedings of the Human Language Technology Conference HLT-2002* (San Diego, California, 2002).
5. Kipper, K., H.T.Dang, and Palmer, M. Class-based construction of a verb lexicon. In *Proceedings of Seventeenth National Conference on Artificial Intelligence AAAI 2000* (Austin,TX, July 2000).
6. Levin, B. *English Verb Classes and Alternation: A Preliminary Investigation.* The University of Chicago Press, 1993.
7. Miller, G. Wordnet: A lexical database. *Communication of the ACM 38*, 11 (1995), 39–41.
8. Shi, L., and Mihalcea, R. An algorithm for open text semantic parsing. In *Proceedings of the ROMAND 2004 workshop on "Robust Methods in Analysis of Natural language Data"* (Geneva, Switzerland, August 2004).
9. Shi, L., and Mihalcea, R. Semantic parsing using FrameNet and WordNet. In *Proceedings of the Human Language Technology Conference (HLT/NAACL 2004)* (Boston, May 2004).
10. Swier, R., and Stevenson, S. Unsupervised semantic role labelling. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP 2004)* (Barcelona, Spain, 2004).

[4] Data available from `http://lit.csci.unt.edu/~spot`

Assigning Function Tags with a Simple Model

Vasile Rus and Kirtan Desai

Department of Computer Science, The University of Memphis,
Institute for Intelligent Systems, Fedex Institute of Technology,
Memphis, TN 38120, USA
vrus@memphis.edu

Abstract. This paper presents a method to assign function tags based on a Naive Bayes approach. The method takes as input a parse tree and labels certain constituents with a set of functional marks such as logical subject, predicate, etc. The performance reported is promising, given the simplicity of a Naive Bayes approach, when compared with similar work.

1 Introduction

Syntactic structure is an important phase in a variety of language tasks since it provides important information for semantic interpretation. State-of-the-art statistical parsers, the tools that generate syntactic structures, are freely available nowadays but their output is limited to basic structures and are not able to deliver richer syntactic information such as logical subject or predicate. Most of the available statistical parsers are trained on Penn Treebank [7] and are only able to identify simple phrases such as NP, VP or S although Penn Treebank contains function tags and remote dependencies coded as traces. This paper presents a naive Bayes approach to augment the output of Treebank-style syntactic parsers with functional information.

In Section 2.2 of Bracketing Guidelines for Treebank II [7], there are 20 function tags grouped in four categories: form/function discrepancies, grammatical role, adverbials, and miscellaneous. Up to 4 function tags can be added to the standard syntactic label (NP, ADVP, PP, etc.) of each bracket. Those tags were necessary to distinguish words or phrases that belong to one syntactic category and is used for some other function or when it plays a role that is not easily identified without special annotation. We rearrange the four categories into four new categories based on corpus evidence, in a way similar to [1]. The new four categories are given in Table 1 and were derived so that no two labels from same new category can be attached to the same bracket.

We present in this paper a naive Bayes approach to build a system that automatically assigns function tags to constituents in parse trees. The function tags assignment problem is viewed as a classification problem, where the task is to select the correct tag from a list of candidate tags. The results are reported per category based on the new categories mentioned above.

Simple Bayesian classifiers have been gaining popularity lately, and have been found to perform surprisingly well [3]. These probabilistic approaches make

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 112–115, 2005.

Table 1. Categories of Function Tags

Category	Function Tags
Grammatical	DTV, LGS, PRD, PUT, SBJ, VOC
Form/Function	NOM, ADV, BNF, DIR, EXT, LOC, MNR, PRP, TMP
Topicalisation	TPC
Miscellaneous	CLR, CLF, HLN, TTL

strong assumptions about how the data is generated, and posit a probabilistic model that embodies these assumptions; then they use a collection of labeled training examples to estimate the parameters of the generative model. Classification of new examples is performed with Bayes' rule by selecting the class that is most likely to have generated the example. The naive Bayes classifier is the simplest of these models, in that it assumes that all attributes of the examples are independent of each other given the context of the class. This is the so-called "naive Bayes assumption". While this assumption is clearly false in most real-world tasks, naive Bayes often performs classification very well. This paradox is explained by the fact that classification estimation is only a function of the sign (in binary cases) of the function estimation; the function approximation can still be poor while classification accuracy remains high [3]. Because of the independence assumption, the parameters for each attribute can be learned separately, and this greatly simplifies learning, especially when the number of attributes is large [6].

2 Related Work

There has been no previous work, to our knowledge, so far that attempted to build a system that assigns function tags using a naive Bayes approach.

There was only one project detailed in [1] to address the task of function tagging. They use a statistical algorithm based on a set of features grouped in *trees*, rather than *chains*. The advantage is that features can better contribute to overall performance for cases when several features are sparse. When such features are conditioned in a chain model the sparseness of a feature can have a dilution effect of a ulterior (conditioned) one.

Previous to that, Michael Collins [2] only used function tags to define certain constituents as complements. The technique was used to train an improved parser.

Related work on enriching the output of statistical parsers, with remote dependency information, were exposed in [5], [4].

3 The Model

Our approach is to map the function tags assignment task into a classification task and then use a naive Bayes model to build a classifier for it. Classifiers are

programs that assign a class from a predefined set to an instance or case under consideration based on the values of attributes used to describe this instance. Naive Bayes classifiers use a probabilistic approach, i.e. they try to compute a conditional distribution of classes and then predict the most probable class.

4 Experimental Setup and Results

We trained our model on sections 1-21 from Wall Street Journal (WSJ) part of Penn Treebank. The set of attributes/features used was automatically extracted from trees together with their classification. In those experiments punctuation was mapped to a unique tag PUNCT and traces were left unresolved and replaced with TRACE. We used a set of features inspired from [1] that includes the following: label, parent's label, right sibling label, left sibling label, parent's head pos, head's pos, grandparent's head's pos, parent's head, head. We did not use the alternative head's pos and alternative head (for prepositional phrases that would be the head of the prepositional object) as explicit features but rather modified the phrase head rules so that the same effect is captured in pos and head features, respectively. A simple add-one smoothing method was used.

To generate the training data, we only considered nodes with functional tags, ignoring constituents unlabeled with such tags. Since a node can have several tags a training example is generated for each tag. There are two types of experiments we played with: (1) each instance is assigned a single tag from the joint set of all categories (2) each instance is assigned a tag from each of four categories. While the first type is more convenient the second is similar to what Treebank does, i.e. assigning tags from multiple categories.

4.1 Results

The results in Table 2 were obtained by testing the Naive Bayes classifier on section 23 from WSJ in Treebank 2. The performance measure reported is precision, defined as the number of correctly tagged test instances divided by the number of attempted instances. Since the input was a perfectly parsed tree the results are an accurate measurement of the actual potential of Naive Bayes for the function tags assignment task. Blaheta [1] parses the test section 23 using a state-of-the-art parser and considers only correct constituents in the output when reporting results of the functional tags assignment classifier. Our method provides state-of-the-art results. Another advantage of our method is its simplicity. For the Topicalisation category the Naive Bayes approach provides perfect tagging (100% accuracy) for the second type of experiments due mainly to the fact that the *Topicalisation* category contains a single possible tag. The precision is considerably higher for the second type of experiment (see last column in the table).

Table 2. Performance Measures

Category	Performance (%)	
	Exp 1	Exp 2
All Categories	94.12	-
Grammatical	97.04	97.91
Form/Function	51.97	59.22
Topicalisation	1.87	100
Miscellaneous	66.93	93.67

5 Conclusion

We presented in this paper a Naive Bayes approach to the task of assigning function tags to constituents in parse trees. Our experiments show that the method is robust enough to offer competitive performance for the Grammatical and Miscellaneous categories of function tags. The results reported are on perfectly parsed trees.

References

1. Blaheta, D., Johnson, M. Assigning Function Tags to Parsed Text Proceedings of the 1st Annual Meeting of the North American Chapter of the Association for Computational Linguistics, Seattle, May 2000, pp. 234-240
2. Collins, M. Three Generative, Lexicalised Models for Statistical Parsing Proceedings of the Thirty-Fifth Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics
3. Friedman, J. On bias, variance, 0/1 - loss, and the curse-of-dimensionality. Data Mining and Knowledge Discovery, 1:55-77, 1997
4. Jijkoun, V. de Rijke, M. Enriching the Output of a Parser Using Memory-Based Learning. Proceedings of the ACL 2004
5. Johnson, M. A simple pattern-matching algorithm for recovering empty nodes and their antecedents Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics
6. McCallum, A., Nigam, K. A Comparison of Event Models for Naive Bayes Text Classification Workshop on Learning for Text Categorization, AAAI (1998)
7. Bies, A., Ferguson, M., Katz, K., MacIntyre, R. Bracketing Guidelines for Treebank II Style. Penn Treebank Project.

Finding Discourse Relations in Student Essays

Rohana Mahmud and Allan Ramsay

School of Informatics, University of Manchester
R.Mahmud@postgrad.umist.ac.uk,
Allan.Ramsay@manchester.ac.uk

Abstract. The aim of the work reported here is to provide a tool to help secondary school (high school) age students to reflect on the structure of their essays. Numerous tools are available to help students check their spelling and grammar. Very little, however, has been done to help them with higher level problems in their texts. In order to do this, we need to be able to analyse the discourse relations within their texts. This is particularly problematic for texts of this kind, since they contain few instances of explicit discourse markers such as '*however*', '*moreover*', '*therefore*'. The situation is made worse by the fact that many texts produced by such students contain large numbers of spelling and grammatical errors, thus making linguistic analysis extremely challenging. The current paper reports on a number of experiments in classification of the discourse relations in such essays. The work explores the use of machine learning techniques to identify such relations in unseen essays, using a corpus of manually annotated essays as a training set.

1 Introduction

Secondary school students have numerous problems when trying to compose extended texts. They have low-level errors with spelling and grammar, for which a range of support tools exists. But they also have problems with organising their texts into coherent well-structured discourses, and they have particular problems using the devices that the language places at their disposal for indicating the structure to the reader (e.g. lexical cohesion relations and careful construction of referential chains). The underlying aim of the work reported here, then, is to provide a tool which will reveal the discourse cues that are present in such essays, and hence to allow students to reflect on what they have written.

There is, clearly, no such thing as the 'right' way to structure an essay. There are correct ways to spell things (though no extant spell-checker gets them all right), and there are correct and incorrect grammatical forms (though no extant parser can be relied on to pass all grammatically correct constructions and flag all grammatically incorrect ones). So it is, at least in principle, possible to produce a tool which will tell you whether you have spelt all the words in some document correctly, and whether all your sentences are grammatically acceptable. But there is no right or wrong structure for an extended text, so it simply makes no sense to talk of showing students where they have made 'errors' in the organisation of their texts. The best we can hope for is to show them where they have used discourse structuring markers, and to show them

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 116 – 119, 2005.

the structure that the clues they have used impose on their texts. The hope is that by making these things manifest we can help students see what the choices are and what their consequences are. If they can at least come to appreciate the importance of discourse clues, we will have provided them with a useful tool.

The problem is that identifying discourse relations is a very complex task and to date there is still no robust discourse parser (Marcu and Echihabi 2002). This process is more difficult when the texts under analysis contain large numbers of grammatical and spelling errors. However, students at a lower secondary school level (age approximately 12-14) frequently make such mistakes, and we have to be able to cope with texts containing low-level errors. To make matters even worse, such students seldom use explicit cue words such as '*however*' and '*even*' in their essays, so that algorithms that depend heavily on such terms will not work in this context.

Despite the absence of explicit cues, essays by students at this level clearly do display structure. Some students make appropriate use of lexical cohesion relations and of appropriate referential chains in organising their essays. Others, however, are less successful, and would clearly benefit from feedback in this area, thus providing motivation for the current study (Mahmud 2004), see also (Burstein, Marcu et al. 2003), (White and Arndt 1991).

2 The Experiment

A number of essays were collected from a school in North West England The essays were segmented into independent sentences simply using the standard delimiters '.', '!' and '?' (paying due attention to the use of '.' for marking the end of standard abbreviations, as in '*Mr.*'). The essays were then parsed using the PARASITE robust parser (Ramsay 2001) and a certain amount of linguistic information was recorded. This information was used as attributes for finding discourse relations. The key features are shown in table 1.0. A human annotator then determined which sentences were related and classified the relations that were found. This annotation did not require the discourse to be structured as a well-formed tree. The assumption underlying our work is that student essays do not always take the shape of well-formed trees (if they did, the tool we are building would not be needed!), so the links proposed during annotation were allowed to cross (which they did occasionally), and sentences were allowed to be unconnected to the remainder of the discourse (which happened quite frequently).

We then used the WEKA machine learning tool (Witten and Frank 2000) to acquire rules for classifying relations between sentences, based on the manual annotation. We used the following small set of relations, since it seemed very unlikely that the texts contained enough information to make learning a finer-grained classification possible:

i. Narrative (a sequence relation)
ii. Elaboration (gives more explanation of the other sentence)
iii. Contrast (if the pair sentences are contrast to each other)
iv. otherRelations (any other types of rhetorical relations (other than the above three))
v. noRelations (if the sentence is not related to any other sentences)

Table 1. Recorded Linguistic Information

Nucleus attributes (n)	Satellite attributes (s)
– n-position id	– s-position id
– n-cue words	– s-cue words
– referential status of n-subject NP	– referential status of s-subject NP
– n-mood	– s-mood
– n-head Verb	– s-head Verb
Pair-sentences attributes (p)	
– p-distance: distances between the nucleus and the satellite, can be negative or positive	
– p-cohesive: superordinate, subordinate, same or none based on the semantic relations of the head Verb; these relations were obtained from WordNet (A. Miller, Beckwith et al. 1993)	
– p-centers: the referential connection between the subject NP of the nucleus and satellite; cf. Centering Theory (Grosz, Joshi et al. 1995), but looking only at the subjects of the two sentences	

3 Results

In this experiment, we were trying to find other possible attributes that can help in identifying discourse relations if the corpus is contains few cue words. Using the RandomForest and RandomTree algorithms and 155 instances of pair-sentences produced 88.4% accuracy in classifying the discourse relations compared to the human annotator (the annotation was not carried out all that rigorously, so all we have actually shown is that the algorithms can learn this annotator's intuitions. However, if at least one annotator intuitions can be learnt then it is likely that the common intuitions of a wider group can also be learnt). The most important attributes turn out to be the s-subject and the p-distances. It is likely that lexical relations, particularly between the main verbs and between the head nouns of the subjects, are also significant, but the mechanisms we had for detecting such relations simply were not powerful enough to capture them.

4 Conclusion

In this paper, we have presented an experiment of finding discourse relations from noisy corpus. Although most of the literature on discourse relations uses cue words as the main attributes in finding discourse relations (Hutchinson 2003), (Marcu 2000), (Corston-Oliver 1998), (Knott 1996), we found other features like sentences-distances and the transitions of the subject NP can also be used as clue in developing a better discourse parser.

References

1. A.Miller, G., R. Beckwith, et al. *Introduction to WordNet: An On-line Lexical Database*, University of Princeton**:** 85, 1993
2. Burstein, J., D. Marcu, et al. Finding the WRITE Stuff: Automatic Identification of Discourse Structure in Student Essays. IEEE Intelligent Systems, 18 (January-February): 32-39, 2003
3. Corston-Oliver, S. H. *Computing Representations of the Structure of Written Discourse*. Santa Barbara, University of California**:** 256, 1998
4. Grosz, B. J., A. K. Joshi, et al. "Centering: A Framework for Modelling the Local Coherence of Discourse." *Computational Linguistics* 21(2): 203-225, 1995
5. Hutchinson, B. Automatic classification of discourse markers by their co-occurrences. *Proceedings of the ESSLLI'03 workshop on The Meaning and Implementation of Discourse Particles*, Vienna, Austria**:** 65-72, 2003
6. Knott, A. A Data-Driven Methodology for Motivating a Set of Coherence Relations. Department of Artificial Intelligence. Edinburgh, University of Edinburgh, 1996
7. Mahmud, R. *Possible attributes for identifying discourse relations structure*. CLUK2004, Birmingham, 2004
8. Marcu, D. *The Theory and Practice of Discourse Parsing and Summarization*. London England, The MIT Press, 2000
9. Marcu, D. and A. Echihabi. An Unsupervised Approach to Recognizing Discourse Relations. *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics* (ACL-2002), Philadelphia, 2002
10. Ramsay, A. PrAgmatics = ReAsoning about the Speaker's InTEnsions—The Parasite Manual. Manchester, Computation Department, UMIST**:** 48, 2001
11. White, R. and V. Arndt. *Process Writing*. Essex, Addison Wesley Longman Limited, 1991
12. Witten, I. H. and E. Frank. *Data Mining - Practical machine Learning Tools and Techniques with Java Implementation*. San Francisco, Morgan Kaufmann, 2000

Regional Versus Global Finite-State Error Repair*

M. Vilares[1], J. Otero[1], and J. Graña[2]

[1] Department of Computer Science, University of Vigo,
Campus As Lagoas s/n, 32004 Ourense, Spain
{vilares, jop}@uvigo.es
[2] Department of Computer Science, University of A Coruña,
Campus de Elviña s/n, 15071 A Coruña, Spain
grana@udc.es

Abstract. We focus on the domain of a regional least-cost strategy in order to illustrate the viability of non-global repair models over finite-state architectures. Our interest is justified by the difficulty, shared by all repair proposals, to determine how far to validate. A short validation may fail to gather sufficient information, and in a long one most of the effort can be wasted. The goal is to prove that our approach can provide, in practice, a performance and quality comparable to that attained by global criteria, with a significant saving in time and space. To the best of our knowledge, this is the first discussion of its kind.

1 Introduction

A classic problem in error repair is how far into the string to validate the process. Given that it is not possible to ensure that the correction and the programmer's intention are the same, the goal is to find the least-cost one. This can only be judged in the context of the entire input, and global methods [4, 5] are not necessarily the best option, due to their inefficiency, but are the most commonly used and for this reason considered to be the most appropriate. An alternative consists of examining the non-global context and attempting to validate repairs by tentatively recognizing ahead, following a successful approach on context-free grammars (CFGs) [7].

In this sense, although all proposals on error repair in the Chomsky's hierarchy are guided by some kind of linguistic data, whether grammar or automaton-based, each level strongly conditions the strategy to follow. So, requests on regular grammars (RGs) are different from those dealing with CFGs [8], where parses are not usually performed in depth, but breadth-wise; whilst the number of states in the associated push-down automaton is often

* Research partially by the Spanish Government under projects TIN2004-07246-C03-01, TIN2004-07246-C03-02 and HP2002-0081, and the Autonomous Government of Galicia under projects PGIDIT03SIN30501PR and PGIDIT02SIN01E.

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 120–131, 2005.

small in practice. Our proposal takes this into account by limiting the search space associated to the repair. We explore the alternatives according to the topology of the corresponding finite automaton (FA). This allows us to restrict the error hypotheses to areas close to the point where the standard recognizer comes to a halt, which translates into a significant reduction in time and space costs in relation to global approaches.

2 The Operational Model

Our aim is to parse a word $w_{1..n} = w_1 \dots w_n$ according to an RG $\mathcal{G} = (N, \Sigma, P, S)$. We denote by w_0 (resp. w_{n+1}) the position in the string, $w_{1..n}$, previous to w_1 (resp. following w_n). We generate from $\mathcal{G}$ a *numbered minimal acyclic finite automaton* for the language $\mathcal{L}(\mathcal{G})$. In practice, we choose a device [3] generated by GALENA [2]. A *finite automaton* (FA) is a 5-tuple $\mathcal{A} = (\mathcal{Q}, \Sigma, \delta, q_0, \mathcal{Q}_f)$ where: $\mathcal{Q}$ is the set of states, Σ the set of input symbols, δ is a function of $\mathcal{Q} \times \Sigma$ into $2^{\mathcal{Q}}$ defining the transitions of the automaton, q_0 the initial state and $\mathcal{Q}_f$ the set of final states. We denote $\delta(q, a)$ by $q.a$, and we say that $\mathcal{A}$ is *deterministic* iff $\mid q.a \mid \leq 1,\ \forall q \in \mathcal{Q},\ a \in \Sigma$. The notation is transitive, $q.w_{1..n}$ denotes the state $(\overset{n-2}{\dots}(q.w_1)\overset{n-2}{\dots}).w_n$. As a consequence, w is *accepted* iff $q_0.w \in \mathcal{Q}_f$, that is, the *language accepted by* $\mathcal{A}$ is defined as $\mathcal{L}(\mathcal{A}) = \{w, \text{ such that } q_0.w \in \mathcal{Q}_f\}$. An FA is *acyclic* when the underlying graph is. We define a *path in the* FA as a sequence of states $\{q_1, \dots, q_n\}$, such that $\forall i \in \{1, \dots, n-1\},\ \exists a_i \in \Sigma,\ q_i.a_i = q_{i+1}$.

In order to reduce the memory requirements, we apply a minimization process [1]. In this sense, we say that two FA's are *equivalent* iff they recognize the same language. Two states, p and q, are *equivalent* iff the FA with p as initial state, and the one that starts in q recognize the same language. An FA is *minimal* iff no pair in $\mathcal{Q}$ is equivalent. Although the standard recognition is deterministic, the repair one could introduce non-determinism by exploring alternatives associated to possibly more than one recovery strategy. So, in order to get polynomial complexity, we avoid duplicating intermediate computations in the repair of $w_{1..n} \in \Sigma^+$, storing them in a table $\mathcal{I}$ of *items*, $\mathcal{I} = \{[q, i],\ q \in \mathcal{Q},\ i \in [1, n+1]\}$, where $[q, i]$ looks for the suffix $w_{i..n}$ to be analyzed from $q \in \mathcal{Q}$.

We describe our work using *parsing schemata* [6], a triplet $\langle \mathcal{I}, \mathcal{H}, \mathcal{D} \rangle$, with $\mathcal{H} = \{[a, i],\ a = w_i\}$ a set of items called *hypothesis* that encodes the word to be recognized[1], and $\mathcal{D}$ a set of *deduction steps* that allow to items to be derived from previous ones. These are of the form $\{\eta_1, \dots, \eta_k \vdash \xi \,/\, conds\}$, meaning that if all antecedents η_i are present and the conditions *conds* are satisfied, then the consequent ξ is generated. In our case, $\mathcal{D} = \mathcal{D}^{\text{Init}} \cup \mathcal{D}^{\text{Shift}}$, where:

$$\mathcal{D}^{\text{Init}} = \{\vdash [q_0, 1]\} \qquad \mathcal{D}^{\text{Shift}} = \{[p, i] \vdash [q, i+1] \,/\, \exists [a, i] \in \mathcal{H},\ q = p.a\}$$

The recognition associates a set of items S_p^w, called *itemset*, to each $p \in \mathcal{Q}$; and applies these deduction steps until no new application is possible. The word

[1] A word $w_{1\dots n} \in \Sigma^+$, $n \geq 1$ is represented by $\{[w_1, 1],\ [w_2, 2],\ \dots\ ,\ [w_n, n]\}$.

is recognized iff a *final item* $[q_f, n+1]$, $q_f \in \mathcal{Q}_f$ has been generated. We can assume, without loss of generality, that $\mathcal{Q}_f = \{q_f\}$, and that there is only one transition from (resp. to) q_0 (resp. q_f). To get this, it is sufficient to augment the original FA with two states becoming the new initial and final states, and linked to the original ones through empty transitions, our only concession to the notion of minimal FA.

3 The Error Repair Frame

Let us assume that we are dealing with the first error in a word $w_{1..n} \in \Sigma^+$. We extend the item structure, $[p, i, e]$, where now e is the error counter accumulated in the recognition of w at position w_i in state p. We talk about the *point of error*, w_i, as the point at which the difference between what was intended and what actually appears in the word occurs, that is, $q_0.w_{1..i-1} = q$ and $q.w_i \notin \mathcal{Q}$. The next step is to locate the origin of the error, limiting the impact on the analyzed prefix to the context close to the point of error, in order to save computational effort. To do so, we introduce some topological properties. Since we work with acyclic FAs, we can introduce a simple order in $\mathcal{Q}$ by defining $p < q$ iff there exists a path $\rho = \{p, \ldots, q\}$; and we say that q_s (resp. q_d) is a *source* (resp. *drain*) for ρ iff $\exists a \in \Sigma,\ q_s.a = p$ (resp. $q.a = q_d$). In this manner, the pair (q_s, q_d) defines a *region* $\mathcal{R}_{q_s}^{q_d}$ iff $\forall \rho,\ \text{source}(\rho) = q_s$, we have that $\text{drain}(\rho) = q_d$ and $\mid \{\forall \rho,\ \text{source}(\rho) = q_s\} \mid > 1$. So, we can talk about $\mathit{paths}(\mathcal{R}_{q_s}^{q_d})$ to refer to the set $\{\rho / \text{source}(\rho) = q_s,\ \text{drain}(\rho) = q_d\}$ and, given $q \in \mathcal{Q}$, we say that $q \in \mathcal{R}_{q_s}^{q_d}$ iff $\exists \rho \in \text{paths}(\mathcal{R}_{q_s}^{q_d}),\ q \in \rho$. We also consider $\mathcal{A}$ as a global region. So, any state, with the exception of q_0 and q_f, is included in a region.

This provides a criterion to place around a state in the underlying graph a zone for which any change applied on it has no effect on its context. So, we say that $\mathcal{R}_{q_s}^{q_d}$ is the *minimal region in* $\mathcal{A}$ *containing* $p \in \mathcal{Q}$ iff it verifies that $q_s \geq p_s$ (resp. $q_d \leq p_d$), $\forall \mathcal{R}_{p_s}^{p_d} \ni p$, and we denote it as $\mathcal{M}(p)$.

We are now ready to characterize the point at which the recognizer detects that there is an error and calls the repair algorithm. We say that w_i is *point of detection* associated to a point of error w_j iff $\exists q_d > q_0.w_{1..j},\ \mathcal{M}(q_0.w_{1..j}) = \mathcal{R}_{q_0.w_{1..i}}^{q_d}$, that we denote by $\mathit{detection}(w_j) = w_i$. We then talk about $\mathcal{R}_{q_0.w_{1..i}}^{q_d}$ as the *region defining the point of detection* w_i.

The error is located in the left recognition context, given by the closest source. However, we also need to locate it from an operational viewpoint, as an item in the process. We say that $[q, j] \in S_q^w$ is an *error item* iff $q_0.w_{j-1} = q$; and we say that $[p, i] \in S_p^w$ is a *detection item* associated to w_j iff $q_0.w_{i-1} = p$.

Once we have identified the beginning of the repair region, we introduce a *modification* to $w_{1..n} \in \Sigma^+$, $M(w)$, as a series of edit operations, $\{E_i\}_{i=1}^n$, in which each E_i is applied to w_i and possibly consists of a sequence of insertions before w_i, replacement or deletion of w_i, or transposition with w_{i+1}.

This topological structure can be used to restrict the notion of modification, looking for conditions that guarantee the ability to recover the error. So, given

$x_{1..m}$ a prefix in $\mathcal{L}(\mathcal{A})$, and $w \in \Sigma^+$, such that xw is not a prefix in $\mathcal{L}(\mathcal{A})$, we define a *repair of* w *following* x as $M(w)$, so that:

(1) $\mathcal{M}(q_0.x_{1..m}) = \mathcal{R}_{q_s}^{q_d}$ (the minimal region including the point of error, $x_{1..m}$)
(2) $\exists\{q_0.x_{1..i} = q_s.x_i, \ldots, q_s.x_{i..m}.M(w)\} \in \text{paths}(\mathcal{R}_{q_s}^{q_d})$

denoted by *repair*(x, w), and $\mathcal{R}_{q_s}^{q_d}$ by *scope*(M). We now organize the concept around a point of error, $y_i \in y_{1..n}$, in order to take into account all possible repairs, by defining the *set of repairs for* y_i, as *repair*$(y_i) = \{xM(w) \in \text{repair}(x, w)/w_1 = \text{detection}(y_i)\}$. Next, we focus on filtering out undesirable repairs, introducing criteria to select minimal costs. For each $a, b \in \Sigma$ we assume insert, $I(a)$; delete, $D(a)$, replace, $R(a, b)$, and transpose, $T(a, b)$, costs. The *cost of a modification* $M(w_{1..n})$ is given by *cost*$(M(w_{1..n})) = \Sigma_{j\in J_\dashv} I(a_j) + \Sigma_{i=1}^{n}(\Sigma_{j\in J_i} I(a_j) + D(w_i) + R(w_i, b) + T(w_i, w_{i+1}))$, where $\{a_j,\ j \in J_i\}$ is the set of insertions applied before w_i; $w_{n+1} = \dashv$ the end of the input and $T_{w_n,\dashv} = 0$. So, we define the set of *regional repairs* for $y_i \in y_{1..n}$, a point of error, as

$$\text{regional}(y_i) = \{xM(w) \in \text{repair}(y_i) \Big/ \begin{matrix} \text{cost}(M) \leq \text{cost}(M'),\ \forall M' \in \text{repair}(x, w) \\ \text{cost}(M) = \min_{L\in \text{repair}(y_i)}\{\text{cost}(L)\} \end{matrix} \}$$

Before dealing with cascaded errors, precipitated by previous erroneous repairs, it is necessary to establish the relationship between recovery processes. So, given w_i and w_j points of error, $j > i$, we define the set of *viable repairs* for w_i in w_j as $\text{viable}(w_i, w_j) = \{xM(y) \in \text{regional}(w_i)/xM(y) \ldots w_j \text{ prefix for } \mathcal{L}(\mathcal{A})\}$. Repairs in *viable*$(w_i, w_j)$ are the only ones capable of ensuring the recognition in $w_{i..j}$ and, therefore, the only ones possible at the origin of cascaded errors. In this sense, we say that a point of error $w_k,\ k > j$ is a *point of error precipitated by* w_j iff $\forall xM(y) \in \text{viable}(w_j, w_k),\ \exists \mathcal{R}_{q_0.w_{1..i}}^{q_d}$ defining $w_i = \text{detection}(w_j)$, such that $\text{scope}(M) \subset \mathcal{R}_{q_0.w_{1..i}}^{q_d}$. This implies that w_k is precipitated by w_j when the region defining the point of detection for w_k summarizes all viable repairs for w_j in w_k. That is, the information compiled from those repairs has not been sufficient to give continuity to a process locating the new error in a region containing the preceding ones and therefore depending on these. We then conclude that the origin of the current error could be a wrong study of previous ones.

4 The Algorithm

Although most authors appeal to global methods to avoid distortions due to unsafe error location [4, 5], our proposal applies a dynamic estimation of the repair region, guided by the linguistic knowledge present in the underlying FA. Formally, we extend the item structure, $[p, i, e]$, where now e is the error counter accumulated in the recognition of w at position w_i in state p. Once the point of error has been located, we apply all possible transitions beginning at both, the point of error and the corresponding point of detection, which corresponds to the following deduction steps in error mode, $\mathcal{D}_{\text{error}} = \mathcal{D}_{\text{error}}^{\text{Shift}} \cup \mathcal{D}_{\text{error}}^{\text{Insert}} \cup \mathcal{D}_{\text{error}}^{\text{Delete}} \cup \mathcal{D}_{\text{error}}^{\text{Replace}} \cup \mathcal{D}_{\text{error}}^{\text{Transpose}}$.

$$\mathcal{D}^{\text{Shift}}_{\text{error}} = \{[p,i,e] \vdash [q,i+1,e],\ \exists [a,i] \in \mathcal{H},\ q = p.a\}$$
$$\mathcal{D}^{\text{Insert}}_{\text{error}} = \{[p,i,e] \vdash [p,i+1,e+I(a)],\ \nexists\, p.a\}$$
$$\mathcal{D}^{\text{Delete}}_{\text{error}} = \{[p,i,e] \vdash [q,i-1,e+D(w_i)] \Big/ \begin{array}{l} \mathcal{M}(q_0.w_{1..j}) = \mathcal{R}^{q_d}_{q_s} \\ p.w_i = q_d \in \mathcal{R}^{q_d}_{q_s} \textbf{ or } q = q_d \end{array} \}$$

$$\mathcal{D}^{\text{Replace}}_{\text{error}} = \{[p,i,e] \vdash [q,i+1,e+R(w_i,a)], \Big/ \begin{array}{l} \mathcal{M}(q_0.w_{1..j}) = \mathcal{R}^{q_d}_{q_s} \\ p.a = q \in \mathcal{R}^{q_d}_{q_s} \textbf{ or } q = q_d \end{array} \}$$
$$\mathcal{D}^{\text{Transpose}}_{\text{error}} = \{[p,i,e] \vdash [q,i+2,e+T(w_i,w_{i+1})] \Big/ \begin{array}{l} \mathcal{M}(q_0.w_{1..j}) = \mathcal{R}^{q_d}_{q_s} \\ p.w_i.w_{i+1} = q \in \mathcal{R}^{q_d}_{q_s} \textbf{ or } q = q_d \end{array} \}$$

where $w_{1..j}$ looks for the current point of error. Observe that, in any case, the error hypotheses apply on transitions behind the repair region. The process continues until a repair covers the repair region.

When dealing with an error which is not the first one in the word, it could condition a previous repair. This arises when we realize that we come back to a detection item for which some recognition branch includes a previous recovery process. The algorithm re-takes the error counters, adding the cost of new error hypotheses to profit from the experience gained from previous repairs. This permits us to deduce that if w_l is a point of error precipitated by w_k, then:

$$q_0.w_{1..i} < q_0.w_{1..j},\ \mathcal{M}(q_0.w_l) = \mathcal{R}^{q_d}_{q_0.w_{1..i}},\ w_j = y_1,\ xM(y) \in \text{viable}(w_k, w_l)$$

which proves that the state associated to the point of detection in a cascaded error is strictly smaller than the one associated to the source of the scope in the repairs precipitating it. So, the minimal possible scope of a repair for the cascaded error includes any scope of the previous ones, that is,

$$\max\{\mathit{scope}(M),\ M \in \text{viable}(w_k, w_l)\} \subset \max\{\mathit{scope}(\tilde{M}),\ \tilde{M} \in \text{regional}(w_l)\}$$

This allows us to get an asymptotic behavior close to that obtained by global repair methods, and with a comparable quality, but in practice at the cost of a local one.

5 Asymptotic Behavior

Our aim now is to validate the practical interest of our proposal in relation to classic global ones, putting into evidence the theoretical results previously advanced. We think that it is an objective criterion to measure the quality of a repair algorithm, since the point of reference is a technique that guarantees the best quality for a given error metric when all contextual information is available.

5.1 The Running Language

We choose to work with a lexicon for Spanish built from Galena [2], which includes 514,781 different words, to illustrate this aspect. The lexicon is recognized by an FA containing 58,170 states connected by 153,599 transitions, of sufficient size to allow us to consider this automaton as a representative starting point for our purposes. Although Spanish is a non-agglutinative language, it

shows a great variety of morphological processes, making it adequate for our description. The most outstanding features are to be found in verbs, with a highly complex conjugation paradigm, including nine simple tenses and nine compound tenses, all of which have six different persons. If we add the present imperative with two forms, the infinitive, the compound infinitive, the gerund, the compound gerund, and the participle with four forms, then 118 inflected forms are possible for each verb. In addition, irregularities are present in both stems and endings. So, very common verbs, such as `hacer` (*to do*), have up to seven different stems: `hac-er`, `hag-o`, `hic-e`, `har-é`, `hiz-o`, `haz`, `hech-o`. Approximately 30% of Spanish verbs are irregular, and can be grouped around 38 different models. Verbs also include enclitic pronouns producing changes in the stem due to the presence of accents: `da` (*give*), `dame` (*give me*), `dámelo` (*give it to me*). We have considered forms with up to three enclitic pronouns, like `tráetemelo` (*bring it for you and me*). There exist some highly irregular verbs that cannot be classified in any irregular model, such as `ir` (*to go*) or `ser` (*to be*); and others include gaps in which some forms are missing or simply not used. For instance, meteorological verbs such as `nevar` (*to snow*) are conjugated only in third person singular. Finally, verbs can present duplicate past participles, like `impreso` and `imprimido` (*printed*).

This complexity extends to gender inflection, with words considering only one gender, such as `hombre` (*man*) and `mujer` (*woman*), and words with the same form for both genders, such as `azul` (*blue*). In relation to words with separate forms for masculine and feminine, we have a lot of models: `autor`, `autora` (*author*); `jefe`, `jefa` (*boss*); `poeta`, `poetisa` (*poet*); `rey`, `reina` (*king*) or `actor`, `actriz` (*actor*). We have considered 20 variation groups for gender. We can also refer to number inflection, with words presenting only the singular

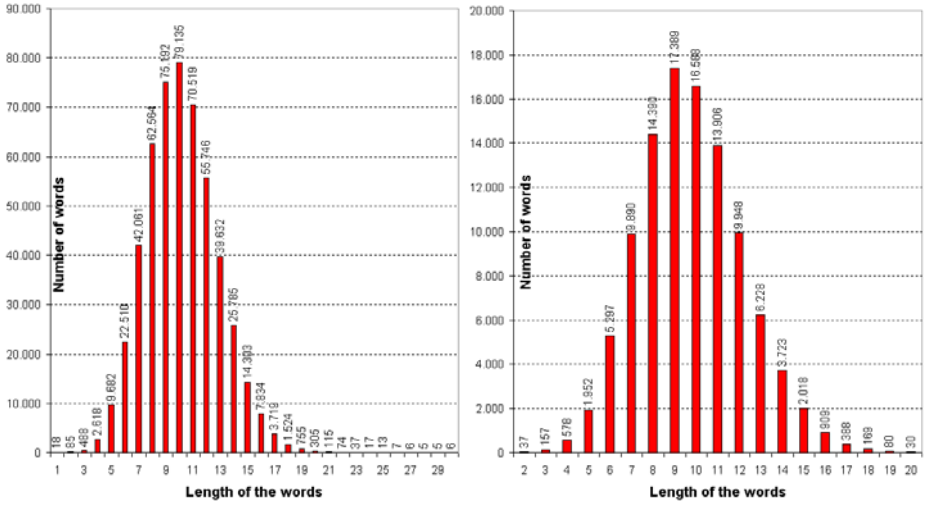


Fig. 1. Statistics on the general and error lexicons

form, as `estrés` (*stress*), and others where only the plural form is correct, as `matemáticas` (*mathematics*). The construction of different forms does not involve as many variants as in the case of gender, but we can also consider a certain number of models: `rojo`, `rojos` (*red*); `luz`, `luces` (*light*); `lord`, `lores` (*lord*) or `frac`, `fraques` (*dress coat*). We have considered 10 variation groups for number.

5.2 The Operational Testing Frame

From this lexicon, we select a representative sample of morphological errors for practical evaluation. This can be verified from Fig. 1, which shows the equitative distribution of both the original lexicon and the running sample, in terms of lengths of the words dealt with. For each length-category, errors have been randomly generated in a number and position for the first error in the input string as is shown in Fig. 3. This is of some importance since, as the authors claim, the performance of previous proposals depend on these factors, which makes no practical sense. No other dependencies, for example in terms of lexical categories, have been detected at morphological level and, therefore, they have not been considered.

In this context, our testing framework seems to be well balanced, from both an operational and linguistic viewpoint, in order to estimate the practical performance of error repair algorithms on FA architectures. It only remains to decide which repair algorithms will be tested. We choose to compare our proposal with the Savary's global approach [5], an evolution of the Oflazer's algorithm [4] and, to the best of our knowledge, the most efficient method of error-tolerant look-up in finite-state dictionaries. The comparison has been made from three complementary viewpoints: the size of the repair region considered, the computational cost and the quality achieved. We consider the editing distance [5] as error metric, the same proposed by Savary.

5.3 The Error Repair Region

We focus on the evolution of this region in relation to the location of the point of error, in opposition to static strategies associated to global repair

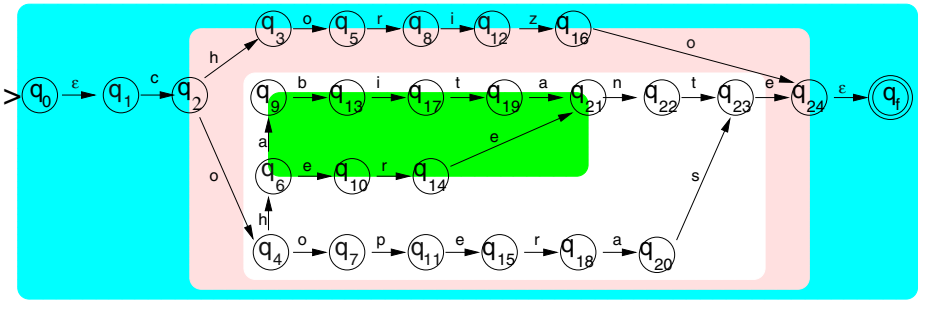


Fig. 2. The concept of region in error repair

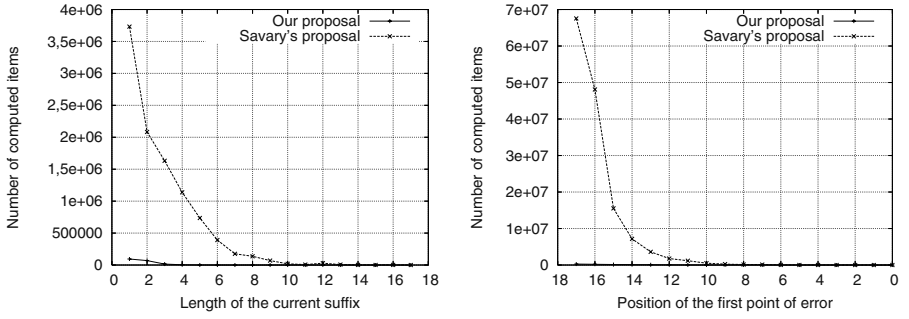


Fig. 3. Number of items generated in error mode

approaches. To illustrate it, we take as running example the FA represented in Fig. 2, which recognizes the following words in Spanish: *"chorizo"* (sausage) , *"cohabitante"* (a person who cohabits with another one) , *"coherente"* (coherent) and *"cooperase"* (I cooperated). We consider as input string the erroneous one *"coharizo"*, resulting from transposing *"h"* with *"o"* in *"chorizo"* (sausage), and later inserting the character *"a"*. We shall describe the behavior from both viewpoints, the Savary's [5] algorithm and our proposal, proving that in the worst case, when precipitated errors are present, our proposal can retake the repair process in order to recover the system from cascaded errors.

In this context, the recognition comes to an halt on state q_9, for which $\mathcal{M}(q_9) = \mathcal{R}_{q_6}^{q_{21}}$ and no transition is possible on *"r"*. So, our approach locates the error at q_6 and applies from it the error hypotheses looking for the minor editing distance in a repair allowing the state q_{21} to be reached. In this case, there are two possible regional repairs consisting in first replacing *"a"* by *"e"* and later inserting an *"e"* after *"r"* (resp. replace *"i"* by *"e"*), to obtain the modification on the entire input string *"coherezo"* (resp. *"cohereizo"*), which is not a word in our running language.

As a result, although we return to the standard recognition in q_{21}, the next input character is now *"i"* (resp. *"z"*), for which no transition is possible and we come back to error mode on the region $\mathcal{M}(q_{21}) = \mathcal{R}_{q_4}^{q_{23}}$ including $\mathcal{M}(q_9) = \mathcal{R}_{q_6}^{q_{21}}$. We then interpret that the current error is precipitated by the previous one, possibly in cascade. As result of this new process none of the regional repairs generated allow us to retake the standard recognition beyond the state q_{23}. At this point, $\mathcal{M}(q_{23}) = \mathcal{R}_{q_2}^{q_{24}}$ become the new region, and the only regional repair is now defined as the transposition of the *"h"* with *"o"*, and the deletion of *"a"*; which agrees with the global repair proposed by Savary, although the repair region is not the total one, as is the case for the latter algorithm. This repair finally allows acceptance by the FA.

The process described demonstrates that we do not need to extend the repair region to the entire FA in order to get the least-cost correction and, secondly, the risk of errors in cascade can be efficiently solved in the context of non-global approaches. Also, in the worst case, our running example illustrates

the convergence of our regional strategy towards the global one from both viewpoints, that of computational cost and that of quality of the correction.

5.4 Computational Cost

These practical results are compiled in Fig. 3, using the concept of item previously defined as a unit for measuring the computational effort. We here consider two complementary approaches illustrating the dependence on both the position of the first point of error in the word and the length of the suffix from it. So, in any case, we ensure that we take into account the degree of penetration in the FA at that point, which determines the effectiveness of the repair strategy. In effect, working on regional methods, the penetration determines the number of regions in the FA including the point of error and, as a result, the possibility of considering a non-global resolution.

In order to clearly show the detail of the tests on errors located at the end of the word, which is not easy to observe from the decimal scale of Fig. 3, we include in Fig. 4 the same results using a logarithmic scale. So, both graphics perfectly illustrate our contribution, in terms of computational effort saved, from two viewpoints which are of interest in real systems. Firstly, our proposal shows in practice a linear-like behavior, in contrast to the Savary's one, which seems to be of the exponential type. In particular, this translates into an essential property in industrial applications, the independence of the time of response from the initial conditions for the repair process. Secondly, in any case, the number of computations is significantly reduced when we apply our regional criterion.

5.5 Performance

However, statistics on computational cost only provide a partial view of the repair process, which must also take into account data related to the performance from both the user's and the system's viewpoint. In order to get this, we have introduced the following two measures, for a given word, w, containing an error:

$$performance(w) = \frac{useful\ items}{total\ items} \qquad recall(w) = \frac{proposed\ corrections}{total\ corrections}$$

that we complement with a global measure on the *precision* of the error repair approach in each case, that is, the rate reflecting when the algorithm provides the correction needed by the user. We use the term *useful items* to refer to the number of generated items that finally contribute to obtaining a repair, and *total items* to refer to the number of these structures generated during the process. We denote by *proposed corrections* the number of corrections provided by the algorithm, and by *total corrections* the number of possible ones, absolutely.

These results are shown in Fig. 5, illustrating some interesting aspects in relation with the asymptotic behavior we want to demonstrate in the regional approach. So, considering the running example, the performance in our case is not only better than Savary's, but the difference existing between them also increases with the location of the first point of error. Intuitively this is due to

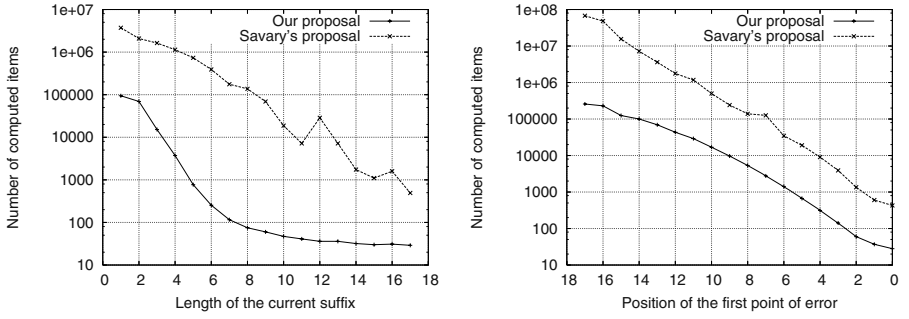


Fig. 4. Number of items generated in error mode. Logarithmic scale

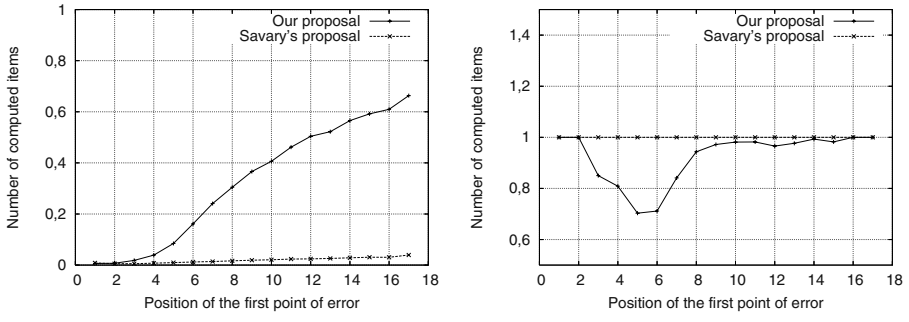


Fig. 5. Performance and recall results

the fact that the closer this point is to the beginning of the word, the greater is the number of useless items generated in error mode, a simple consequence of the higher availability of different repair paths in the FA when we are working in a region close to q_0. In effect, given that the concept of region is associated to the definition of corresponding source and drain points, this implies that this kind of region is often equivalent to the total one since the lay-out of these regions is always concentric. At this point, regional and repair approaches apply the same error hypotheses not only on a same region, but also from nearby states given that, in any case, one of the starting points for these hypotheses would be q_0 or a state close to it. That is, in the worst case, both algorithms converge.

The same reasoning could be considered in relation to points of error associated to a state in the recognition that is close to q_f, in order to estimate the repair region. However, in this case, the number of items generated is greater in the case of the global technique, which is due to the fact that the morphology of the language often leads to the generation of regions which concentrate near q_f, a simple consequence of the common derivational mechanisms applied on suffixes defining gender, number or verbal conjugation groups. So, it is possible to find a regional repair by just implying some error hypotheses from the state associated

to the point of error or from the associated detection point and, although this regional repair may be different from the global one, its computational cost would usually be lower.

A similar behavior can be observed with respect to the recall relation. Here, Savary's algorithm shows a constant graph since the approach applied is global and consequently the set of corrections provided is always the entire one for a fixed error counter. In our proposal, the results prove that the recall is smaller than that for Savary's, which illustrates the gain in computational efficiency in comparison with the global method. With regard to the convergence between both approaches, we must again search around points of detection close to the beginning of the word, which also often implies repair regions being equivalent to the total one and repairs starting around q_0, as is illustrated in Fig. 5.

However, in contrast to the case of performance, it can be seen that for recall the convergence between global and regional proposals also seems to extend to processes where the point of error is associated to states close to q_f, that is, when this point is located near the end of the word. To understand this, it is sufficient to take into account that we are not now computing the number of items generated in the repair, but the number of corrections finally proposed. So, given that the closer to the end of the word we are, the smaller is the number of alternatives for a repair process, both global and regional approaches also converge towards the right of the graph for recall.

Finally, the regional (resp. the global) approach provided as correction the word from which the error was randomly included in 77% (resp. 81%) of the cases. Although this could be interpreted as a justification for using global methods, it is necessary to remember that we are now only taking into account morphological information, which has an impact on precision for a regional approach, but not for a global one, which always provides all the repair alternatives without exclusion. So, the consideration of the precision concept represents, in the exclusive morphological context considered, a clear disadvantage for our proposal since it bases its efficiency in the limitation of the search space. We expect that the integration of linguistic information from both syntactic and semantic viewpoints will significantly reduce this gap of less than 4% in precision, or may even eliminate it.

6 Conclusion

We have illustrated how a least-cost error repair method can be applied to a finite-state architecture in order to recover the recognition at the point of each error, to avoid the possibility of non-detection of any subsequent errors. So, although the correctness of a symbol can only be judged in the context of the entire string, which can be extremely time-consuming, our proposal minimizes the impact by dynamically graduating the size of the repair zone on the basis of underlying grammatical structure. In this sense, the practical results seem promising, demonstrating as they do a significant reduction in time and space costs with no apparent loss of quality.

References

1. J. Daciuk, S. Mihov, B.W. Watson, and R.E. Watson. Incremental construction of minimal acyclic finite-state automata. *Computational Linguistics*, 26(1):3–16, 2000.
2. J. Graña, F.M. Barcala, and M.A. Alonso. Compilation methods of minimal acyclic automata for large dictionaries. *Lecture Notes in Computer Science*, 2494:135–148, 2002.
3. C.L. Lucchesi and T. Kowaltowski. Applications of finite automata representing large vocabularies. *Software-Practice and Experience*, 23(1):15–30, January 1993.
4. K. Oflazer. Error-tolerant finite-state recognition with applications to morphological analysis and spelling correction. *Computational Linguistics*, 22(1):73–89, 1996.
5. A. Savary. Typographical nearest-neighbor search in a finite-state lexicon and its application to spelling correction. *Lecture Notes in Computer Science*, 2494:251–260, 2001.
6. K. Sikkel. *Parsing Schemata*. PhD thesis, Univ. of Twente, The Netherlands, 1993.
7. M. Vilares, V.M. Darriba, and M.A. Alonso. Searching for asymptotic error repair. *Lecture Notes in Computer Science*, 2608:276–281, 2003.
8. M. Vilares, V.M. Darriba, and F.J. Ribadas. Regional least-cost error repair. *Lecture Notes in Artificial Intelligence*, 2088:293–301, 2001.

Lexicalized Beam Thresholding Parsing with Prior and Boundary Estimates*

Deyi Xiong[1,2], Qun Liu[1], and Shouxun Lin[1]

[1] Institute of Computing Technology, Chinese Academy of Sciences,
P.O. Box 2704, Beijing 100080, China
[2] Graduate School of Chinese Academy of Sciences
{dyxiong, liuqun, sxlin}@ict.ac.cn

Abstract. We use prior and boundary estimates as the approximation of outside probability and establish our beam thresholding strategies based on these estimates. Lexical items, e.g. head word and head tag, are also incorporated to lexicalized prior and boundary estimates. Experiments on the Penn Chinese Treebank show that beam thresholding with lexicalized prior works much better than that with unlexicalized prior. Differentiating completed edges from incomplete edges paves the way for using boundary estimates in the edge-based beam chart parsing. The beam thresholding based on lexicalized prior, combined with unlexicalized boundary, runs faster than that only with lexicalized prior by a factor of 1.5, at the same performance level.

1 Introduction

In the recent development of parsing technology, lexicalized grammars has been used in several state-of-the-art parsers(see [1][2] etc.) to pursue high accuracy because they control not only structural dependencies, but also lexical dependencies, lexico-structural dependencies(see [3]). In this paper, we just consider lexicalized context-free grammars(LCFG). LCFG is a CFG with its nonterminals lexicalized by some lexical items(see [4]). For example, in Collins' bilexical grammars, each nonterminal is associated with a word(called the head of the corresponding constituent) and a POS tag of the head.

When CKY chart parsing techniques are used to bilexical context-free grammars, the time complexity is not $O(n^3)$, but $O(n^5)$. A CKY chart parser can be considered as a two-dimensional matrix of cells. In each chart cell, there are $O(n)$ edges because of the $O(n)$ possible choices for head words to be associated with nonterminals of edges. When fundamental rule is used between two neighbor cells, the algorithm requires additional time $O(n^2)$.

Because of the heavy work load for lexicalized parsers, edge pruning techniques, like beam thresholding, are usually used by practical parsing algorithms.

* This work was supported in part by National High Technology Research and Development Program under grant #2001AA114010.

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 132–141, 2005.

The key problem for beam thresholding is how to select a evaluation function which removes less likely edges from cells. A good evaluation function should make reasonable tradeoff between accuracy and efficiency, which means pruning as many edges which are not part of the correct parse as possible. The ideal evaluation function should consider not only inside probability but also outside probability of constituents. However, outside probability can only be computed after a full parse is completed. This is very difficult for bottom-up chart parsing. Approximate estimates of outside probability are therefore used as alternatives.

We check prior probability and boundary estimate of constituents as our approximation of outside probability. Prior probability measures the likelihood of the lexicalized/unlexicalized nonterminal without considering any contexts where the nonterminal occurs. Boundary estimates compute the prior probability in the context of neighbor word sequences.

Although unlexicalized prior probability was used by Goodman(see [5]), and lexicalized prior probability was used in Collins' thesis work(see [2]), we give an experimental comparison between lexicalized and unlexicalized prior probability in section 4. What's more, different thresholds are used for complete and incomplete edges, which make the curves of accuracy vs. the number of produced edges more smoothing.

Boundary estimates were used in best-first chart parsing(see [6]), which were proved to be the best figures of merit. However, to the best of our knowledge, it is the first time to use them in beam thresholding parsing. When boundary estimates used in the lexicalized beam thresholding parsing, two changes must be made. One is lexicalized extension which is discussed in section 2, the other is the conversion from constituent-based parsing into edge-based parsing. We use a very simple way to do this conversion, and discuss it in section 3. Finally, the combination of lexicalized prior probability and unlexicalized boundary estimate is totally new beam thresholding technique, which gains a speedup by a factor of 1.5 compared with lexicalized prior beam thresholding, at the same performance level.

2 Prior and Boundary Estimates

According to the wisdom of the parsing literature, the best way to measure the likelihood of a constituent given the entire sentence should maximize not only the total probability of that constituent appearing in isolation, but also the likelihood of sentence as a whole. We denote the probability as $P(N^X_{j,k}|w_{0,n})$, here $N^X_{j,k}$ is a constituent of type X (e.g. NP, VP for delexicalized nonterminal, NP(week,NN), VP(bought,VBD) for lexicalized nonterminal, etc.) that covers the span of words $w_j, ..., w_k$. We can rewrite the conditional probability as follows:

$$\begin{aligned} P(N^X_{j,k}|w_{0,n}) &= \frac{P(N^X_{j,k}, w_{0,n})}{P(w_{0,n})} \\ &\approx \frac{P(N^X_{j,k}, w_{0,j-1}, w_{k+1,n})P(w_{j,k}|N^X_{j,k})}{P(w_{0,n})} \quad . \end{aligned} \tag{1}$$

where the left part of numerator of (1) is the so-called outside probability $\alpha(N^X_{j,k})$ and the right part is the inside probability $\beta(N^X_{j,k})$. For the outside probability, we can rewrite it as follows:

$$\alpha(N^X_{j,k}) = P(w_{0,j-1}, w_{k+1,n})P(N^X_{j,k}|w_{0,j-1}, w_{k+1,n}) \ . \tag{2}$$

Finally, we get:

$$P(N^X_{j,k}|w_{0,n}) \approx \frac{P(N^X_{j,k}|w_{0,j-1}, w_{k+1,n})\beta(N^X_{j,k})}{P(w_{i,j}|w_{0,j-1}, w_{k+1,n})} \ . \tag{3}$$

If we assume that $P(N^X_{j,k}|w_{0,j-1}, w_{k+1,n}) \approx P(N^X_{j,k})$, we get the prior probability of the constituent of type X. If $N^X_{j,k}$ is a lexicalized nonterminal, denoted as a triple $< l, hw, ht >$, where l is the delexicalized nonterminal, hw, ht are the head word and head tag of the constituent respectively, we call the probability $P(l, hw, ht)$ the lexicalized prior. Otherwise, we call the probability $P(l)$ the unlexicalized prior.

If we assume that $P(N^X_{j,k}|w_{0,j-1}, w_{k+1,n}) \approx P(N^X_{j,k}|w_{j-1})$, we get the boundary estimate of the constituent of type X. If $N^X_{j,k}$ is a lexicalized nonterminal, we refer to the probability $P(l, hw, ht|w_{j-1})$ as the lexicalized boundary estimate. Otherwise, we refer to the probability $P(l|w_{j-1})$ as the unlexicalized boundary estimate.

Of course, we can also use the right side word sequence $w_{k+1,n}$, just like Caraballo and Charniak (see [6], henceforth C&C). According to their derivation and independent assumption, we can get our lexicalized version:

$$P(N^X_{j,k}|w_{0,n}) \approx \frac{P(N^X_{j,k}|w_{j-1})\beta(N^X_{j,k})P(w_{k+1}|N^X_{j,k})}{P(w_{j,k+1}|w_{0,j-1})} \ . \tag{4}$$

However, when we calculate the probability $P(w_{k+1}|N^X_{j,k})$, we have to face serious data sparseness, especially for lexicalized nonterminals. Therefore, we just ignore the word context on the right side of constituent $N^X_{j,k}$.

The other difference between our version and the work of C&C is that there is no need of computing the denominator of formula (3) since all edges in the same cell have the same value of the denominator. In C&C's parser, all constituents in the agenda were compared to all other constituents, so the denominator is different for different constituents in the agenda. Although our work is greatly simplified without the normalization of two distributions of numerator and denominator, global information from the denominator is lost. Maybe comparing edges from different cells with boundary estimates is our further work.

The calculation of inside probability $\beta(N^X_{j,k})$ will be discussed in section 4.1, here we give a brief introduction of calculation of prior and boundary estimates. Unlexicalized prior and boundary probabilities are estimated from our training data using the maximum likelihood estimate by collecting all counts from events where they appear. For the lexicalized prior, we divide it into two parts:

$$P(l, hw, ht) = P(hw, ht)P(l|hw, ht) \ . \tag{5}$$

The lexicalized boundary estimate is similarly decomposed as:

$$P(l, hw, ht|w_{j-1}) = P(hw, ht|w_{j-1})P(l|hw, ht, w_{j-1}) \ . \tag{6}$$

All conditional probabilities are smoothed through Witten-Bell interpolation just like Collins (see [2]).

3 Edge-Based Extension

Boundary estimates were originally used on constituents, or completed edges in the approach taken in C&C. Only constituents are pushed into the agenda and ranked by boundary figure of merit.[1] Charniak et al.(see [7]) extended C&C to edge-based parsing by grammar binarization. In their work, tree-bank grammars were transformed to be unary or binary. However, our parser uses Markov grammars (see [1][2]) which decompose the right-hand side (henceforth RHS) of CFG rules into one unique head and several modifiers. During bottom-up parsing, heads are firstly generated and then their parents added upon them. Later modifiers to the left/right of heads will be found and attached according to fundamental rules. In beam thresholding parsing, cells are filled with completed edges (no modifiers to be attached) and incomplete edges (some modifiers waiting for bing attached) at any time. For incomplete edges, there is no sense of using boundary estimates, but prior estimates can still be used. Therefore, in our parser, boundary estimates are only used on completed edges, prior estimates are used on both incomplete and complete edges. And correspondingly, different thresholds are assigned for completed edges and incomplete edges.

Along this line, we take two different beam thresholds for completed edges and incomplete edges even if only prior estimates are used. And we find double beam thresholding (with two different thresholds for completed and incomplete edges) is better than single beam thresholding (with the same threshold for completed and incomplete edges), which is shown in Fig. 1. The curve of double beam thresholding is more smoothing than that of single beam thresholding. We think it is because completed edges and incomplete edges do need different beam width to prune less likely edges. Just one single beam threshold is too strict and therefore fits in with incomplete edges but not with completed edges or vice versa.

Since we use double beam thresholds, a practical consideration is how to choose the best set of thresholds which make the best speed versus performance tradeoff. Here we use Goodman's (see [5]) automatic thresholding parameter optimization algorithm with some little changes. We use the total entropy as the metric of performance and measure the amount of work done by the parser in terms of the total number of edges produced by the parser (including edges to

[1] There is some difference between our definition of boundary estimates and that in C&C. By boundary estimates, we just mean $P(l, ht, hw|w_{j-1})$, or $P(l|w_{j-1})$, not including inside probability, and the denominator.

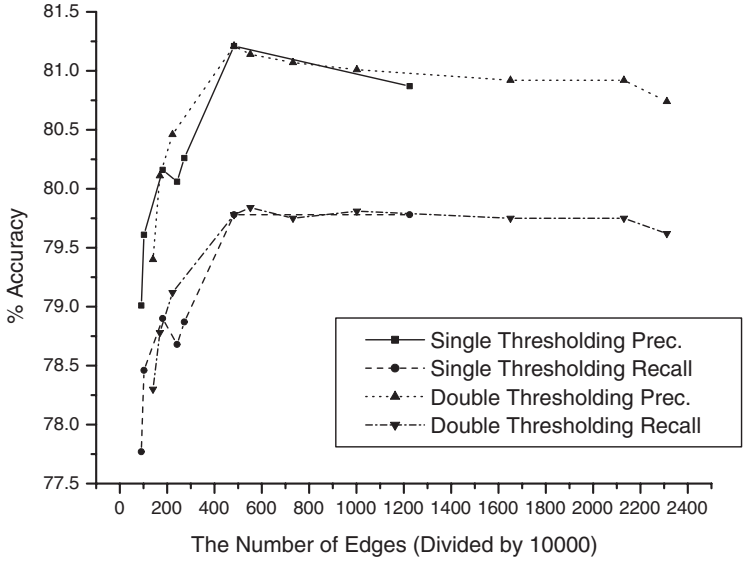


Fig. 1. Double beam thresholding vs. Single beam thresholding

be pruned and those to be kept or replaced in dynamic programming). In fact, we do obtain different beam thresholds for completed and incomplete edges by the beam thresholds optimization algorithm when we just use the prior beam thresholding.

4 The Experiments

4.1 The Parser, Data and Measurement

Our parsing model is similar to Collins' model 2. Nonterminals are lexicalized with the corresponding head word and head tag. Markov grammars are used, which decompose the RHS of CFG rules as follows:

$$P(h) \rightarrow \#L_n(l_n)...L_1(l_1)H(h)R_1(r_1)...R_m(r_m)\# \ . \tag{7}$$

The uppercase letters are delexicalized nonterminals, while the lowercase letters are lexical items corresponding to delexicalized nonterminals. $H(h)$ is the head constituent of the rule from which the head lexical item h is derived according to some head percolation rules (here we use the modified head percolation table for Chinese from Xia (see [8])). The special termination symbol "#", which indicates that there is no more symbols to the left/right, makes Markov process model the left and right modifiers sequences. When rules expanded, the head constituent H is firstly generated, then in order $L_1(l_1)$ through L_{n+1} $(=\#)$, and similarly for $R_1(r_1)$ through the right termination symbol. The probability of guessing H is conditioned on the parent P and the head

word hw and head tag ht, while the probability of generating modifiers $M_i(m_i)$ (eg., $L_i(l_i)$ or $R_i(r_i)$) is conditioned on P, H, ht, hw, M_{i-1} and the direction and distance features. Our distance definitions are different for termination symbol and non-termination symbol, which are similar to Klein and Manning (see [9]).

We do some linguistically motivated re-annotations. The first one is marking non-recursive noun phrases from other common noun phrases without introducing any extra unary levels (see [2][10]). We find this basic NP re-annotation is very helpful for the performance. The second re-annotation is marking basic VPs, which we think is beneficial for reducing multilevel VP adjunction ambiguities (see [11]). The last one is distinguishing single clauses from compound clauses which are constituted with several single clauses bundled up by some logical relationships such as causality. In the Penn Chinese Treebank (version 1.0, henceforth CTB for short; see [12]), all simple clauses are labelled as IP. Since the paper focuses on the beam thresholding parsing, we just give a brief description about these re-annotations.

All experiments are trained on articles 1-270 of CTB just like Bikel and Chiang (see [13]). Input trees are preprocessed under standard normalizations with punctuation items apart from commas or colons removed. Articles 271-300 are used for test and the automatic beam thresholds optimization algorithm. The first 30 sentences of length at most 30 are extracted from articles 271-300 for optimizing beam thresholds with Goodman's algorithm, which are called optimization sentences. Then the next 15 sentences of length at most 30 are used as interval separating the optimization sentences from the next 200 sentences of length at most 30 which are used as test data.

For the measurement of correctness, we use the labelled precision/recall just like Collins (see [2]) except that entropy is used as the metric of performance in beam thresholds optimization algorithm. As for the metric of speed, we use the total number of edges (divided by 10000) produced by the parser just described in the last section.

4.2 Lexicalized Prior Versus Unlexicalized Prior

Our first experiment is designed to show what's the role lexical items (e.g. head word hw and head tag ht) play in the prior estimate, and thus in beam thresholding. On the 200 sentences test set, we run two parsers. One uses the unlexicalized prior probability $P(l)$ to prune competed edges and incomplete edges, while the other uses the lexicalized prior probability $P(l, hw, ht)$ to remove less likely completed and incomplete edges. Beam thresholds of both parsers for completed and incomplete edges are optimized on the 30 sentences optimization set. The curves of precision and recall versus the number of edges are graphed as we sweep the set of optimized beam thresholds, which are shown in Fig. 2. As can be seen, the prior estimate with lexical items is much more efficient than that without them. For example, to reach the 79.1% recall level, the parser with unlexicalized prior estimate produces edges nearly 6 times as many as those produced by the parser with lexicalized prior estimate.

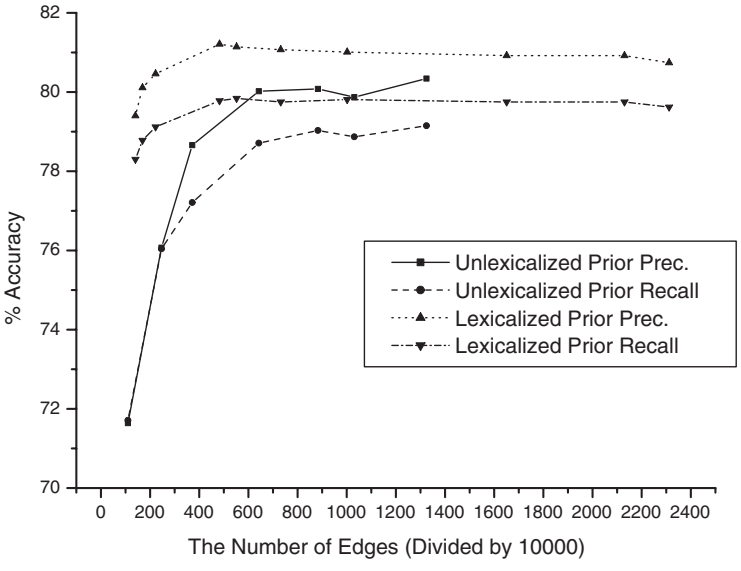


Fig. 2. Lexicalized Prior vs. Unlexicalized Prior

The reason that lexical items are indeed helpful is very obvious. For a certain delexicalized nonterminal, the choice for its corresponding head word in the cell (or the span of sentences) is very limited. Less likely combination, for instance, a VP headed by a preposition word "of" or "in" will get very small lexicalized prior probability even though it may have a high inside probability. And on the other hand, the words in the span which are to be selected as head word press some conditions on the selection of delexicalized nonterminal. If there are not any verb words in the span, VP may be less likely to be selected as the nonterminal dominating the span. Therefore, even we increase the number of edges in the cell by expanding the set of nonterminals through lexicalization, the pruning by lexical items maybe overwhelmingly offset the increase.

4.3 Lexicalized Boundary Versus Lexicalized Prior

We try experiments comparing beam thresholding with lexicalized boundary estimate to that with lexicalized prior estimate. In the experiment with lexicalized boundary pruning, according to the way discussed in section 3, boundary estimates are only used on completed edges while prior estimates are used on incomplete edges. In the experiment with lexicalized prior pruning, prior estimates are used on both completed and incomplete edges. The results of these experiments are shown in Fig. 3. Unfortunately, we find that lexicalized boundary pruning is totally worse than lexicalized prior pruning. Our intuition was that we would see a improvement from the boundary estimate. We think data sparseness may lead to this failure. In the next experiments, we will use unlexicalized bound-

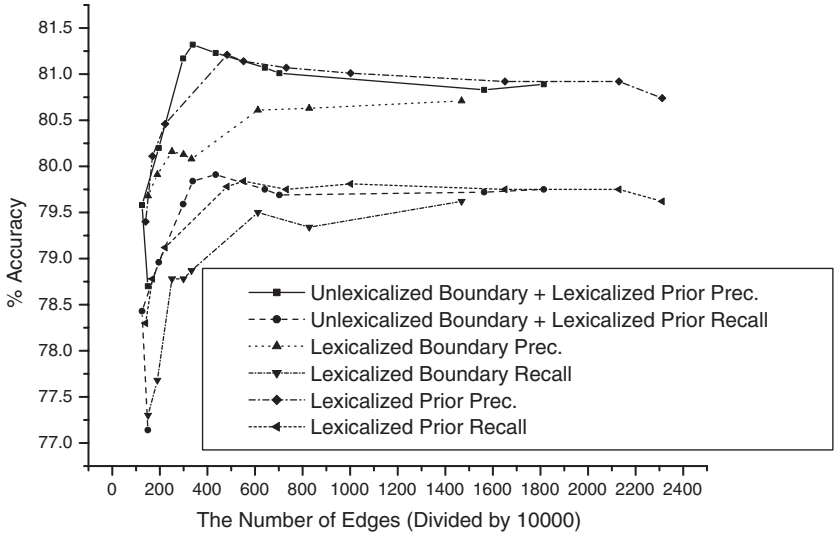


Fig. 3. Lexicalized Boundary vs. Lexicalized Prior vs. Combination of Unlexicalized Boundary and Lexicalized Prior

ary estimate combined with lexicalized prior estimate to replace the lexicalized boundary estimate hoping that it is helpful to alleviate data sparseness.

4.4 Combining Unlexicalized Boundary and Lexicalized Prior

From the first experiment, we can see the interdependency between delexicalized nonterminal and lexical items is very important for efficient pruning. However, in the formula (6), the conditional probability $P(l|hw, ht, w_{j-1})$ will be very small because of serious data sparseness even if we use complicated smoothing techniques such as Witten-Bell smoothing (see [14]). Since we want to calculate the prior probability of delexicalized nonterminal l conditioned on both head items hw, ht and lexical boundary item w_{j-1}, we just separate them. We will use the following to approximate the outside probability.

$$\begin{aligned}\alpha(N^X_{j,k}) &\approx P(l, hw, ht)P(l|w_{j-1}) \\ &= P(hw, ht)P(l|hw, ht)P(l|w_{j-1}) \ . \end{aligned} \tag{8}$$

Thus, we not only model the interdependency between delexicalized nonterminal l and head items hw, ht and boundary item w_{j-1}, but also reduce data sparseness. Then we try experiment to check the new pruning with the new approximation. Similarly, lexicalized prior estimates are used on incomplete edges, and the new approximation is used on completed edges. Figure 3 shows the results of this experiment. As can be seen, beam thresholding with lexicalized prior probability times unlexicalized boundary estimate is much better than that with lexicalized boundary estimate, and also better than lexicalized

beam thresholding. Since the parsing time is nearly proportional to the number of edges produced by the parser, the combined thresholding runs averagely 1.5 times faster than lexicalized prior beam thresholding alone.

5 Related Work

Among the previous related work, the most similar to our approaches is Goodman's work (see [5]). He also used beam thresholding with prior probability. The biggest difference is that his parser used unlexicalized grammars and therefore lexical items can't be incorporated into his prior probability. In fact, our experiments show that lexical items are very helpful for edge pruning.

Another similar work was done by C&C. They used boundary estimates in best-first constituent-based parsing. Compared to their approach, our boundary estimates calculation need not consider trigram probability and normalization. And other differences include edge-based extension and lexicalization in our boundary estimate pruning strategy.

Compared to Collins's work, our lexicalized prior pruning distinguishes completed edges and incomplete edges and therefore optimizes two different beam width for them. Additionally, our combined beam thresholding pruning works better than lexicalized prior pruning alone.

6 Conclusions and Further Work

We check prior and boundary estimates as the approximation of outside probability and incorporate them into beam thresholding pruning strategies. We have found that lexical items (e.g. head word and head tag) are very beneficial for edge pruning. After edge-based conversion and lexicalized extension, boundary estimates are used in beam thresholding. To our knowledge, the beam thresholding with boundary estimates is novel. Although lexicalized boundary estimates work worse than lexicalized prior estimates, the combination of unlexicalized boundary and lexicalized prior estimates works better.

Our future work involves pruning edges from different cells. Goodman's global thresholding is very interesting, though it works better only on simpler grammars. Maybe we will use boundary estimates with trigram probability which provides global information in some sense to achieve this goal.

References

1. Charniak Eugene. 2000. A maximum-entropy-inspired parser. In Proceedings of the 1st Conference of the North American Chapter of the Association for Computational Linguistics. Seattle.
2. Michael Collins. Head-Driven Statistical Models for Natural Language Parsing. PhD thesis, University of Pennsylvania, 1999.
3. Daniel M. Bikel. 2004. Intricacies of Collins' Parsing Model. See http://www.cis.upenn.edu/ dbikel/.

4. Giorgio Satta. 2000. Parsing Techniques for Lexicalized Context-free Grammars. Invited talk in the Sixth International Workshop on Parsing Technologies. Trento, Italy.
5. Joshua Goodman. 1997. Global thresholding and multiple-pass parsing. In Proceedings of the Second Conference on Empirical Methods in Natural Language Processing, pages 11-25.
6. Sharon Caraballo and Eugene Charniak. 1998. New figures of merit for best-first probabilistic chart parsing. Computational Linguistics, 24(2):275-298, June.
7. Eugene Charniak, Sharon Goldwater, and Mark Johnson. 1998. Best-first edge-based chart parsing. In 6th Annual Workshop for Very Large Corpora, pages 127-133.
8. Fei Xia. Automatic Grammar Generation from Two Different Perspectives. PhD thesis, University of Pennsylvania, 1999.
9. Dan Klein and Christopher D. Manning. Fast Exact Natural Language Parsing with a Factored Model. Advances in Neural Information Processing Systems 15 (NIPS-2002), 2002.
10. Dan Klein, Christorpher D.Manning. 2003. Accurate Unlexicalized Parsing. In Proceedings of the 42th Association for Computational Linguistics.
11. Roger Levy, Christopher Manning. 2003. Is it harder to parse Chinese, or the Chinese Treebank? In Proceedings of the 42th Association for Computational Linguistics.
12. Nianwen Xue and Fei Xia. 2000. The Bracketing Guidelines for Chinese Treebank Project. Technical Report IRCS 00-08, University of Pennsylvania.
13. Daniel M. Bikel and David Chiang. 2000. Two statistical parsing models applied to the chinese treebank.In Proceedings of the Second Chinese Language Processing Workshop, pages 1-6.
14. Stanley F. Chen, Joshua Goodman. An Empirical Study of Smoothing Techniques for Language Modeling. In Proceedings of the Thirty-Fourth Annual Meeting of the Association for Computational Linguistics.

Unsupervised Evaluation of Parser Robustness

Johnny Bigert[1], Jonas Sjöbergh[1], Ola Knutsson[1], and Magnus Sahlgren[2]

[1] KTH Nada, 100 44 Stockholm, Sweden
{johnny, jsh, knutsson}@nada.kth.se
[2] SICS, Box 1263, 164 29 Kista, Sweden
mange@sics.se

Abstract. This article describes an automatic evaluation procedure for NLP system robustness under the strain of noisy and ill-formed input. The procedure requires no manual work or annotated resources. It is language and annotation scheme independent and produces reliable estimates on the robustness of NLP systems. The only requirement is an estimate on the NLP system accuracy. The procedure was applied to five parsers and one part-of-speech tagger on Swedish text. To establish the reliability of the procedure, a comparative evaluation involving annotated resources was carried out on the tagger and three of the parsers.

1 Introduction

Automatic parsing of text is a popular field of research. Many of the applications where parsing is used, such as parsing human input to a computer system, handle text that is not proofread. Depending on the application, the text can be relatively error free (e.g. parsing newspaper articles from the internet) or contain large amounts of errors (e.g. using a parser as a tool for second language learners when writing essays). If the intended use of a parser is domains with many errors, it must be robust enough to produce useful output despite noisy input. It is not sufficient to achieve a good performance on error-free text. Usually, the accuracy of a parser on error-free text is known, but the accuracy on texts containing errors is often unknown.

Carroll and others give a comprehensive overview of different parser evaluation methods and discuss some shortcomings [1]. Evaluation of parsers is usually carried out by comparing the parser output to a manually annotated or manually corrected version of a test text. Manual work is expensive, and not necessarily error free. If the NLP system is under development, the evaluation has to be carried out repeatedly. Thus, very large amounts of annotated resources may be required to avoid data exhaustion. Many languages have no large manually annotated resources at all, and those existing often contain only error-free texts.

Manual annotation is not only expensive, but often hard to reuse when evaluating a new parser. Generally, it is non-trivial to map the output of one parser to the output of another [2]. Thus, the effort of manually annotating text with one type of parse information is not generally reusable for other parsers.

To carry out the evaluation of NLP system robustness while avoiding the above-mentioned drawbacks, we propose a procedure that requires no manual work or annotated

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 142–154, 2005.

resources. There are, as pointed out by Menzel [3], many types of robustness. Robustness in this context is defined as the system's reluctance to change its output when the input becomes increasingly noisy and ill-formed. The only requirements of the evaluation method are a (relatively error-free) text and an estimate of the accuracy of the parser (on error-free text, which is usually known). Despite the modest requirements, the evaluation procedure provides accurate estimates of the robustness of an NLP system.

The method is an extension of a supervised approach to parser robustness evaluation [4]. It is unsupervised and based on introduction of artificial spelling errors in error-free text. We have chosen to use spelling errors to simulate noisy input for several reasons. First, performance (keyboard) spelling errors are language independent. Hence, anyone can use the framework and apply it to their parser in their language without modification. Second, performance spelling errors are easily described and widely understood and thus, does not obscure the important parts of the evaluation procedure. Also, to keep the description of the error model as straightforward as possible, we have refrained from applying an automatic spelling corrector. Please keep in mind that the evaluation method is not restricted to spelling errors, but applicable to any error type, such as incomplete sentences in the sense of e.g. [5].

Another approach to evaluation of parser robustness is provided by Foster [6]. There, parser robustness is evaluated by running a parser on ungrammatical text and comparing the output to the output when run on the same text after it has been manually corrected. Also, Li [7] proposes a method based on an annotated corpus of low quality language use, in this case transcribed phone calls.

We assessed the reliability of the evaluation method by using five different parsers and one part-of-speech tagger. All five parsers process written Swedish text, even though the evaluation method is language independent. The tagger and three of the parsers had resources annotated with the correct tagger/parser output, allowing us to verify the results of the unsupervised evaluation.

2 Proposed Method

We are given an NLP system processing and outputting row-based data, that is, reading one input (e.g. a word) per row and producing one output (e.g. a parse string) per row. We want to assess the robustness of the system. To this end, we need to evaluate the performance of the system when applied to input with increasing amounts of noise. The proposed method is applicable to most NLP system, but parsers will be used here for the clarity of exposition.

Naturally, the performance of an NLP system can be better assessed with an annotated resource. To begin with, the discussion here will include such a resource. The aim is to establish how much information can be gained concerning the performance of the NLP system *without* the annotated resource.

We require a text to be used in the evaluation. The text will be processed by the NLP system (i.e. a parser). Even though the text can be chosen arbitrarily, we simplify the exposition of the method by using the text from a treebank; but keep in mind that the method does not require an annotated resource. We introduce spelling errors in the text to determine the performance of the NLP system under the influence of noisy and

ill-formed input. To this end, we use a freeware program called MISSPLEL [8], producing human-like spelling errors. We introduce spelling errors simulating keyboard mistypes. To avoid alternate interpretations of a sentence, the spelling errors result only in words not present in a dictionary. For more details on the introduction of spelling errors, we refer to [4].

Three different data sources are involved in the discussion of the evaluation method. The three files have the same number of rows since they all originate from the same text (i.e. the text in the treebank). For each row, they contain a data pair: a word (that may or may not be misspelled) and a parse string for that word. Only the parse part is used here.

The first file, denoted `m`, is the manually checked annotated resource (e.g. a tree bank). The second file, denoted `0` (zero), is the output of the NLP system when applied to the original treebank text (0% errors). The third file, denoted `n`, is the output of the NLP system when applied to the text containing errors (e.g. $\texttt{n} = 5\%$ of the words in the file are misspelled). Clearly, a file containing $\texttt{n}\%$ errors is more difficult to parse than an error-free text and we want to determine how difficult.

2.1 Five Cases

Given one row of the treebank, the 0% file and the $\texttt{n}\%$ file, we analyze the different cases that may occur. Say that the treebank parse (i.e. the correct answer) is `a`. The 0% file either contains the correct answer `a`, or an incorrect answer `b`. Furthermore, the $\texttt{n}\%$ file may contain the correct answer `a`, the same incorrect answer as the 0% file `b` or even another incorrect answer `c`. From this, we obtain several different combinations.

We introduce a notation (denoted `m0n`) consisting of three columns. The first position is the parse found in the treebank `m`, the second is the 0% file `0` and the third is the $\texttt{n}\%$ file `n`. For example, `abc` means that the parse from the treebank was `a`, the parse from the 0% file was `b` and the parse found in the $\texttt{n}\%$ file was `c`.

Thus, using the new notation, we get five different cases when comparing parses of a single word: `aaa`, `aab`, `aba`, `abb` and `abc`. See Table 1 for an example. The first case `aaa` is the most common, where all three files agree on the same parse. Second, `aab` is the case where an error nearby in the text corrupted the parsing process of this row. The third case `aba` is unusual, but not negligibly so. This may occur when the parser is uncertain and chooses between two equal alternatives and arbitrarily chooses the correct one at the $\texttt{n}\%$ level due to a nearby error in the text. The fourth case `abb` is common and occurs when the parser does not know how to parse a correct grammatical construction. The last case `abc` may be caused by an error introduced near a correct grammatical construction that the parser cannot parse correctly. This case is uncommon.

Let $x_{aaa}, x_{aab}, x_{aba}, x_{abb}$ and x_{abc} correspond to the relative frequencies of the five cases. For example, if `abb` occupies 10% of the rows, $x_{abb} = 0.10$. Clearly,

$$x_{aaa} + x_{aab} + x_{aba} + x_{abb} + x_{abc} = 1, \tag{1}$$

since they cover all possible outcomes. Let acr_{m0} denote the accuracy when comparing the `m` file (treebank) to the `0` file (error-free text). We see that

$$acr_{m0} = x_{aaa} + x_{aab} \tag{2}$$

Table 1. An example of the different cases resulting from parsing a single word. Translation: Vi (We) kan (can) välja (choose) att (to) säga upp (cancel) avtalet (the agreement)

(treebank) word	manual annotation	(error-free text) word	parser output	(n% errors) word	parser output	case
Vi	NP begin	Vi	NP begin	Vi	NP begin	aaa
kan	VP begin	kan	VP begin	*kna*	VP begin	aaa
välja	VP end	välja	VP end	välja	**VP begin**	aab
att	NP(inf) begin	att	**Outside**	att	NP(inf) begin	aba
säga	VP begin in NP	säga	**VP begin**	säga	VP begin in NP	aba
upp	VP end in NP	upp	**VP end**	*upö*	**NP begin in NP**	abc
avtalet	NP begin in NP	avtalet	**NP begin**	avtalet	**NP begin**	abb

since only in cases aaa and aab, the two columns m and 0 contain the same output a. Furthermore, by the same reasoning,

$$acr_{mn} = x_{aaa} + x_{aba} \quad \text{and} \tag{3}$$

$$acr_{0n} = x_{aaa} + x_{abb}. \tag{4}$$

The x_{abb} is included in the last equality since 0 equals n in abb even though they both differ from m. The fact that they differ from the treebank cannot be established without the correct answer m.

We say that the performance of the NLP system *degrades* when the performance decreases with increasing levels of errors in the text. The degradation $degr_n$ is a comparison between the performance at the n% error level and the performance at the 0% error level. Let

$$degr_n = 1 - \frac{acr_{mn}}{acr_{m0}}. \tag{5}$$

Clearly, this is calculable only if you have access to acr_{mn} and acr_{m0}.

Normally, some sort of evaluation has been carried out to estimate the accuracy of the parser on error-free text, denoted *acr*. High accuracy is obtained when the correct answer m often corresponds to the output 0. Thus, the accuracy is a very good estimate for acr_{m0} and we will use $acr_{m0} = acr$. Nevertheless, without the annotated resource, we do not have access to or estimates for acr_{mn}.

2.2 Upper and Lower Bounds

We want to estimate the degradation $degr_n$ without knowing acr_{mn}. Without the annotated resource, we only have access to acr_{0n} and $acr_{m0} = acr$. We will use these to establish an upper bound $degr_n^{upr}$ for $degr_n$. We want the value $degr_n^{upr}$ to be an expression including *acr* and acr_{0n} that can be proven to be greater than $degr_n$.

We propose

$$degr_n^{upr} = \frac{1 - acr_{0n}}{acr} \tag{6}$$

as an upper bound. We prove that $degr_n^{upr}$ is always greater than $degr_n$ by letting

$$degr_n^{upr} = degr_n + \epsilon. \tag{7}$$

Equations (1)–(2) and (4)–(6) give us

$$\epsilon = \frac{2x_{aba} + x_{abc}}{acr}. \tag{8}$$

We see that $\epsilon \geq 0$ since all $x \geq 0$ and thus, $degr_n^{upr} \geq degr_n$ as required.

The smaller the value of ϵ, the better. From the discussion, we see that x_{aba} and x_{abc} are normally quite small, which is promising.

We now turn to a lower bound for $degr_n$. We propose

$$degr_n^{lwr} = \frac{1}{2} degr_n^{upr} = \frac{1 - acr_{0n}}{2acr}. \tag{9}$$

Again, as for the upper bound, the expression must be proven to be less than $degr_n$. To this end, we let

$$degr_n^{lwr} + \delta = degr_n. \tag{10}$$

From Equations (1)–(2), (4)–(5) and (9), we obtain

$$\delta = \frac{x_{aab} - 3x_{aba} - x_{abc}}{2acr}, \tag{11}$$

which is non-negative when $x_{aab} \geq 3x_{aba} + x_{abc}$.

Both cases `aab`, `aba` and `abc` are the result of an introduced spelling error. With no errors, x_{aab}, x_{aba} and x_{abc} are all zero and with increased levels of introduced errors, they will all increase. Hence, x_{aab}, x_{aba} and x_{abc} are positively correlated. Furthermore, it is clear that case `aab` is much more common than `aba` and `abc` since it involves correctly parsed text at the 0% error level. The accuracy *acr* determines the amount of correctly parsed text and thus, with reasonable accuracy, the above inequality holds with a good margin of error. See Appendix A for details on the conditions under which the above inequality holds. Section 3 further support that the inequality holds, since in all experiments the left-hand side is more than twice the right-hand side.

From the above discussion and given the conditions, we have obtained

$$degr_n^{lwr} \leq degr_n \leq degr_n^{upr}. \tag{12}$$

2.3 Estimation of the Degradation

The simple relationship between the upper and lower bounds allows us to deduce some further information. Given an upper bound $degr_n^{upr}$ and a lower bound $degr_n^{lwr}$, we want to estimate the position of the true value $degr_n$. Clearly, $degr_n$ is somewhere in between $degr_n^{lwr}$ and $degr_n^{upr}$ from Equation (12). Let $degr_n^{est}$ be the center of the interval contained by the lower and upper bound, that is,

$$degr_n^{est} = \frac{1}{2}(degr_n^{lwr} + degr_n^{upr}) \tag{13}$$

and let γ be the distance from $degr_n$ to $degr_n^{est}$. Then,

$$degr_n + \gamma = degr_n^{est}. \tag{14}$$

Equations (7), (10) and (13) yield $\gamma = (\epsilon - \delta)/2$. Using Equations (8) and (11) results in the explicit form

$$\gamma = \frac{7x_{aba} + 3x_{abc} - x_{aab}}{4acr}. \tag{15}$$

We see that γ is small if $7x_{aba} + 3x_{abc} \approx x_{aab}$.

As the discussion above about the lower bound illustrated, x_{aab}, x_{aba} and x_{abc} are correlated. See Appendix A for a discussion on the conditions required to make γ small. Though the experiments in Section 3 show that γ is quite small, we make no claims that γ is equally small for all NLP systems. The estimations here are just theoretical indications where the true value of $degr_n$ may reside.

We have indicated that $degr_n^{est}$ is, in theory, close to $degr_n$. By using Equations (6) and (9), we simplify and obtain an explicit formula for the estimated degradation:

$$degr_n^{est} = \frac{3}{4} degr_n^{upr} = \frac{3(1 - acr_{0n})}{4acr}. \tag{16}$$

Hence, without having an annotated resource, we can estimate the robustness (degradation) of the system quite accurately.

2.4 Accuracy

Now that the degradation of the performance has been established, we turn to the accuracy. The definition of $degr_n$ in Equation (5) states that $degr_n = 1 - acr_{mn}/acr$. We are interested in the accuracy of the NLP system on the $\mathrm{n}\%$ file, that is, acr_{mn}. Rearranging the above equation yields

$$acr_{mn} = acr(1 - degr_n). \tag{17}$$

Since $degr_n$ is unknown, we use $degr_n^{upr}$, $degr_n^{lwr}$ and $degr_n^{est}$ to obtain bounds on the accuracy:

$$acr_{mn}^{lwr} = acr(1 - degr_n^{upr}), \tag{18}$$

$$acr_{mn}^{upr} = acr(1 - degr_n^{lwr}), \tag{19}$$

$$acr_{mn}^{est} = acr(1 - degr_n^{est}). \tag{20}$$

The estimation in Equation (20) is not precise, so we let

$$acr_{mn} + \lambda = acr_{mn}^{est}. \tag{21}$$

From Equations (14), (17) and (20), we obtain

$$\lambda = acr \cdot (-\gamma). \tag{22}$$

Thus, if $|\gamma|$ is small, $|\lambda|$ is even smaller, and thus, acr_{mn}^{est} is a good approximation of the accuracy of the NLP system when applied to a file containing $\mathrm{n}\%$ errors.

3 Empirical Results

Five different parsers were used to assess the accuracy of the evaluation method.

GTA [9] is a rule-based shallow parser. It relies on hand-crafted rules of which a few are context-sensitive. The rules are applied to part-of-speech tagged text. GTA identifies constituents and assigns phrase labels but does not build full trees with a top node.

FDG [10], Functional Dependency Grammar, is a commercial dependency parser. It builds a connected tree structure, where every word points at a dominating word. Dependency links are assigned a function label. FDG produces other information too, such as morphological analysis and lemma of words, which is not used here.

A dependency parser by Nivre [11] uses a manually constructed grammar and assigns dependency links between words, working from part-of-speech tagged text. We denoted it the MCD parser (manually constructed dependency).

The Malt parser [12], another dependency parser, is based on the same algorithm as MCD but uses a memory-based classifier trained on a treebank instead of a manually constructed grammar. Unlike MCD, the Malt parser not only assigns dependency links between words but also attaches function labels to these links.

A manually constructed context-free grammar for Swedish was used with an implementation of Earley's parsing algorithm, as described in [13]. We denoted it the Earley parser.

3.1 Parser Robustness Evaluation

In the evaluation, we used 100 000 words from the Stockholm-Umeå Corpus (SUC) [14]. The SUC is a balanced collection of written Swedish, well proofread. The SUC is annotated with part-of-speech information. It does not contain any parse annotation.

The 100 000 word text was parsed using each of the parsers. The parse results from this error-free text (0% errors) constituted the `0` file, as defined in the first part of Section 2. Spelling errors (resulting in non-existing words only) were randomly inserted into the text, using a tool that emulates errors produced by a human, as described in Section 2. The parse results from the misspelled text (containing e.g. 5% errors) constituted the `n` file, also from Section 2. For the GTA, the MCD and the Malt parser, manually annotated resources were available. The experiments on these are reported in the next section.

To see how the parser behaves with increasing amounts of errors, $\mathtt{n} = 1\%$, 2%, 5%, 10% and 20% of all words were randomly misspelled. To reduce the influence of chance, 10 different misspelled files were created for each error level. Using these, we calculated the mean for the degradation, the accuracy and so forth. The variance between different files was low. To simplify the evaluation, a freeware program called AutoEval [8] was used for input and output handling and data processing.

The degradation estimates for a particular file were obtained by calculating acr_{0n}, that is, by comparing how many of the parses in the `0` file that corresponded to the parses in the `n` file. From acr_{0n} we calculated the upper and lower bounds as well as estimates on the degradation and accuracy, as seen in Section 2.

The results for the five parsers are presented in Tables 2 through 6, which also present the accuracy *acr* on error-free text. The first column reports on the amount of errors in the text. The second is the amount of parse output that differs between the rows of the `0` file and the `n` file. This value is $1 - acr_{0n}$. The third column presents the degradation of the parser. The first value is the lower bound $degr_n^{lwr}$ and the second is the upper bound $degr_n^{upr}$. The figure in parentheses is the estimated degradation $degr_n^{est}$. The fourth column contains the estimations on the accuracy: lower bound acr_{mn}^{lwr}, upper bound acr_{mn}^{upr} and estimated value acr_{mn}^{est}.

Table 2. Estimated robustness of the GTA parser on 100 000 words. All figures are given in per cent. Estimated accuracy on error-free text was 89%

Error level	Output differs	Estimated degradation	Estimated accuracy
1	1.2	0.7 - 1.3 (1.0)	88 - 88 (88)
2	2.4	1.3 - 2.6 (2.0)	87 - 88 (87)
5	5.7	3.2 - 6.4 (4.8)	83 - 86 (85)
10	11	6.2 - 12 (9.4)	78 - 83 (81)
20	21	12 - 24 (18)	68 - 78 (73)

Table 3. Estimated robustness of the MCD parser on 100 000 words. Estimated accuracy on error-free text was 82%

Error level	Output differs	Estimated degradation	Estimated accuracy
1	0.9	0.5 - 1.1 (0.8)	81 - 82 (82)
2	1.7	1.1 - 2.1 (1.6)	81 - 81 (81)
5	4.3	2.6 - 5.3 (4.0)	78 - 80 (79)
10	8.6	5.2 - 10 (7.8)	74 - 78 (76)
20	17	10 - 20 (15)	66 - 74 (72)

Table 4. Estimated robustness of the Malt parser on 100 000 words. Estimated accuracy on error-free text was 79%

Error level	Output differs	Estimated degradation	Estimated accuracy
1	1.8	1.2 - 2.4 (1.8)	77 - 78 (77)
2	3.7	2.3 - 4.7 (3.5)	75 - 77 (76)
5	8.9	5.7 - 11 (8.5)	70 - 74 (72)
10	17	11 - 22 (16)	61 - 70 (66)
20	31	20 - 39 (29)	48 - 63 (55)

Table 5. Estimated robustness of the Earley parser on 100 000 words. Estimated accuracy on error-free text was 90%

Error level	Output differs	Estimated degradation	Estimated accuracy
1	0.8	0.5 - 0.9 (0.7)	89 - 90 (89)
2	1.7	0.9 - 1.8 (1.4)	88 - 89 (89)
5	4.1	2.3 - 4.5 (3.4)	86 - 88 (87)
10	8.2	4.5 - 9.1 (6.8)	82 - 86 (84)
20	16	9.1 - 18 (14)	74 - 82 (78)

The proposed method evaluates the robustness on one row at the time. For example, if the first column says 5%, we have introduced errors in 5% of the words (with one word per row). Similarly, if we report 11% in the second column (parse differs), then 11% of the parse output (with one parse per row) is different between the two files.

In the experiments, any deviation from the correct parse was considered an error, even if it was "almost" correct (though the evaluation method could just as easily use a more sophisticated analysis). Hence, parsers that provide richer information will generally be

Table 6. Estimated robustness of the FDG parser on 100 000 words. Estimated accuracy on error-free text was 90%

Error level	Output differs	Estimated degradation	Estimated accuracy
1	2.1	1.2 - 2.3 (1.7)	88 - 89 (88)
2	4.2	2.3 - 4.6 (3.5)	86 - 88 (87)
5	10	5.5 - 11 (8.3)	80 - 85 (83)
10	19	11 - 21 (16)	71 - 81 (76)
20	34	19 - 37 (28)	56 - 73 (65)

Table 7. Estimated robustness of the PoS tagger TnT on 100 000 words. All figures are given in per cent. Estimated accuracy on error-free text was 96%

Error level	Output differs	Estimated degradation	Estimated accuracy
1	0.7	0.4 - 0.7 (0.6)	95 - 96 (95)
2	1.4	0.7 - 1.5 (1.1)	95 - 95 (95)
5	3.6	1.9 - 3.7 (2.8)	92 - 94 (93)
10	7.2	3.7 - 7.5 (5.6)	89 - 92 (91)
20	14	7.5 - 15 (11)	82 - 89 (85)

less robust than parsers that return less information, since there are more possibilities for errors.

Parsers base much of their decisions on the part-of-speech information assigned to a word. Since part-of-speech taggers often guess the correct tag for regularly inflected unknown words, the part-of-speech tagger is responsible for a large part of the robustness. In Table 7, the estimated degradation of the part-of-speech (PoS) tagger TnT [15] is shown. TnT was used for all parsers but FDG, which includes its own tagger.

Comparing the output of FDG on different versions of the same text is non-trivial, since the tokenization may be altered by a misspelled word. Here, any tokens without a directly corresponding token in the other text were ignored. All other tokenization difficulties were interpreted to give FDG as many "correct" parses as possible. The 90% accuracy for FDG is our estimation. Malt and MCD are similar in their construction but their results are not really comparable since Malt assigns function labels and MCD does not. On unlabeled output, Malt is more accurate than MCD.

3.2 Evaluating the Evaluation Method

Text with correctly annotated parse output was available for some of the parsers, though only in small amounts. By using these, we wanted to assess the accuracy of the proposed method.

For the GTA parser and the TnT part-of-speech tagger, we had a 14 000 word file of manually corrected parse and tag data. For the MCD parser, we had a 4 000 word file and for Malt we had 10 000 words. We used the text from these files and carried out the same procedure as in the previous subsection, that is, introduced errors and evaluated. We also had the correct answers from the annotated resource. From this, we calculated the real degradation and accuracy.

Table 8. Estimated and actual robustness of the GTA parser on 14 000 words of manually annotated text. All figures are given in per cent. Estimated parser accuracy on error-free text was 89%

Error level	Output differs	Estimated degradation	**Real degr.**	Estimated accuracy	**Real accur.**
1	1.2	0.7 - 1.4 (**1.0**)	**0.9**	88 - 88 (**88**)	**88**
2	2.3	1.3 - 2.6 (**1.9**)	**1.8**	87 - 88 (**87**)	**87**
5	5.1	2.9 - 5.7 (**4.3**)	**4.2**	84 - 86 (**85**)	**85**
10	9.9	5.5 - 11 (**8.3**)	**8.1**	79 - 84 (**81**)	**82**
20	19	10 - 21 (**16**)	**16**	70 - 80 (**75**)	**75**

Table 9. Estimated and actual robustness of the MCD parser on 4 000 words of manually annotated text. Estimated parser accuracy on error-free text was 82%

Error level	Output differs	Estimated degradation	**Real degr.**	Estimated accuracy	**Real accur.**
1	0.7	0.4 - 0.8 (**0.6**)	**0.6**	82 - 82 (**82**)	**82**
2	1.7	1.0 - 2.0 (**1.5**)	**1.4**	81 - 82 (**81**)	**81**
5	4.0	2.5 - 4.9 (**3.7**)	**3.2**	78 - 80 (**79**)	**80**
10	8.3	5.0 - 10 (**7.6**)	**6.6**	74 - 78 (**76**)	**77**
20	16	9.6 - 19 (**14**)	**13**	67 - 74 (**71**)	**72**

Table 10. Estimated and actual robustness of the Malt parser on 10 000 words of manually annotated text. Estimated parser accuracy on error-free text was 79%

Error level	Output differs	Estimated degradation	**Real degr.**	Estimated accuracy	**Real accur.**
1	1.8	1.1 - 2.3 (**1.7**)	**1.3**	77 - 78 (**77**)	**78**
2	3.4	2.2 - 4.3 (**3.2**)	**2.4**	75 - 77 (**76**)	**77**
5	8.7	5.5 - 11 (**8.3**)	**6.1**	70 - 74 (**72**)	**74**
10	16	11 - 21 (**16**)	**12**	62 - 70 (**66**)	**69**
20	30	19 - 38 (**29**)	**23**	48 - 64 (**56**)	**60**

Table 11. Estimated and actual robustness of the TnT part-of-speech tagger on 14 000 words of manually annotated text. Estimated tagger accuracy on error-free text was 96%

Error level	Output differs	Estimated degradation	**Real degr.**	Estimated accuracy	**Real accur.**
1	1.1	0.6 - 1.1 (**0.9**)	**0.9**	95 - 95 (**95**)	**95**
2	1.9	1.0 - 2.0 (**1.5**)	**1.6**	94 - 95 (**94**)	**94**
5	3.9	2.0 - 4.1 (**3.1**)	**3.6**	92 - 94 (**93**)	**92**
10	7.3	3.8 - 7.6 (**5.7**)	**6.7**	88 - 92 (**90**)	**89**
20	14	7.4 - 15 (**11**)	**13**	82 - 89 (**85**)	**83**

The results are provided in Tables 8 through 11. As guaranteed by the proposed method, the real degradation and accuracy are always between the lower and upper bound. We see that the estimated degradation and accuracy are close or equal to the real degradation and accuracy, as indicated in the discussion about γ in Section 2.3 and λ in Section 2.4. Hence, there is strong reason to believe that the estimations on the 100 000 word files in Section 3.1 are also accurate. Furthermore, by using the results from a small

annotated resource (if available), we obtain a good estimate on the relation γ between the real and the estimated degradation for the 100 000 file.

We note that rich information is a liability for at least two of the parsers, FDG and Malt. Thus, comparing the robustness figures between two parsers is not entirely fair. Nevertheless, if the objective is reluctancy to change the output when facing unrestricted and noisy text, the figures are accurate.

We also note that the proposed method could easily be adapted to other types of output besides the row-based used here. This might require a small adjustment of the estimations in the theory section.

4 Conclusions

We have presented a method to estimate the robustness of an NLP system. The method provides lower and upper bounds as well as estimates on the actual robustness. The main strength of the evaluation is that neither manual work nor annotated resources are required. The only requirements are an arbitrary (unannotated) text and an estimate of the accuracy of the parser on error-free text. Thus, we have eliminated the need for expensive and time-consuming manual labor.

The proposed method is applicable to any language and most annotation schemes and NLP systems. Even though spelling errors have been used here as an example in the presentation of the method, any error type can be used to simulate noise. Using annotated resources, we have assessed the reliability of the unsupervised evaluation and found that the estimates were quite accurate. We conclude that the proposed method is a reliable and highly timesaving tool for the evaluation of NLP system robustness.

A Conditions

We want to determine the circumstances under which the restriction on δ holds, that is, when

$$\delta = \frac{x_{aab} - 3x_{aba} - x_{abc}}{2acr} \geq 0, \tag{23}$$

as discussed in Section 2.2. Furthermore, we will establish the requirements for γ to be small, i.e. when

$$\gamma = \frac{7x_{aba} + 3x_{abc} - x_{aab}}{4acr} \approx 0. \tag{24}$$

A few assumptions are required. We know from Equations (1) and (4) that

$$x_{aab} + x_{aba} + x_{abc} = 1 - acr_{0n}. \tag{25}$$

We are interested in an approximation of x_{aab}. We will assume that $x_{aab}/(1 - acr_{0n}) = acr$. That is, we assume that x_{aab} compared to the three cases $x_{aab} + x_{aba} + x_{abc}$ is about the same as the accuracy *acr* compared to one (the sum of all cases). The reader should take a moment to recognize that this is not an unreasonable estimation. We rearrange the above approximation and obtain

$$x_{aab} = acr(1 - acr_{0n}). \tag{26}$$

From this and Equation (25), we get

$$x_{aba} + x_{abc} = (1 - acr)(1 - acr_{0n}). \tag{27}$$

Our second assumption is that

$$x_{aba} \leq x_{abc}. \tag{28}$$

The two cases `aba` and `abc` originate from a grammatical construct that could not be parsed by the system. When an error is introduced, the parser changes its output. The most probable is that the change results in something erroneous, as in `abc`.

We use the assumptions with δ in Equation (23):

$$\begin{gathered}\delta = (x_{aab} - 3x_{aba} - x_{abc})/2acr \geq \\ (x_{aab} - 2(x_{aba} + x_{abc}))/2acr \geq 0 \\ \iff acr - 2(1 - acr) \geq 0.\end{gathered}$$

Hence, the inequality in Equation (23) is satisfied if $acr \geq 2/3$. If we have an accuracy of more than 67%, the lower bound for the degradation is valid.

We repeat the above process with γ in Equation (24) and obtain

$$\begin{gathered}\gamma = (7x_{aba} + 3x_{abc} - x_{aab})/4acr \leq \\ (5(x_{aba} + x_{abc}) - x_{aab})/4acr \leq 0 \\ \iff 5(1 - acr) - acr \leq 0.\end{gathered}$$

Hence, γ in Equation (24) is negative if $acr \geq 5/6 = 83.3\%$. On the other hand,

$$\begin{gathered}\gamma = (7x_{aba} + 3x_{abc} - x_{aab})/4acr \geq \\ (3(x_{aba} + x_{abc}) - x_{aab})/4acr \geq 0 \\ \iff 3(1 - acr) - acr \geq 0.\end{gathered}$$

Now, γ is positive if $acr \leq 3/4 = 75\%$. Thus, for parsers with reasonable accuracy, γ will be small and the approximation of the degradation will be accurate.

References

1. Carroll, J., Briscoe, T., Sanfilippo, A.: Parser evaluation: a survey and a new proposal. In: Proceedings of LREC 1998, Granada, Spain (1998) 447–454
2. Hogenhout, W.I., Matsumoto, Y.: Towards a more careful evaluation of broad coverage parsing systems. In: Proceedings of Coling 1996, San Francisco, USA (1996) 562–567
3. Menzel, W.: Robust processing of natural language. In: Proceedings of 19th Annual German Conference on Artificial Intelligence, Berlin, Germany (1995) 19–34
4. Bigert, J., Knutsson, O., Sjöbergh, J.: Automatic evaluation of robustness and degradation in tagging and parsing. In: Proceedings of RANLP 2003, Bovorets, Bulgaria (2003)
5. Vilares, M., Darriba, V.M., Vilares, J., Rodriguez, R.: Robust parsing using dynamic programming. Lecture Notes in Computer Science **2759** (2003) 258–267
6. Foster, J.: Parsing ungrammatical input: An evaluation procedure. In: Proceedings of LREC 2004, Lisbon, Portugal (2004) 2039–2042
7. Li, X., Roth, D.: Exploring evidence for shallow parsing. In Daelemans, W., Zajac, R., eds.: Proceedings of CoNLL 2001, Toulouse, France (2001) 38–44
8. Bigert, J., Ericson, L., Solis, A.: Missplel and AutoEval: Two generic tools for automatic evaluation. In: Proceedings of Nodalida 2003, Reykjavik, Iceland (2003)

9. Knutsson, O., Bigert, J., Kann, V.: A robust shallow parser for Swedish. In: Proceedings of Nodalida 2003, Reykjavik, Iceland (2003)
10. Voutilainen, A.: Parsing Swedish. In: Proceedings of Nodalida 2001, Uppsala, Sweden (2001)
11. Nivre, J.: An efficient algorithm for projective dependency parsing. In: Proceedings of IWPT 2003, Nancy, France (2003) 149–160
12. Nivre, J., Hall, J., Nilsson, J.: Memory-based dependency parsing. In: Proceedings of CoNLL, Boston, MA (2004)
13. Megyesi, B.: Data-Driven Syntactic Analysis – Methods and Applications for Swedish. PhD thesis, KTH, Stockholm, Sweden (2002)
14. Ejerhed, E., Källgren, G., Wennstedt, O., Åström, M.: The Linguistic Annotation System of the Stockholm-Umeå Project. Department of Linguistics, University of Umeå, Sweden (1992)
15. Brants, T.: TnT – a statistical part-of-speech tagger. In: Proceedings of ANLP 2000, Seattle, USA (2000)

Mutual Information Independence Model Using Kernel Density Estimation for Segmenting and Labeling Sequential Data

Guodong Zhou, Lingpeng Yang, Jian Su, and Donghong Ji

Institute for Infocomm Research, 21 Heng Mui Keng Terrace, Singapore 119613
{zhougd, lpyang, sujian, dhji}@i2r.a-star.edu.sg

Abstract. This paper proposes a Mutual Information Independence Model (MIIM) to segment and label sequential data. MIIM overcomes the strong context independent assumption in traditional generative HMMs by assuming a novel pairwise mutual information independence. As a result, MIIM separately models the long state dependence in its state transition model in a generative way and the observation dependence in its output model in a discriminative way. In addition, a variable-length pairwise mutual information-based modeling approach and a kNN algorithm using kernel density estimation are proposed to capture the long state dependence and the observation dependence respectively. The evaluation on shallow parsing shows that MIIM can effectively capture the long context dependence to segment and label sequential data. It is interesting to note that using kernel density estimation leads to increased performance over using a classifier-based approach.

1 Introduction

A Hidden Markov Model (HMM) is a model where a sequence of observations is generated in addition to the Markov state sequence. It is a latent variable model in the sense that only the observation sequence is known while the state sequence remains "hidden". In recent years, HMMs have enjoyed great success in many tagging applications, most notably part-of-speech (POS) tagging [1,2,3] and named entity recognition [4,5]. Moreover, there have been also efforts to extend the use of HMMs to word sense disambiguation [6] and shallow/full parsing [7,8,9].

Given an observation sequence $O_1^n = o_1 o_2 \cdots o_n$, the goal of a HMM is to find a stochastic optimal state sequence $S_1^n = s_1 s_2 \cdots s_n$ that maximizes $P(S_1^n, O_1^n)$:

$$S^* = \arg\max_{S_1^n} \log P(S_1^n, O_1^n) = \arg\max_{S_1^n} \{\log P(S_1^n) + \log P(O_1^n \mid S_1^n) \quad (1)$$

Traditionally, HMM segments and labels sequential data in a generative way by making a context independent assumption that successive observations are independent given the corresponding individual state [10]:

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 155 – 166, 2005.

$$P(O_1^n \mid S_1^n) = \prod_{i=1}^{n} P(o_i \mid s_i) \tag{2}$$

By applying the assumption (2) and using the chain rule, equation (1) can be rewritten as:

$$\begin{aligned} S^* &= \arg\max_{S_1^n} \{\log P(S_1^n) + \sum_{i=1}^{n} \log P(o_i \mid s_i)\} \\ &= \arg\max_{S_1^n} \{\sum_{i=2}^{n} \log P(s_i \mid S_1^{i-1}) + \log P(s_1) + \sum_{i=1}^{n} \log P(o_i \mid s_i)\} \end{aligned} \tag{3}$$

More formally, a generative (first-order) HMM is given by a finite set of states S including an designated initial state and an designated final state, a set of possible observation O, two conditional probability distributions: a state transition model $P(s \mid s')$ from s' to s for $s', s \in S$ and an output model $P(o \mid s)$ for $o \in O, s \in S$. A sequence of observations is generated by starting from the designated initial state, transmiting to a new state according to $P(s \mid s')$, emitting an observation selected by that new state according to $P(o \mid s)$, transmiting to another new state and so on until the designated final state is generated.

There are several problems with this generative approach. First, many tasks would benefit from a richer representation of observations—in particular a representation that describes observations in terms of many overlapping features, such as capitalization, word endings, part-of-speech in addition to the traditional word identity. Note that these features always depends on each other. Furthermore, to define a joint probability over the observation and state sequences, the generative approach needs to enumerate all the possible observation sequences. However, in some tasks, the set of all the possible observation sequences is not reasonably enumerable. Second, the generative approach fails to effectively model the dependence in the observation sequence. Third, the generative approach normally estimates the parameters to maximize the likelihood of the observation sequence. However, in many NLP tasks, the goal is to predict the state sequence given the observation sequence. In other words, the generative approach inappropriately applies a generative joint probability model for a conditional probability problem. In summary, the main reasons behind these problems of the generative approach are the strong context independent assumption and the generative nature in modeling sequential data. While the dependence between successive states can be directly modeled by its state transition model, the generative approach fails to directly capture the observation dependence in the output model.

To resolve the inherent problems in generative HMMs, some researches (please see related works in Section 6 for details) have been done to move from generative HMMs to discriminative Markov models (DMMs). DMMs do not expend modeling effort on the observation sequnce, which are fixed at test time. Instead, DMMs model the state sequence depending on arbitrary, non-independent features of the observation sequence, normally without forcing the model to account for the distribution of those dependencies.

This paper proposes a Mutual Information Independence Model (MIIM), which separates the dependence of a state on the previous states and the observation sequence. Compared with generative HMMs, MIIM explicitly models the long state dependence in a generative way and the observation dependence in a discriminative way. In addition, a variable-length pairwise mutual information based modeling is proposed to capture the long state dependence of a state on the previous states while a kNN algorithm using kernel density estimation is proposed to capture the observation dependence of a state on the observation sequence.

The layout of this paper is as follows. Section 2 proposes the Mutual Information Independence Model (MIIM) and presents the variable-length pair-wise mutual information-based modeling approach to capture the long state dependence. Section 3 presents the kNN algorithm using kernel density estimation to capture the observation dependence. Section 4 introduces the shallow parsing task while Section 5 gives experimental results. Section 6 describes some of the related works in discriminative Markov modeling. Finally, some conclusion will be drawn in Section 7.

2 Mutual Information Independence Model

In principle, given an observation sequence $o_1^n = o_1 o_2 \cdots o_n$, the goal of a conditional probability model is to find a stochastic optimal state sequence $s_1^n = s_1 s_2 \cdots s_n$ that maximizes $\log P(s_1^n \mid o_1^n)$

$$s^* = \arg\max_{s_1^n} \{\log P(s_1^n \mid o_1^n)\} = \arg\max_{s_1^n} \{\log P(s_1^n) + \log \frac{P(s_1^n, o_1^n)}{P(s_1^n) \cdot P(o_1^n)}\} \quad (4)$$

Obviously, the second term $\log \frac{P(s_1^n, o_1^n)}{P(s_1^n) \cdot P(o_1^n)}$ captures the pairwise mutual information (PMI) between the state sequence s_1^n and the observation sequence o_1^n. Here, we define $PMI(s_1^n, o_1^n) = \log \frac{P(s_1^n, o_1^n)}{P(s_1^n) \cdot P(o_1^n)}$. To compute $PMI(s_1^n, o_1^n)$ efficiently, we propose a novel pairwise mutual information independence assumption:

$$PMI(s_1^n, o_1^n) = \sum_{i=1}^{n} PMI(s_i, o_1^n) \text{ or } \log \frac{P(s_1^n, o_1^n)}{P(s_1^n) \cdot P(o_1^n)} = \sum_{i=1}^{n} \log \frac{P(s_i, o_1^n)}{P(s_i) \cdot P(o_1^n)} \quad (5)$$

That is, we assume a state is only dependent on the observation sequence o_1^n and independent on other states in the state sequence s_1^n. This assumption is reasonable because the dependence among the states in the state sequence s_1^n has been directly captured by the first term $\log P(s_1^n)$ in equation (4).

By applying the assumption (5) into the equation (4), we have:

$$
\begin{aligned}
s^* &= \arg\max_{s_1^n}\{\log P(s_1^n) + \sum_{i=1}^{n} \log \frac{P(s_i, o_1^n)}{P(s_i)\cdot P(o_1^n)} \\
&= \arg\max_{s_1^n}\{\log P(s_1^n) - \sum_{i=1}^{n} \log P(s_i) + \sum_{i=1}^{n} \log \frac{P(s_i, o_1^n)}{P(o_1^n)}\} \\
&= \arg\max_{s_1^n}\{\sum_{i=2}^{n} \log P(s_i \mid s_1^{i-1}) + \log P(s_1) - \sum_{i=1}^{n} \log P(s_i) + \sum_{i=1}^{n} \log P(s_i \mid o_1^n)\} \qquad (6) \\
&= \arg\max_{s_1^n}\{\sum_{i=2}^{n} \log P(s_i \mid s_1^{i-1}) - \sum_{i=2}^{n} \log P(s_i) + \sum_{i=1}^{n} \log P(s_i \mid o_1^n)\} \\
&= \arg\max_{s_1^n}\{\sum_{i=2}^{n} PMI(s_i, s_1^{i-1}) + \sum_{i=1}^{n} \log P(s_i \mid o_1^n)\}
\end{aligned}
$$

The above model consists of two models: the state transition model $\sum_{i=2}^{n} PMI(s_i, s_1^{i-1})$ which measures the state dependence of a state given the previous states in a generative way, and the output model $\sum_{i=1}^{n} \log P(s_i \mid o_1^n)$ which measures the observation dependence of a state given the observation sequence in a discriminative way. This is done by assuming a novel pair-wise mutual information independence model. Therefore, we call the above model as in equation (6) a Mutual Information Independence Model (MIIM). The main difference between a generative HMM and a MIIM lies in their output models in that the output model of a MIIM directly captures the context dependence between successive observations in determining the "hidden" states while the output model of the generative HMM fails to do so. That is, the output model of a MIIM overcomes the strong context independent assumption in the generative HMM and becomes observation context dependent. Alternatively, we can have equation (7) by rewriting equation (3) using the Bayes's rule:

$$
\begin{aligned}
S^* &== \arg\max_{S_1^n}\{\sum_{i=2}^{n} \log P(s_i \mid S_1^{i-1}) + \log P(s_1) + \sum_{i=1}^{n} \log P(o_i \mid s_i)\} \\
&= \arg\max_{S_1^n}\{\sum_{i=2}^{n} \log P(s_i \mid S_1^{i-1}) + \log P(s_1) \\
&+ \sum_{i=1}^{n} \{\log P(s_i \mid o_i) + \log P(o_i) - \log P(s_i)\} \\
&= \arg\max_{S_1^n}\{\sum_{i=2}^{n} \log P(s_i \mid S_1^{i-1}) + \log P(s_1) + \sum_{i=1}^{n} \{\log P(s_i \mid o_i) - \log P(s_i)\} \qquad (7) \\
&= \arg\max_{S_1^n}\{\sum_{i=2}^{n} \{\log P(s_i \mid S_1^{i-1}) - \log P(s_i)\} + \sum_{i=1}^{n} \log P(s_i \mid o_i)\} \\
&= \arg\max_{S_1^n}\{\sum_{i=2}^{n} PMI(s_i, S_1^{i-1}) + \sum_{i=1}^{n} \log P(s_i \mid o_i)\}
\end{aligned}
$$

Compared with MIIM as equation (6) and the generative HMM rewritten as equation (7), we can see that MIIM extends the notion of an observation and estimates the output model based on the observation sequence rather than the corresponding individual observation.

Computation of a MIIM consists of two parts. The first is to compute the state transition model: $\sum_{i=2}^{n} PMI(s_i, s_1^{i-1})$. Traditionally, ngram modeling(e.g. bigram for the first-order HMM and trigram for the second-order HMM) is used to estimate the state transition model. However, such approach fails to capture the long state dependence since it is not reasonably practical for ngram modeling to be beyond trigram. In this paper, a variable-length pairwise mutual information-based modeling approach is proposed as follow: For each $i(2 \le i \le n)$, we first find a minimal $k(0 \le k \prec i)$ where the frequency of s_k^i is not smaller than a threshold (e.g. 3) and then estimate $PMI(s_i, s_1^{i-1})$ using $PMI(s_i, s_k^{i-1}) = \log \frac{P(s_k^i)}{P(s_i)P(s_k^{i-1})}$. In this way, the long state dependence can be captured in a dynamical way. Here, the frequencies of variable-length state sequences are smoothed using the simple Good-Turing approach [11].

The second is to estimate the output model: $\sum_{i=1}^{n} \log P(s_i \mid o_1^n)$. Ideally, we would have sufficient training data for every event whose conditional probability we wish to calculate. Unfortunately, there is rarely enough training data to compute accurate probabilities when decoding on new data. Traditionally, there are two existing approaches to resolve this problem: linear interpolation [12] and back-off [13]. However, these two approaches only work well when the number of different information sources is limited. When a long context is considered, the number of different information sources is exponential and not reasonably enumerable. The current trend is to recast it as a classification problem and use the output of a classifier, e.g. the maximum entropy classifier (ME) [14] to estimate the state probability distribution given the observation sequence. In the next section, we will propose a more effective kNN algorithm using kernel density estimation to resolve this problem.

3 *kNN* Using Kernel Density Estimation

The main challenge for the above MIIM is how to reliably estimate $P(s_i \mid o_1^n)$ in its output model. For efficiency, we can always assume $P(s_i \mid o_1^n) \approx P(s_i \mid E_i)$, where the pattern entry $E_i = o_{i-N} \cdots o_i \cdots o_{i+N}$. That is, we only consider the observation dependence in a window of 2N+1 observations (e.g. we only consider the current observation, the previous observation and the next observation when N=1). For convenience, we denote $P(\bullet \mid E_i)$ as the conditional state probability distribution of the states given E_i and $P(s_i \mid E_i)$ as the conditional state probability of s_i given E_i.

The kNN algorithm calculates $P(\bullet \mid E_i)$ by first finding the K nearest neighbors of frequently occurring pattern entries $kNN(E_i) = \{E_i^k \mid k = 1,2,..., K\}$ and then aggregating them to make a proper estimation of $P(\bullet \mid E_i)$ using kernel density estimation. Here, the conditional state probability distribution is estimated instead of the classification in a traditional kNN classifier.

3.1 Finding K Nearest Neighbors

To do so, all the frequently occurring pattern entries are extracted exhaustively from the training corpus and stored in a dictionary *FrequentEntryDictionary*. Here, the dictionary *FrequentEntryDictionary* is indexed using the tree-based indexing scheme as in TiMBL[1] [15]. In order to limit the dictionary size and keep efficiency, we constrain a valid set of pattern entry forms *ValidEntryForm* to consider only the most informative information sources. Obviously, *ValidEntryForm* defines the possible feature conjunctions. Generally, *ValidEntryForm* can be determined manually or automatically according to the applications. In Section 5, we will give an example.

Given a pattern entry E_i and the indexed dictionary *FrequentEntryDictionary*, the K nearest neighbors of the pattern entry E_i is found as follows:

- Extract all the compatible entries with E_i from the indexed dictionary
- Compute the similarity between E_i and each of the compatible entries using a kernel function
- Sort out the K nearest neighbors according to their similarities

Here, the kernel function $k(E_i^k, E_i)$ between E_i and E_i^k is determined by their shared feature conjunctions:

$$k(E_i^k, E_i) = \sum_{f^j \in E_i, f^j \in E_i^k} w^j \tag{8}$$

with the parameter vector $\boldsymbol{w}$. Here, w^j is the weight for the j-th possible feature conjunction $f^j \in ValidEntryForm$. Here, the parameter vector $\boldsymbol{w}$ is determined as follows: For each entry in *FrequentEntryDictionary*, we first find the 2*K nearest neighbors from the indexed dictionary using the above same algorithm (Here, all the w^j in $\boldsymbol{w}$ is initialized to 1 divided by $|ValidEntryForm|$, the number of possible feature conjunctions). For each possible feature conjunction, we calculate its weight (averaged over all the entries in *FrequentEntryDictionary*) as its non-occurrence frequency in the second K nearest neighbors divided its occurrence frequency in the first K neighbors. The intuition is that, if a feature conjunction occurs more in the first K nearest neighbors, and less in the second K nearest neighbors, such feature

[1] http://ilk.kub.nl/

conjunction contributes more. Finally, $\boldsymbol{w}$ is normalized ($\sum_{all\ f^j \in ValidEntryForm} w^j = 1$). The above training algorithm is repeated until $\boldsymbol{w}$ becomes stable (e.g. the change in $|\boldsymbol{w}|$ is less than 1%).

3.2 Kernel Density Estimation

After the K nearest neighbors have been found, the conditional state probability distribution $P(\bullet \mid E_i)$ of the pattern entry E_i is calculated using kernel density estimation. That is, $P(\bullet \mid E_i)$ is estimated by the weighted average of its K nearest neighbors:

$$P(\bullet \mid E_i) = \frac{\sum_{k=1}^{K} k(E_i^k, E_i) \cdot F(E_i^k) \cdot P(\bullet \mid E_i^k)}{\sum_{k=1}^{K} k(E_i^k, E_i) \cdot F(E_i^k)} \tag{9}$$

where the kernel function $k(E_i^k, E_i)$ measures the similarity between the pattern entry E_i and its nearest neighbor E_i^k and $F(E_i^k)$ is the occurring frequency of E_i^k.

4 Shallow Parsing

In order to evaluate the MIIM, we have applied it in the application of shallow parsing.

For shallow parsing, we have $o_1 = p_i w_i$, where $w_1^n = w_1 w_2 \cdots w_n$ is the word sequence and $p_1^n = p_1 p_2 \cdots p_n$ is the part-of-speech (POS) sequence, while the states are represented as structural tags to bracket and differentiate various categories of phrases. The basic idea of using the structural tags to represent the states is similar to Skut et al [8] and Zhou et al [9]. Here, a structural tag consists of three parts:

- Boundary Category (BOUNDARY): it is a set of four values: "O"/"B"/"M"/"E", where "O" means that current word is a whOle phrase and "B"/"M"/"E" means that current word is at the Beginning/in the Middle/at the End of a phrase.
- Phrase Category (PHRASE): it is used to denote the category of the phrase.
- Part-of-Speech (POS): Because of the limited number of boundary and phrase categories, the POS is added into the structural tag to represent more accurate state transition model.

For example, given the following POS tagged sentence as the observation sequence:

He/PRP reckons/VBZ the/DT current/JJ account/NN deficit/NN will/MD narrow/VB to/TO only/RB $/$ 1.8/CD billion/CD in/IN September/NNP ./.

We can have a corresponding sequence of structural tags as the state sequence:

O_NP_PRP(He/PRP) O_VP _VBZ (reckons/VBZ) B_NP _DT (the/DT) M_NP _JJ (current/JJ) M_NP _NN (account/NN) E_NP _NN (deficit/NN) B_VP _MD (will/MD) E_VP _VB (narrow/VB) O_PP _TO (to/TO) B_QP _RB (only/RB) M_QP _$ ($/$) M_QP _CD (1.8/CD) E_QP _CD (billion/CD) O_PP _IN (in/IN) O_NP _NNP(September/NNP) O_O _. (./.)

and an equivalent phrase chunked sentence as the shallow parsing result:

[NP He/PRP] [VP reckons/VBZ] [NP the/DT current/JJ account/NN deficit/NN] [VP will/MD narrow/VB] [PP to/TO] [QP only/RB $/$ 1.8/CD billion/CD] [PP in/IN] [NP September/NNP] [O ./.]

5 Experimental Results

We have used the CoNLL'2000 standard chunking corpus in our experimentation. This corpus was first used in the CoNLL-2000 shared chunking task [16], which aims to annotate 10 base phrase classes (NP, VP, PP, ADJP, etc). This corpus consists of four sections (15-18) of the WSJ part of the Penn TreeBank [17] for the training data (211727 tokens) and one section (20) for the test data (47377 tokens)[2].

All the evaluations are measured using the F-measure. Here, the F-measure is the weighted harmonic mean of the precision (P) and the recall (R): $F = \frac{(\beta^2+1)RP}{\beta^2 R + P}$ with $\beta^2 = 1$ [18], where the precision (P) is the percentage of predicted phrase chunks that are actually correct and the recall (R) is the percentage of correct phrase chunks that are actually found.

In this paper, the valid set of pattern entry forms *ValidEntryForm* is defined to include those pattern entry forms within a window of 7 observations(including current, left 3 and right 3 observations) where for w_j to be included in a pattern entry, all or one of the overlapping features in each of $p_j, p_{j+1}..., p_i (j \le i)$ or $p_i, p_{i+1}..., p_j (i \le j)$ should be included in the same pattern entry while for p_j to be included in a pattern entry, all or one of the overlapping features in each of $p_{j+1}, p_{j+2}..., p_i (j \prec i)$ or $p_i, p_{i+1}..., p_{j-1} (i \prec j)$ should be included in the same pattern entry. For example of a window of 3:

$$ValidEntryForm = \{p_i, p_i w_i, p_{i-1} p_i, p_{i-1} p_i w_i, p_{i-1} w_i p_i, p_{i-1} w_{i-1} p_i w_i, p_i p_{i+1}, p_i w_i p_{i+1},$$
$$p_i p_{i+1} w_{i+1}, p_i w_i p_{i+1} w_{i+1}, p_{i-1} p_i p_{i+1}, p_{i-1} w_{i-1} p_i p_{i+1}, p_{i-1} p_i w_i p_{i+1}, p_{i-1} p_i p_{i+1} w_{i+1},$$
$$p_{i-1} w_{i-1} p_i w_i p_{i+1}, p_{i-1} w_{i-1} p_i p_{i+1} w_{i+1}, p_{i-1} p_i w_i p_{i+1} w_{i+1}, p_{i-1} w_{i-1} p_i w_i p_{i+1} w_{i+1}\}$$

Table 1 shows the effect of different number of nearest neighbors in the kNN algorithm and considered previous states in the variable-length pair-wise mutual information modeling approach of the MIIM on the CoNLL'2000 chunking corpus. It

[2] http://cnts.uia.ac.be/conll2000/chunking/

shows that finding 3 nearest neighbors in the kNN algorithm using kernel density estimation performs best. It also shows that further increasing the number of nearest neighbors does not increase or even decrease the performance. This may be due to introduction of noisy neighbors when the number of nearest neighbors increases. Moreover, Table 1 shows that the MIIM performs best when six previous states is considered in the variable-length pair-wise mutual information-based modeling approach and further considering more previous states does not increase the performance. This suggests that the state dependence exists well beyond traditional ngram modeling (e.g. bigram and trigram) to six previous states and the variable-length pair-wise mutual information-based modeling approach can capture the long state dependence. In the following experimentation, we will use the MIIM with 3 nearest neighbors used in the kNN algorithm and 6 previous states considered in the variable-length pair-wise mutual information modeling approach.

Table 1. Effect of different numbers of nearest neighbors in the kNN algorithm and previous states considered in the variable-length pair-wise mutual information modeling approach of the MIIM

Shallow Parsing		Number of nearest neighbors				
		1	2	3	4	5
Number of considered previous states	1	92.06	92.51	92.83	92.82	92.83
	2	92.55	93.02	93.35	93.36	93.30
	4	92.82	93.34	93.72	93.67	93.61
	6	93.01	93.63	**93.96**	93.91	93.88
	8	93.14	93.68	93.92	93.85	93.83

Table 2. Comparison of the MIIM with generative HMMs and the kNN algorithm using kernel density estimation with a classifier-based approach

Model		CoNLL'2000 chunking
HMM	First order	91.84
	Second order	92.01
MIIM	kNN	**93.96**
	MaxEnt	93.59
	SNoW	93.74

Table 2 compares the MIIM with generative HMMs. It also compares the kNN algorithm using kernel density estimation with a classifier-based approach, such as SNoW [19,20] and MaxEnt [14] in estimating the output model of the MIIM. Here, all the classifiers (SNoW and MaxEnt) use the same observation history as the kNN algorithm in the MIIM with a windows of 7 observations including current, left 3 and right 3 observations, and use the same feature conjunctions as defined in *ValidEntryForm*. It shows that the MIIM significantly outperforms generative HMMs due to the modeling of the observation dependence and allowing for non-independent, difficult to enumerate observation features. It also shows that the kNN algorithm using kernel density estimation outperforms these classifier-based

approaches. This may be because kernel density estimation can output better probability distribution than a classifier-based approach. This suggests that the kNN algorithm using kernel density estimation captures the dependence between the features of the observation sequence more effectively by forcing the model to account for the distribution of those dependencies.

Table 3. Comparison of the MIIM with the best-reported systems on shallow parsing

Model	CoNLL'2000 chunking
Zhang et al [23](ensemble)	94.13
LSD-DMM(individual)	**93.96**
Kudoh et al [21] (ensemble)	93.91
Zhou et al [9](individual)	92.12

Table 3 compares the MIIM with the best-reported systems on shallow parsing, where one best individual system (using a single classifier) and two ensemble systems (using an ensemble of classifiers) are included. It shows that our system based on an individual MIIM significantly outperforms other best-reported individual systems and gains comparable performance with the best-reported ensemble systems.

6 Related Work

To resolve the inherent problems in generative HMMs, some researches have been done to move from generative HMMs to discriminative Markov models (DMMs). Punyakanok and Roth [22] proposed a projection-based DMM (PDMM) which represents the probability of a state transition given not only the current observation but also past and future observations and used the SNoW classifier [19,20] to estimate it. McCallum et al [24] proposed the extact same model and used maximum entropy to estimate it in the application of information extraction. Lafferty et al [25] extanded ME-PDMM using conditional random fields by incorporating the factored state representation of the same model (that is, representing the probability of a state given the observation sequence and the previous state) to alleviate the label bias problem in PDMMs, which can be biased towards states with few successor states. Similar work can also be found in Bouttou [26]. McCallum et al [27] further extended Lafferty et al [25] using dynamic conditional random fields. Punyakanok and Roth [22] also proposed a non-projection-based DMM which separates the dependence of a state on the previous state and the observation sequence, by rewriting the generative HMM in a discriminative way and heuristically extending the notation of an observation to the observation sequence. Zhou et al [9] systematically derived the exact same model as in Punyakanok and Roth [22] and used back-off modeling with error driven learning to esimate the probability of a state given the observation sequence.

Compared with the above DMMs, the MIIM explicitly models the long state dependence using a variable length pair-wise mutual information-based modeling approach and the non-projection nature of the MIIM alleviates the label bias problem inherent in projection-based DMMs. Another difference is the use of the kernel density estimation in estimating the output model of the MIIM. It is interesting to

show that the kernel density estimation leads to increased performance over a classifier-based approach.

7 Conclusion

Hidden Markov Models (HMM) are a powerful probabilistic tool for modeling sequential data and have been applied with success to many text-related tasks, such as shallow parsing. In these cases, the observations are usually modified as multinomial distributions over a discrete dictionary and the HMM parameters are set to maximize the likelihood of the observations. This paper presents a Mutual Information Independence Model (MIIM) that allows observations to be represented as arbitrary overlapping features and defines the conditional probability of the state sequence given the observation sequence. It does so by assuming a novel pair-wise mutual information independence to separate the dependence of a state given the observation sequence and the previous states. Finally, the long state dependence and the observation dependence can be effectively captured by a variable-length pair-wise mutual information model and a kNN algorithm using kernel density estimation respectively. It is also interesting to note that kernel density estimation leads to increased performance over a classifier-based approach in estimating the output model of the MIIM.

In future work, we will explore the effect of an ensemble in MIIM and its application in other tasks, such as named entity recognition and full parsing.

References

1. Church K.W. (1998). A Stochastic Pars Program and Noun Phrase Parser for Unrestricted Text. *Proceedings of the Second Conference on Applied Natural Language Processing (ANLP'1998).* Austin, Texas.
2. Weischedel R., Meteer M., Schwartz R., Ramshaw L. & Palmucci J. (1993). Coping with Ambiguity and Unknown Words through Probabilistic Methods. *Computational Linguistics.* 19(2): 359-382.
3. Merialdo B. (1994). Tagging English Text with a Probabilistic Model. *Computational Linguistics.* 20(2): 155-171.
4. Bikel D.M., Schwartz R. & Weischedel R.M. (1999). An Algorithm that Learns What's in a Name. *Machine Learning* (Special Issue on NLP). 34(3): 211-231.
5. Zhou G.D. & Su J. (2002). Named Entity Recognition Using a HMM-based Chunk Tagger, *Proceedings of the Conference on Annual Meeting for Computational Linguistics (ACL'2002).* 473-480, Philadelphia.
6. Segond F., Schiller A., Grefenstette & Chanod F.P. (1997). An Experiment in Semantic Tagging using Hidden Markov Model Tagging. *Proceedings of the Joint ACL/EACL workshop on Automatic Information Extraction and Building of Lexical Semantic Resources.* pp. 78-81. Madrid, Spain.
7. Brants T., Skut W., & Krenn B. (1997). Tagging Grammatical Functions. *Proceedings of the Conference on Empirical Methods on Natural Language Processing (EMNLP'1997).* Brown Univ.
8. Skut W. & Brants T. (1998). Chunk Tagger – Statistical Recognition of Noun Phrases. *Proceedings of the ESSLLI'98 workshop on Automatic Acquisition of Syntax and Parsing.* Univ. of Saarbrucken. Germany.

9. Zhou G.D. & Su J. (2000). Error-driven HMM-based Chunk Tagger with Context-Dependent Lexicon. *Proceedings of the Joint Conference on Empirical Methods on Natural Language Processing and Very Large Corpus (EMNLP/ VLC'2000)*. Hong Kong.
10. Rabiner L.R. (1989). A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE,* 77(2): 257-286.
11. Gale W.A. & Sampson G. 1995. Good-Turing frequency estimation without tears. *Journal of Quantitative Linguistics*. 2:217-237.
12. Jelinek F. (1989). Self-Organized Language Modeling for Speech Recognition. In Alex Waibel and Kai-Fu Lee (Editors). *Readings in Speech Recognitiopn*. Morgan Kaufmann. 450-506.
13. Katz S.M. (1987). Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer. *IEEE Transactions on Acoustics. Speech and Signal Processing*.35:400-401.
14. Ratnaparkhi A. (1999). Learning to parsing natural language with maximum entropy models. *Machine Learning*. 34:151-175.
15. Daelemans W. Buchholz S. & Veenstra. (1999). Memory-based shallow parsing. In *Proceedings of CoNLL'1999*. Bergen, Norway.
16. Tjong K.S. Daelemans, Dejean H. etc. (2000). Applying system combination to base noun noun phrase indentification. In *Proceedings of COLING 2000*. Saarbrucken Germany.
17. Marcus M., Santorini B. & Marcinkiewicz M.A. (1993). Buliding a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*. 19(2):313-330.
18. van Rijsbergen C.J. (1979). *Information Retrieval*. Buttersworth, London.
19. Roth D. (1998). Learning to resolve natural language ambiguities: A unified approach. In *Proceedings of the National Conference on Artificial Intelligence.* 806-813.
20. Carlson A, Cumby C. Rosen J. & Roth D. 1999. The SNoW learning architecture. *Techinical Report* UIUCDCS-R-99-2101. UIUC.
21. Kudoh T. & Matsumoto Y. (2001). Chunking with support vector machines. In *Proceedings of NAACL'2001*. Pittsburgh, PA, USA.
22. Punyakanok V. & Roth D. (2000). The Use of Classifiers in Sequential Inference *NIPS-13.*
23. Zhang T., Damerau F. & Johnson D. (2002). Text chunking based on a generalization of winnow. Journal of Machine Learning Research. Vol. 2 (March). Pp 615-637).
24. McCallum A. Freitag D. & Pereira F. 2000. Maximum entropy Markov models for information extraction and segmentation. ICML-19. 591-598. Stanford, California.
25. Lafferty J. McCallum A & Pereira F. (2001). Conditional random fields: probabilistic models for segmenting and labeling sequence data. *ICML-20.*
26. Bottou L. (1991). Une approche theorique de l'apprentissage connexionniste: Applications a la reconnaissance de la parole. *Doctoral dissertation, Universite de Paris XI.*
27. McCallum A., Rohanimanesh K. & Sutton C. (2003). Dynamic conditional random fields for jointly labeling multiple sequences. In Proceedings of IJCAI 2003.

Applying Conditional Random Fields to Chinese Shallow Parsing

Yongmei Tan, Tianshun Yao, Qing Chen, and Jingbo Zhu

Natural Language Processing Lab,
Northeastern University, Shenyang 110004
yongmeitan@hotmail.com, chenqing@ics.neu.edu.cn
{tsyao, zhujingbo}@mail.neu.edu.cn

Abstract. Chinese shallow parsing is a difficult, important and widely-studied sequence modeling problem. CRFs are new discriminative sequential models which may incorporate many rich features. This paper shows how conditional random fields (CRFs) can be efficiently applied to Chinese shallow parsing. We employ using CRFs and HMMs on a same data set. Our results confirm that CRFs improve the performance upon HMMs. Our approach yields the F1 score of 90.38% in Chinese shallow parsing with the UPenn Chinese Treebank. CRFs have shown to perform well for Chinese shallow parsing due to their ability to capture arbitrary, overlapping features of the input in a Markov model.

1 Introduction

Chinese shallow parsing is an important component of most text analysis systems in applications such as information extraction and summary generation. This problem has been widely studied and approached from different aspects. There are two main types of approaches to shallow parsing. One is base on rule-based methods; the other based on statistical methods. There is now a growing interest in applying machine-learning techniques to chunking, as they can avoid tedious manual work and are helpful in improving performance.

Much work has been done by researchers in this area. Li et al. used Maximum Entropy (ME) model to conduct Chinese chunk parsing [1], Zhang and Zhou used the inner structure and lexical information of base phrases to disambiguate border and phrase type [2]. Zhou et al. introduced the Chinese chunk parsing scheme and separated constituent recognition from full syntactic parsing, by using words boundary and constituent group information [3]. Zhao and Huang systematically defined Chinese base noun phrase from the linguistic point of view and presented a model for recognizing Chinese base noun phrases [4]. The model integrated Chinese base noun phrase structure templates and context features. These studies achieved promising results. However, comparing Chinese shallow parsing performance is difficult because those papers use different chunk definition and different data sets.

In this paper, we explore the practical issues in Chinese shallow parsing and present results on Chinese shallow parsing using Conditional Random Fields (CRFs).

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 167–176, 2005.

CRFs [5] are models proposed recently that have the ability to combine rich domain knowledge, with finite-state decoding, sophisticated statistical methods, and discriminative, supervised training. In their most general form, they are arbitrary undirected graphical models trained to maximize the conditional probability of the desired outputs given the corresponding inputs. This method has been successfully applied in many NLP fields, such as POS tagging [5], noun phrase segmentation [6], Chinese word segmentation [7], named entity extraction [8] and Information Extraction [9][10].

In what follows, first, we briefly describe the general framework of Chinese shallow parsing and explore the practical issue in Chinese shallow parsing. Then, we describe CRFs, including how to conduct parameter estimation. Finally, we present experimental results and draw conclusions with possible future directions.

2 Chinese Shallow Parsing

Shallow parsing is the process of identifying syntactical phrases in natural language sentences. Several types of chunks – phrases that are derived from parse trees of Chinese sentences by flattening down the structure of the parse trees - provide an intermediate step to natural language understanding.

The pioneer work of Ramashaw and Marcus [11] has been proved to be an important inspiration source for shallow parsing. They formulate the task of NP-chunking as a tagging task where a large number of machine learning techniques are available to solve the problem. Therefore shallow parsing can be regarded as of as a sequence segmentation problem in which each word is a token in a sequence to be assigned a label. Without loss of generality, let X be a set of word sequences and Y be a set of syntactic labels. The training set is then a sequence of pairs of the form $(X\,1, Y\,1), (X\,2, Y\,2) \ldots (X\,n, Y\,n)$, where $X\,i \in \mathrm{X}$, $Y\,i \in \mathrm{Y}$. On the basis of such a training set, a shallow parser could be trained, and then it can make predictions on future, unlabelled examples.

2.1 Chinese Chunk Definition

Chunks were first introduced by Abney [12], who used them for syntactic parsing. According to his definition, a chunk is the nonrecursive core of an intra-clausal constituent, extending from the beginning of constituent to its head, but not including post-head dependents.

Like the definition of English chunk given by Abney, we define Chinese chunk as a single semantic and non-recursive core of an intra-clausal constituent, with the restriction that no chunks are included in another chunk.

To be able to represent the whole hierarchical phrase structure, 10 types of Chinese chunks are defined. The phrase categories are listed below, each followed by a simple explanation and an example.

Table 1. The Chinese chunk categories

No	Category	Explanation	Example
1	VP	verb phrase	[ADVP 先后/ad] [VP 颁布/vv 实行/vv 了/as]
2	DP	determiner phrase	[DP 这些/dt] [NP 经济/nn 活动/nn]
3	ADJP	adjective phrase	[ADJP 对内/jj 和/cc 对外/jj] [NP 政策/nn]
4	QP	quantifier phrase	[VP 增长/vv] [QP 一成/cd 至/cc 两成/cd]
5	FRAG	fragment phrase	[FRAG（/pu 完/vv ）/pu]
6	NP	noun phrase	[NP 德国/nr 政府/nn 发言人/nn] [NP 福格尔/nr]
7	PP	preposition phrase	[PP 在/p 2 月/nt 2 6 日/nt] [VP 举行/vv]
8	LCP	phrase formed by “LC”	[VP 是/vc] [LCP 近年/nt 来/lc]
9	ADVP	adverbial phrase	[ADVP 已经/ad 或/cc 正在/ad] [VP 研究/vv]
10	CLP	classifier phrase	[QP 二十五/cd] [CLP 米/m] [NP 口径/nn]

To represent Chinese chunks clearly, we use 3 types of chunk border tags in this paper.

1. B-XP XP ∈ {VP, DP, ADJP, QP, FRAG, NP, PP, LCP, ADVP, CLP} denotes that the current word is the first word of chunk XP.
2. I-XP XP ∈ {VP, DP, ADJP, QP, FRAG, NP, PP, LCP, ADVP, CLP} denotes that the current word is inside of chunk XP.
3. O denotes that the current word is outside any chunk.

Using these chunk border tags, we can consider the Chinese shallow parsing as a tagging task.

2.2 Independency Assumption

HMMs learn a generative model over input sequence and labeled sequence pairs. While enjoying wide historical success, standard HMMs have difficulties in modeling multiple non-independent features of the observation sequence. They are generative, in the sense which that they represent a joint probability distribution $P(X, Y)$. Because this includes a distribution $P(X)$ over the input features, it is difficult to use arbitrary, overlapping features while maintaining tractability.

2.3 Label Bias

Classical probabilistic automata [13], discriminative Markov models [14], maximum entropy taggers [15], and MEMMs, as well as non-probabilistic sequence tagging and segmentation models with independently trained next-state classifiers [16] are all potential victims of the label bias problem [5]. This is because the per-state normalization requirement of next-state classifiers – the probability transitions leaving any given state must sum to one. Each transition distribution defines the conditional probabilities of possible next states given the current state and next observation element. Therefore, the per-state normalization requirement means that observations are only able to affect which successor state is selected, and not the probability mass passed onto that state which results in a bias towards states with low entropy

transition and, in the case of states with a single outgoing transition, causes the observation to be effectively ignored [17]. In Chinese parsing, this problem is extremely severe.

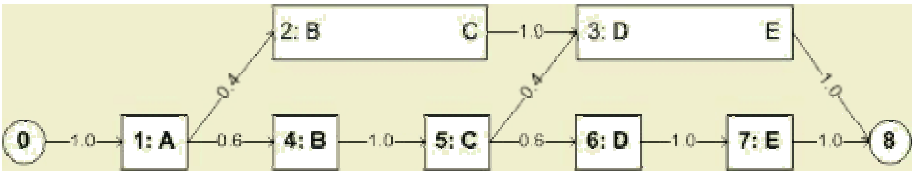


Fig. 1. Label bias problem

For example, Figure 1 represents a simple finite-state model designed to shallow parsing. The optimal path 0-1-4-5-6-7-8 is indicated by bold font. But the path 0-1-2-3-8 will have a higher probability and then be selected in decoding, because P (0, 1, 4, 5, 6, 7, 8|X) = 1.0*0.6*1.0*0.6*1.0*1.0 = 0.36, P(0, 1, 2, 3, 8|X)=1.0*0.4*1.0*1.0=0.4, P (0, 1, 4, 5, 3, 8|X) = 1.0*0.6*1.0*0.4*1.0 = 0.24, P (0, 1, 4, 5, 6, 7, 8|X) < P (1, 2, 3, 8|X) and P (0, 1, 4, 5, 3, 8|X) < P (1, 2, 3, 8|X). This is case that the states with a single outgoing transition effectively ignore their observations. More generally, states with low-entropy next state distributions will take little notice of observations.

3 Conditional Random Fields (CRFs)

CRFs are a recently introduced [5] from of conditional model that allow the strong independence assumptions of HMMs to be relaxed, as well as overcoming the label-bias problem exhibited by MEMMs [18]. This allows the specification of a single joint probability distribution over the entire label sequence given the observation sequence, rather than defining per-state distributions over the next states given the current state. The conditional nature of the distribution over label sequences allows CRFs to model real-world data in which the conditional probability of a label sequence can depend on non-independent, interacting features of the observation sequence. In addition to this, the exponential nature of the distribution chosen by Lafferty et al. enables features of different states to be traded off against each other, weighting some states in a sequence as being more important than others.

CRFs are defined as follows. Let $X = x_1 x_2 ... x_T$ denote some observed input data sequences, such as a sequence of words in training data. Let $Y = y_1 y_2 ... y_T$ be a set of finite state machine (FSM) states, each of which is associated with a label. By the Hammersley-Clifford theorem, CRFs define the conditional probability of a state sequence given an input sequence X

$$p(Y \mid X) = \frac{1}{Z_X} \exp(\sum_{i=1}^{T} \sum_k \lambda_k f_k(y_{i-1}, y_i, X, t)) \quad (1)$$

where Z_X is a normalization factor over all candidate paths. In other words, it is the sum of the "scores" of all possible state sequence.

$$Z_X = \sum_{y \in Y} \exp(\sum_{i=1}^{T} \sum_k \lambda_k f_k(y_{i-1}, y_i, X, t))$$

$f_k(y_{i-1}, y_i, X, t)$ is a feature function. The feature functions can measure any aspect of a state transition $y_{t-1} \to y_t$, and the observation sequence X, centered at the current time step t.

λ_k is a learned weight associated with feature f_k. Large positive values for λ_k indicate a preference for such an event, while large negative values make the event unlikely.

Given such a model as defined in Equ.1, the most probable labeling sequence for an input X is Y^* which maximizes a posterior probability.

$$Y^* = \arg\max_Y P_\lambda(Y \mid X) \tag{2}$$

It can be found with dynamic programming using the Viterbi algorithm.

In the case of the commonly used graph structure for modeling sequential data, the general form of Equ. 1 can be expanded to

$$p(Y \mid X) = \frac{1}{Z_X} \exp(\sum_{i=1}^{T} \sum_k \lambda_k f_k(y_{i-1}, y_i, X) + \sum_{i=1}^{T} \sum_k \mu_k g_k(y_i, X)) \tag{3}$$

Where each $f_k(y_{i-1}, y_i, X)$ is a feature of the entire observation sequence and the labels at position i and $i-1$ in the corresponding label sequence, each $g_k(y_i, X)$ is a feature of the label at position i and the observation sequence, and λ_k and μ_k are feature weights.

In this situation, the parameters λ_k and μ_k corresponding to these features are equivalent to the algorithm of HMMs transition and emission probabilities. Although it encompasses HMM-like models, the class of CRFs is much more expressive, because it allows arbitrary dependencies on the observation sequence [5].

3.1 Parameter Estimation

Given the parametric from of a CRF in Equ.3, fitting empirical distribution involves identifying the values of parameters λ_k and μ_k which can be estimated by maximum likelihood, i.e. maximizing the loglikelihood L_Λ – maximizing the conditional probability of a set of label sequences, each given their corresponding input sequence.

$$\begin{aligned} L_\Lambda &= \sum_i \log P_\Lambda(Y_i \mid X_i) \\ &= \sum_i (\sum_{t=1}^{T} \sum_k \lambda_k f_k(y, y, X, t) - \log Z_{x_i}) \end{aligned} \tag{4}$$

To maximize L_Λ, we have to maximize the difference between the correct path and those of all other candidates. CRFs are thus trained to discriminate the correct path from all other candidates, which reduces the inference of label bias.

Lafferty et al. introduced an iterative scaling algorithm for Equ. 4 L_Λ and reported that it was exceedingly slow. Several researchers have implemented gradient ascendant methods, but naïve implementations are also very slow, because the various λ and μ parameters interact with each other increasing one parameter may require compensating changes in others. McCallum 2003 employs the BFGS algorithm, which is an approximate second-order method that deals with these parameter interactions.

4 Experiments

We conducted experiments comparing CRFs to HMMs on Chinese shallow parsing. Also, we compared the performance of the model trained using CRFs from different training data size.

4.1 Experimental Setting

We use the Penn Wall Street Journal Chinese Treebank (LDC-2001T11) as experimental data. It consists of about 100K words, 325 Xinhua newswire articles on a variety of subjects. We consider each sentence to be a training instance, with single words as tokens. Sections 1-300 were used as training set, sections 301-325 was used as the test set. Table 2 summarizes the information on the dataset. Table 3 shows the detail information of training set and test set. In this experiment we only use the pos tag of the current word t_i and the current word w_i as features.

Table 2. The simple statistics on dataset

Information	Value
# articles	325
# sentences	4185
# words	100k
# chunk types	10
# chunks	62633

Table 3. The number of each chunk type in dataset

Type	Data set	Training set	Test set
VP	13619	13211	408
DP	1322	1275	47
ADJP	3132	3082	50
QP	4008	3735	273
FRAG	593	564	29
NP	26807	25782	1025
PP	3754	3618	136
LCP	1358	1305	53
ADVP	4893	4747	146
CLP	3147	2922	225

4.2 Evaluation Metrics

We measure the performance in terms of tagging accuracy, precision, recall and F-score, which are standard measures for the chunk recognition.

$$accuracy = \frac{\#of\ correct\ tagged\ words}{\#of\ words\ in\ test\ corpus}$$

$$recall = \frac{\#of\ correct\ words}{\#of\ words\ in\ test\ corpus}$$

$$precision = \frac{\#of\ correct\ words}{\#of\ words\ in\ system\ output}$$

$$F_{\beta=1} = \frac{2 \times recall \times precision}{recall + precision}$$

4.3 Experimental Results

We first report the overall results by comparing CRFs with HMMs. Table 4 shows the results on the dataset described before with the best results in bold. Compared with the result of the HMMs the result based on CRFs leads to an improved performance on most types of Chinese chunks, except ADVP, DP, LCP and PP chunks. The precision of ADJP is 1.42% lower than that of HMMs, but the FB1 is 0.18% higher than that of the HMMs and the recall of QP is 0.73% lower than that of HMMs, but the FB1 is higher than that of the HMMs, which shows that CRFs significantly outperforms HMMs.

Table 4. The results based on CRFs and based on HMMs

	CRFs accuracy: **91.90**			HMMs-bigram accuracy: 90.17		
	precision	recall	FB1	precision	recall	FB1
ADJP	87.04	94.00	**90.38**	88.46	92.00	90.20
ADVP	100.00	99.32	**99.66**	100.00	99.32	**99.66**
CLP	98.25	100.00	**99.12**	96.98	100.00	98.47
DP	97.92	100.00	**98.95**	97.92	100.00	**98.95**
FRAG	96.55	96.55	**96.55**	72.97	93.10	81.82
LCP	100.00	100.00	**100.00**	100.00	100.00	**100.00**
NP	83.79	80.53	**82.13**	78.35	77.66	78.00
PP	100.00	100.00	**100.00**	100.00	100.00	**100.00**
QP	84.46	91.58	**87.87**	82.08	92.31	86.90
VP	94.03	96.57	**95.28**	93.64	93.87	93.76
ALL	89.74	89.89	**89.82**	86.65	88.21	87.42

In the following experiment, we use 25 files as test set, but the training set range from 25 files to 300 files. Figure 2 shows the performance curve on the same test set in terms of the FB1, P and R measure with respect to the size of training data. We can see that precision, recall, and FB1 improve rapidly when the size of training set has not reached 75 folders. After that, the improvement slows down significantly. From

this figure, we can see that more training data can help improve Chinese shallow parsing performance.

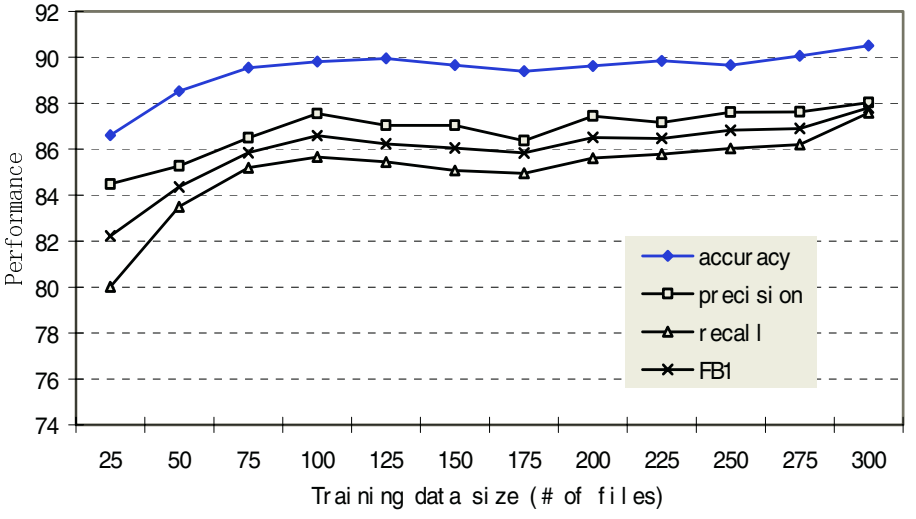


Fig. 2. The results based on CRFs vs. training set size

5 Discussion

From the experimental results we observed that the performance for Chinese shallow parsing do not look as good as those for English. One of the reasons might be that Chinese syntactic structure is more flexible and more ambiguous.

Looking through the errors in the results, we see that BAP, NP and QP internal structure is more flexible than another Chinese chunk type. The ambiguities of these syntactic structures lead to their poor experimental results. E.g. "[NP 支撑/nn 和/cc 推动/nn 作用/nn]" is likely to be selected compared to multiple tokens "[NP 支撑/nn 和/cc 推动/nn] [NP 作用/nn]" while the multiple tokens "[NP 美国/nr 网球/nn] [NP 公开赛/nn]" is likely to be selected compared to "[NP 美国/nr 网球/nn 公开赛/nn]". "[ADJP 老/jj] 、/ pu [ADJP 少/jj] 、/pu [ADJP 边/ jj] 、/pu [ADJP 穷/jj]" or "[QP 4 /cd] ： /pu [QP 3 /cd]" is likely to be selected compared to "[ADJP 老/jj 、/ pu 少/jj 、/pu 边/ jj 、/pu 穷/jj]" or "[QP 4 /cd ： /pu 3 /cd]".

Another question is that CRFs have many promising properties, but their main limitation is the slow convergence of the training algorithm relative HMMs, for which training on fully observed data is efficient.

6 Conclusion and Future Work

As far as we know the presented work is the first to apply CRFs to Chinese shallow parsing. In this paper, we present how conditional random fields can be applied to Chinese shallow parsing in which Chinese chunk boundary ambiguity exists. By virtue

of CRFs, a number of correlated features can be incorporated and label bias can be minimized. We compare results between CRFs and HMMs in Upenn Chinese Treebank, and CRFs outperform the other approaches. Although we discuss Chinese shallow parsing, the proposed approach can be applicable to other language such as Thai.

From the experimental results we observed that more linguistic knowledge incorporated into the models may further improve the performance as well. Thus, our future work is to incorporate more contextual information into the models, including the boundary information of the phrases, semantic, collocation and co-occurrence information, aiming at further improvement of chunking in terms of the precision, recall and F score.

Another attractive aspect of CRFs is that one can implement efficient feature selection and feature induction algorithm for them. In the future we can start from features generating rules and evaluate the benefit of generated features automatically on data instead of specifying in advance which features of (X，Y) to use.

Acknowledgments. The authors would like to thank Cong Li and the anonymous reviewers for their helpful comments. This research was supported in part by National Natural Science Foundation (Beijing) (60083006), National 973 Project (Beijing) (G19980305011) and National Natural Science Foundation (Beijing) & Microsoft (60203019).

References

1. Sujiang Li, Qun Liu and Zhifeng Yang. Chunk Parsing with Maximym Entropy Principle. Chinese Journal of Computers. 2003. 26(12) 1734-1738.
2. Yuqi Zhang and Qiang Zhou. Automatic identification of Chinese base phrases. 2002. Journal of Chinese Information Processing, 16(16).
3. Qiang Zhou, Maosong Sun and Changning Huang. Chunking parsing scheme for Chinese sentences. 1999. Chinese J.Computers, 22(1):1158–1165.
4. Jun Zhao and Changning Huang. A transformation-based model for Chinese basenp recognition. 1998. Journal of Chinese Information Processing, 13(2):1–7.
5. John Lafferty, Andrew McCallum and Fernando Pereira. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. 2001. Proc. 18th International Conf. on Machine Learning.
6. Fei Sha and Fernando Pereira. Shallow Parsing with Conditional Random Fields. 2003. In Proceedings of Human Language Technology-NAACL, Edmonton, Canada.
7. Andrew McCallum and Fang-fang Feng. Chinese Word Segmentation with Conditional Random Fields and Integrated Domain Knowledge.2003.
8. Andrew McCallum and Wei Li. Early results for named entity recognition with conditional random fields, Feature Induction and Web-Enhanced Lexicons. 2003. In Proceedings of Seventh Conference on Natural Language Learning (CoNLL).
9. David Pinto, Andrew McCallum, Xing Wei and W. Bruce Croft. 2003. Table extraction using conditional random fields. In Proc. of SIGIR, pages 235-242.
10. Fuchun Peng and Andrew McCallum. Accurate information extraction from research papers. 2004. In Proc. of HLT/NAACL.
11. L. A. Ramashaw and M. P. Marcus. Text chunking using transformation-based learning . 1995. Proceedings of the Third ACL Workshop on Very Large Corpora.

12. S. Abney. Parsing by chunks. 1991. In Robert C.Berwick, Steven P. Abney, and Carol Tenny, editors, Principle-Based Parsing: Computation and Psycholinguistics, Kluwer Academic Publishers, Boston, pages 257–278.
13. A. Paz. Introduction to probabilistic automata. Academic Press. 1971.
14. L. Bottou. Une approche theorique del'apprentissage connexionniste: Applications a la reconnaissance de la parole. 1991. Doctoral dissertation, Universite de Paris XI.
15. A. Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. Proc. EMNLP. New Brunswick, New Jersey: Association for Computational Linguistics.
16. V. Punyakanok. 2001. The use of classifiers in sequential inference. NIPS 13.
17. Hanna Wallach. Efficient Training of Conditional Random Fields. 2002. Thesis. Master of Science School of Cognitive Science, Division of Informatics. University of Edinburgh.
18. Andrew McCallum, D. Freitag and F. Pereira. Maximum entropy Markov models for information extraction and segmentation. 2000. Proc. ICML 2000(pp. 591-598). Stanford, California.
19. Andrew Kachites McCallum. "MALLET: A Machine Learning for Language Toolkit." http://mallet.cs.umass. edu. 2002.
20. Tianshun Yao, et al, Natural Language Processing – A research of making computers understand human languages, Tsinghua University Press, 2002, (In Chinese).

Distributional Thesaurus Versus WordNet: A Comparison of Backoff Techniques for Unsupervised PP Attachment*

Hiram Calvo[1], Alexander Gelbukh[1], and Adam Kilgarriff[2]

[1] Center for Computing Research, National Polytechnic Institute, Mexico
hcalvo@sagitario.cic.ipn.mx, gelbukh@gelbukh.com
[2] Lexical Computing Ltd., United Kingdom
adam@lexmasterclass.com

Abstract. Prepositional Phrase (PP) attachment can be addressed by considering frequency counts of dependency triples seen in a non-annotated corpus. However, not all triples appear even in very big corpora. To solve this problem, several techniques have been used. We evaluate two different backoff methods, one based on WordNet and the other on a distributional (automatically created) thesaurus. We work on Spanish. The thesaurus is created using the dependency triples found in the same corpus used for counting the frequency of unambiguous triples. The training corpus used for both methods is an encyclopaedia. The method based on a distributional thesaurus has higher coverage but lower precision than the WordNet method.

1 Introduction

The Prepositional Phrase (PP) attachment task can be illustrated by considering the canonical example *I see a cat with a telescope*. In this sentence, the PP *with a telescope* can be attached to *see* or *cat*. Simple methods based on corpora address the problem by looking at frequency counts of word-triples or dependency triples: *see with telescope* vs. *cat with telescope*. In order to find enough occurrences of such triples, a very large corpus is needed. Such corpora are now available, and the Web can also be used [4, 27]. However, even then some combinations of words do not occur. This is a familiar effect of Zipf's law: few words are very common and there are many words that occur with a low frequency [14], and the same applies to word combinations.

To address the problem, several backoff techniques have been explored. In general, 'backing off' consists of looking at statistics for a set of words, when there is insufficient data for the particular word. Thus *cat with telescope* turns into ANIMAL *with* INSTRUMENT and *see with telescope* turns into *see with* INSTRUMENT (capitals denote sets of instrument-words, animal-words, etc.) One way to identify

* Work done under partial support of Mexican Government (CONACyT, SNI, PIFI-IPN, CGEPI-IPN) and RITOS-2.

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 177–188, 2005.

Table 1. State of the art for PP attachment disambiguation

Human (without context)		Use WordNet backoff		Use thesaurus backoff	
Ratnaparkhi [20]	88.2	Stetina and Nagao [24]	88.1	Pantel and Lin [19]	84.3
Mitchell [16]	78.3	Li and Abe 1998 [12]	85.2	McLauchlan [15]	85.0

the set of words associated with a given word is to use WordNet, and another is to use a distributional thesaurus. A distributional thesaurus is a thesaurus generated automatically from a corpus by finding words which occur in similar contexts to each other [8, 25, 26]. Both approaches have already been explored (for English) and have been shown to yield results close to human disambiguation, see Table 1.

Experiments using different techniques have been carried out independently, and to date there are no evaluations which compare WordNet with distributional thesauruses. In this paper we compare those two approaches, as proposed in [10]. We use a single corpus in both cases to enable us to compare results. The same corpus is used for generating the thesaurus and the WordNet generalizations. The corpus is also used for counting the dependency triples.

Our work is on Spanish. This is, to the best of our knowledge, the first work exploring backoff methods for PP attachment for a language other than English.

2 PP Attachment with No Backoff

2.1 Building the Resources

The main resource is the count of dependency triples (DTC). In order to increase coverage, instead of considering strictly adjacent words, we consider dependency relations between word types (lemmas). Only unambiguous dependency relations are considered. For example the following two sentences: *I see with a telescope. A cat with three legs is walking*, will provide the dependency triples *see, with, telescope* and *cat, with, legs*, respectively. However, the sentence *I see a cat with a telescope* will not provide any dependency triple, as it is an ambiguous case.

We extract all dependency triples from our corpus in a batch process. We first tag the text morphologically and then group adjectives with nouns, and adverbs with verbs. Then, we search for the patterns *verb preposition noun*, *noun preposition noun*, *noun verb*, and *verb noun*. Determiners, pronouns and other words are ignored.

Following Lin [13], dependency triples consist of two words and the grammatical relationship, including prepositions, between two words in the input sentence. To illustrate the kind of dependency triples extracted, consider a micro-corpus (μC) consisting of two sentences: *A lady sees with a telescope*; and *The lady with a hat sees a cat*. The triples corresponding to this μC are shown in Figure 1. We then denote the number of occurrences of a triple <*w,r,w'*> as |*w,r,w'*|. From μC, |*lady,SUBJ,see*|=2 and |*lady*,*with*,*hat* |=1. |*,*,*| denotes the total number of triples (10 in μC), an asterisk * represents any word or relation. In μC, |*see,*,*| = 4, |*,*with,*| = 2, |*,*,*lady*| = 2.

see, SUBJ, lady	see, SUBJ, lady	see, OBJ, cat
lady, SUBJ-OF, see	lady, SUBJ-OF, see	cat, OBJ-OF, see
see, with, telescope	lady, with, hat	
telescope, with_r, see	hat, with_r, lady	

Fig. 1. Dependency triples extracted from μC

$$x=|x,*,*| \quad p=|*,p,*| \quad n=|*,*,n_2| \quad t=|*,*,*| \quad \bar{x}=t-x, \quad \bar{p}=t-p, \quad \bar{n}=t-n$$

$$xpn=|x,p,n_2|, \quad \bar{x}pn=|*,p,n_2|-xpn, \quad x\bar{p}n=|x,*,n_2|-xpn, \quad xp\bar{n}=|x,p,*|-xpn$$

$$\bar{x}\bar{p}n=n-xpn-\bar{x}pn-x\bar{p}n, \quad \bar{x}p\bar{n}=p-xpn-\bar{x}pn-xp\bar{n}$$

$$x\bar{p}\bar{n}=x-xpn-x\bar{p}n-xp\bar{n}, \quad \bar{x}\bar{p}\bar{n}=t-(xpn+\bar{x}pn+x\bar{p}n+xp\bar{n}+\bar{x}\bar{p}n+\bar{x}p\bar{n}+x\bar{p}\bar{n})$$

$$score=xpn\cdot\log[xpn/(x\cdot p\cdot n/t^2)]+\bar{x}pn\cdot\log[\bar{x}pn/(\bar{x}\cdot p\cdot n/t^2)]+$$
$$x\bar{p}n\cdot\log[x\bar{p}n/(x\cdot\bar{p}\cdot n/t^2)]+xp\bar{n}\cdot\log[xp\bar{n}/(x\cdot p\cdot\bar{n}/t^2)]+$$
$$\bar{x}p\bar{n}\cdot\log[\bar{x}p\bar{n}/(\bar{x}\cdot p\cdot\bar{n}/t^2)]+\bar{x}\bar{p}n\cdot\log[\bar{x}\bar{p}n/(\bar{x}\cdot\bar{p}\cdot n/t^2)]+$$
$$x\bar{p}\bar{n}\cdot\log[x\bar{p}\bar{n}/(x\cdot\bar{p}\cdot\bar{n}/t^2)]+\bar{x}\bar{p}\bar{n}\cdot\log[\bar{x}\bar{p}\bar{n}/(\bar{x}\cdot\bar{p}\cdot\bar{n}/\mathrm{t}^2)]$$

for VScore, x is v, for NScore, x is n_1

Fig. 2. Formulae for calculating three-point log-likelihood

The grammatical relationships without prepositions will be useful later for thesaurus-building, where word similarity will be calculated based on contexts shared between two words. By now, we will use this resource (DTC) only to count triples of (*verb*, *preposition*, $noun_2$) and ($noun_1$, *preposition*, $noun_2$) to decide a PP attachment. This is explained in the following section.

2.2 Applying the Resources

The task is to decide the correct attachment of p,n_2 given a 4-tuple of verb, $noun_1$, preposition, $noun_2$: (v,n_1,p,n_2). The attachment of p,n_2 can be either to the verb v or the noun n_1. The simplest unsupervised algorithm attaches according to which is the highest of VScore = $|v,p,n_2|$ and NScore = $|n_1,p,n_2|$. When both values are equal we say that this attachment is not decidable by this method.

The corpus used for counting dependency triples (DTC) in this experiment was the whole Encarta encyclopaedia 2004 in Spanish [1]. It has 18.59 M tokens, 117,928 types in 73MB of text, 747,239 sentences, and 39,685 definitions. The corpus was tagged using the TnT Tagger trained with the manually tagged (morphologically) corpus CLiC-TALP[1] and lemmatized using the Spanish Anaya dictionary [11].

Once the corpus is morphologically tagged and lemmatized, the dependency triples are extracted. Encarta produced 7M dependency triple tokens, amongst which there were 3M different triples, i.e. 3M dependency-triple types. 0.7M tokens (0.43M types) involved prepositions.

[1] http://clic.fil.ub.es. The TnT tagger trained with the CLiC-TALP corpus has a performance of over 92% 17.

Table 2. Different formulae for calculating VScore and NScore

	description	VScore	NScore
S	the simplest one	$\lvert v,p,n_2\rvert$	$\lvert n_1,p,n_2\rvert$
S2	considering doubles too	$\lvert v,p,n_2\rvert \times \lvert v,p,*\rvert$	$\lvert n_1,p,n_2\rvert \times \lvert n_1,p,*\rvert$
LL3	Log likelihood ratio	See Figure 2	
Feat	Simplified Roth features 19 and 23	$\log(\lvert *,p,*\rvert/\lvert *,*,*\rvert) +$ $\log(\lvert v,p,n_2\rvert/\lvert *,*,*\rvert) +$ $\log(\lvert v,p,*\rvert/\lvert v,*,*\rvert) +$ $\log(\lvert *,p,n_2\rvert/\lvert *,*,n_2\rvert)$	$\log(\lvert *,p,*\rvert/\lvert *,*,*\rvert) +$ $\log(\lvert n_1,p,n_2\rvert/\lvert *,*,*\rvert) +$ $\log(\lvert n_1,p,*\rvert/\lvert v,*,*\rvert) +$ $\log(\lvert *,p,n_2\rvert/\lvert *,*,n_2\rvert)$

We used four different formulae for calculating VScore and NScore, listed in Table 2. The first two formulae can be seen as the calculus of the probability of each triplet, e.g. $p(v,p,n_2)=\lvert v,p,n_2\rvert/\lvert *,*,*\rvert$. Since both VScore and NScore are divided by the same number $\lvert *,*,*\rvert$, it can be omitted without any difference. For log-likelihood[2] formulae, see Figure 2.

Following the PP attachment evaluation method by Ratnaparkhi *et al.* [20], the task is to determine the correct attachment given a 4-tuple (v,n_1,p,n_2). We extracted 1,137 4-tuples, along with their correct attachment (N or V), from the manually tagged corpus Cast-3LB[3] [18]. Sample 4-tuples are shown in Table 3.

Table 3. Example of 4-tuples (v,n_1,p,n_2) used for evaluation

4-tuples	English gloss
informar comunicado del Banco_Central N	inform communication of Central_Bank N
producir beneficio durante periodo V	produce benefit during period V
defender resultado de elección N	defend results of election N
recibir contenido por Internet V	receive contents by Internet V
planchar camisa de puño N	iron shirt of cuff N

The baseline can be defined in two ways. The first is to assign all attachments to $noun_1$. This gives precision of 0.736. The second is based on the fact that the preposition *de* 'of' attaches to a noun in 96.9% of the 1,137 4-tuples.[4] This gives a

Table 4. Comparison of formulae for calculating VScore and NScore

Method	Coverage	Precision
Baseline	1.000	0.661
S	0.127	0.750
S2	0.127	**0.773**
LL3	0.127	0.736
Feat	0.127	0.717

[2] Log-likelihood was calculated using the Ngram statistics package, see [2].

[3] Cast-3LB is part of the 3LB project, financed by the Science and Technology Ministry of Spain. 3LB, (FIT-150500-2002-244 and FIT 150500-2003-411)

[4] This is valid also for English. For the training set provided by Ratnaparkhi, the preposition *of* attaches to a noun in 99.5% of the 20,801 4-tuples.

precision of 0.855, a high value for a baseline, considering that the human agreement level is 0.883. To avoid this highly biased baseline, we opted for excluding all 4-tuples with preposition *de*—no other preposition presents such a high bias. Then all our evaluations are done using only 419 of the 1,137 4-tuples extracted. The baseline in this case consists of assigning all attachments to the verb, which gives 66.1% precision. The human inter-tagger agreement for 4-tuples excluding preposition *de* is 78.7%, substantially lower than human agreement for all 4-tuples. Results are shown in Table 4.

The highest precision is provided by formula S2, so from now on we will use this formula to compare results with backoff methods.

3 WordNet Backoff

3.1 Building the Dictionary

We are looking for a wider coverage of dependency relations in order to decide a correct PP attachment. To achieve this, we construct a dictionary which uses WordNet to find a generalization of dependency relations. For example, we seek the generalization of *eat with fork*¸ *eat with spoon* and *eat with knife* into *eat with* {*tableware*}. Note that {*tableware*} is not a word, but a concept in WordNet. WordNet provides the knowledge that *fork*¸ *spoon* and *knife* are {*tableware*}. This way, if an unseen triple is found, such as *eat with chopsticks,* WordNet can help by saying that *chopsticks* are a {*tableware*} too, so that we can apply our knowledge about *eat with* {*tableware*}.

Before we describe our method, let us introduce some notation. Every word *w* is linked to one or more synsets in WordNet corresponding to its different senses. W_n denotes the synset corresponding to the *n*-th sense of *w,* and *N* the total number of senses. Each one of these synsets has several paths to the root by following their hypernyms. W_n^m denotes the *m-th* hypernym of the *n-th* sense of *w',* and M_n the depth, i.e. the number of hypernyms to the root for sense number *n*.

For example, *glass* in WordNet has 7 senses. The third hypernym of the fourth sense of *glass* is denoted by W_4^3 = *astronomical_telescope*. See below an extract for *glass* from WordNet to illustrate this.

sense 2: *glass* (drinking glass) → container → instrumentality → artifact → object → whole → object → entity

sense 4: *glass* (spyglass) → refracting_telescope → optical_telescope → **astronomical_telescope** → telescope → magnifier → scientific_instrument → instrument → device → instrumentality → artifact → object → entity

Our WordNet backoff method is based on [5] and [6]. To extend a score (NScore or VScore) through WordNet, we must consider all triples involving the same *w* and *r*, varying *w'* (as in the case of learning *eat with* {*tableware*} from several examples of *eat with* *). This set of triples is denoted by <*w,r,**>. For each involved w', we

distribute evenly[5] each score s(*w,r,w'*) among each one of its senses of *w'* (as in [22]). Then this result is propagated to all hypernyms W_n^m. This value is accumulative: higher nodes in WordNet collect information from all their daughters. This way, more general concepts summarize the usage (frequency of triples) of their specific concepts (hyponyms).

To avoid over-generalization (that is, the excessive accumulation at top levels,) depth must be considered. Sometimes the depth of hypernyms' chain is very large (as in *glass*' sense 4) and sometimes small (sense 2 of *glass*). A useful propagation formula that allows generalization and considers depth of chains of hypernyms is:

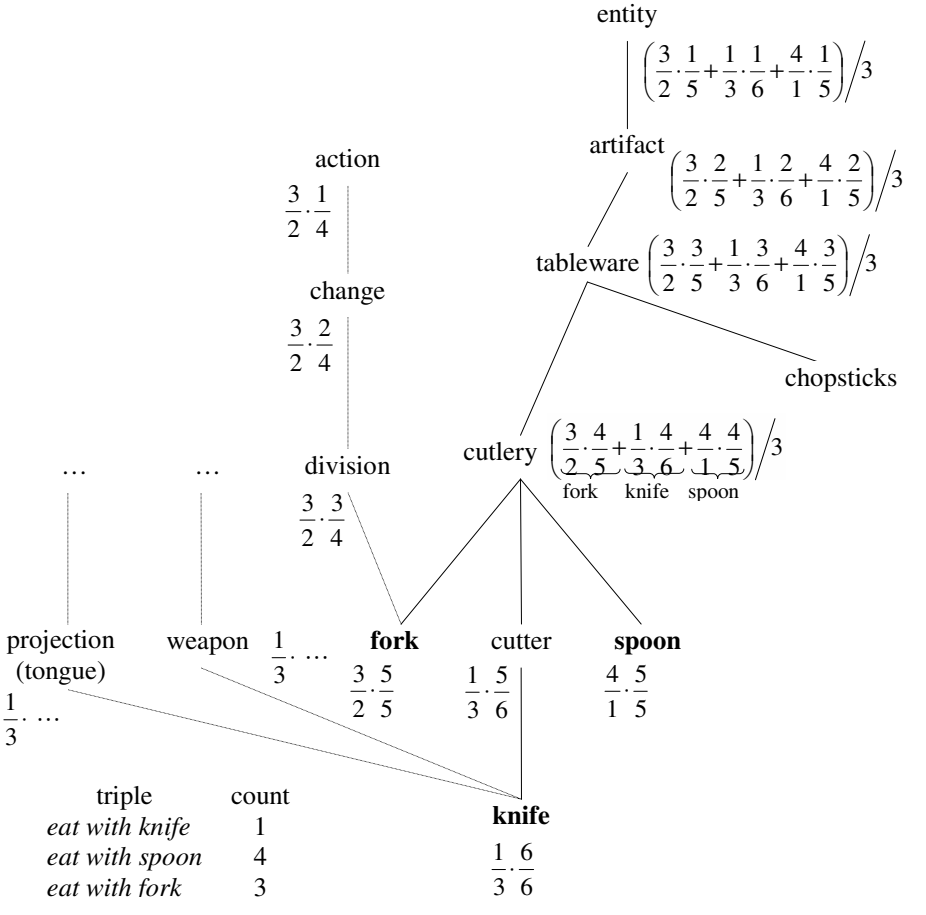


Fig. 3. Example of propagation of triple's counts in WordNet

[5] We assume an equiprobable distribution, which is problematic. However, there are currently no comprehensive sense tagged texts for Spanish from which we could extract sense distributions.

Table 5. Examples of relation triples (*w,r,w'*) with WordNet backoff

w	*r*	*w'*	English	score
comer	*con*	*mano*	hand	3.49
'eat'	'with'	*cubiertos*	cutlery	1.31
		tenedor	fork	1.19
matar	*con*	*arma*	weapon	0.27
'kill'	'with'	*armamento*	armaments	0.23
		utillaje	utensil	0.18

$$s(w,r,W_n^m) = [s(w,r,w')/\mathrm{N}] \times [1-(m-1/M_n)] \quad (1)$$

In addition, the number of triples contributing to a certain WordNet node is counted for averaging at upper levels. That is, after considering the *k* triples <*w,r,**>, we count the number of triple types contributing to each node. Then, the value of each node is divided by such number.

To illustrate our algorithm, see Figure 3. For this example suppose we only have three triples—each one is listed along with its count in Figure 3. The frequency count for each triple is added to the corresponding word in WordNet. For *eat with fork*, the node for the word *fork* is labeled with 3 counts for *eat with*. *fork* may be used with other combinations of words, but we show here only values for *eat with,* i.e., <*w,r,**>. Accordingly to Formula (1), this value is divided by the number of senses of *fork*. In this example we assume two different senses of *fork*, with different hypernyms each: {*division*} and {*cutlery*}. Focusing on the {*cutlery*} branch, we can see how this value is propagated towards to {*entity*}. For this branch there are 5 levels of depth from {*entity*} to *fork* (M_2=5)—the other branch has 4 levels (M_1=4). Following the propagation of *fork* up in the tree, it can be seen how each level has a lower weight factor—for {*tableware*} is 3/5 and for {*entity*} only 1/5. Each node is accumulative; because of this, {*cutlery*} accumulates the values for *fork*, *knife* and *spoon.* The value for {*cutlery*} is divided by 3 because the number of types of contributing triples to this node is 3. If we had another triple *eat with chopsticks* then {*cutlery*} would remain untouched, but {*tableware*} would be divided by 4.

For this experiment we used Spanish EuroWordNet[6] 1.0.7 (S-EWN) [7]. It has 93,627 synsets (62,545 nouns, 18,517 adjectives, 12,565 verbs), 51,593 hyponym/ hypernym relations, 10,692 meronym relations and 952 role information entries (noun agent, instrument, location or patient). We propagated all dependency triples in DTC using Formula (1) (creation of DTC was explained in Section 0.)

The WordNet backoff algorithm presented in this section produces subjectively good results. In Table 5 the first three top qualifying triples with *con* as relation for two common Spanish verbs are listed.

3.2 Using the Dictionary

To decide a PP attachment in a 4-tuple (v,n_1,p,n_2), we calculate NScore for (n_1,p,n_2), and VScore for (v,p,n_2) as in Section 2.2. The highest score determines the

[6] S-EWN was Developed jointly by the University of Barcelona (UB), the Nat University of Open Education (UNED), and the Polytechnic University of Cataluña (UPC), Spain.

attachment. WordNet backoff is applied when a triple is not found. In this case, n_2 is substituted by its hypernyms until the score from the new triple (x,p, W_n^m) is found in the previously calculated WordNet-extended-scores. When calculating NScore, x is n_1, and when calculating VScore, x is v. The highest score determines the attachment. Note that we are backing off only n_2. We decided not to back off v because the verb structure in S-EWN has very few hypernym relations for verbs (7,172) and the definition of a hypernym for a verb is not clear in many cases. Since we do not back off v, we cannot back off n_1 as this would introduce a bias of NScores against VScores. Also note that W_n^m is a specific synset in the WordNet hierarchy, and hence it has a specific sense. The problem of disambiguating the sense of n_2 is solved by choosing the highest value from each set of senses in each hypernym layer; see [5] and [24] for WSD using PP attachment information. Results for this method will be presented in Section 5.

Following the example from Figure 3, suppose we want to calculate the VScore for *eat with chopsticks*. Since this triple is not found in our corpus of frequency counts, we search for the hypernyms of *chopsticks*, in this case, {*tableware*}. Then, the value of this node is used to calculate VScore.

4 Thesaurus Backoff

4.1 Building the Dictionary

Here we describe the automatic building of a thesaurus so that words not found in the dependency triples can be substituted by similar words. This similarity measure is based on Lin's work [19]. This thesaurus is based on the similarity measure described in [13]. The similarity between two words w_1 and w_2 as defined by Lin is:

$$sim_{lin}(w_1, w_2) = \frac{\sum_{(r,w)\in T(w_1)\cap T(w_2)} (I(w_1, r, w) + I(w_2, r, w))}{\sum_{(r,w)\in T(w_1)} I(w_1, r, w) + \sum_{(r,w)\in T(w_2)} I(w_2, r, w)}$$

$$I(w, r, w') = \log \frac{|w, r, w| \times |*, r, *|}{|w, r, *| \times |*, r, w'|}$$

$T(w)$ is the set of pairs (r,w') such that $I(w,r,w')$ is positive. The algorithm for building the thesaurus is the following:

```
for each word type w1 in the corpus
|    for each word type w2 in the corpus
|    |      sims(w1) ← {simlin(w1,w2), w2}
|    sort sims(w1) by similarity in descending order
```

Table 6. Example of similar words using Lin similarity method

word *w*	similar word *w'*	English	$sim_{lin}(w,w')$
guitarrista	*pianista*	pianist	0.141
'guitarist'	*fisiólogo*	physiologist	0.139
	educador	teacher	0.129
devoción	*afecto*	affection	0.095
'devotion'	*respeto*	respect	0.091
	admiración	admiration	0.078
leer	*editar*	to edit	0.078
'to read'	*traducir*	to translate	0.076
	publicar	to publish	0.072

Like the WordNet method, this gives subjectively satisfactory results: Table 6 lists the 3 most similar words to *guitarrista* 'guitarrist', *devoción* 'devotion', and *leer* 'to read'.

4.2 Using the Dictionary

To decide a PP attachment in a 4-tuple (v,n_1,p,n_2), our algorithm calculates NScore for (n_1,p,n_2), and VScore for (v,p,n_2) as in Section 2.2. The highest score determines the attachment. When a triple is not found, the backoff algorithm is applied. In this case, n_2 is substituted by its most similar word n'_2 calculated using $sim_{lin}(n_2, n'_2)$. If the new triple (x,p,n'_2) is found in the count of dependency triples (DTC), then it is used for calculating the score. If it is not found, then the next most similar word is tried for a substitution, until the new triple (x,p,n'_2) is found. When calculating NScore, x is n_1; when calculating VScore, x is v. The highest score determines the attachment. The algorithm is shown below. When n=1, the n-th most similar word corresponds to the first most similar word—for example *pianist* for *guitarist*. For n=2 it would be *physiologist*, and so on.

```
To decide the attachment in (v,n1,p,n2):

      VSCore = count(v,p,n2)
      NScore = count(n1,p,n2)
      n, m ← 1
      if NScore = 0
      |   while NScore = 0 & exists n-th word most similar to n2
      |   |       simn2  ← n-th word most similar to n2
      |   |       factor ← sim(n2,simn2)
      |   |       NScore ← count(n1,p,simn2) × factor
      |   └       n      ← n + 1
      if VScore = 0
      |   while VScore = 0 & exists n-th word most similar to n2
      |   |       simn2  ← m-th word most similar to n2
      |   |       factor ← sim(n2,simn2)
      |   |       VScore ← count(n1,p,simn2) × factor
      |   └       m      ← m + 1
      |
      if NScore = VScore then cannot decide
      if NScore > Vscore then attachment is to n1
      if NScore < Vscore then attachment is to v
```

Table 7. Results of our experiments for PP attachment disambiguation

Method	Coverage	Precision	Average
Manual agreement (human)	1.000	0.787	0.894
Default to verb (baseline)	1.000	0.661	0.831
No backoff	0.127	**0.773**	0.450
WordNet backoff	0.661	0.693	0.677
Distributional thesaurus backoff	**0.740**	0.677	**0.707**

5 Comparison of Methods

In this section we compare results of the three methods: no backoff, WordNet backoff and thesaurus backoff. The results are listed in Table 7, along with the baseline and manual agreement results. The third column shows the average between coverage and precision. Note that the baseline shown in Table 7 involves some supervised knowledge: most of attachments, after excluding *de* cases, are to noun. The highest precision, coverage and average values are in boldface. After excluding *de* cases, we have 419 instances. For 12.7% of them all three algorithms do the same thing, so the differences between WordNet backoff and distributional thesaurus backoff are based on the remaining 366 cases.

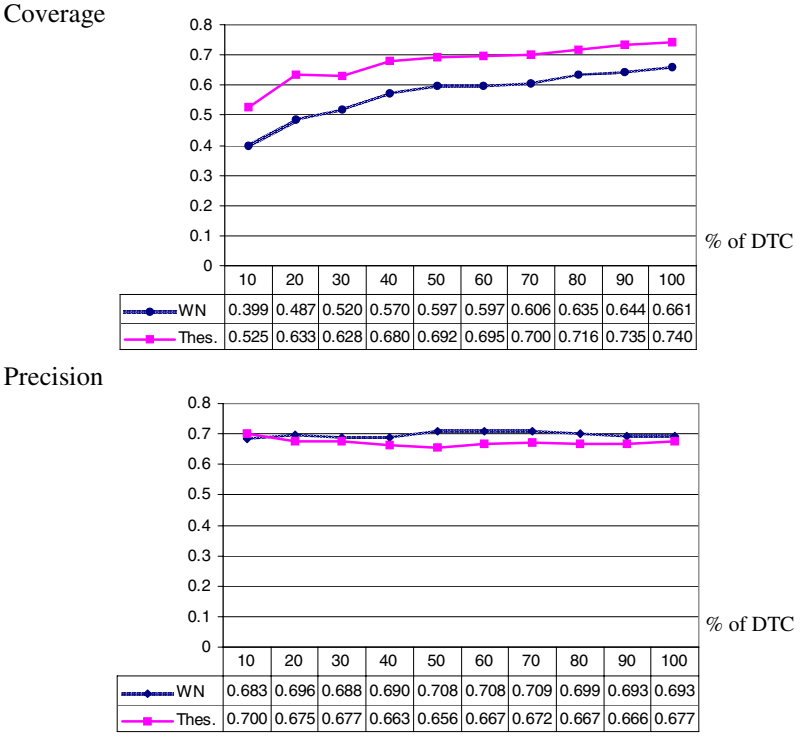


Fig. 4. Precision and coverage using different percentages of triple counts (0–100%)

Not all cases are covered by these backoff methods either because no substitution can be found for a certain word (such as several acronyms or proper names), or because even after trying all possible substitutions the triple was not found in DTC. In general, this coverage is low because of the size of the corpus for counting attachment frequencies. Although an encyclopaedia provides a text with many different words, the number of prepositional attachments extracted is rather low. We believe that using a bigger corpus will yield higher coverage measures but will keep the same relationship between the backoff methods studied, as suggested by our experiments which use only randomly chosen partial percentages of the DTC corpus. This is shown in Figure 4. Note that we are using a totally unsupervised model. That is, in both algorithms we do not use any other backoff technique for not covered cases.

6 Conclusions

Amongst the three methods evaluated for PP attachment, the best average measure was 0.707 using thesaurus backoff, due to its greater coverage compared with other methods. However, it has lower precision than WordNet backoff. The method with no backoff had a very low coverage (0.127) but for the attachments covered the results were the best, at 0.773 close to manual agreement. (Remember that this agreement is calculated excluding a highly biased preposition: *de* 'of', which practically is always attached to nouns.) Performance of WordNet backoff could be increased by adding information of the sense distribution for each word, instead of assuming an equiprobable distribution, although this would render this method closer to a supervised approach, and moreover no resource providing sense distributions for Spanish is available.

Our results indicate that an automatically built resource (in this case, a thesaurus) can be used instead of a manually built one and still obtain similar results.

In our future work we shall explore using much larger corpora for gathering counts of triples, and we shall experiment with more sophisticated algorithms for using the thesaurus to determine attachments.

References

1. Biblioteca de Consulta Microsoft Encarta 2004, Microsoft Corporation, 1994–2004
2. Banerjee, S. and Ted Pedersen, T., The Design, Implementation, and Use of the Ngram Statistic Package, in *Proceedings of the Fourth International Conference on Intelligent Text Processing and Computational Linguistics*, Mexico City, 2003, 370–381
3. Brants, Thorsten. TnT: A Statistical Part-of-Speech Tagger. *Proceedings of the 6th Applied Natural Language Processing Conference*, Seattle, WA, USA, 2000
4. Calvo, Hiram and Alexander Gelbukh. Improving Disambiguation of Prepositional Phrase Attachments Using the Web as Corpus, Procs. of CIARP'2003, Cuba, 592–598, 2003
5. Calvo, Hiram and Alexander Gelbukh. Unsupervised Learning of Ontology-Linked Selectional Preferences, Procs. *of* CIARP'2004, Puebla, Mexico, 2004
6. Clark, Stephen and David Weir. Class-based Probability Estimation Using a Semantic Hierarchy, *Computational Linguistics* 28(2), 2002

7. Farreres, X., G. Rigau and H. Rodríguez. Using WordNet for Building WordNets. In *Proceedings of COLING-ACL Workshop "Usage of WordNet in Natural Language Processing Systems*", Montreal, Canada. 1998.
8. Grefenstette, G. *Explorations in Automatic Thesaurus Discovery*. Kluwer, 1994
9. Hindle, Don and Mats Rooth. Structural ambiguity and lexical relations. *Computational Linguistics* 19:103–120, 1993
10. Kilgarriff, Adam. Thesauruses for Natural Language Processing. *Proceedings of NLP-KE 03*, Beijing, China, 5–13, 2003
11. Lázaro Carreter, F. (Ed.) Diccionario Anaya de la Lengua, Vox, 1991
12. Li, Hang and Naoki Abe. Word clustering and disambiguation based on co-ocurrence data. *Proceedings of COLING '98*, 749–755, 1998
13. Lin, Dekang. An information-theoretic measure of similarity. *Proceedings of ICML'98*, 296–304, 1998
14. Manning, Chris and Hinrich Schutze. Chapter 1 of *Foundations of Statistical Natural Language Processing*, MIT Press, Cambridge, MA, 1999
15. McLauchlan, Mark. Thesauruses for Prepositional Phrase Attachment. *Proceedings of CoNLL-2004*, Boston, MA, USA, 73–80, 2004
16. Mitchell, Brian. *Prepositional phrase attachment using machine learning algorithms*. Ph.D. thesis, University of Sheffield, 2003
17. Morales-Carrasco, Raúl and Alexander Gelbukh. Evaluation of TnT Tagger for Spanish. *Proc. Fourth Mexican International Conference on Computer Science*, Mexico, 2003
18. Navarro, Borja, Montserrat Civit, M. Antonia Martí, R. Marcos, B. Fernández. Syntactic, semantic and pragmatic annotation in Cast3LB. *Shallow Processing of Large Corpora (SProLaC)*, a Workshop of Corpus Linguistics, Lancaster, UK, 2003
19. Pantel, Patrick and Dekang Lin, An Unsupervised Approach to Prepositional Phrase Attachment using Contextually Similar Words. *Proceedings of Association for Computational Linguistics (ACL-00)*, Hong Kong, 101–108, 2000
20. Ratnaparkhi, Adwait, Jeff Reynar and Salim Roukos. A maximum entropy model for prepositional phrase attachment. *Proceedings of the ARPA Workshop on Human Language Technology*, 250–255, 1994
21. Ratnaparkhi, Adwait. Unsupervised Statistical Models for Prepositional Phrase Attachment. *Proceedings of COLINGACL98*, Montreal, Canada, 1998
22. Resnik, Philip. Selectional preference and sense disambiguation, *ACL SIGLEX Workshop on Tagging Text with Lexical Semantics*, Washington, D. C., USA, 1997
23. Roth, D. Learning to Resolve Natural Language Ambiguities: A Unified Approach. Ini *Proceedings of AAAI-98*, Madison, Wisconsin, 806–813, 1998
24. Stetina, Jiri and Makoto Nagao. Corpus based PP attachment ambiguity resolution with a semantic dictionary. *Proceedings of WVLC '97*, 66–80, 1997
25. Sparck Jones, Karen. *Synonymy and Semantic Classification*. Edinburgh University. Press, 1986
26. Weeds, Julie, *Measures and Applications of Lexical Distributional Similarity*. Julie Weeds, Ph.D. thesis. University of Sussex. 2003
27. Volk, Martin. Exploiting the WWW as a corpus to resolve PP attachment ambiguities. Proceeding of *Corpus Linguistics 2001*, Lancaster, 2001

Automatic Recognition of Czech Derivational Prefixes

Alfonso Medina Urrea[1] and Jaroslava Hlaváčová[2]

[1] GIL IINGEN UNAM,
Apartado Postal 70-472, 04510 Coyoacán, DF, Mexico
amedinau@iingen.unam.mx
[2] ÚFAL MFF UK
Malostranské náměstí 25, 11800, Praha, Czech Republic
hlava@ufal.mff.cuni.cz

Abstract. This paper describes the application of a method for the automatic, unsupervised recognition of derivational prefixes of Czech words. The technique combines two statistical measures — Entropy and the Economy Principle. The data were taken from the list of almost 170 000 lemmas of the Czech National Corpus

1 Introduction

Our contribution concerns only those languages where words are created by means of affixes. Usually, there exists a quite stable vocabulary, but it is possible to create entirely new words adding suffixes and/or prefixes to already existing ones. If the derivation follows common rules for word creation, everybody understands them, even if they have never seen them before.

The Czech language belongs to the group of languages that derive their vocabulary mainly by means of adding affixes. While the set of suffixes is very stable and does not change during long periods of time, prefixes are much more vivid. Of course, there is a set of old, traditional prefixes, that have been used for a very long time and do not change. But one can very easily add a morph, usually borrowed from other languages, in front of an existing word and create an entirely new word. The old prefixes can be found in every grammar, but the new ones cannot.

Everybody who understands the language understands new prefixes. Everybody but computers. And for any analysis of language, it is very important to know them. Without a sufficiently large list of prefixes, we cannot run successfully enough a morphological analysis, which usually stands on the basis of all automatic language processing.

To be specific — the morphological analyzer always encounters unknown words; that is, words for which it does not recognize their basic forms nor their morphological categories. It is possible to design a "guesser" that uses special properties of the language which could help to guess those basic features of the unknown word. In Czech, we usually take suffixes as the basis [1].

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 189–197, 2005.

The list of prefixes can serve as another type of such guesser: if we get a word that is not included in our morphological dictionary, we would try to see if any of the prefixes matches the beginning of the word. If so, it is probable, that the rest of the word, after tearing the prefix off, will be recognized by the morphological analyzer.

2 Word Sample

The empirical word source for this paper is a list of around 170 000 word types. The basis of our experiment was the Czech National Corpus (CNC) with 100 million word forms. As prefixes do not change with word declensions, we worked with basic word forms — lemmas. There are more than 800 000 different lemmas in the CNC, but the great majority of them have very low frequency. We selected only the lemmas with frequency of at least 5, mainly because words with lower frequencies are very often typos or other rubbish. Their total number is 166 733. In order to make the list smaller, we left out those parts of speech that do not have prefixes, namely prepositions and conjunctions. We took into account only lemmas not beginning with a capital letter, because these are almost 100% proper nouns that usually do not have prefixes.

There are some letters that are untypical for Czech — a Czech word cannot begin with y; letters g, f, w and x are very unusual and there is only a limited number of words containing them. In our list, it would be easy to go through them manually and check whether there is a foreign new prefix or not. However, since the method is unsupervised, we decided not to intervene manually into the process and let the method do it automatically.

3 Method

A full description of the method applied to Spanish can be found in Medina [2]. In that paper, several quantitative measures are explored to compare their desirability as methods to discover affixes. The methods were very successful for suffixes, but not so good for prefixes, surely because in Spanish the former constitute a compact, highly organized inflectional and derivational system, whereas prefixes do not. As the method is general and language independent,[1] we tried to use it for Czech prefixes which, as mentioned above, are very productive.

Let us quickly outline the approach applied. It combines two quantitative methods: measurement of entropy — one of the topics of information theory [4] — and the principle of economy of signs [5, 6]. We will examine some of the reasons why these two methods work well together.

[1] It was applied successfully to a small corpus of Chuj, a Mayan language spoken in Chiapas and Guatemala [3] (essentially with respect to entropy measurement); and recently to a small corpus of Raramuri (Tarahumara), a Yuto-Aztecan language spoken in Northern Mexico (both entropy and economy measurements). Because of space constraints, results for those languages are not presented here.

High entropy measurements have been reported repeatedly as successful indicators of borders between bases and affixes [2, 3, 7, 8, 9, 10]. These measurements are relevant because, as it was pointed out as early as the fifties by linguists like Joseph Greenberg[2] [11], shifts of amounts of information (in the technical sense) can be expected to correspond to the amounts of information that a reader or hearer is bound to obtain from a text or spoken discourse. Frequent segments must contain less information than those occurring rarely. Hence, affixes must accompany those segments of a text (or discourse) which contain the highest amounts of information. And this has been in fact observed for a very wide range of affixes [2, 10, 3], including those whose structural evidence —like that behind the economy principle described below— is not fully provided by a corpus, either because the corpus is too small or not representative of the language [3] or because the affixes in question are old and unproductive [10].

The other important measure used in this experiment is based on the principle of economy of signs (some experiments using measurements based on this principle — either maximum or minimum approaches — are [5, 6, 12, 2, 13]). In essence, for this approach this is a quantity representing how much linguistic structure there is in a given expression. If natural languages are systems, they and their components must be economical to some degree. Thus, we can expect certain signs to be more economical than others because they relate to other signs in an economical way. One aspect of sign economy is evident in that a sign at one level of language, say the lexical one, may be composed of more than one sign of the lower level, say the morphological ones. In this manner, a language can refer at the lexical level to a great number of things using considerably fewer signs than it would be necessary if it had exactly one sign for each thing named.

Affixes can combine with bases to produce a number (virtually infinite) of lexical signs. It is clear that affixes do not combine with every base. Certain ones combine with many bases, others with only a few. Nevertheless, it makes sense to expect more economy where more combinatory possibilities exist.

This refers to the syntagmatic dimension. The paradigmatic dimension can also be considered: affixes alternate in a corpus with other affixes to accompany bases. If there is a relatively small set of alternating signs (paradigms) which adhere to a large set of unfrequent signs (to form syntagms) the relations between the former and the latter must be considered even more economical. This is naturally pertinent for both derivation and inflection; that is, this is as true for lemma affixes, as it is for affixes of the inflected words of discourse.

4 Building a Catalog of Czech Prefixes

The program basically takes the words of the word sample and determines the best segmentation for each one (according to the two measurements discussed

[2] Zellig Harris relied on phoneme counts before and after a given word segmentation (according to a corpus), a matter undoubtedly related to entropy measurement. But he did not specifically refer to information theory, like Greenberg did.

above and described bellow). Each best segmentation represents a hypothesis postulating a base and an prefix. Thus, the presumed prefix (and the values associated with it) are fed into a structure called Catalog. The more frequent a presumed prefix is, the more likely it is really a prefix.[3]

4.1 Information Content

Information content of a set of word fragments is typically measured by applying Shannon's method.[4] As mentioned above, high entropy measurements have been reported repeatedly as successful indicators of borders between bases and affixes.

For this experiment in particular, the task was to measure the entropy of the word segments which follow a prefix candidate, according to the word sample: borders between prefixes and stems tend to exhibit peaks of entropy. Thus, looking for peaks of information meant taking each left-hand substring of each word of the sample, determining the probability of everything that follows, and applying Shannon's formula to obtain an entropy measurement for the right hand substrings related to each left-hand substring examined.

4.2 Economy Principle

The economy of segmentations can be measured by comparing the following sets of word beginnings and endings[5] from a word sample. Given a prefix candidate, there are two groups of word segments:

1. *companions* — word endings which follow the given prefix candidate (syntagmatic relation).
2. *alternants* — word beginnings which alternate (occur in complementary distribution) with the prefix candidate.

The following fraction is a simplified example of how these can be compared to capture the essence of the method proposed in [5, 6, 2, 10]:

$$k = \frac{companions}{alternants} \tag{2}$$

More formally, let $B_{i,j}$ be the set of word endings which follow, according to a corpus, the left-hand word segment $a_{i,j}$, which consists of the first jth letters

[3] Other possibilities, like selecting several best segmentations per word or including some threshold criteria to filter forms with low values, are discussed in Medina [2].

[4] Recall the formula

$$H = -\sum_{i=1}^{n} p_i \log_2 p_i \tag{1}$$

where p_i stands for the relative frequency of word fragment i [4]. See Oakes [9] or Manning and Shütze [14] —among many others— for brief descriptions of the method.

[5] It is worth noting that with the term 'ending' we do not mean here the grammatical ending of a word, but just the substring of letters towards its end.

of the ith word of the corpus. Let $B^s_{i,j}$ be the subset of $B_{i,j}$ consisting of the word endings which are suffixes of the language in question. Let $A^p_{i,j}$ be the set of word beginnings which are, also according to the corpus, prefixes of the language and occur in complementary distribution with the word beginning $a_{i,j}$. One way to estimate the economy of a segmentation between a set word beginnings and a set of word endings, in such a way that the word beginnings are prefixes is:

$$k^p_{i,j} = \frac{|B_{i,j}| - |B^s_{i,j}|}{|A^p_{i,j}|} \tag{3}$$

As established in (2), the numerator of (3) can be described as the set of right-hand *companions* of the left-hand word segment $a_{i,j}$ and the denominator the set of left-hand segments or *alternants* of (in paradigmatic relation to) $a_{i,j}$.

In this way, when an initial word fragment is given, a very large number of companions and a relatively small number of alternants yield a high economy value. Meanwhile, a small number of companions and a large one of alternants indicate a low economy measurement. In the latter case, the word fragment in question is not very likely to represent exactly a morpheme (nor, as we will see, a sequence of them).

4.3 Entropy and Economy Combined

Word segmentation methods can be compared in order to determine how successful they are [7, 2]. But they can also be combined to improve their effectiveness. The methods described above complement each other in the estimation of what can be called the affixality of word fragments. In fact, the values obtained for a given word fragment can be averaged or multiplied. For this experiment, they were normalized and averaged. That is, we estimated *prefixality* by means of the arithmetic average of the relative values of entropy and economy: $(\frac{h_i}{\max h} + \frac{k_i}{\max k}) * \frac{1}{2}$, where h_i stands for the entropy value associated to prefix candidate i; k_i represents the economy measurement associated to the same candidate; and $\max h$ returns the maximum quantity of h calculated for all prefixes (same idea for $\max k$).

The important fact is that the highest values (those expected to occur at the borders between prefixes and bases, and between bases and suffixes) are good criteria to include word fragments as items in the Catalog of Prefixes.

5 Results

The results are shown in Table 1 which contains the ninety prefix candidates with highest affixality values. Candidates are presented in the second column. The third column exhibits frequency of all lemmas from the original word sample, where the candidate comes out as the best prefix. The fourth and fifth columns contain the normalized measurements of entropy and economy. Candidates showing values of less than 0.5 (of either measurement) were filtered.

Table 1. Catalog of Czech Prefixes

	prefix	fr	econ	entr	affty
1.	severo∼	75	0.974	0.93	0.952
2.	proti∼	457	0.928	0.968	0.948
3.	jiho∼	76	0.946	0.922	0.934
4.	mezi∼	199	0.923	0.922	0.922
5.	super∼	263	0.857	0.965	0.911
6.	dvoj∼	233	0.863	0.948	0.905
7.	mimo∼	154	0.879	0.93	0.905
8.	troj∼	136	0.858	0.944	0.901
9.	mnoho∼	103	0.913	0.888	0.901
10.	osmi∼	97	0.929	0.872	0.901
11.	spolu∼	267	0.896	0.902	0.899
12.	video∼	138	0.93	0.868	0.899
13.	východo∼	47	0.926	0.871	0.899
14.	devíti∼	59	0.961	0.833	0.897
15.	při∼	1361	0.910	0.882	0.896
16.	více∼	151	0.886	0.899	0.892
17.	radio∼	102	0.862	0.92	0.891
18.	šesti∼	113	0.93	0.844	0.887
19.	nad∼	437	0.774	1	0.887
20.	celo∼	123	0.871	0.902	0.886
21.	šéf∼	45	0.938	0.833	0.885
22.	pěti∼	168	0.886	0.885	0.885
23.	západo∼	44	0.888	0.881	0.884
24.	sedmi∼	82	0.943	0.817	0.880
25.	několika∼	67	0.957	0.803	0.88
26.	pseudo∼	149	0.82	0.939	0.879
27.	třiceti∼	39	0.944	0.811	0.878
28.	velko∼	172	0.917	0.835	0.876
29.	elektro∼	168	0.802	0.947	0.875
30.	od∼	2393	0.814	0.935	0.875
31.	vy∼	4389	0.838	0.91	0.874
32.	dvanácti∼	51	0.983	0.761	0.872
33.	polo∼	448	0.794	0.95	0.872
34.	středo∼	75	0.849	0.892	0.871
35.	dvaceti∼	48	0.942	0.798	0.870
36.	deseti∼	84	0.895	0.845	0.87
37.	patnácti∼	36	0.968	0.768	0.868
38.	před∼	861	0.803	0.933	0.868
39.	vnitro∼	55	0.894	0.842	0.868
40.	jedno∼	280	0.827	0.908	0.868
41.	tří∼	240	0.867	0.869	0.868
42.	pod∼	1236	0.756	0.977	0.866
43.	dvou∼	304	0.838	0.894	0.866
44.	vysoko∼	52	0.802	0.928	0.865
45.	osmnácti∼	14	0.964	0.761	0.863
46.	horno∼	20	0.887	0.836	0.862
47.	za∼	4052	0.805	0.916	0.860
48.	čtrnácti∼	29	0.939	0.775	0.857
49.	čtyřiceti∼	33	0.92	0.791	0.856
50.	rychlo∼	62	0.856	0.836	0.846
51.	jedenácti∼	24	0.935	0.754	0.845
52.	česko∼	38	0.892	0.792	0.842
53.	foto∼	181	0.787	0.893	0.840
54.	vele∼	84	0.813	0.864	0.839
55.	roz∼	2431	0.769	0.901	0.835
56.	bio∼	164	0.782	0.886	0.834
57.	vodo∼	58	0.782	0.878	0.830
58.	znovu∼	129	1	0.654	0.827
59.	žluto∼	17	0.848	0.804	0.826
60.	mikro∼	256	0.740	0.912	0.826
61.	plno∼	39	0.826	0.826	0.826
62.	nízko∼	54	0.757	0.893	0.825
63.	půl∼	127	0.797	0.853	0.825
64.	roze∼	215	0.851	0.796	0.823
65.	arci∼	31	0.919	0.726	0.823
66.	šedesáti∼	22	0.914	0.730	0.822
67.	na∼	3580	0.730	0.912	0.821
68.	ode∼	126	0.853	0.789	0.821
69.	anti∼	398	0.7	0.939	0.819
70.	malo∼	91	0.781	0.858	0.819
71.	čtvrt∼	55	0.760	0.874	0.817
72.	do∼	2374	0.445	0.924	0.815
73.	staro∼	151	0.742	0.888	0.815
74.	ultra∼	53	0.860	0.769	0.814
75.	euro∼	106	0.722	0.903	0.812
76.	mnoha∼	35	0.881	0.743	0.812
77.	čtyřiadvaceti∼	14	0.881	0.743	0.812
78.	samo∼	229	0.758	0.864	0.811
79.	padesáti∼	39	0.884	0.734	0.809
80.	šestnácti∼	22	0.852	0.766	0.809
81.	modro∼	21	0.775	0.839	0.807
82.	vše∼	169	0.736	0.873	0.805
83.	pů∼	204	0.749	0.858	0.803
84.	osma∼	22	0.886	0.719	0.802
85.	pětiset∼	11	0.820	0.784	0.802
86.	třinácti∼	17	0.911	0.691	0.801
87.	psycho∼	105	0.803	0.795	0.799
88.	popo∼	51	0.836	0.757	0.797
89.	tisíci∼	18	0.901	0.684	0.792
90.	červeno∼	30	0.735	0.849	0.792

The last column contains the affixality index, which was calculated as the arithmetic average of the entropy and economy values of the fourth and fifth columns. Finally, the first column shows the rank of the candidates according to this index.

It is interesting that within the first ninety items there is one segment constituted by two prefixes joined together (*popo-* no. 88). As can be expected, there are more catalog items representing sequences of prefixes down the rest of the Catalog (for example, within the first hundred, *zne-* no. 99).

It is worth noting that there are no false prefixes within the first one hundred candidates (which means that the precision is 1.0 for this set).

In order to calculate recall, a set of productive Czech prefixes was compiled. Thus, the 45 most traditional prefixes were considered in order to determine how many important prefixes the method did not miss. We refer to this set as **T**.

Table 2 shows precision and recall for the first five hundreds of candidates. The first column, labelled **N**, refers to the number of the catalog items considered. The second column, **E**, shows the number of erroneous candidates (mistakes) within the first **N** candidates. The third column shows the precision — (**N-E**)/**N**. The fourth column, **NF**, displays the number of Czech traditional prefixes from the set **T** that were not found within the first **N** candidates (omissions). The recall was calculated as (45-NF)/45. Naturally, the precision decreases with the increasing number of candidates, while the recall exhibits the opposite tendency.

Some of the prefixes are not real prefixes. They could be regarded as word stems that combine with other stems to create new words, by means of composition. However, these (pre)stems behave like prefixes — they are common to more words modifying their meaning. This is, among others, the case of "numerical prefixes" — a special inflective form (usually identical with genitive) of numerals added to a word (mainly adjectives) modifying them numerically. In fact, every number can serve as a prefix in that sense, but it is usually used only for short numerals. If we wanted to say for instance *a dragon that has seven heads* we can say *sedmihlavý drak* — something like *seven headed dragon*, but the *seven headed* corresponds to one word unit in Czech containing a prefix *sedmi* meaning *seven*.

One can object that the prefixes are not divided into groups according to the parts of speech they can join. It is true that some prefixes cannot prefix any word base. That remains an interesting task for future work. In this experiment we just wanted to recognize everything that could serve as a prefix, no matter

Table 2. Evaluation measurements

N	E	precision	NF	recall
100	0	1.0000	23	0.4889
200	12	0.9400	12	0.7333
300	72	0.7600	10	0.7778
400	149	0.6275	7	0.8889
500	229	0.5420	5	0.9556

the context of the rest of the word. The sorting into groups should be a part of a further analysis. Other automatic processings would benefit from it, for instance the guesser mentioned in the Introduction.

6 Conclusions

From the results we can see that it is possible to recognize prefixes independently of the language represented by the corpus (provided they constitute an organized subsystem in that language). There was no false prefix among the first hundred of recognized prefixes. As the list becomes longer (and as the measure of affixality becomes lower), there naturally appear more mistakes.

We can examine how long the Catalog must be in order to be relatively sure that we will have recognized most prefixes. From the very essence of statistics, we can never be sure. But according to the number of wrong prefixes occurring among the items with lower affixality, it seems to us, that 500 would be a good compromise. Although there are probably some more prefixes with lower affixality, their number would be small. As it is always necessary to check the prefixes manually before using them in further analyses, the list should not be too long. We have found that the first 500 items include almost all the traditional prefixes and many new ones.

The comparison of this method with other methods (minimal and maximal distance techniques) is certainly an interesting task for future work. Nevertheless, our approach has shown that it is possible to make a list of prefixes using exact methods. If we had wanted to make the list manually, we would have had to engage in tedious work — searching dictionaries, old grammar books, checking large corpora manually. The method described is useful for everybody who is concerned with morphology of an inflectional language. Moreover, it can recognize even the most modern prefixes that have entered the language quite recently.

Acknowledgments

The work reported on this paper has been supported by a grant of the Czech Ministry of Education LN00A063, by Conacyt's Project R37712A and by DGAPA PAPITT's Project IX402204. Also, thanks to the Institute of the Czech National Corpus for allowing us to use the Czech National Corpus.

References

1. Hlaváčová, J.: "Morphological Guesser of Czech Words". In: Proc. TSD 2001, Berlin Heidelberg, Springer Verlag (2001) 70–75
2. Medina Urrea, A.: "Automatic Discovery of Affixes by Means of a Corpus: A Catalog of Spanish Affixes". Journal of Quantitative Linguistics **7** (2000) 97–114

3. Medina Urrea, A., Buenrostro Díaz, E.C.: "Características cuantitativas de la flexión verbal del chuj". Estudios de Lingüística Aplicada **38** (2003) 15–31
4. Shannon, C.E., Weaver, W.: The Mathematical Theory of Communication. University of Illinois Press, Urbana (1949)
5. de Kock, J., Bossaert, W.: Introducción a la lingüística automática en las lenguas románicas. Volume 202 of Estudios y Ensayos. Gredos, Madrid (1974)
6. de Kock, J., Bossaert, W.: The Morpheme. An Experiment in Quantitative and Computational Linguistics. Van Gorcum, Amsterdam, Madrid (1978)
7. Hafer, M.A., Weiss, S.F.: "Word Segmentation by Letter Successor Varieties". Information Storage and Retrieval **10** (1974) 371–385
8. Frakes, W.B.: "Stemming Algorithms". In: Information Retrieval, Data Structures and Algorithms. Prentice Hall, New Jersey (1992) 131–160
9. Oakes, M.P.: Statistics for Corpus Linguistics. Edinburgh University Press, Edinburgh (1998)
10. Medina Urrea, A.: Investigación cuantitativa de afijos y clíticos del español de México. Glutinometría en el Corpus del Español Mexicano Contemporáneo. PhD thesis, El Colegio de México, Mexico (2003)
11. Greenberg, J.H.: Essays in Linguistics. The University of Chicago Press, Chicago (1957)
12. Goldsmith, J.: "Unsupervised Learning of the Morphology of a Natural Language". Computational Linguistics **27** (2001) 153–198
13. Gelbukh, A., Alexandrov, M., Han, S.Y.: "Detecting Inflection Patterns in Natural Language by Minimization of Morphological Model". In: CIARP-2004. Volume 3287 of Lecture Notes in Computer Science. Springer (2004) 432–438
14. Manning, C., Schütze, H.: Foundations of Statistical Natural Language Processing. The MIT Press, Cambridge (Mass.) (1999)

Korma 2003: Newly Improved Korean Morpheme Analysis Module for Reducing Terminological and Spacing Errors in Document Analysis*

Ho-cheol Choi and Sang-yong Han

Graduate School of Information, Chungang University, Korea
hansy@cau.ac.kr

Abstract. The paper describes the newly improved Korean morpheme analysis module KorMa 2003. This new module applies the custom user dictionary for analyzing new and unknown words and special terms and operates an automatic word spacing module during post-processing to prevent failures of sentence analysis due to incorrect spacing between words. KorMa 2003 has accuracy enhanced by 15% in comparison with the previously reported version.

1 Introduction

With the emergence of the Internet industry, the amount of documents produced and distributed online is increasing tremendously. One notable fact is that such documents are generally ungrammatical and also contain many newly coined words. What is more, in Korean language most words are composed of several roots, each root corresponding to one syllable, i.e., one Korean glyph, e.g., *dehanminguk*—the official name of South Korea: *de* 'great', *han* 'Korean', *min* 'democracy', *guk* 'country'. This causes many errors consisting in incorrect spacing between words, such as **dehanmin guk* or **de hanminguk*. Thus, there is a need for a morpheme analysis module with improved analysis of newly coined words, special terms used in special fields, and words with spacing errors.

Recently, much effort has been devoted to correcting such word spacing errors. Many of such proposals use various heuristics or correct word spacing errors during preprocessing before morpheme analysis is executed. However, if word spacing corrections are made through heuristics, it is difficult to handle every single error among the countless mistakes the writer makes. In other words, corrections are restricted to the common mistakes made by a relatively large number of people [1, 3].

Our previous morpheme analysis module KorMa2000 [5] generated a list of candidate morphemes and then formed a final list of the most appropriate morphemes according to the probability of joints within or between phrasal units (*eojeols*—Korean phrasal units composed of one or more words) according to the equations:

* This research was supported by the MIC (Ministry of Information and Communication), Korea, under the Chung-Ang University HNRC-ITRC (Home Network Research Center) support program supervised by the IITA (Institute of Information Technology Assessment).

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 198 – 201, 2005.

$$\mathbf{P}\tau(P) \approx \prod \mathbf{P}\tau(P_i \mid P_{i-1}),$$
$$\mathbf{P}\tau(W \mid P) \approx \prod \mathbf{P}\tau(W_i \mid P_i),$$
$$p' = \arg\max \prod \mathbf{P}\tau(P_i \mid P_{i-1}) \prod \mathbf{P}\tau(W_i \mid P_i) = \arg\max \sum \log \mathbf{P}\tau(P_i \mid P_{i-1}) + \prod \mathbf{P}\tau(W_i \mid P_i).$$

Such analysis-through-generation architecture (successfully applied by other authors to inflective languages [1, 2]) made our Korean analysis module very sensitive to word spacing. In this paper we describe a newly revised morpheme analysis module, which applies a custom user dictionary for newly coined words, special terminology, and automatic word spacing. With this dictionary, the user can classify special words such as compound nouns and words adopted from foreign language as correct words beforehand in order to reduce errors in analysis of certain words.

2 The Korean Morpheme Analysis Module KorMA 2003

The design of the Korean morpheme analysis module KorMa2003 is shown in Fig. 1.

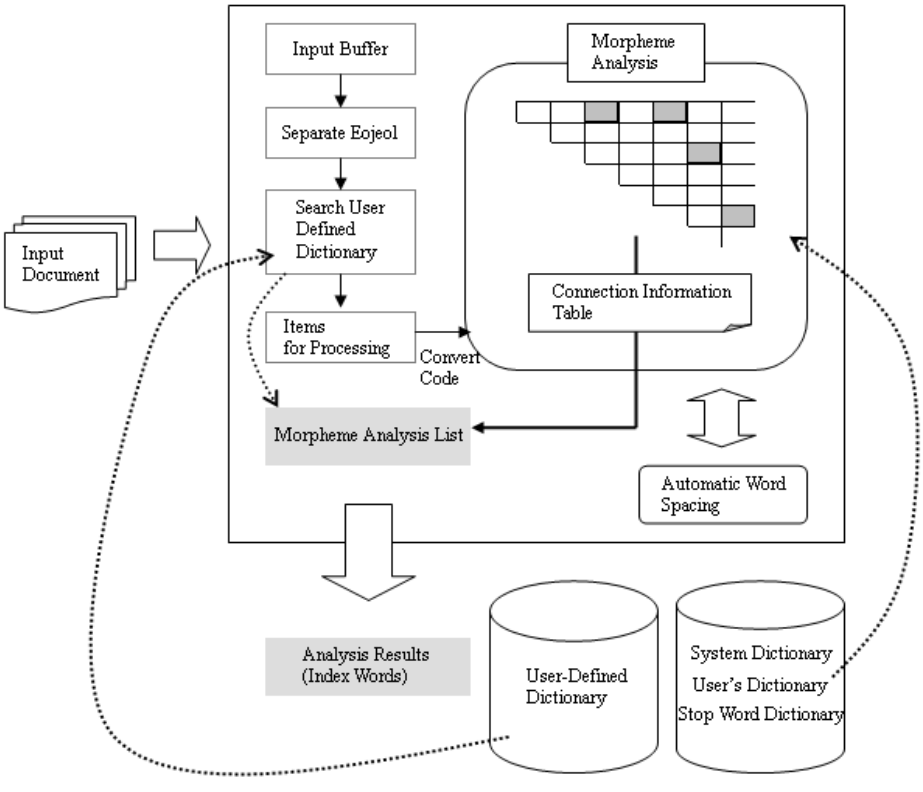


Fig. 1. Design of User-Defined Dictionary and Automatic Spacing Suppor

2.1 User-Defined Dictionary

Among the many kinds of dictionaries, the present study applies the user-defined dictionary to allow the user to define words themselves. We use to enhance accuracy in analyzing certain words or newly coined words. Although there are a variety of ways to deal with unregistered words, newly coined words, special terminology, etc., the most widely used methods are the user dictionary and special processing of suffixes and postpositions. However, these methods are insufficient in handling the matter. Table 1 exemplifies the difficulties in analyzing unregistered and special words using the former morpheme analysis module [5].

Table 1. Incorrect Results of Analysis of Unregistered Words and Special Words by the Former Morpheme Analysis Module

Compound Nouns	▪ 한국(Korea)사회(social)보장(security)제도(policy) *vs.* 한국(Korea)/사회보장제(social security system)/도(island) ▪ 한국(Korea)사회(Social)보장(Security)법(Law)론(discourse) *vs.* 한국사(Korean History)/회보(Bulletin)/장법/(law against stolen goods)/론(theory)
Unknown Words	▪ 공동정범(common/criminal) *vs.* (ball)/동정(sympathy,virginity)/범(tiger) *공동/common , 정범/crime, criminal
Special Words	▪ 뇌사자 *vs.* 뇌(brain)/사자(lion) *뇌 brain/사 dead /자 person ▪ 대어음 *vs.* 대어(big fish)/음(negative) *대 fictitious/ 어음 bill

2.2 Automatic Word Spacing

The automatic word spacing checking function looks for incorrect spacing between each syllable of an incorrect phrase using right-to-left and left-to-right search and the longest and shortest match methods [5]. Additionally, a system has been implemented to correct any mistakes in spacing, for both space insertion and space omission errors.

3 Experimental Results

We used three document collections for our tests. Table 2 and Figure 3 show the comparison of the number of index words and correct words extracted by each system.

Table 2. Percentage of Correct Words

Text collection	Correct set	KorMa2003	KorMa2000
Chamber of Commerce (unregistered)	20132	97.9%	83.6%
Military Terms (special words)	6891	97.9%	83.0%
Court (special words)	4958	97.9%	84.3%

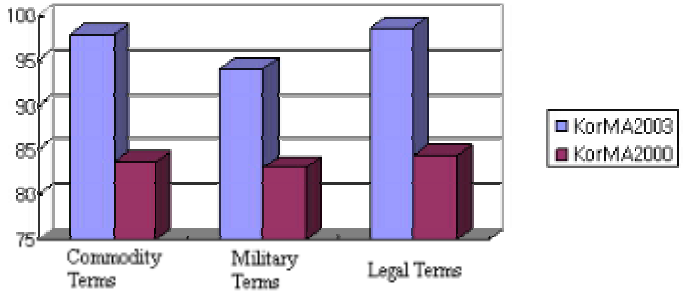


Fig. 2. Percentage of correct words after application of user-defined dictionary

4 Conclusion

To prevent errors in analysis of grammatically correct sentences as well as sentences with incorrect word spacing, we implemented an independent word spacing function for making spacing corrections when an analysis error has occurred after morpheme analysis. The Korean morpheme analysis module KorMa2003 is 15% to 17% more accurate than the former module supporting only user dictionary and post-processing (presuming pure Korean words, postposition processing).

The described morpheme analysis module—an indexing module for information retrieval systems—can be used more efficiently in cases such as documents of special fields using special terms, such as National Assembly legislation proposals that have no spaces between words, or Internet message boards where incorrect spacing occurs frequently, especially when a massive amount of documents must be indexed.

For the future work, for efficient operation of the new morpheme analysis module in indexing a vast amount of documents a method for automation should be developed to efficiently implement the user correct word set.

References

1. A. Gelbukh, G. Sidorov. Morphological Analysis of Inflective Languages through Generation. *Procesamiento de Lenguaje Natural*, No 29, Spain, p. 105–112.
2. A. Gelbukh, G. Sidorov. Approach to construction of automatic morphological analysis systems for inflective languages with little effort. In: A. Gelbukh (ed.), Computational Linguistics and Intelligent Text Processing (CICLing-2003). Lecture Notes in Computer Science, N 2588, Springer-Verlag, pp. 215–220.
3. DoSam Hwang, KiSun Choi, and TaeSuk Kim, *Natural Language Processing*.
4. JongKeun Kwak, J.J. Eun, Y.S. Kang. Structure and Characteristics of LGKNA. In: *1st workshop on Evaluation of Morpheme Analysis and Tagging*.
5. SungJin Lee, DukBong Kim, JungYun Seo, KiSun Choi, and GilChang Kim, Construction and Analysis of Korean Morpheme based on two-level model. In: Proc. of KISS Fall Conference, Volume 19.

Word Extraction Based on Semantic Constraints in Chinese Word-Formation

Maosong Sun[1], Shengfen Luo[1], and Benjamin K T'sou[2]

[1] National Lab. of Intelligent Tech. & Systems,
Tsinghua University, Beijing 100084, China
sms@mail.tsinghua.edu.cn
[2] Language Information Sciences Research Centre, City University of Hong Kong
rlbtsou@cityu.edu.hk

Abstract. This paper presents a novel approach to Chinese word extraction based on semantic information of characters. A thesaurus of Chinese characters is conducted. A Chinese lexicon with 63,738 two-character words, together with the thesaurus of characters, are explored to learn semantic constraints between characters in Chinese word-formation, forming a semantic-tag-based HMM. The Baum-Welch re-estimation scheme is then chosen to train parameters of the HMM in the way of unsupervised learning. Various statistical measures for estimating the likelihood of a character string being a word are further tested. Large-scale experiments show that the results are promising: the F-score of this word extraction method can reach 68.5% whereas its counterpart, the character-based mutual information method, can only reach 47.5%.

1 Introduction

Processing of unknown words is important for Chinese word identification in running texts. New words are generated quite often with the rapid development of Chinese society. In experience, the accuracy of a word identification system will decrease about 10% if unknown words are not treated properly [12].

Chinese is an isolating language. Methods for processing of unknown words in inflective languages, like, for example [5], may not be appropriate for Chinese because of its different morphological structure. A Chinese word is composed of either single or multiple Chinese characters. In most cases, a Chinese character has at least one sense, and can stand independently at the morphological level. The task of extracting Chinese words with multi-characters from texts is quite similar to that of extracting phrases (e.g., compound nouns) in English, if we regard Chinese characters as English words.

Researches in this field have been done extensively. Generally, there are two kinds of methods for word/phrase extraction, i.e., rule-based and statistic-based. The latter has become the mainstream of the state-of-the-art. In statistic-based approaches, the soundness of an extracted item being a word/phrase is usually estimated by the associative strength between constituents of it. Two widely used statistical measures for

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 202–213, 2005.

quantifying the associative strength are frequency and mutual information [1, 2, 7, 10]. Some variations/derivations of these two basic types, log-likelihood for instance, are also exploited [3, 4, 9, 11].

All work so far on Chinese word extraction has depended directly on characters involved in extracted items to measure the associative strength. These approaches ignored an important characteristic of Chinese words: each character of a word is usually meaningful, thus the sense sequence of the involved characters may reflect the semantic constraint 'hidden' in the word to some extent. Consequently, all sense sequences over a lexicon would constitute complete semantic constraints underlying Chinese word-formation. This suggests that semantic constraints in the lexicon implicitly may be helpful for validating Chinese words. The biggest advantage achieved by taking the semantic information into consideration is that we can make certain degree of inference in word extraction. For example, suppose '美军' (American army), '日军' (Japanese army) and '苏军' (Soviet army) are contained in the lexicon, whereas '俄军' (Russian army) is not. We find that all these four words bear the same sense sequence 'country+army' ('美' for the United States, '日' for Japan, '苏' for Soviet Union, and '俄' for Russia), so a hypothesis comes: '俄军' is possibly a word. The idea is simple and straightforward, but it is radically different from previous ones: word extraction will depend heavily on senses of characters, rather than on characters. Furthermore, the associative strength can also be determined statistically using senses of characters. A side effect of doing so is that the data sparseness problem in word extraction may be better settled.

The paper will focus on this novel approach. Section 2 introduces the key linguistic resources used, Section 3 describes the proposed method in detail, and Section 4 gives experimental results and analyses. We conclude in Section 5.

2 Key Linguistic Resources Used

Two key linguistic resources are mainly used in this research: *THSCS*, a thesaurus of Chinese characters, and, *THW2*, a Chinese lexicon.

We firstly developed *THSCS* (short for the Semantic Coding System for Chinese Characters), a thesaurus of Chinese characters. It covers all 6,763 Chinese characters defined in GB-2312, a National Standard of China for Chinese character set in information exchange. In *THSCS*, each character is assigned its possible semantic categories (semantic tags) manually. The principle in designing *THSCS* is that its semantic hierarchy is as compatible as possible with that of *TYCCL*, a well-known thesaurus of Chinese words [8].

There are totally 1,380 semantic categories in *THSCS*. Their distributions are not balanced. As shown in Fig. 1, the most frequent category occurs 927 times, but a majority of categories occur only a few times: 36.4% no more than 5 times, and 87.0% no more than 20 times.

About 54.12% of the 6,763 characters are polysemous according to *THSCS*. Table 1 gives the distributions of these polysemous characters. Note that these polysemous characters are more active than those with single category. An observation over all 32,624 two-character words in *TYCCL* shows that only 1.40% of them do not contain any polysemous characters.

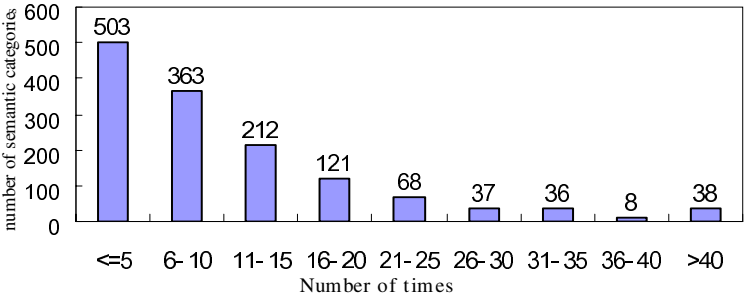


Fig. 1. Distribution of semantic categories in THSCS

Table 1. Distribution of polysemous characters

# of senses per character	# of characters	Percentage in polysemous characters
2	1,556	42.7%
3	787	21.6%
4	457	12.5%
5	285	7.8%
6	181	5.0%
7	124	3.4%
8	79	2.17%
9	51	0.85%
More than 9	123	3.9%

THW2, a lexicon with 63,378 two-character words, is used to learn semantic constraints underlying Chinese word-formation. The reason for choosing two-character words is that they comprise the largest proportion in a Chinese lexicon and represent the most popular word-formation of Chinese.

3 The Proposed Method

3.1 Representing Semantic Constraints in Word-Formation by Hidden Markov Model

Let *C* be the set of Chinese characters, *T* be a thesaurus over *C*, *S* be the set of semantic tags derived from *T*, *W* be the set of Chinese wordlist of two-character words, and *WS* be the set of pairs <word, semantic tags> over *W* in which every character in a

word is assigned a unique semantic tag (though the character may possibly have multiple semantic tags in terms of *T*). Then we can construct a Hidden Markov Model (HMM) accordingly as a five-tuple $WF_{sem} = (S, S_0, C, P_S, P_C)$:

S serves as the set of states; $S_0 \in S$ is the set of initial states associated with the initials of semantic tag sequences of *W*; C serves as the set of output alphabet; $P_S = \{p(s_j \mid s_i)\}$ ($s_i \in S$, $s_j \in S$) is the set of transition probabilities among states; $P_C = \{p(c_k \mid s_i, s_j)\}$ ($s_i \in S$, $s_j \in S$, $c_k \in C$) is the set of observation probabilities.

Both P_S and P_C will be trained using *WS*.

This five-tuple (S, S_0, C, P_S, P_C) describes the semantic constraints in word formation underlying *W* statistically and systematically.

Given any character string c_1c_2 ($c_1 \in C$, $c_2 \in C$), the following derivations hold for $LW(c_1c_2)$, the likelihood of this string being a word, according to properties of HMM and Bayes theorem:

$$\begin{aligned} LW(c_1c_2) &= \sum_{s_1s_2} p(s_1)p(s_2 \mid s_1)p(c_1 \mid s_1, s_2)p(c_2 \mid s_1, s_2) \\ &\approx \sum_{s_1s_2} p(s_1)p(s_2 \mid s_1)p(c_1 \mid s_1)p(c_2 \mid s_2) \\ &= \sum_{s_1s_2} p(s_1, s_2)\frac{p(s_1 \mid c_1)p(c_1)}{p(s_1)}\frac{p(s_2 \mid c_2)p(c_2)}{p(s_2)} \\ &= \sum_{s_1s_2} \frac{p(s_1, s_2)}{p(s_1)p(s_2)} p(s_1 \mid c_1)p(s_2 \mid c_2)p(c_1)p(c_2)\cdot \end{aligned} \tag{1}$$

where s_1s_2 ($s_1 \in S, s_2 \in S$) is any semantic tag sequence generated by c_1c_2 in a combinatorial way.

We ignore $p(c_1)$ and $p(c_2)$ in (1) in order to increase the generalization power of $LW(c_1c_2)$:

$$LW(c_1c_2) = \sum_{s_1s_2} \frac{p(s_1, s_2)}{p(s_1)p(s_2)} p(s_1 \mid c_1)p(s_2 \mid c_2)\cdot \tag{2}$$

For sake of clarity, let:

$$MI^*(s_1, s_2) = \frac{p(s_1, s_2)}{p(s_1)p(s_2)}\cdot \tag{3}$$

then an equivalent of formula (2), denoted $LW_{MI^*}(c_1c_2)$, is obtained consequently:

$$LW_{MI^*}(c_1c_2) = \sum_{s1s2} MI^*(s_1, s_2)p(s_1 \mid c_1)p(s_2 \mid c_2)\cdot \tag{4}$$

Note that $MI^*(s_1, s_2)$ is exactly the inner part of $MI(s_1, s_2)$:

$$MI(s_1, s_2) = \log_2 \frac{p(s_1, s_2)}{p(s_1)p(s_2)}. \tag{5}$$

We thus put forward a variation of formula (4), denoted $LW_{MI}(c_1c_2)$, as an alternative of the likelihood, though the derivation from formula 4 to 6 does not hold mathematically:

$$LW_{MI}(c_1c_2) = \sum_{s_1 s_2} MI(s_1, s_2)p(s_1 \mid c_1)p(s_2 \mid c_2). \tag{6}$$

And, another alternative $LW_P(c_1c_2)$ is presented for the purpose of comparisons:

$$LW_P(c_1c_2) = \sum_{s_1 s_2} p(s_1, s_2)p(s_1 \mid c_1)p(s_2 \mid c_2). \tag{7}$$

Now we have three alternatives for measuring the likelihood of c_1c_2 being a word: $LW_{MI^*}(c_1c_2)$, $LW_{MI}(c_1c_2)$ and $LW_P(c_1c_2)$. We shall choose the most appropriate one in Section 4.

3.2 Estimation of HMM Parameters

If we already have a manually annotated *WS*, the training of WF_{sem} will be easy. Unfortunately, we do not have it yet. In fact, we only have *C*, *T*, *S* and *W*. It is very tough to handcraft such a *WS* because the related linguistic study is poor, resulting in a lack of theoretical preparations necessary to do so. We have to seek for strategies to make some degree of approximations in parameter estimation. We try three schemes.

3.2.1 The Mean Scheme

For any word $w = c_1c_2 \in W$, suppose c_i has n_i possible semantic tags $\{ s_{i,1},...,s_{i,n_i} \}$ according to *T* (*i*=1,2, $n_i \geq 1$):

$$\begin{array}{lll} w = & c_1 & c_2 \\ & s_{1,1} & s_{2,1} \\ & s_{1,2} & s_{2,2} \\ & \cdots\cdots & \\ & s_{1,n_1} & s_{2,n_2} \end{array}$$

The mean scheme will simply set:

$$p(s_{i,j} \mid c_i) = \frac{1}{n_i} \quad (i=1, 2, j=1,\ldots, n_i). \tag{8}$$

Let *f(x)* and *f(x, y)* stand for the number of times *x* occurs and *xy* co-occurs over *W* respectively, then the contribution of semantic tag $s_{i,j}$ of character c_i of this *w* to the frequency counting of P_C would be:

$$f(s_{i,j}) = f(s_{i,j}) + \frac{1}{n_i} \quad (i=1,2). \tag{9}$$

and the contribution of semantic tag sequence $s_{1,j}s_{2,k}$ of this w to the frequency counting of P_S would be:

$$f(s_{1,j}, s_{2,k}) = f(s_{1,j}, s_{2,k}) + \frac{1}{n_1 n_2} \quad (j=1, \ldots, n_1, k=1,\ldots,n_2). \tag{10}$$

We shall obtain P_S and P_C after the above process has been done over all w in W.

3.2.2 The Bias Scheme

The bias scheme will apply the mean scheme first, and then adjust $p(s_{i,j} \mid c_i)$ by the resulting $f(s_{i,j})$:

$$p(s_{i,j} \mid c_i) = \frac{f(s_{i,j})}{\sum_j f(s_{i,j})}. \tag{11}$$

3.2.3 The Baum-Welch Re-estimation Scheme

Baum-Welch re-estimation algorithm is often used in unsupervised learning of HMM parameters [6]. The algorithm is re-paraphrased to fit the need here:

Step 1. Initialize P_S and P_C with the mean scheme.

Step 2. Apply Baum-Welch algorithm one pass through W based on P_S and P_C.

Step 3. Calculate new $P_S^{'}$ and $P_C^{'}$ according to the results of step 2.

Step 4. Let:

$$Q_W^{'} = \sum_{c_1c_2 \in W} \sum_{s_1s_2 \in c_1c_2} p^{'}(s_1, s_2) p^{'}(s_1 \mid c_1) p^{'}(s_2 \mid c_2)$$

$$Q_W = \sum_{c_1c_2 \in W} \sum_{s_1s_2 \in c_1c_2} p(s_1, s_2) p(s_1 \mid c_1) p(s_2 \mid c_2)$$

calculate:

$$\delta_Q = Q_W^{'} - Q_W$$

If $\delta_Q \le \delta_0$ then return $P_S^{'}$ and $P_C^{'}$ as the final solution;

else do $P_S \leftarrow P_S^{'}$, $P_C \leftarrow P_C^{'}$, go to step 2.

where δ_0 is the desired convergence limit to be determined experimentally.

3.3 Static Versus Dynamic Training

Static training refers to the strategy, as described in Section 3.2, that every word w in W is treated equally in estimating P_S and P_C, while dynamic training refers to another strategy that w in W is weighted by its frequency in a large corpus. In dynamic

training, a frequent word in usage will be given a higher weight, and its corresponding semantic tag sequence will play more important role in word-formation. For example, the character '全' belongs to 'the state of fullness or partialness' with semantic tag 'Eb02' in *THSCS*. There exist only two words, '全国'(the whole country) and '全省'(the whole province), with semantic tag sequence 'Eb02+Di02', in *THW2*(Di02 for 'countries or administrative districts'), thus the importance of sequence 'Eb02+Di02' in word-formation is very low in static training. But these two words appear frequently in a corpus, indicating that the word-building ability of this sequence may be under-estimated. Obviously, its importance will be raised a lot in dynamic training.

All formulae in Section 3.2 still hold in dynamic training.

4 Experiments

A series of experiments are carried out to fix the factors of the framework proposed in Section 3. In static training, *THW2* is used as the training data. In dynamic training, all words in *THW2* are weighted by their string frequencies derived from *RCC*, a very huge raw corpus composed of about 1,000M Chinese characters. The open test is performed on *PDA98J*, a manually word-segmented corpus composed of the People Daily of January 1998 with about 1.3M Chinese characters, developed by the Institute of Computational Linguistics, Peking University: all distinct character bigrams excluding proper nouns in *PDA98J* are exhaustively collected, – in total, we obtain 238,946 such bigrams, among which 23,725 are two-character words. These 238,946 character bigrams form the test set, denoted *TS238946*, of experiments.

To better verify the effectiveness of our semantic-tag-based word extraction method, some typical methods based directly on characters rather than semantic tags are also tested in parallel for comparisons. *PDR9596*, a raw corpus composed of the People Daily of 1995 and 1996 with about 50M Chinese characters is used to train character bigrams on these occasions.

4.1 Determining the Most Appropriate $LW(c_1c_2)$

We need to decide which of $LW_{MI^*}(c_1c_2)$, $LW_{MI}(c_1c_2)$ and $LW_P(c_1c_2)$ is most appropriate for measuring the likelihood of a character string c_1c_2 being a word. Here, we use the Baum-Welch re-estimation scheme to estimate WF_{sem}, because the scheme sounds more refined than the other two, the mean and the bias (experimental results in Sections 4.2 and 4.3 will support this assumption). Then we compare the performance of $LW_{MI^*}(c_1c_2)$, $LW_{MI}(c_1c_2)$ and $LW_P(c_1c_2)$ in word extraction on *TS238946* in the context of static training. As shown in Fig. 2, $LW_{MI}(c_1c_2)$ is the best among the three, achieving a slightly better performance than $LW_{MI^*}(c_1c_2)$, though the latter is most rational mathematically. The performance of $LW_P(c_1c_2)$ is the worst, far away from that of $LW_{MI}(c_1c_2)$ and $LW_{MI^*}(c_1c_2)$. We therefore choose $LW_{MI}(c_1c_2)$.

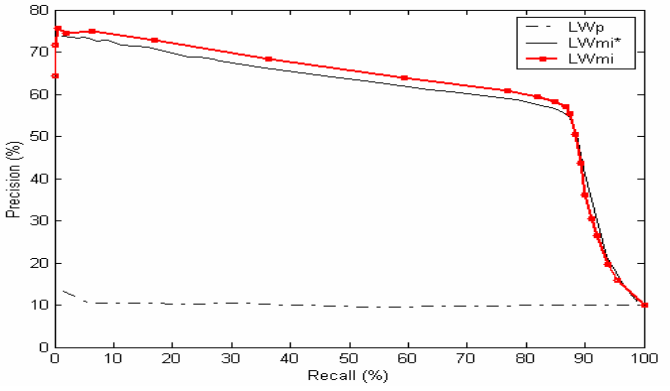


Fig. 2. Performance of $LW_{MI^*}(c_1c_2)$, $LW_{MI}(c_1c_2)$ and $LW_P(c_1c_2)$ in word extraction

4.2 Performance Comparisons Among Various Methods

We experimented with seven candidate methods carefully designed in various settings (the former four are semantic-tag-based, and the latter three are character-based):

- SMean: $LW_{MI}(c_1c_2)$, the mean scheme, static training;
- SBias: $LW_{MI}(c_1c_2)$, the bias scheme, static training;
- SBW: $LW_{MI}(c_1c_2)$, the Baum-Welch re-estimation scheme, static training;
- SDBW: $LW_{MI}(c_1c_2)$, the Baum-Welch re-estimation scheme, dynamic training;
- CP: $p(c_1, c_2)$;
- CMI: $mi(c_1, c_2)$;
- CLL: $\log - likelihood(c_1, c_2)$

Experimental results are given in Fig.3 and Table 2.

The following comparisons can be made based on experimental results from three perspectives:

(1) Comparison among the three schemes for parameter estimation of HMM:

The highest F-measure of SBias, SMean and SBW is 45.0% (at 50.0% recall), 62.0% (at 70.0% recall) and 68.0% (at 80.0% recall), and the 10-point average F-measure is 33.5%, 44.9% and 46.2%, respectively. The fact that SBW increases about 23.0% in the highest F-measure and 12.7% in the average F-measure compared to SBias indicates that the impact of the scheme of HMM parameter estimation on word extraction is obvious. In addition, it is a bit surprise that SBias is much weaker than SMean.

(2) Comparison between static and dynamic training:

The highest F-measure of SDBW is 68.5% (at 80.0% recall), and its 10-point average F-measure is 47.9%. SDBW increases about 0.5% in the highest F-measure and 1.7% in the average F-measure compared to SBW.

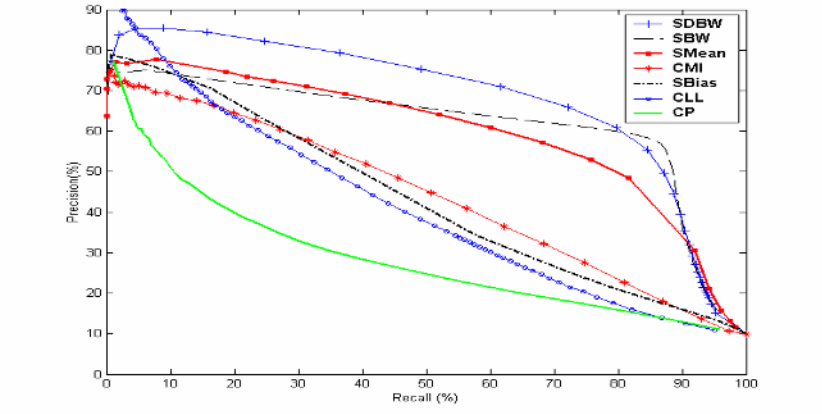


Fig. 3. Performance of various semantic-tag-based and character-based methods

Table 2. 10-point F-measure of various semantic-tag-based and character-based methods

Recall(%)	F-Measure(%)						
	CP	CLL	CMI	SBias	SMean	SBW	SDBW
10	16.7	17.5	17.5	17.5	17.5	17.5	18.0
20	26.6	30.4	30.5	30.5	31.0	30.5	32.0
30	31.5	38.8	39.8	39.0	42.0	40.5	43.0
40	33.3	42.8	45.3	44.0	50.5	50.0	52.5
50	33.0	43.0	47.5	45.0	57.0	56.0	60.0
60	31.5	40.1	46.7	42.3	60.5	62.0	65.0
70	29.4	34.8	43.0	38.5	62.0	65.0	68.0
80	26.5	28.4	36.2	33.0	61.5	68.0	68.5
90	22.5	22.1	26.6	27.2	48.5	54.0	54.0
100	18.1	18.1	18.1	18.1	18.1	18.1	18.1
Average	26.9	31.6	35.1	33.5	44.9	46.2	47.9

Observe a word candidate '全州'(the whole state): its semantic-tag sequence is 'Eb02+Di02'. 'Eb02' is productive in word-formation, resulting in that LW_{MI}(全州) is quite low (-3.26) and therefore rejected to be a word by SBW. However, as stated in Section 3.3, the sequence 'Eb02+Di02' is frequent in dynamic training (because of the presence of '全国' and '全省' in *THW2*), leading to an increasing of LW_{MI}(全州) to 1.04, – '全州' is thus successfully recognized by SDBW.

(3) Comparison between semantic-tag-based and character-based approaches:

The highest F-measure of CP, CLL and CMI is 33.3% (at 40.0% recall), 43.0% (at 50.0% recall) and 47.5% (at 50.0% recall), and the 10-point average F-measure is 26.9%, 31.6% and 35.1%, respectively. CMI outperforms the other two in character-based approaches. Further notice that the performance is improved very significantly as we move from CMI to SDBW: SDBW increases about 21.0% in the highest F-measure and 12.8% in the average F-measure compared to CMI!

Recall the word candidate '俄军' (Russian army) in Section 1: '俄军' occurs only 3 times in *PDR9596*, while its involved characters '俄' and '军' occurs pretty frequently, making CMI(俄军) under 1.00 and rejected to be a word by CMI. In this case, CMI in fact suffers from the data sparseness problem. Our semantic-tag-based approach can resolve this problem in some degree: there exist a number of words with the same semantic-tag sequence 'country+army' in *THW2*, such as '美军'(American army), '日军'(Japanese army) and '苏军'(Soviet army), and those words occur in the corpus quite often, – as a consequence, LW_{MI}(俄军) raises to 4.24 while using SDBW, and '俄军' is accepted as a word.

Summarizing the experimental results, SDBW and SBW outperforms all the other five methods, and SDBW is the best among the all.

4.3 Further Observations on the Baum-Welch Re-estimation Scheme

As said in Section 4.2, both SDBW and SBW explore the Baum-Welch re-estimation scheme to acquire more adequate HMM parameters. Let's have a more detailed look at it.

One look is that the scheme converges after 95 times iteration.

Another look is about why the scheme is quite effective? We tend to partially answer this question from the angle of sense tagging, under an assumption that strong ability in sense disambiguation may lead to good performance in measuring word likelihood. Similar to part-of-speech tagging, we apply Viterbi algorithm to any word $w = c_1c_2$, finding the most likely semantic-tag sequence $s_1's_2'$ for it, according to the HMM obtained from the Baum-Welch re-estimation scheme:

$$
\begin{aligned}
s_1's_2' &= \arg\max_{s_1s_2} p(s_1s_2 \mid c_1c_2) \\
&= \arg\max_{s_1s_2} \frac{p(s_1, s2)}{p(c_1c_2)} p(c_1c_2 \mid s_1s_2) \\
&= \arg\max_{s_1s_2} p(s_1, s_2)\, p(c_1c_2 \mid s_1s_2) \\
&\approx \arg\max_{s_1s_2} p(s_1, s_2)\, p(c_1 \mid s_1)\, p(c_2 \mid s_2) \\
&= \arg\max_{s_1s_2} p(s_1, s_2) \frac{p(s_1 \mid c_1)\, p(c_1)}{p(s_1)} \frac{p(s_2 \mid c_2)\, p(c_2)}{p(s_2)} \\
&= \arg\max_{s_1s_2} \frac{p(s_1, s_2)}{p(s_1)\, p(s_2)} p(s_1 \mid c_1)\, p(s_2 \mid c_2) \\
&= \arg\max_{s_1s_2} MI^*(s_1, s_2)\, p(s_1 \mid c_1)\, p(s_2 \mid c_2).
\end{aligned}
\tag{12}
$$

Note that the inner parts of formulae (4) and (12) are identical.

We randomly extract 2,027 two-character words from *THW2*, and manually annotate those words with a unique semantic-tag sequence each, constituting the test set of

the sense tagging experiment. In the test set, there are totally 4,054 characters, out of them, 3,054 are polysemous. The accuracy of sense tagging is defined as:

$$Accuracy\,1 = \frac{number - of - characters - correctly - tagged}{total - number - of - characters}$$

$$Accuracy\,2 = \frac{number - of - polysemous - characters - correctly - tagged}{total - number - of - polysemous - characters}$$

We take SBias as a baseline of comparison. SBias and SBW will correspond to two classical computational models in part-of-speech tagging, i.e., the unigram model and the bigram model, if we relate sense tagging to part-of-speech tagging. The results are listed in Table 3.

Table 3. SBW and SBias in sense tagging

	Number	SBias	SBW
1	Total number of characters	4054	4054
	Number of correctly tagged characters	1999	2395
	Accuracy1 (%)	49.3	59.1
2	Total number of polysemous characters	3054	3054
	Number of correctly tagged polysemous characters	999	1395
	Accuracy2 (%)	32.7	45.7

The disambiguation ability of SBW is more powerful than that of SBias. This may provide an evidence of why the word extraction performance of the former is much better than the latter. The results also indicate that the difficulty of sense tagging would be larger than that of part-of-speech tagging in Chinese: the bigram models usually achieve over 90% accuracy in part-of-speech tagging, if counted on the total number of words in texts, whereas SBW here can only achieve 59.1% accuracy in sense tagging.

5 Conclusions

This paper presents a semantic-tag-based approach to automatic extraction of two-character words of Chinese. The key feature of this approach is that it tries to capture Chinese word-formation using semantic constraints between characters in words, mainly based on a thesaurus of Chinese characters and a Chinese lexicon. The Baum-Welch re-estimation scheme is used to train parameters of semantic HMM in the way of unsupervised learning. No literature has reported on the similar work so far. The large-scale experiments demonstrate that the proposed method is effective: compared to the character-based methods, the F-measure of SDBW and SBW increases over 20.0%.

Further work will concern some unsolved issues. One issue is on how to minimize the possible negative effect of semantic-tag-based approach. For instance, we use SDBW and CMI to extract 30,000 words out of *TS238946* respectively. SDBW can

recognize 18,568 words and CMI recognize 9,671 words successfully. CMI covers 46.4% of what SDBW has correctly recognized and SDBW covers 89.2% of what CMI has correctly recognized, but SDBW fails to correctly recognize 10.8% of what CMI has correctly recognized. Another issue is on how to expand the method to the task of extracting multiple-character words.

Acknowledgements. This research is supported by the National Natural Science Foundation of China under grant number 60321002 and the National 863 Project of China under grant number 2001AA114210－03.

References

1. Calzolari, N., Bindi, R.: Acquision of Lexical Information from a Large Textual Italian Corpus. In: Proc. of COLING'90, Helsinki, Finland, 1990, 54-59.
2. Chien, L.F.: PAT-tree-based Adaptive Keyphrase Extraction for Intelligent Chinese Information Retrieval. Information Processing and Management, special issue: Information Retrieval with Asian Language, 1998.
3. Daille, B.: Study and Implementation of Combined Techniques Automatic Extraction of Terminology. In: Proc. of the Balancing Act Workshop at 32nd Annual Meeting of the ACL, 1994, 29-36
4. Dunning, T.: Accurate Method for the Statistics of Surprise and Coincidence. Computational Linguistics, 1993, vol 19 no. 1 61-75.
5. Gelbukh, A., Sidorov, G.: Approach to Construction of Automatic Morphological Analysis Systems for Inflective Languages with Little Effort. Lecture Notes in Computer Science, N 2588, Springer-Verlag, (2003) 215-220.
6. Hajic, J.: HMM Parameters Estimation: The Baum-Welch Algorithm. www.cs.jhu.edu/~hajic, 2000.
7. Johansson, C.: Good Bigrams. In: Proc. of COLING'96. Copenhagen, Denmark, 1996
8. Mei, J.J.: Tong Yi Ci Ci Lin. Shanghai Cishu Press, 1983.
9. Merkel, M., Andersson, M.: Knowledge-lite Extraction of Multi-word Units with Language Filters and Entropy Thresholds. In: Proc. of RIAO'2000, Paris, France, 2000, 737-746.
10. Nie, J.Y., Hannan, M.L., Jin, W.: Unknown Word Detection and Segmentation of Chinese Using Statistical and Heuristic Knowledge. Communications of COLIPS, 199, vol 5 47-57
11. Sornlertlamvanich, V., Potipiti, T., Charoenporn, T.: Automatic Corpus-based Thai Word Extraction with the C4.5 Learning Algorithm. In: Proc. of COLING'2000, Saarbrucken, Germany, 2000, 802-807.
12. Sun, M.S., Shen, D.Y., Huang, C.N.: CSeg&Tag1.0: A Practical Word Segmenter and POS Tagger for Chinese Texts. In: Proc. of the 5th Int'l Conference on Applied Natural Language Processing, Washington DC, USA, 1997, 119-126.

Using Directed Graph Based BDMM Algorithm for Chinese Word Segmentation

Yaodong Chen, Ting Wang, and Huowang Chen

National Laboratory for Parallel and Distributed Processing,
Changsha, Hunan, P.R.China 410073
ydchen0104@yahoo.com.cn, wonderwang70@hotmail.com

Abstract. Word segmentation is a key problem for Chinese text analysis. In this paper, with the consideration of both word-coverage rate and sentence-coverage rate, based on the classic Bi-Directed Maximum Match (BDMM) segmentation method, a character Directed Graph with ambiguity mark is designed for searching multiple possible segmentation sequences. This method is compared with the classic Maximum Match algorithm and Omni-segmentation algorithm. The experiment result shows that Directed Graph based BDMM algorithm can achieve higher coverage rate and lower complexity.

1 Introduction

Word segmentation (WS) is a key problem in Chinese text processing. Many methods have been proposed, such as Forward Maximum Match (FMM), Backward Maximum Match (BMM) and Bi-Directed Maximum Match (BDMM), which are fast and simple, but deficiency in disambiguation. The accuracy of segmentation is vital to further processing, such as POS tagging and parsing. Because of the ambiguity in WS, multi-level linguistic knowledge should be considered [1]. It is rational to reserve multi-candidates of segmentation for further processing rather than only one result. On the other hand, too many redundant candidates produced by Omni-segmentation seem to cause low efficiency. Both FMM and BMM are viewed as an extreme in WS (the most simple way, producing only one candidate) and Omni-segmentation is another (the most complex one, producing all the possible candidates but most of which are incorrect), what we need is a tradeoff that gets over the segmentation blindness and the explosion of candidates [2]. This paper applies the BDMM with a character directed graph annotated with ambiguity mark, which aims to include all rational segmentation candidates and exclude the wrong ones for further processing.

2 Directed Graph Based BDMM Algorithm (DGBS)

2.1 The Idea of DGBS

Given a sentence $S = c_1c_2\ldots c_n$ where c_i (i = 1, 2, ..., n) is Chinese character, a segmenting method M can produce a candidate set $T= \{T_1 ,..., T_l\}$ according to S. Let W denote the correct word sequence for S as $w_1w_2...w_m$, and $T_i=w'_1 w'_2 \ldots w'_t$, where both of w_k and w'_j (k=1, 2, ..., m, j = 1, 2…t) are Chinese words.

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 214 – 217, 2005.

Definition 1: A Sentence-Matching for S means $\exists T_i = w'_1 w'_2 ... w'_m \in T$, making $w'_i = w_i$ (i=1, 2, ..., m). If p sentences are sentences-matched in the corpus contains q sentences, the **Sentence-Coverage Rate** (SCR) for M on is p/q**.**

Definition 2: A Word-Matching for w_i (i = 1, 2, ..., m) means $\exists T_i \in T$ and $w'_j \in T_i$ making $w_i = w'_j = c_{x...} c_y$ (1<=x, y<=n). If p' words are word-matched in a corpus contains q' words, the **Word-Coverage Rate** (WCR) for M on is p'/ q'**.**

The SCR illuminates the reciprocity between WS and further steps of text processing better than the accuracy criterion because such steps as parsing, is based on word sequence (i.e. sentence) rather than single word.

The main idea of DGBS is: integrating BDMM and combination ambiguity (CA) mark, multi-candidates of segmentation are obtained by scanning a directed graph in order to improve both the SCR and WCR. The low SCR resulted from most segmenting methods could drop the effect of syntactic parsing though the WCR is relatively high (see Table 1). On the other hand, the result of the Omni-segmentation is too tremendous to be handled by a parser (see Table 2). The DGBS balances correctness of segmentation with the quantity of result. It produces the result as small as possible under the conditions of keeping all possible ambiguities which cannot be disambiguated in WS until further processing such as parsing and semantic analyzing.

All possible CAs in Chinese, as well as their possible segmentation modes, are gathered and stored in lexicon with special mark, with which all the CAs in sentence could be detected. The enumeration of CAs is feasible because their number is finite [3]. We gathered about 1200 CAs automatically from the corpus of People Daily in January 1998 (including 28,603 sentences, 58,201 words). The BDMM is efficient in solving overlapping ambiguity (OA) whose length of ambiguous chain is odd [4]. Based on the statistic of the occurrence frequency of all kinds of ambiguous chain in [4], the BDMM algorithm could detect about 99.7% of OA theoretically.

2.2 The Design of DGBS

The DGBS consists of three parts: constructing directed graph with CA marks (stored in lexicon beforehand) based on the two segmentation sequences produced by BDMM, searching independent chunk (a substring of sentence, but can not be a substring of word in its context) from graph and obtaining multi-candidates.

The directed graph contains nodes (denote characters) and edges (denote connections between characters). There are three kinds of edges: word span edge, word inner edge and CA edge. For example, in sentence "美国会通过进口法案" (means *the American congress pass the import act*.), the ambiguous chunk "美国会" could not be disambiguated in WS until more syntactic or semantic knowledge can help. Fig. 1 shows the conversion of two sequences produced by BDMM into directed graph with CA marks.

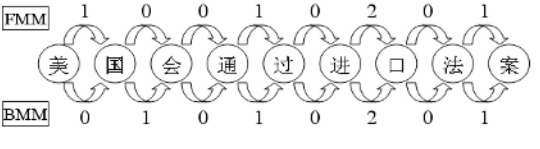


Fig. 1. Directed graph (0 denotes word span edge, 1 denotes word inner edge, 2 denotes CA edge)

Then we search all independent chunks from the graph by the following algorithm.

Algorithm ---Searching Independent Chunks:

Input: two edge sequences in directed graph;
Output: all independent chunks which include several edge sequences (i.e. path);

length denotes the length of input sentence
head denotes the next edge following the last identified chunk, initialized to 1;

for (int *i*=1; *i* ≤ *length*; *i*++)
  if both of the current *i*-th edges of two input sequences are span edges **then**
    1. the edge sequence from *head* to *i* is identified as one chunk;
    2. **if** the two edge sub-sequences are same
      then put them in current chunk;
      else
        2.1 Scan the first edge sub-sequence, **if** type of edge is CA **then**
          Duplicate the paths in chunk, convert CA edge into span edge and inner edge, and append them to tail of paths, respectively;
        2.2 Scan the second edge sub-sequence as the 2.1;
        2.3 Remove the duplicate path in chunk;
    3. set *head* to *i*;

The algorithm produces all the independent chunks for Fig.1 as follows ($Chunk_j[k]$ means the k-th path in the j-th chunk):

$chunk_0[0]$={美国, 会 }, $chunk_0[1]$={美, 国会}, $chunk_1[0]$={ 通过 },
$chunk_2[0]$={ 进口 }, $chunk_2[1]$={ 进, 口 }, $chunk_3[0]$={ 法案 }

Finally, all segmentation candidates are obtained by the combination of all chunks.

candidate[1]={美国/会/通过/进口/法案}, candidate[2]={美/国会/通过/进口/法案},
candidate[3]={美国/会/通过/进/口/法案}, candidate[4]={美/国会/通过/进/口/法案}.

3 Comparisons on the Coverage Rate and the Complexity

We have compared FMM, BMM, BDMM with DGBS on the People Daily Corpus (including 6,987 sentences, 25,607 words) and Lancaster Corpus (including 45,592 sentences, 46,130 words) respectively for both SCR and WCR, as shown in Table 1:

Table 1. Result on the People Daily and Lancaster Corpora

Corpus	Measure	FMM	BMM	BDMM	DGBS
People Daily	Sentence coverage rate	34.01%	35.05%	39.51%	68.54%
	Word coverage rate	90.56%	90.35%	92.07%	95.70%
Lancaster	Sentence coverage rate	48.65%	48.73%	55.02%	75.53%
	Word coverage rate	89.88%	90.01%	92.35%	96.37%

The table indicates both SCR and WCR increase obviously in DGBS. We observe that above 90% of segmentation mistakes in DBGS come from named entities.

Literature [5] discussed the model of Omni-segmentation which has an inevitable disadvantage -- the exponential expansion of its candidates as the sentence length increases. Table 2 shows the complexity between Omni-segmentation and DGBS (for all sentences with the length less than 26 in Lancaster Corpus):

Table 2. Result about complexity on the Lancaster Corpora

	Average of edges	Total of edges	Average of paths	Total of paths
DGBS	8.4390	169675	1.87	37766
Omni-segmentation	16.9309	252393	125.5314	340414

The table indicates the advantage of DGBS: it produces an acceptable quantity of candidates against Omni-segmentation's large garbage candidates. In addition, DGBS can detect most ambiguous chunks very quickly and easily. For instance, a chunk denotes ambiguity if the number of its paths is more than one and the type of ambiguity is identified by the type of chunk's edges.

4 Conclusion

Chinese word segmentation has great effect on the further processing of Chinese text analysis. It is a dilemma whether is prior to efficiency or accuracy. The DGBS, by integrating the BDMM with CA mark to handle both OA and CA, produces multiple segmentation candidates. By the means of reserving the ambiguities which could not be disambiguated in the WS phrase and constructing directed graph, the method not only improve the coverage accuracy of sentence which is valuable for further processing, but also avoid the increase of complexity on a large scale.

This research is supported by the National Natural Science Foundation (60403050) of China.

References

1. Andi Wu, Zixin Jiang. Word Segmentation in Sentence Analysis. In Proc. of 1998 International Conference on Chinese Information Processing, Beijing China, 1998, 169–180.
2. Maosong Sun, Jiayan Zou: A review of the study of Chinese automatic segmentation. Journal of Modern Language, 3, 2001, 22–32
3. Kaiying Liu. Chinese text automatic segmentation and tagging. Commercial Press, Beijing, 2000
4. Yintang Yan, Xiaoqiang Zhou. Study of Segmentation Strategy on Ambiguous Phrases of Overlap Type. Journal of the China Society for Scientific and Technical Information. Vol. 19:6, 2000
5. Jiancheng Wan, Chunhua Yang. An Algorithm Model of Word Omni-segmentation for Written Chinese. Mini-micro Systems V.23:7, 2003, 1247–125.

Entity-Based Noun Phrase Coreference Resolution

Xiaofeng Yang, Jian Su, and Lingpeng Yang

Institute for Infocomm Research, 21 Heng Mui Keng Terrace, Singapore, 119613
{xiaofengy, sujian, lpyang}@i2r.a-star.edu.sg

Abstract. In this paper we propose an NP coreference resolution system which does resolution on the entity-level. The framework of the system is presented and different resolution strategies are investigated.

1 Introduction

Coreference resolution is the process of linking multiple expressions which refer to the same entity. Traditional supervised machine learning approaches (e.g. [1, 2, 3]) do resolution based on the mention-level. Specifically, a pairwise classifier is learned and used to determine whether or not two NPs in a document refer to the same entity in the world. However, as an individual mention usually lacks adequate information about its referred entity (e.g, we could not know the gender or the name of "the president"), it is often difficult to determine whether or not two NPs refer to the same entity simply from the pair itself. Recent research ([4, 5]) has revealed that entity information could help resolution. In our work we would like to further study how to effectively incorporate the entity information into coreference resolution. The framework of such a entity-based system is presented and different resolution strategies are investigated in this paper.

2 Baseline: A Mention-Based System

We built a Mention-Mention based system as the baseline, which adopts a learning framework similar to the paradigm proposed by Soon et al. [2].

Each instance takes the form of $i\{\mathrm{NP}_i, \mathrm{NP}_j\}$, which is associated with a feature vector consisting of 12 features ($f_1 \sim f_{12}$) as described in Table 1. During training, for each anaphor NP_j in a given text, a positive instance is generated by pairing NP_j with its closest antecedent. A set of negative instances is also formed by NP_j and each NP occurring between NP_j and NP_i.

When the training instances are ready, a classifier is learned by C5.0 algorithm [6]. During resolution, each encountered noun phrase, NP_j, is paired in turn with each preceding noun phrase, NP_i. For each pair, a testing instance is created and then presented to the decision tree, which returns a confidence value (CF) indicating the likelihood that they co-refer. NP_j will be linked to the NP with the maximal CF (above 0.5).

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 218–221, 2005.

Table 1. The features used in the coreference resolution system

Features describing the relationships between NP_j and NP_i	
1. Type_1	the type of NP_j (Indefinite NP, Definte NP, Pronoun, ...)
2. Type_1	the type of NP_j (Indefinite NP, Definte NP, Pronoun, ...)
3. NumAgree	NP_i and NP_j are compatible in number
4. GenderAgree	NP_i and NP_j are compatible in gender
5. Sdist	the distance between NPi and NPj in sentences
6. Pdist	the distance between NPi and NPj in paragraphs
7. Appositive	NP_i and NP_j are in an appositive structure
8. NameAlias	NP_i and NP_j are in an alias of the other
9. HeadStrMatch	NP_i and NP_j contain the same head string
10. FullStrMatch	NP_i and NP_j contain the same string
11. StrSim_1	The string similarity of NP_j against NP_i
12. StrSim_2	The string similarity of NP_i against NP_j
Features describing the relationships between NP_j and ENT_i	
13. C_NumAgree	NP_j is compatible in number with any mention of ENT_i
14. C_GenAgree	NP_j is compatible in gender with any mention of ENT_i
15. C_Appositive	NP_j is in an appositive structure with a mention of ENT_i
16. C_NameAlias	NP_j is in an alias of a mention of ENT_i; else 0
17. C_HeadStrMatch	NP_j contains the same head string as a mention of ENT_i
18. C_FullStrMatch	NP_j contains the same string as a mention of ENT_i
19. C_MaxStrSim	The maximal string similarity between NP_j and the mentions of ENT_i
20. C_StrSim	The string similarity of NP_j against ENT_i

3 The Entity-Based System

3.1 Instance Representation

An instance in our approach has the form of $\boldsymbol{i}\{ENT_i, ENT_j\}$, where ENT_i and ENT_j are two partial entities under consideration.

In our system, each instance is represented as a set of 20 features as shown in Table 1. The features are supposed to capture the properties and relationships between two entities. Note that here NP_i is the last mention in ENT_i, while NP_j is the first mention in ENT_j.

An instance is labelled as positive if ENT_i and ENT_j are of the same entity, or negative if otherwise.

3.2 Training Procedure

Given an annotated training document, we process the noun phrases from beginning to end. For each anaphoric noun phrase NP_j, we represent it as a partial entity ENT_j. ENT_j will be paired with each preceding coreferential chain, ENT_i, to form a training instance. The process continues until the chain to which ENT_j belongs is found.

3.3 Resolution Procedure

The resolution could be thought of as a clustering problem. Initially, each NP in a given documents is represented as a single cluster, and then small clusters referring to the partial entities are merged together to form a complete entity.

We use two similarity metrics to evaluate the likelihood that two partial entity, ENT_i and ENT_j, are co-referring:

- **Single Similarity**: it simply uses the confidence returned by the classifier. Suppose function CF is the confidence value of an instance

$$Similarity(ENT_i, ENT_j) = CF_{i\{ENT_i, ENT_j\}} \tag{1}$$

- **Maximal Similarity**: ENT_i is divided into several sub_clusters. The similarity is the maximal confidence between ENT_j and the sub_clusters. Specifically, Suppose ENT_i contains k mentions, $M_{i1}, M_{i2}, \ldots, M_{ik}$. Let $SubSet_i = \{ENT_{id} | ENT_{id} = \{M_{i1}, \ldots, M_{id}\}, 1 \leq d \leq k\}$, then

$$Similarity(ENT_i, ENT_j) = \max_{ENT_{id} \in SubSet_i} CF_{i\{ENT_{id}, ENT_j\}} \tag{2}$$

And three clustering strategies are considered to group the partial entities:

- **Simple Clustering**: Each cluster is simply merged to the best preceding cluster with the highest similarity (above 0.5), if any.

- **Incremental Clustering**: Clusters are processed from left to right. A cluster is merged into the best preceding cluster, if any, before proceeding to subsequent ones.

- **Greedy Clustering**: Clustering is done iteratively. In each iteration, every two clusters are tested and the pair with the highest similarity is merged together. The iteration continues until no remaining clusters could be merged.

4 Evaluation and Discussion

In our study we used the standard MUC-6 and MUC-7 coreference corpora. In each data set, around 30 "dry-run" documents were annotated for training as well as 20-30 documents for testing.

In the experiments we evaluated our system under the two similarity metrics and the three clustering strategies. The performance is listed in Table 2. The Recall and Precision were calculated based on the standard MUC coreference resolution scoring scheme [7].

The baseline system produces the F-measure of 60.7% (MUC-6) and 63.7% (MUC-7). The score is similar to that of Soon et al.'s system (62.6% and 60.4%).

Compared with the baseline, our entity-based system obtains large gain (7.1% for MUC-6 and 2.5% for MUC-7) in Precision, with slight loss (less than 1%)

Table 2. Experimental Results

	MUC-6						MUC-7					
	Single			Maximal			Single			Maximal		
	R	P	F	R	P	F	R	P	F	R	P	F
Baseline	**65.2**	56.8	60.7				**68.4**	59.5	63.6			
Simple	64.7	60.2	62.3	64.7	60.2	62.3	67.8	60.4	63.9	67.8	60.4	63.9
Incremental	63.7	63.5	63.6	64.8	59.8	62.3	66.2	61.8	64.2	67.8	**62.4**	**65.0**
Greedy	63.4	**63.9**	**63.7**	64.7	59.8	62.2	66.5	62.0	64.2	67.8	**62.4**	**65.0**

in Recall. Overall, the system achieves F-measure up to about 3% higher than the baseline. This result indicates that our entity-based system is effective for coreference resolution.

From the table, the performance difference under the two similarity metrics is obscure. For MUC-6, *single-similarity* is slightly better than *maximal-similarity*, while the latter seems to be superior for MUC-7.

In comparing the three clustering strategies, we observe no apparent performance difference between *incremental-clustering* and *greedy-clustering*. By contrast, in most cases these two clustering methods outperform *simple-clustering*, especially in Precision. It should be due to the fact that simple clustering does not use the entity information during resolution. The results further prove that entity information will help not only training, but also resolution.

References

1. McCarthy, J., Lehnert, Q.: Using decision trees for coreference resolution. In: Proceedings of the 14th International Conference on Artificial Intelligences. (1995) 1050–1055
2. Soon, W., Ng, H., Lim, D.: A machine learning approach to coreference resolution of noun phrases. Computational Linguistics **27** (2001) 521–544
3. Ng, V., Cardie, C.: Improving machine learning approaches to coreference resolution. In: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, Philadelphia (2002) 104–111
4. Yang, X., Su, J., Zhou, G., Tan, C.: An np-cluster approach to coreference resolution. In: Proceedings of the 20th International Conference on Computational Linguistics, Geneva (2004)
5. Luo, X., Ittycheriah, A., Jing, H., Kambhatla, N., Roukos, S.: A mention-synchronous coreference resolution algorithm based on the bell tree. In: Proceedings of 42th Annual Meeting of the Association for Computational Linguistics. (2004) 135–142
6. Quinlan, J.R.: C4.5: Programs for machine learning. Morgan Kaufmann Publishers, San Francisco, CA (1993)
7. Vilain, M., Burger, J., Aberdeen, J., Connolly, D., Hirschman, L.: A model-theoretic coreference scoring scheme. In: Proceedings of the Sixth Message understanding Conference (MUC-6), San Francisco, CA, Morgan Kaufmann Publishers (1995) 45–52

The Right Frontier Constraint as Conditional

Claudia Sassen[1] and Peter Kühnlein[2]

[1] Univ. Dortmund
claudia.sassen@uni-dortmund.de
[2] Univ. Bielefeld
p@uni-bielefeld.de

Abstract. The *Right Frontier Constraint* (RFC) claims that antecedents are only available for anaphoric reference if they are located at the right hand side of any level of a linearly ordered discourse parse tree. We show that this constraint does hold only under certain conditions — which, however, apply for most circumstances of everyday talk. The data of our analysis in which the RFC does not hold come from a corpus of chat communication. From our findings we argue that the RFC is best viewed as a conditional constraint.

Most theories of discourse employ one or another way of respecting the *Right Frontier Constraint* (RFC). Polanyi ([1988]) for instance already explicitly built her LDM to respect the RFC, as well as more recent grammars of discourse do (Gardent [1998], Asher and Lascarides [2003]).

An example where the RFC applies is the following short discourse:

(1) a. Max had a great evening yesterday.
b. He had a great meal.
c. He ate salmon.
d. He devoured lots of cheese.
e. He then won a dancing competition.

Example 1 has to be analysed as follows: (1a) is elaborated by (1b) and (1e), which in turn form a narration. (1b) is elaborated by (1c) and (1d), again a narrating sequence. Attempting to attach the sentence

(1) f. It was a beautiful pink.

to the discourse above intuitively and in accordance with the RFC results in a reduced acceptability. The only semantically adequate antecedent, *salmon* in (1c), is not at the right frontier of the discourse and, hence, blocked.

Sassen ([2005]) explored whether chat communication, as an instance of a non-traditional communication system makes an exception when it comes to the RFC. The data used for the analysis was taken from 28 logfiles of the Allegra Chat, a chit chat that has ceased to exist and 8 extracts from the advisory chat of the BeraNet (http://www.beranet.de/). The Allegra-Chat offers its users a whisper lounge, i.e. the opportunity to communicate privately, of which whispered messages could be integrated into the analysis.

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 222–225, 2005.

In order to assess Sassen's procedure, it is helpful to have a rough understanding of Polanyi's LDM. According to Polanyi ([1988]), a discourse is made up from *discourse constituent units* (DCUs), which can either be atomic utterances or be recursively embedded. This results in discourse parsing trees which assign each discourse a structural description on a left-to-right and sentence by sentence base. These allow to make predictions about those discourse units which are structurally available and which are not available as an attachment point Polanyi ([1988]: 611).

For purposes of illustration, we render a Polanyi-type parse tree for the chat fragment (Table 1), see Figure 1. The Polanyi-type tree is an efficient way of representing and tracing violations of the RFC. The nodes of the tree are coordinated or subordinated with regard to others; these relations are the results of discourse relations that obtain.

Sassen ([2005]) applies Polanyi's LDM to chat communication on the explicit assumption of similarity between chat communication and traditional communication systems, in particular spoken language. This assumption, however, is largely undisputed (Yates [1996]). The application of the LDM to chat can also be maintained since Polanyi intended her model for the representation of arbitrary discourse scenarios such as question-answer sequences in service encounters or doctor-patient communication (cp. Polanyi ([1988]: 603)).

The distinction between two communicative units is important for Sassen's analysis of chat, viz. *chat contribution* (and *move*.) A *chat contribution* is an utterance framed by a preceding and subsequent carriage return and hence represents a formal unit. A *move* is a pragmatic unit constituted by its propositional content and illocutionary function, realised by at least one chat contribution. In Figure 1, which represents the parse tree of an advisory chat fragment from Sassen ([2005]) in Table 1, the leaves are contributions.

The dashed arrows in Figure 1 indicate the temporal order in which the contributions were logged from the advisor's perspective. The contributions are

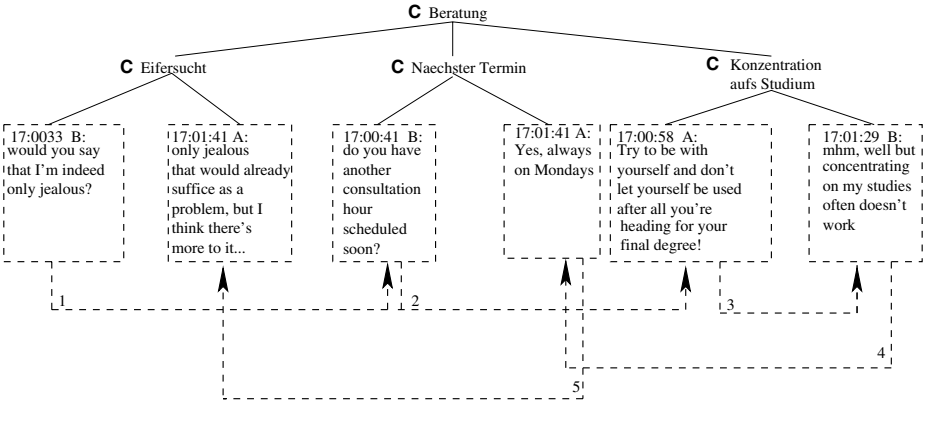


Fig. 1. Parse tree of an advisory chat from ([2005]), cf. Table 1. The labels assigned to the arrows indicate the original order of the contributions in the logfile

Table 1. Translated chat fragment from the advisory chat of the BeraNet A = consultant, B = the person consulted

17:00:33 B:	would you say that I'm indeed only jealous?
17:00:41 B:	do you have another consultation hour scheduled soon?
17:00:58 A:	Try to be with yourself after all you're heading for your final degree!
17:01:29 B:	mhm, well, but concentrating on my studies often doesn't work
17:01:41 A:	yes, always on Monday. only jealous that would already suffice as problem, but I think there is more to it
17:01:57 A:	For this reason, go to the advisory service

grouped according to the DCUs to which they belong. Hence, crossing dashed arrows indicate RFC violations.

What results from Sassen's analysis is that for the public part of the Allegra chat not one single RFC-violation could be diagnosed; however, one instance of an RFC-violation could be found in the whisper lounge. The advisory chat displayed a relatively high density of three RFC-violations compared to the small amount of DCUs communicated. An explanation of this phenomenon runs as follows: whisper lounges are designed for establishing particular contacts and so are advisory chats. Because of the written form of chat communication which is quite awkward compared to spoken interaction there are no backchannel signals. The initiation of a new DCU in order to avoid anticipating responses to the preceding DCU and to wait for the reaction of the interlocutor is an option to keep up the communication, bridge pauses and compensate for the missing backchannel options. In whisper lounges and advisory chats RFC-violations are apparently motivated by the desire to keep the communication channel open and to signal that the interlocutor is still there. For advisory chats this necessity is particulary evident.

In chats of a lower pressure to maintain contact, hardly any violations of the RFC could be located. It seems that in public chats there is a awareness of the communicative actions of the others and chatters pursue them. Whenever ambiguities affect the communication in which they participate, chatters seem to avoid simple pronominal reference and instead use more complex expressions.

The RFC is thus conceived as a conditional constraint that restricts possible antedecents of anaphora to sit on the right frontier only if there is low pressure to keep the communicative channel open. It operates, so conceived, at the interface between pragmatics, syntax and semantics. Accordingly, we re-write the right-frontier constraint as

$$\text{RFC}_c =_{def} \mathit{cond} \rightsquigarrow \text{RFC} \tag{1}$$

where *cond* expresses the condition "absence of pressure" and "RFC" (without subcript) is the "classical" RFC.

Asher and Lascarides ([2003]) discuss an example of RFC violation that is reminiscent of the cases Sassen found in her data. They propose to *modify availability in* SDRT to capture those *structures involving questions that make a strict*

right-frontier constraint unworkable. They *propose that full answers in a* single *turn recapitulate enough of the material in the question that they can attach* [via an indirect question-answer pair relation] *to* any *question node that was available at the start of the turn.* This proposal is unsatisfactory for two reasons: First, the sample chat fragment in Table 1 shows instances of answers to questions which do not in any obvious sense contain *enough of the material* that they could easily attach. Not a single word of the corresponding question is repeated in the answer *yes, always on Monday*. Second, the definition of the modification of availability makes in turn use of availability. According to the RFC at least, the question was blocked.

On the other hand, conceiving the RFC as a conditional constraint in the given sense helps explain why its violation in Asher's and Lascarides' example is tolerable:

(2) a. A: Where were you on the 15th?
b. B: Uh, let me think.
c. A: Do you remember talking to anyone right after the incident?
d. B: I was at home.
I didn't talk to anyone after the incident.

Surely, (2d) is a dialogue produced under pressure. In fact, it sounds like an example from an investigatory inquiry. Besides, the reading that B was at home on the 15th is not the only possible one. For an alternative, imagine the dialogue to consist only of the last three lines. Clearly, what B would be saying is that *after the incident* s/he was at home and there not talking to anyone. For the same effect, imagine a longer pause between B's first answer and the query about B's talking to anyone. But longer pauses are indicative exactly of low pressure to maintain the communication channel — and, thus, the presence of the condition that enables the right-hand side of the RFC_c.

To sum up, using data from a corpus of chat logs we have argued that the (classical) RFC does hold under certain conditions only. This has led us to the reformulation of the right frontier constraint as a conditional constraint that works at the interface of pragmatics, syntax and semantics.

References

[2003] Asher, N., Lascarides, A.: Logics of Discourse. Cambridge UP (2003)
[1998] Gardent, C.: Discourse tree adjoining grammar. CLAUS report 91, Univ. des Saarlandes (1998)
[1988] Polanyi, L.: A formal model of the structure of discourse. Journal of Pragmatics **12** (1988) 601–38
[2005] Sassen, C.: The Right Frontier in Chat Communication. Zeitschrift für Germanistische Linguistik (2005)
[1996] Yates, S.: Oral and written linguistic aspects of computer conferencing. In Herring, S., ed.: Computer-Mediated Communication. John Benjamins (1996) 29–47

Name Discrimination by Clustering Similar Contexts

Ted Pedersen[1], Amruta Purandare[2], and Anagha Kulkarni[1]

[1] University of Minnesota, Duluth, MN 55812, USA
[2] University of Pittsburgh, Pittsburgh, PA 15260, USA
http://senseclusters.sourceforge.net

Abstract. It is relatively common for different people or organizations to share the same name. Given the increasing amount of information available online, this results in the ever growing possibility of finding misleading or incorrect information due to confusion caused by an ambiguous name. This paper presents an unsupervised approach that resolves name ambiguity by clustering the instances of a given name into groups, each of which is associated with a distinct underlying entity. The features we employ to represent the context of an ambiguous name are statistically significant bigrams that occur in the same context as the ambiguous name. From these features we create a co–occurrence matrix where the rows and columns represent the first and second words in bigrams, and the cells contain their log–likelihood scores. Then we represent each of the contexts in which an ambiguous name appears with a second order context vector. This is created by taking the average of the vectors from the co–occurrence matrix associated with the words that make up each context. This creates a high dimensional "instance by word" matrix that is reduced to its most significant dimensions by Singular Value Decomposition (SVD). The different "meanings" of a name are discriminated by clustering these second order context vectors with the method of Repeated Bisections. We evaluate this approach by conflating pairs of names found in a large corpus of text to create ambiguous pseudo-names. We find that our method is significantly more accurate than the majority classifier, and that the best results are obtained by having a small amount of local context to represent the instance, along with a larger amount of context for identifying features, or vice versa.

1 Introduction

The problem of name ambiguity exists in many forms. It is common for different people to share the same name. For example, there is a George Miller who is a prominent Professor of Psychology, another who is a Congressman from California, and two more who are film directors from Australia. Locations may have the same name. For example, Duluth is a city in Minnesota and also a city in Georgia. The acronyms associated with organizations may also be ambiguous. UMD can refer to the University of Michigan – Dearborn, the University of Minnesota, Duluth or the University of Maryland .

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 226–237, 2005.

The effects of name ambiguity can be seen when carrying out web searches or retrieving articles from an archive of newspaper text. For example, the top 10 hits of a Google search for "George Miller" mention five different people. While it may be clear to a human that the Congressman from California, the Professor from Princeton, and the director of the film *Mad Max* are not the same person, it is difficult for a computer program to make the same distinction. In fact, a human may have a hard time organizing this information such that they find all the material relevant to the particular person they are interested in.

The problem of grouping occurrences of a name based on the underlying entity's identity can be approached using techniques developed for word sense discrimination. This is the process of examining a number of sentences that contain a given polysemous word, and then grouping those instances based on the meaning of that word. Note that this is distinct from word sense disambiguation, which is the process of assigning a sense to a polysemous word from a predefined set of possibilities, usually defined by a dictionary or some other well established resource. However, it is not likely that we will have a complete inventory of the possible identities associated with each name, so our immediate objective is to group the occurrences of a name into clusters based on the underlying identity. We are currently developing methods that will examine the content of each cluster to automatically create a descriptive label that will identify the entity represented. This paper is only concerned with discriminating among the different entities, while the labeling step is an area of ongoing work for us.

Approaches to word sense discrimination generally rely on the strong contextual hypothesis of Miller and Charles [10], who hypothesize that words with similar meanings are often used in similar contexts. This is equally true for names, where a particular entity will likely be mentioned in certain contexts. For example, George Miller the film director may not be mentioned with Princeton University very often, while George Miller the Professor will be. Thus, our approach to name discrimination reduces to the problem of finding classes of similar contexts such that each class represents a distinct entity. In other words, contexts that are grouped together in the same class represent a particular entity.

In this paper we show how the unsupervised word sense discrimination methods of Purandare and Pedersen (e.g., [12], [13]) can be applied to the problem of name discrimination. We begin with a summary of related work on the problem of name discrimination, and then describe our approach, which is based on clustering second-order context vectors whose dimensions have been reduced by Singular Value Decomposition (SVD). We present an evaluation of our approach based on pseudo-names that we create by conflating two related names in a large corpus of newswire text.

2 Related Work

The problem of name discrimination is a natural extension to work that identifies named entities in text. This was shown in early work by Wacholder, et. al. [15], who developed an integrated approach to identifying named entities and resolv-

ing any ambiguities that might be present based on knowledge of co–occurring names in the context, and a database of known names.

Cross document co–reference resolution is closely related to name discrimination, in that it seeks to resolve referents across multiple documents. There are several variations to this problem. For example, there may be multiple forms of the same name (*J. Smith* and *John Smith* and *Mr. Smith*), or there may be titles, pronouns, etc. that refer to an entity (*J.Smith, the President, him*). We focus on the more specific problem of identifying which entities the particular form of a name refer to. For example, *John Smith* may be mentioned in 30 documents. Our objective is to determine how many different individuals this entails. While we do not explicitly find chains of references, in fact this would be easy to reconstruct from our results since each occurrence of a name will appear in a cluster. All of the members of a single cluster can then be considered to form a chain of references.

Bagga and Baldwin [1] propose a method based on creating first order context vectors that represent each instance in which an ambiguous name occurs. Each vector contains exactly the words that occur within a 55 word window around the ambiguous name, and the similarity among names is measured using the cosine measure. In order to evaluate their approach, they created the *John Smith* corpus, which consists of 197 articles from the New York Times that mention 35 different *John Smiths*.

Gooi and Allan [5] present a comparison of Bagga and Baldwin's approach to two variations of their own. They used the *John Smith* Corpus, and created their own corpus which is called the *Person-X* corpus. Since it is rather difficult to obtain large samples of data where the actual identity of a truly ambiguous name is known, the *Person-X* corpus consists of pseudo-names that are ambiguous. These are created by disguising known names as *Person-X*, thereby introducing ambiguities. There are 34,404 mentions of *Person-X*, which refer to 14,767 distinct underlying entitles. Gooi and Allan re–implement Bagga and Baldwin's context vector approach, and compare it to another context vector approach that groups vectors together using agglomerative clustering. They also group instances together based on the Kullback–Liebler Divergence. Their conclusion is that the agglomerative clustering technique works particularly well.

Mann and Yarowsky [9] have proposed an approach for disambiguating personal names using a Web based unsupervised clustering technique. They rely on a rich feature space of biographic facts, such as date or place of birth, occupation, relatives, collegiate information, etc. A seed fact pair (e.g., Mozart, 1776), is queried on the Web and the sentences returned as search results are used to generate the patterns which are than used to extract the biographical information from the data. Once these features are extracted clustering follows. Each instance of an ambiguous name is assigned a vector of extracted features, and at each stage of cluster the two most similar vectors are merged together to produce a new cluster. This step is repeated until all the references to be disambiguated are clustered.

There has also been work on name disambiguation using supervised learning approaches in a number of different domains. These approaches rely on having some number of examples available, where the underlying entity for an ambiguous name is known prior to learning.

For example Han et. al. [6] address the problem of resolving ambiguity in bibliography entries, such as *J. Smith* versus *John Smith* versus *J.Q. Smith*. They rely on the use of co–occurrence relations among the names. For example, if *J.Q. Smith* and *Johnny Smith* both wrote articles with *H. L. Hutton*, then they might conclude that *J.Q.* and *Johnny* are one in the same. They compare the use of Naive Bayesian classifiers and Support Vector Machines, and conclude that both methods are effective in certain circumstances.

Name disambiguation is also a problem in the medical domain. For example, Hatzivassiloglou, et. al. [7] point out that genes and proteins often share the same name, and that it's important to be able to identify which is which. They employ a number of well known word sense disambiguation techniques and achieve excellent results. Ginter, et. al. [4] develop an algorithm for disambiguation of protein names based on weighted features vectors derived from surface lexical features and achieve equally good results.

3 Discrimination by Clustering Similar Contexts

Purandare and Pedersen (e.g., [12], [13]) have developed methods of clustering multiple occurrences of a given word into senses based on their contextual similarity. In this paper we adapt those techniques to the problem of name discrimination.

We begin by collecting some number of instances of an ambiguous name. Each instance consists of approximately 50 words, where the ambiguous name is found in the center of the context.

Then we identify significant bigrams in the contexts to be clustered[1]. A bigram is a sequence of two words that may or may not be adjacent. In our work we generally allow bigrams to be an ordered pair of non–consecutive words and permit one intermediate word between them, or bigrams with a window size of 3. A bigram is judged significant by measuring the log–likelihood ratio between the two words. If that score is greater than 3.814 then the bigram is significant and selected as a feature. Note that we employ a technique known as *OR stop–listing* and remove any bigram that is made up of one or two stop–words. Thus, the bigrams we select are made up of two content words.

We build a matrix based on the set of significant bigrams that we identify. The rows in this matrix represent the first word in the bigram, and the columns represent the second word. Each cell in the matrix contains the log–likelihood ratio associated with the bigram represented by the row and column. Thus, each

[1] It would be possible to identify these features in a separate large corpus of training data. However, in this work we are identifying the features in the instances that are to be clustered.

row of this matrix can be viewed as a word vector made up of log–likelihood ratios, where the word is represented by words with which it co–occurs. Since the bigrams are ordered, this matrix is not symmetric. This matrix is also very sparse, since many words that form bigrams only occur with a small number of other words.

Because of it's large size and sparsity, we employ Singular Value Decomposition (SVD) to reduce the dimensionality. We reduce the matrix to 10% of it's original number of columns, or 300 columns, whichever is least. Thus, any matrix of 3,000 or more columns will be reduced to 300 columns, while those less than 3,000 columns are reduced to 10% of their number of columns. Note that SVD reduces the number of columns, but not the number of rows. The reduction has two effects. First, it acts as a smoothing operation, where the resulting matrix will have very few (if any) zero values. Second, it has the effect of reducing the words that make up the columns from a word level feature space into a concept level semantic space.

The SVD reduced bigram matrix is used to create *second order context vectors* ([14]) that will represent the instances to be clustered. Each word in the context of the ambiguous name that has a row vector in the SVD reduced matrix will be represented by that vector. All the vectors associated with the context that are found in the SVD reduced matrix are averaged together to create an overall representation of the context.

The use of SVD and the averaging of word vectors to create a second order context representation has been employed by Schütze ([14]) in the context of word sense discrimination research, and in Latent Semantic Analysis [8], and Latent Semantic Indexing [2]. Our approach is certainly related to this, although our use of bigram features and the log–likelihood scores makes it somewhat distinct, since the usual technique is to create a word co–occurrence matrix that employs frequency counts.

The general intuition behind the second order representation is that it captures indirect relationships between words. For example, suppose that the word *shoot* forms significant bigrams with the words *murder*, *bullets*, and *weapon*, and that *gun* forms significant bigrams with *fire*, *bullets*, and *murder*. Our intuitive understanding that *shoot* and *gun* are related is confirmed by the shared second order relationships they have with *murder* and *bullets*.

4 Clustering

Once the instances to be discriminated are represented by second order context vectors, they are clustered such that the instances that are similar to each other are placed into the same cluster.

Clustering algorithms are typically classified into three main categories: hierarchical, partitional, and hybrid. It is generally believed that the quality of clustering by partitional algorithms such as k–means is inferior to that of the agglomerative methods such as average link. However, a recent study by Zhao and Karypis [16] has suggested that these conclusions are based on experiments

conducted with smaller data sets, and that with larger data sets partitional algorithms are not only faster but lead to better results.

In particular, Zhao and Karypis recommend a hybrid approach known as Repeated Bisections. This overcomes the main weakness with partitional approaches, which is the instability in clustering solutions due to the choice of the initial random centroids. Repeated Bisections starts with all instances in a single cluster. At each iteration it selects one cluster whose bisection optimizes the given criteria function. The cluster is bisected using standard K-means method with K=2, while the criteria function maximizes the similarity between each instance and the centroid of the cluster to which it is assigned. As such this is a hybrid method that combines a hierarchical divisive approach with partitioning.

5 Experimental Data

Our experimental data is made up of six pairs of pseudo-names that are generated by identifying pairs of names that occur in a large corpus of newswire text. Six pairs of names were selected that represent different frequency distributions and types of names. Once selected, all of the instances associated with each pair were extracted from the corpus and placed in separate files (one file per pair). Each instance consists of approximately 25 words to the left and right of the ambiguous name. After the pairs were extracted, they were conflated in each file by creating an obfuscated form of the name that is used in place of both names. For example, one of our pairs was "David Beckham" and "Ronaldo". All of the instances in the corpus that included either name were extracted, and then all occurrences of both name were replaced with the obfuscated form "RoBeck". Discrimination is then carried out in a completely unsupervised way, meaning that we don't use the knowledge of the correct name until evaluation.

The corpus employed in these experiments is the Agence France Press English Service (AFE) portion of the GigaWord English Corpus, as distributed by the Linguistic Data Consortium. The AFE corpus consists of 170,969,000 words of English text which appeared in the AFE newswire from May 1994 to May 1997, and from December 2001 until June 2002. In all this represents approximately 1.2 GB of text (uncompressed).

The pairs of names we selected and their frequency of occurrence are shown in Table 5. This also shows the combined frequency of the pseudo-name, and the percentage which the more common of the two names occurs in reality. This last value represents the majority class, and is the level of accuracy that a baseline clustering algorithm could achieve by simply placing all instances in one cluster.

These pairs were selected to try and force our methods to make relatively fine grained distinctions between the words/senses that make up the pair. One known drawback of pseudo–words arises when the component words are randomly selected. In such a case, it is very likely that the two senses represented will be quite distinct ([3]). We have adopted a solution to this problem that is somewhat similar to that of Nakov and Hearst [11], who suggest creating psuedo words of words that are individually unambiguous, and yet still related in some way.

Table 1. Conflated Pairs of Names

Name1	Count1	Name2	Count2	Conflated	Total	Majority
Ronaldo	1,652	David Beckham	740	RoBeck	2,452	69.3%
Tajik	3,002	Rolf Ekeus	1,071	JikRol	4,073	73.7%
Microsoft	3,401	IBM	2,406	MSIBM	5,807	58.6%
Shimon Peres	7,846	Slobodan Milosevic	6,176	MonSlo	13,734	56.0%
Jordan	25,539	Egyptian	21,762	JorGypt	46,431	53.9%
Japan	118,712	France	112,357	JapAnce	231,069	51.4%

For example, we make distinctions between two soccer players (RoBeck), an ethnic group and a diplomat (JikRol), two computer companies (MSIBM), two political leaders (MonSlo), a nation and a nationality (JorGypt), and two countries (JapAnce). Note that our task has now become finding the original and correct name that was in the corpus before it was obfuscated. In general the names we have selected have only one underlying entity, for example, "David Beckham" always refers to the soccer player, and Microsoft always refers to the software company. However, "Jordan" is an exception. The dominant sense is that of the country (given the nature of the news wire text) but there are also occurrences of the famous American basketball player. This may well have an impact on the results of clustering, which we will discuss in our analysis.

Each pair of words is processed separately, so we are making a 2 class distinction in this study. In future work we will conflate larger number of names so that we are making distinctions between more underlying entities.

The two clusters are evaluated by replacing the conflated form of the word with the correct original, and determining which name should be assigned to which cluster in order to maximize accuracy. This can be thought of as similar (but not exactly equivalent) to measuring the purity of the clusters. We can find the maximum accuracy by considering the results (once the known identities are available) as a two-by-two cross classification table, that shows the distribution of names and clusters. An example is shown in Figure 5. Each row represents the distribution of the instances in the clusters as compared to their actual identity, and each column shows the distribution of the actual identities in the clusters. We can find the assignment of clusters to identities that maximizes the accuracy by simply reordering the columns of the matrix such that the main diagonal sum is maximized.

From Figure 5, we can see that the assignment of Peres to C1, and Milosevic to C2, results in an accuracy of 91.4% ((6,573 + 6,012)/13,734), while an

	Milosevic	Peres	
C1	36	6,537	6,573
C2	6,012	1,149	7,161
	6,048	7,686	13,734

	Peres	Milosevic	
C1	6,573	36	6,573
C2	1,149	6,012	7,161
	7,648	6,048	13,734

Fig. 1. Assigning Cluster to Name

assignment of Peres to C2 and Milosevic to C1 results in accuracy of 8.6% (36 + 1,149)/13,734).

We measure the precision and recall based on the maximally accurate assignment of names to clusters. Precision is defined as the number of instances that are clustered correctly divided by the number of instances clustered, while recall is the number of instances clustered correctly over the total number of instances[2]. From these values we compute the F–measure, which is two times the product of precision and recall, divided by the sum of precision and recall.

6 Experimental Methodology

There are several significant issues that determine how accurate this approach can be. First, we must determine the size of the context around the ambiguous name to be clustered. We refer to this as the *test scope*. This size of the test scope determines how many words make up the averaged vector that represents the context. Note that when we set our test scope to a value of N, it means use all of the words within N positions of the target word on both sides that have a row vector associated with them in the SVD reduced bigram matrix.

A small test scope is predicated on the idea that the words nearest the ambiguous name will be the most important indicators of how it should be clustered. For example, in the case of names of people, titles or affiliations might be located in close proximity. However, a larger test scope brings in more context, and allows for more content to be included in the averaged vector, potentially making it possible to make finer grained distinctions.

As there are good arguments in favor of both approaches, we will experiment with test scopes of 5 and 20, where a test scope of N means represent the context with the average of all the vectors found for words within N positions to the left and right of the ambiguous name.

The *training scope* is also a significant factor. This determines how large a context around the ambiguous name will be used for identifying the bigram features. If the training scope is set to N, it means that we restrict consideration of bigrams to those that occur within N positions of the ambiguous name.

A smaller training scope will focus the search for bigrams on those that are near or include the ambiguous name (in the case of one word names). This can result in a small number of very reliable collocational features. However, a larger training scope may find bigrams related to the identity of the ambiguous name that do not necessarily include the name itself.

Again, since there are interesting possibilities with both larger and smaller training scopes, we will run experiments with that scope set to 5 and 30.

Note that in this experiment the test and training data are the same. We use the training data for feature identification, and then the test data is what we use to determine how we build the second order context representation.

[2] The clustering algorithm that we use has the option of not placing an instance in any cluster, which is why precision and recall may differ.

In addition, we hypothesize that the potential role of SVD is unclear. The second order co–occurrence features already help to represent indirect relationships, so it's not clear that the smoothing and identification of principle dimensions done by SVD adds significantly to the results.

In all cases we used bigram features and selected those by taking all bigrams that occurred 5 or more times, and had an associated log-likelihood score of 3.814 or above. We used a standard stop–list of function words, and discard any bigram as a feature if it consists of 1 or 2 stop words.

7 Experimental Results and Discussion

For each of our six pseudo-names, we run eight different experiments. We run all possible combinations of experiments where the test scope is set to 5 and 20, the training scope is set to 5 and 20, and we may or may not use SVD.

We show the results of all eight experiments for each conflated pseudo-word in Table 7. This table shows the F-measure for each combination of settings, and also provides general information about the conflated word such as the total number of instances to be clustered, and the percentage of those that belong to the majority identity. Remember that this value can serve as a lower bound for these approaches, since a method that placed every instance in a single cluster would attain this level of accuracy.

Table 2. F-Measures for Name Discrimination

			test scope 5		test scope 20	
			training scope		training scope	
			5	20	5	20
RoBeck	2,452 to cluster	no SVD	57.3	72.7	**85.9**	64.7
	(69.3) majority	SVD	78.4	71.0	81.9	64.9
JikRol	4,073 to cluster	no SVD	94.7	**96.2**	91.0	90.4
	(73.7) majority	SVD	90.9	93.5	87.2	89.3
MSIBM	5,807 to cluster	no SVD	47.7	51.3	**68.0**	60.0
	(58.6) majority	SVD	52.8	52.6	57.2	58.5
MonSlo	13,734 to cluster	no SVD	62.8	**96.6**	54.6	91.4
	(56.0) majority	SVD	80.0	91.4	82.2	94.2
JorGypt	46.431 to cluster	no SVD	56.6	59.1	57.0	53.0
	(53.9) majority	SVD	56.8	**62.2**	61.5	61.5
JapAnce	231,069 to cluster	no SVD	51.1	51.1	50.3	50.3
	(51.4) majority	SVD	51.1	51.1	50.3	50.3

For smaller samples of data, we observe that SVD will at times offer an improvement, but in general does not lead to significant improvements. RoBeck (test=5, training=5) is one case where SVD offers a significant improvement, from 57.3 to 78.4. Given this relatively small amount of data (using small windows for both test and training purposes) the resulting bigram vector is very sparse, and using SVD helps to smooth that out and make it possible to still draw distinctions between contexts.

We note that SVD shows a benefit for those psuedo-names with neither a very large nor a very small number of instances. For example, it results in an improvement for 3 of 4 cases for MonSlo, and all 4 cases for JorGypt. However, JapAnce shows no such improvement, and in fact the overall results are somewhat disappointing in that they are less than the majority sense. We hypothesized that the very large size (more than 200,000 instances) of the data may have had a negative impact, but upon reducing the size of the experiment to 40,000 instances we found essentially identical results. Thus, we believe that this pair might represent a very hard sense distinction to make. While Japan and France are clearly distint geographically and culturally, it may be that they arise in so many different contexts in news text that there are no consistently strong discriminating features that can be identified. This remains an interesting issue for future exploration.

The effect of the variations in the test and training scope are quite interesting. First, the best results for each pair of words came about by either using a small test scope with a large training scope (test = 5, train = 20) or a large test scope with a small training scope (test=20, train = 5). There was no case where the small scopes or large scopes alone gave the best results. We believe that this shows that the scopes are complementary.

A large test scope means that there are many words in the context that will be used to create the averaged vector. If those words are represented by a feature vector that is derived from a large training scope, then the combination of these two wide scopes leads to overly general information. However, if the training scope is small, then the words that occur in the context vector are all represented relative to words that are known to occur near the ambiguous name in the training data. A similar argument can be made for the case of a small test scope and a large training scope. The small test scope means that the averaged context vector will be made up of a small number of words that occur near the ambiguous noun. The words that make up the contexts may all be fairly distinct, but the co–occurrence information derived from a larger training scope will make it possible to identify them as being similar with other words in the test scope.

8 Future Work

The experiments in this paper all focus on binary distinctions, between two relatively distinct entities. We will extend these experiments in future to make distinctions among a larger number of underlying individuals. Rather than sim-

ply using pseudo-names, we will use the *John Smith* corpus as described in the work of Bagga and Baldwin, as well as the data used in the Mann and Yarowsky.

The use of this data will also introduce the other side of the name discrimination problem, that is in identifying two different names that refer to the same person (e.g., *Mr. Smith* and *John Smith*). Fortunately our techniques can be used without modification for this particular problem, and we are optimistic that they will perform well.

We are also developing techniques for looking at the content of the clusters to identify the entity associated with a particular cluster. We have experimented with identifying the most significant features in the clusters of text, and these provide very simple descriptive terms that might describe the entity. However, we wish to improve this approach to the point where it is more analogous to generating a summary of the text in the cluster, and thereby become a tool for knowledge discovery.

9 Conclusions

We have found that the method of Purandare and Pedersen for discriminating word senses by clustering similar contexts performs well in discriminating among ambiguous names. This is an unsupervised approach, so the fact that it nearly always out performs the majority baseline clustering method is significant. We observed that the test and training scopes are complementary, and should be set such that one is small and the other is large in order to get optimal results.

Acknowledgments

This research is supported by a National Science Foundation Faculty Early CAREER Development Award (#0092784).

All of the experiments in this paper were carried out with version 0.55 of the SenseClusters package, freely available from the URL shown on the title page.

References

1. A. Bagga and B. Baldwin. Entity–based cross–document co–referencing using the vector space model. In *Proceedings of the 17th international conference on Computational linguistics*, pages 79–85. Association for Computational Linguistics, 1998.
2. S. Deerwester, S.T. Dumais, G.W. Furnas, T.K. Landauer, and R. Harshman. Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science*, 41:391–407, 1990.
3. T. Gaustad. Statistical corpus-based word sense disambiguation: Pseudowords vs. real ambiguous words. In *Companion Volume to the Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL/EACL 2001) – Proceedings of the Student Research Workshop*, pages 61–66, Toulouse, France, 2001.

4. F. Ginter, J. Boberg, J. Jrvine, and T. Salakoski. New techniques for disambiguation in natural language and their application to biological text. *Journal of Machine Learning Research*, 5:605–621, June 2004.
5. C. H. Gooi and J. Allan. Cross-document coreference on a large scale corpus. In S. Dumais, D. Marcu, and S. Roukos, editors, *HLT-NAACL 2004: Main Proceedings*, pages 9–16, Boston, Massachusetts, USA, May 2 - May 7 2004. Association for Computational Linguistics.
6. H. Han, L. Giles, H. Zha, C. Li, and K. Tsioutsiouliklis. Two supervised learning approaches for name disambiguation in author citations. In *Proceedings of the 2004 joint ACM/IEEE conference on Digital libraries*, pages 296–305, 2004.
7. V. Hatzivassiloglou, P. Duboue, and A. Rzhetsky. Disambiguating proteins, genes, and rna in text: A machine learning approach. In *Proceedings of the 9th International Conference on Intelligent Systems for Molecular Biology*, Tivoli Gardens, Denmark, July 2001.
8. T.K. Landauer, P.W. Foltz, and D. Laham. An introduction to Latent Semantic Analysis. *Discourse Processes*, 25:259–284, 1998.
9. G. Mann and D. Yarowsky. Unsupervised personal name disambiguation. In W. Daelemans and M. Osborne, editors, *Proceedings of CoNLL-2003*, pages 33–40. Edmonton, Canada, 2003.
10. G.A. Miller and W.G. Charles. Contextual correlates of semantic similarity. *Language and Cognitive Processes*, 6(1):1–28, 1991.
11. P. Nakov and M. Hearst. Category-based pseudowords. In *Companion Volume to the Proceedings of HLT-NAACL 2003 - Short Papers*, pages 67–69, Edmonton, Alberta, Canada, May 27 - June 1 2003.
12. A. Purandare. Discriminating among word senses using McQuitty's similarity analysis. In *Companion Volume to the Proceedings of HLT-NAACL 2003 – Student Research Workshop*, pages 19–24, Edmonton, Alberta, Canada, May 27 - June 1 2003.
13. A. Purandare and T. Pedersen. Word sense discrimination by clustering contexts in vector and similarity spaces. In *Proceedings of the Conference on Computational Natural Language Learning*, pages 41–48, Boston, MA, 2004.
14. H. Schütze. Automatic word sense discrimination. *Computational Linguistics*, 24(1):97–123, 1998.
15. N. Wacholder, Y. Ravin, and M. Choi. Disambiguation of proper names in text. In *Proceedings of the fifth conference on Applied natural language processing*, pages 202–208. Morgan Kaufmann Publishers Inc., 1997.
16. Y. Zhao and G. Karypis. Evaluation of hierarchical clustering algorithms for document datasets. In *Proceedings of the 11th Conference of Information and Knowledge Management (CIKM)*, pages 515–524, 2002.

Word Sense Disambiguation by Semi-supervised Learning

Zheng-Yu Niu[1], Donghong Ji[1], Chew-Lim Tan[2], and Lingpeng Yang[1]

[1] Institute for Infocomm Research, 21 Heng Mui Keng Terrace, 119613 Singapore
{zniu, dhji, lpyang}@i2r.a-star.edu.sg
[2] Department of Computer Science, National University of Singapore, 3 Science Drive 2, 117543 Singapore
tancl@comp.nus.edu.sg

Abstract. In this paper we propose to use a semi-supervised learning algorithm to deal with word sense disambiguation problem. We evaluated a semi-supervised learning algorithm, local and global consistency algorithm, on widely used benchmark corpus for word sense disambiguation. This algorithm yields encouraging experimental results. It achieves better performance than orthodox supervised learning algorithm, such as kNN, and its performance on monolingual benchmark corpus is comparable to a state of the art bootstrapping algorithm (bilingual bootstrapping) for word sense disambiguation.

1 Introduction

In this paper, we address the problem of word sense disambiguation (WSD), which is to assign an appropriate sense to an occurrence of a word in a given context. Many learning algorithms have been proposed or investigated to deal with this problem, including knowledge or dictionary based algorithms, and corpus based algorithms. Corpus based algorithms can be categorized as supervised learning algorithms, weakly supervised learning algorithms [1, 3, 5, 6, 7, 8], and unsupervised learning algorithms. In WSD task, we often face a shortage of labeled training data, but there is a large amount of unlabelled data which can be cheaply acquired. As a result, a great deal of work [1, 3, 5, 6, 7, 8] have been devoted to effective usage of unlabeled data for improving the performance of WSD systems.

Here we use a semi-supervised learning algorithm [9] to perform WSD. Compared with other weakly supervised learning based WSD algorithms, such as bootstrapping or co-training, semi-supervised learning algorithm explores the manifold structure to determine the labels of unlabeled points. Secondly, bootstrapping and co-training require that the class distribution should be fixed during the iteration procedure to avoid degenerate solutions.

This paper is organized as follows. In section 2 we will define feature vector and distance measure for WSD. In section 3 we will describe the semi-supervise

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 238–241, 2005.

learning algorithm used for WSD. Section 4 will give out the experimental results of a semi-supervised learning algorithm on widely used benchmark corpus. In section 5 we will conclude our work and suggest possible improvements.

2 Feature Set and Distance Measure

We use three types of features to capture contextual information: part-of-speech of neighboring words, unordered single words in topical context, and local collocation, following [2]. In later experiment, we conduct a simple feature selection by deleting features if they co-occurred less than three times with ambiguous word.

Let $V = \{v_i\}_{i=1}^{N}$, where v_i represents the feature vector of the i-th occurrence of ambiguous word w, and N is the total number of this ambiguous word's occurrences. Then the distance between symbol-valued vector v_i and v_j can be calculated using a modified Hamming distance:

$$\hat{d}_{ij} = \sum_{k} 1\{v_{ik} == v_{jk}, if\ v_{ik}\ \neq 0\ or\ v_{jk} \neq 0\}. \tag{1}$$

3 Semi-supervised Learning Algorithm

We will give a brief summary of the semi-supervised learning method, local and global consistency algorithm (LGC), introduced in [9].

Given a data set $X = \{x_1, ..., x_l, x_{l+1}, ..., x_n\}$, and a class label set $L = \{1, ..., c\}$, the first l points $x_i (1 \leq i \leq l)$ are labeled as y_i ($y_i \in L$) and remaining points $x_u (l+1 \leq u \leq n)$ are unlabeled. Define $Y \in N^{N \times c}$ with $Y_{ij} = 1$ if point x_i has label j and 0 otherwise. Let $\mathsf{F} \in R^{N \times c}$ denote all the matrices with nonnegative entries. A matrix $F \in \mathsf{F}$ is a matrix that labels all points x_i with a label $y_i = argmax_{j \leq c} F_{ij}$. Define the series $F(t+1) = \alpha SF(t) + (1-\alpha)Y$ with $F(0) = Y$, $\alpha \in (0, 1)$. The entire algorithm is defined as follows:

1. Form the affinity matrix W by $W_{ij} = 1 - exp(-\frac{\hat{d}_{ij}}{2\sigma^2})$ if $i \neq j$ and $W_{ii} = 0$;
2. Compute $S = D^{-1/2} W D^{-1/2}$ with $D_{ii} = \sum_j W_{ij}$ and $D_{ij} = 0$ if $i \neq j$;
3. Compute the limit of series $lim_{t \to \infty} F(t) = F^* = (I - \alpha S)^{-1} Y$. Label each point x_i as $argmax_{j \leq c} F^*_{ij}$. I is $N \times N$ identity matrix.

The regularization framework for this method follows. The cost function associated with the matrix F with regularization parameter $\mu > 0 (\alpha = \frac{1}{1+\mu})$ is defined as:

$$Q(F) = \frac{1}{2}\left(\sum_{i,j=1}^{N} W_{ij} \left\| \frac{1}{\sqrt{D_{ii}}} F_i - \frac{1}{\sqrt{D_{jj}}} F_j \right\|^2 + \mu \sum_{i=1}^{N} \|F_i - Y_i\|^2\right). \tag{2}$$

Then the classifying function is

$$F^* = argmin_{F \in \mathsf{F}} Q(F). \tag{3}$$

In later experiments, we let Y be consistent with classification result of a supervised learning algorithm, such as kNN.

Table 1. Accuracy in [3] and accuracy of kNN and LGC with the size of labeled examples as $c \times b$. MB-D denotes monolingual bootstrapping with decision list as the classifier, MB-B monolingual bootstrapping with ensemble of Naive Bayes as the classifier, and BB bilingual bootstrapping with ensemble of Naive Bayes as the classifier

	Accuracies in [3]				Our Results		
Ambiguous Words	Major	MB-D	MB-B	BB	#labeled examples	kNN	LGC
interest	54.6%	54.7%	69.3%	75.5%	60	72.9%	76.6%
line	53.5%	55.6%	54.1%	62.7%	90	56.8%	61.9%

4 Experiments and Results

For comparison of semi-supervised learning algorithm with other weakly supervised learning method, such as bootstrapping algorithm, we evaluated it on widely used benchmark corpus, the corpora of four ambiguous words "hard", "interest", "line", and "serve".

We used kNN (k=1) as baseline, and ran kNN and LGC algorithm using all three types of features on four data sets. The α in LGC algorithm was simply fixed as 0.90. The width of the RBF kernel, σ, was set as 5. After calculation of affinity matrix, we use minimum spanning tree method to construct a connected and sparse graph for LGC.

In [3], they adopted "interest" and "line" corpora as test data. To the word "interest", they used its four major senses. For comparison to their results, we ran kNN and LGC on reduced "interest" corpus (constructed by retaining four major senses) and complete "line" corpus with the number of labeled examples as $c \times b$. c is the number of senses of ambiguous word, and b is the number of examples augmented in each iteration of bootstrapping procedure [3]. $c \times b$ can be deemed as the size of initial labeled examples in their bootstrapping algorithm. All the accuracies were averaged over 10 trials calculated on unlabeled data.

Figure 1 shows the accuracy curves of kNN and LGC versus different percentage of labeled examples. We see that LGC consistently outperformed the orthodox supervised learning algorithm kNN. It indicates that the incorporation of unlabeled data in learning procedure improves the classification results.

Table 1 shows that the performance of LGC algorithm is comparable to the bilingual bootstrapping algorithm (BB) and better than monolingual bootstrapping algorithms (MB-D and MB-B). It should be noted that LGC algorithm utilized only monolingual corpus. However BB achieved their performance with the requirement of two monolingual corpora (English text and Chinese text) and bilingual translation lexicon.

5 Conclusion and Future Work

In this paper we investigated the application of a semi-supervised learning algorithm for word sense disambiguation. In future work, we would like adopt feature

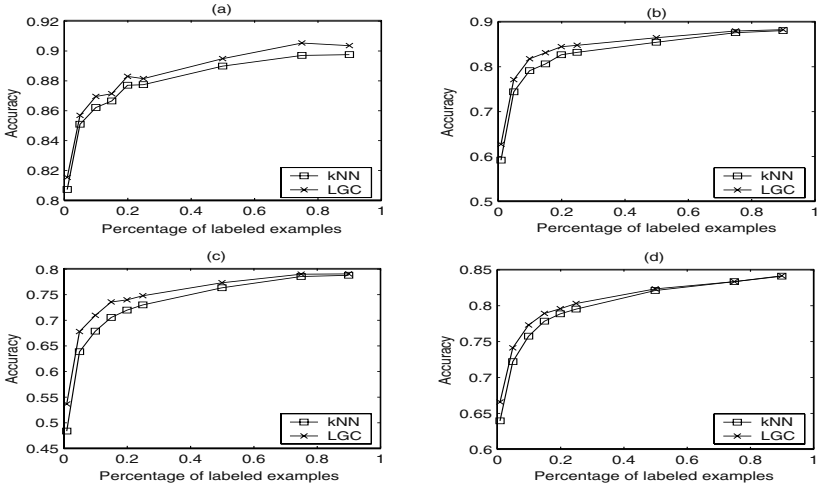


Fig. 1. Accuracy (axis Y) of kNN and LGC versus various percentage of labeled examples (axis X) on (a) hard, (b) interest, (c) line, and (d) serve corpus

clustering technique to deal with high dimensionality problem in feature vector representation of WSD.

References

1. Dagan, I. & Alon I.: Word Sense Disambiguation Using A Second Language Monolingual Corpus. Computational Linguistics, Vol. 20(4), pp. 563-596.(1994)
2. Lee, Y.K. & Ng, H.T.: An Empirical Evaluation of Knowledge Sources and Learning Algorithms for Word Sense Disambiguation. Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing, (pp. 41-48).(2002)
3. Li, H. & Li, C.: Word Translation Disambiguation Using Bilingual Bootstrapping. Computational Linguistics 30(1), 1-22.(2004)
4. Mihalcea R.: Bootstrapping Large Sense Tagged Corpora. Proceedings of the 3rd International Conference on Languages Resources and Evaluations.(2002)
5. Mihalcea R.: Co-training and Self-training for Word Sense Disambiguation. Proceedings of the Conference on Natural Language Learning.(2004)
6. Park, S.B., Zhang, B.T., & Kim, Y.T.: Word Sense Disambiguation by Learning from Unlabeled Data. Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics.(2000)
7. Su, W., Carpuat, M., & Wu, D.: Semi-Supervised Training of A Kernel PCA-Based Model for Word Sense Disambiguation. Proceedings of the 20th International Conference on Computational Linguistics.(2004)
8. Yarowsky, D.: Unsupervised Word Sense Disambiguation Rivaling Supervised Methods. Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics, pp. 189-196.(1995)
9. Zhou D., Bousquet, O., Lal, T.N., Weston, J., & Schölkopf, B.: Learning with Local and Global Consistency. Advances in Neural Information Processing Systems 16, pp. 321-328.(2003)

Crossing Parallel Corpora and Multilingual Lexical Databases for WSD

Alfio Massimiliano Gliozzo, Marcello Ranieri, and Carlo Strapparava

ITC-irst, Istituto per la Ricerca Scientifica e Tecnologica,
I-38050 Trento, Italy
{gliozzo, ranieri, strappa}@itc.it

1 Introduction

Word Sense Disambiguation (WSD) is the task of selecting the correct sense of a word in a context from a sense repository. Typically, WSD is approached as a supervised classification task to get state-of-the-art performance (e.g. [1]), and thus a large amount of sense-tagged examples for each sense of the word is needed, according to the word-expert approach. This requirement makes the supervised approach unfeasible for "all-words" tasks, consisting on disambiguating all the words in texts. This problem has been called the Knowledge Acquisition Bottleneck and many solutions have been proposed for it (see for example [2]) .

In this paper we propose the use of aligned corpora and multilingual lexical databases to automatically acquire sense tagged data, exploiting the polisemic differential between two (or more) languages.

Even though the underlying idea of the approach proposed in this paper is not totally original in the WSD literature (see for example [3, 4]) our basic contribution is to show how far we can go in using parallel corpora to collect sense tagged data, by reporting both a quantitative and a qualitative evaluation. It will be shown that having an "ideal" aligned wordnet (i.e. a lexical resource such that all the sense distinctions in one language are reflected in the other), our simple strategy allows to disambiguate 51% of the English/Italian aligned pairs of words with 100% precision, while with the available resources this figures decreases to 67% precision for a subset of 40% words. In the rest of the paper we will evaluate this technique by exploiting two resources recently developed at ITC-irst: MultiWordNet and MultiSemCor.

2 MultiWordNet and MultiSemCor

MultiWordNet (`http://multiwordnet.itc.it`) is a multilingual computational lexicon, conceived to be strictly aligned with the Princeton WordNet. In our experiment we used the English and the Italian components. The last version of the Italian WordNet contains around 58,000 Italian word senses and 41,500 lemmas organized into 32,700 synsets aligned whenever possible with WordNet English synsets.

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 242–245, 2005.

The MultiSemCor [5] (`http://multisemcor.itc.it`) corpus originates from the Princeton SemCor corpus. SemCor texts were taken from the Brown Corpus, and were semantically annotated according with the synsets of WordNet. MultiSemCor has been built starting from a subset of the SemCor texts. 116 English texts were translated into Italian by professional translators. Then, the original texts and their translations were automatically aligned at the word level. Finally the annotations were transferred from each text to its alignment, creating a bilingual parallel corpus endowed with semantic annotation (about 116,000 semantically annotated English tokens, about 90,000 semantically annotated Italian tokens, being MultiWordNet the shared repository of senses).

3 A Bilingual WSD Algorithm

In this section we describe an unsupervised WSD technique that uses aligned corpora and multilingual lexical databases to automatically acquire sense tagged data, exploiting the polisemic differential between two languages. The basic assumption is that if two texts are one the translation of the other, they should refer to the same facts, and then words contained in them should refer to the same concepts. An aligned multilingual lexical resource (e.g. MultiWordNet) allows us to automatically disambiguate aligned words in both languages by simply intersecting their senses. If the intersection contains only one sense, then the words in both languages will be fully disambiguated, while if the cardinality of the intersection is higher, the words still remain ambiguous. In any case the number of possible senses is often sensibly reduced. For instance if the English word *soccer* is aligned with the Italian word *calcio*, the correct sense is "a football game" and not, for example, "a white metallic chemical element" (one of the four senses of the Italian word).

More formally let $S = \{c_1, c_2, \ldots, c_n\}$ be the set of aligned pairs of English/Italian lemmas such that $c_i = (l_i^E, l_i^I)$, $s(l)$ a function returning the set of senses corresponding to the lemma l, and $I(c_i) = s(l_i^E) \bigcap s(l_i^I)$ the intersection of the synsets corresponding to the two lemmas in the two languages. The function $WSD_{strict}(c_i)$, defined by equation 1, fully disambiguate the word pair if the intersection is a singleton.

$$WSD_{strict}(c_i) = \begin{cases} I(c_i) & : \texttt{if } |I(c_i)| = 1 \\ \emptyset & : \texttt{otherwise} \end{cases} \tag{1}$$

Equation 2 returns the set of all the possible senses.

$$WSD_{soft}(c_i) = I(c_i) \tag{2}$$

4 Evaluation and Discussion

We compared the results with a random baseline, being our method completely unsupervised. We also try to define an upper bound, assuming that all the senses

Table 1. Multilingual WSD evaluation on word pairs and on polysemous words

Evaluation	Language	Precision	Coverage	F1	#Valid
Ideal	both	1	0.51	0.68	39983
All	both	0.67	0.40	0.38	71421
Ideal-polysemous	English	1	0.39	0.56	32277
All-polysemous	English	0.56	0.37	0.30	61712
All-polysemous (random baseline)	English	0.22	1	0.22	61712
Ideal-polysemous	Italian	1	0.32	0.48	28890
All-polysemous	Italian	0.61	0.31	0.29	49206
All-polysemous (random baseline)	Italian	0.17	1	0.17	49206

Table 2. Polysemy reduction using soft multilingual disambiguation

WSD	Pol-ENG	Pol-ITA	Pol-RES	Precision	Coverage	#Valid
Ideal	5.62	3.35	1.98	1	1	39983
All	6.72	3.28	1.54	0.56	1	71421

annotated in the corpus are actually in the Italian WordNet. We evaluated our WSD method on the following two subsets of the original aligned pairs of lemmas in MultiSemCor. Let $G(c_i)$ be the gold standard function returning the correct sense annotated in MultiSemCor for c_i.

Ideal: Only couples such that the gold standard annotation is a possible sense for the lemmas in both languages $S_C = \{c_i | G(c_i) \in s(l_i^E) \text{ and } G(c_i) \in s(l_i^I)\}$.

All: Only couples such that both lemmas are contained in MultiWordNet $S_A = \{c_i | s(l_i^E) \neq \emptyset \text{ and } s(l_i^I) \neq \emptyset\}$.

We distinguish among results for word pairs, polysemous terms in English and polysemous terms in Italian (see Table 1). As expected our WSD method is perfect (i.e. precision 100%) in the **Ideal** evaluation dataset, in which the sense in the Gold Standard is also a possible sense for the Italian lemma. Unfortunately this is not a realistic case, because the Italian resource does not still have the coverage of the English one (i.e. in the Italian part of MultiSemCor a word could be annotated with a sense not reachable from the lemma in Italian MultiWordNet). Thus in the **All** dataset, the precision of the algorithm drops to 0.67 for word pairs. We also evaluated the precision and coverage of the WSD algorithm only by considering polysemous words in English and Italian, and the results where encouraging (i.e. precision is the important feature in the case of acquisition of sense-tagged examples). Table 2 displays the polysemy reduction using the formula 2.

We showed that in the "ideal" case the methodology allows to disambiguate with 100% precision, while with available lexical resources the precision drastically drops to about 60%. A qualitative analysis (see Table 3) of 100 errors (randomly selected) showed that in about 77% of the errors are caused by *not covered* senses (i.e. senses in Italian that should be included in the resource even though they are not actually represented in MultiWordNet).

Table 3. Qualitative analysis of 100 errors (randomly selected)

Causes of errors	# of cases
Senses not covered by the Italian WordNet	77
Alignment errors	7
Inter-lingual differences	16

5 Conclusions and Future Works

In this paper an unsupervised WSD methodology has been presented. This methodology can be applied to parallel corpora allowing to fully disambiguate about the 50% of words, without requiring any external knowledge. Obviously the same approach can be applied also to aligned corpora composed by texts written in more than two languages. Intuitively the probability to obtain a smaller intersection among senses of a translated word in three (or more) languages is higher than the one for only two languages. For the future we plan to automatically acquire the most frequent *not covered* senses by exploiting MultiSemCor in order to improve the WSD performances in "real" parallel corpora, and to apply it extensively to disambiguate large scale parallel corpora (e.g. EuroParl [6]), in order to automatically acquire sense tagged data to train a supervised disambiguation system to be used in an "all-words" task.

Acknowledgments

We would like to thank Emanuele Pianta for useful discussions. This work was partially supported by the *Meaning* European Project (IST-200134460).

References

1. Strapparava, C., Gliozzo, A., Giuliano, C.: Pattern abstraction and term similarity for word sense disambiguation: Irst at senseval-3. In: Proc. of SENSEVAL-3, Barcelona, Spain (2004)
2. Mihalcea, R., Moldovan, D.: An automatic method for generating sense tagged corpora. In: Proc. of AAAI 99, Orlando, FL (1999)
3. Magnini, B., Strapparava, C.: Experiments in word domain disambiguation for parallel texts. In: Proc. of "Word Senses and Multi-Linguality", Hong Kong, Workshop held in conjunction of ACL2000 (2000)
4. Diab, M., Resnik, P.: An unsupervised method for word sense tagging using parallel texts. In: Proc. of ACL 02, Philadelphia (2002)
5. Bentivogli, L., Pianta, E.: Exploiting parallel texts in the creation of multilingual semantically annotated resources: the MultiSemCor corpus. Journal of Natural Language Engineering (NLE), Special Issue on Parallel Texts. (To appear)
6. Koehn, P.: EuroParl: A multilingual corpus for evaluation of machine translation. (Unpublished (http://people.csail.mit.edu/~ koehn/publications/europarl.ps))

A Mapping Between Classifiers and Training Conditions for WSD

Aarón Pancardo-Rodríguez[1], Manuel Montes-y-Gómez[1,2], Luis Villaseñor-Pineda[1], and Paolo Rosso [2]

[1] National Institute of Astrophysics, Optics and Electronics, Mexico
{aaron_cyberman, mmontesg, villasen}@inaoep.mx
[2] Polytechnic University of Valencia, Spain
{mmontes, prosso}@dsic.upv.es

Abstract. This paper studies performance of various classifiers for Word Sense Disambiguation considering different training conditions. Our preliminary results indicate that the number and distribution of training examples has a great impact on the resulting precision. The Naïve Bayes method emerged as the most adequate classifier for disambiguating words having few examples.

1 Introduction

The objetive of Word Sense Disambiguation (WSD) is to distinguish between the different senses of a word, that is, to identify the correct sense of a word in a context. The state of the art of WSD [1] shows that the supervised paradigm is the most efficient. Under this approach, the disambiguation process is carried out using information that is estimated from data. Several statistical and machine learning techniques have been applied to learn classifiers from disambiguated corpora. For instance, statistical classifiers, decision trees, decision lists, memory-based learners, and kernel methods such as Support Vector Machines (SVM).

The comparison among the different approaches to WSD is difficult. The last edition of the Senseval competition showed that the SVM is emerging as one of the most powerful supervised techniques for WSD [3]. Although important, this comparison focuses on the entire systems as black boxes, and does not consider the details about the individual classifiers and the fine tunning of their paramethers.

Some researchers have attempted to compare the performace of classifiers under equal training conditions. For instance, Paliouras et al [2] disambiguated all content words from Semcor using various classifiers (e.g., J48, Naïve Bayes, PART, k-nn and a decision table). Their results indicated that the decision tree induction outperforms other algorithms. Zavrel et al [4] investigated the performance of some classifiers (neuronal networks, memory-based leraning, rule induction, decision trees, maximum entropy, winnow perceptrons, Naïve-Bayes, and SVM) and some ensembles on a diverse set of natural language processing tasks. Their results showed that the SVM algorithm is the most prommising for WSD.

In the study of the global execution of some classifiers, we focus our attention on providing information about the behaviour of the classifiers under different training

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 246 – 249, 2005.

conditions. Basically, in this paper we analyze the influence of the number of training examples and context words over the output precision for each classifier.

2 Analysis of the Semantically Tagged Corpora

The supervised methods for WSD requiere a semantically tagged corpus in order to learn the disambiguation rules. Traditionally, the Semcor[1] corpus has been used for this purpose. It is a subset of the English Brown corpus containing almost 700,000 running words tagged by POS, and more than 200,000 content words lemmatized and sense-tagged according to Wordnet.

The Senseval[2] corpora are other common resources for WSD. The Senseval-3 English all words corpus consists of approximately 5,000 words of running text from two Wall Street Journal articles and one excerpt from the Brown corpus. It contains a total of 2,212 words tagged with the Wordnet senses.

Table 1 shows some statistics from the Semcor 2.0 and the Senseval-3 English all words joint corpora. The statistcs indicate that: (i) the available training corpora are very small, smaller than supposed. Just 21% of the nouns of the corpora are polysemic; (ii) the corpora are very unbalanced. The majority of the examples correspond to the first sense of each noun. The rest of the sense has on average less than five examples.

Table 1. Some statistics from Semcor plus Senseval-3 English all words

Sense	n-secmic Nouns	Average number of examples
1	9082	13.51
2	1368	4.61
3	544	3.68
4	228	3.55
5	117	3.24
6	59	2.74
7	43	3.52
8	22	3.13
9	8	3.17
10	4	2.33
>10	11	1.75

3 Experimental Results

3.1 Experimental Setup

Learning Methods. Naïve Bayes, decision tables, LWL –locally weighted learning–, SVM –support vector machines–,and KNN.

[1] http://www.cs.unt.edu/~rada/downloads.html#semcor
[2] http://www.senseval.org/

Test Set. 10 nouns from the Semcor corpus (refer to table 2). The selection of these nouns was based on two criteria: (i) different number of average examples per sense, and (ii) a more or less balanced distribution of the examples.

Evaluation. It was based on the precision measure (i.e., the porcentage of correctly classified word senses), and on the technique of ten-cross fold validation.

Table 2. Statistics from the test set

Noun	Senses	Examples	Average examples per sense	Distribution of the examples per sense
adult	2	10	5.0	[5 5]
Link	2	10	5.0	[5 5]
formation	5	18	3.6	[4 3 4 4 3]
Dirt	2	20	10.0	[10 10]
stone	3	25	8.3	[8 8 9]
Hope	4	46	11.5	[16 15 14 1]
discussion	2	49	24.5	[27 22]
activity	3	92	30.7	[43 36 13]
plant	2	99	49.5	[63 36]
experience	3	125	41.7	[51 47 27]
state	4	200	50.0	[26 116 21 37]
thing	10	271	27.1	[52 40 32 27 24 20 28 27 17 4]

3.2 Results

Each classifier was tested over the set of selected nouns, and trained using context windows of different sizes (of 4, 6, and 8 words around the noun). Table 3 shows the obtained results. These results demostrate that, even when the classifiers had a similar average precision, their behaviour is altered depending on the training conditions.

The results indicate the following: (i) The size of the context window – number of neighboring words used on the training process – has minor effects on the output average precision; (ii) It seems that for the nouns having few examples most classifiers worked better considering more contextual information; (iii) The Naïve Bayes classifier emerged as the most adequate method for disambiaguting the nouns having few training examples per sense.

In addition, we observed that the majority of the used classifiers had a poor performace when dealing with high polysemic nouns.

4 Conclusions

In this paper we analyzed the coverage and example distribution of the Semcor and Senseval-3 English all word joint corpora. Our results are worrying: the available training corpora is smaller that supossed and unbalanced. This condition greatly affects the performance of most classifiers.

The majority of the supervised methods requiered several examples in order to construct an “accurate” classifier for WSD. According to our results, the Naïve Bayes

algorithm outperforms the others on the disambiguation of nouns having few examples. We consider that this is because it compensates the lack of training examples using more contextual information.

Currently we are studying the performance of the classifiers disambiguating a selection of verbs and adjectives from the Semcor corpus. We believe that this kind of analysis will facilitate the selection of the more appropriate classifier for disambiguating a word depending on its characteristics, which probably would have important repercussions on the construction of hybrid systems for WSD.

Table 3. Peformance of different classifiers on WSD

Classifier	N. Bayes			D.T.			LWL			SVM			KNN; K=1		
W. size	2	4	6	2	4	6	2	4	6	2	4	6	2	4	6
Adult	.40	**.60**	.50	.40	.40	.10	.50	.40	.40	.30	.40	**.60**	.40	.50	.50
Link	.60	**.80**	**.80**	.60	.30	.30	.40	.30	.50	.20	.50	.50	.30	.70	.60
Formation	.38	.61	**.66**	.11	.05	.05	.38	.16	.22	.33	.16	.16	.28	.28	.28
Dirt	**.80**	.70	.60	.70	.70	.70	**.80**	.75	.75	.65	.75	**.80**	.65	.60	.55
Stone	.60	**.64**	**.64**	.36	.40	.40	.44	.44	.48	.48	.44	.48	.48	.48	.48
hope	.37	.39	.37	.37	.34	.32	.34	.32	.32	**.54**	.45	.39	.33	.30	.22
discussion	.59	.63	**.69**	.53	.51	.61	.49	.49	.53	.55	.57	.57	.57	.63	.55
activity	.57	.59	.51	.50	.45	.46	.56	.48	.47	.59	.56	**.60**	**.60**	.61	.48
plant	**.68**	.59	.56	.60	.61	.60	.63	.64	.66	.57	.62	.59	.63	.64	.66
experiencie	**.52**	.45	.46	.44	.43	.42	.42	.42	.43	.50	.48	.51	.46	.47	.49
state	.65	.66	.64	.68	.65	.65	.68	.68	.68	**.69**	.66	.65	.66	.65	.62
thing	**.29**	.23	.24	.21	.21	.21	.26	.25	.24	.25	.22	.22	.28	.23	.24
Average precision	**.50**	.48	.47	.45	.43	.43	.47	.45	.46	.48	.47	.47	.48	.47	.45

Acknowledgements. We would like to thank CONACyT (43990A-1), R2D2 CICYT (TIC2003-07158-C04-03) and ICT EU-India (ALA/95/23/2003/077-054), as well as the *Secretaría de Estado de Educación y Universidades de España* for partially supporting this work.

References

1. Mihalcea, R., Edmonds, P. (Eds.): Proc. of Senseval-3: The 3rd Int. Workshop on the Evaluation of Systems for the Semantic Analysis of Text. Barcelona, Spain. (2004)
2. Paliouras, G., Karkaletsis, V., Androutsopoulos, I., Spyropoulos, C. D.: Learning Rules for Large-Vocabulary Word Sense Disambiguation: a comparison of various classifiers. Proc. of the 2nd International Conference on Natural Language Processing, Patra, Greece (2000).
3. Snyder, B., Palmer, M.: The English All-Words Task. SENSEVAL-3: Third International Workshop on the evaluation of Systems for the Semantic Analysis of Text, Barcelona, Spain (2004).
4. Zavrel, J., Degroeve, S., Kool, A., Daelemans, W., Jokinen, K.: Diverse Classifiers for NLP Disambiguation Tasks: Comparison, Optimization, Combination, and Evolution. Proceedings of the 2nd CEvoLE Workshop "Learning to Behave" (2000).

Multiwords and Word Sense Disambiguation

Victoria Arranz, Jordi Atserias, and Mauro Castillo

TALP Research Center, Universitat Politècnica de Catalunya,
Jordi Girona Salgado, 1-3, E-08034 Barcelona, Catalonia
{varranz, batalla, castillo}@lsi.upc.es

Abstract. This paper[1] studies the impact of multiword expressions on Word Sense Disambiguation (WSD). Several identification strategies of the multiwords in WordNet2.0 are tested in a real Senseval-3 task: the disambiguation of WordNet glosses. Although we have focused on Word Sense Disambiguation, the same techniques could be applied in more complex tasks, such as Information Retrieval or Question Answering.

1 Introduction

In the past years there has been a growing awareness of Multiword Expressions (MWEs). Due to their complexity and flexible nature, many NLP applications have chosen to ignore them. However, given their frequency in real language data, and their importance in areas such as terminology [1], they represent a problem that needs to be addressed. Whilst there has been considerable research on the extraction of MWEs [2], little work has been carried out on their identification. This is particularly so in the framework of Word Sense Disambiguation (WSD). However, in order to face WSD in free running text, we should handle MWEs.

The traditional approach to deal with MWEs has been searching for the longest word-sequence match. An exception can be found in the work of Kenneth C. Litkowski. His research on both Question-Answering [3] and Word-Sense Disambiguation [4] explores the idea of inflection in MWEs, even if just by reducing inflected forms to their root forms. Other works have aimed, for instance, at automatically generating MWEs based on some knowledge source. This is the case of Aline Villavicencio's work [5], where she uses regular patterns to productively generate Verb-Particle Constructions.

In the current work, we have gone further in our MWE detection and selection by lemmatizing our MWEs and allowing some inflection of their subparts (cf. section 3). Bearing in mind that our final goal is the WSD of free running text, where segmentation of word units will not be provided, we have applied our treatment of MWEs to a real NLP task: the Senseval-3 Word-Sense Disambiguation of WordNet glosses. The system here described has participated in the Senseval-3 task achieving the third best results. Further, the same WSD system but using the gold tokenization of the solution (including MWEs) has obtained the best scores in the competition.

[1] This work is supported by the European Commission (MEANING IST-2001-34460).

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 250–262, 2005.

1.1 Senseval-3 Task: WSD of WordNet Glosses

WSD can be defined as the process of deciding the meaning of a word in its context. The possible senses of a word are defined *a priori* in a sense repository. Wordnet [6] has become the *de facto* standard sense repository in the Natural Language Processing community. However, a number of improvements can be considered for this resource, such as the disambiguation of its glosses.

Senseval[2] is the international organization devoted to the evaluation of WSD Systems. Its mission is to organise and run evaluation and related activities to test the strengths and weaknesses of WSD systems in different tasks.

While most of these tasks overlook MWEs and focus on other issues (e.g. in the all-word task, text has been already tokenized and MWEs identified), this does not happen in the WSD of WordNet glosses.

Many WordNet improvements are currently underway. Among these, there is the hand-tagging of the WordNet glosses with their WordNet senses. At the same time, sense-tagging of the glosses is being performed in the eXtended WordNet (XWN) project under development at the University of Texas[3].

More generally, sense disambiguation of definitions in any lexical resource is an important objective in the language engineering community. While substantial research has been performed on machine-readable dictionaries, technology has not yet been developed to make systematic use of these resources. It seems appropriate for the lexical research community to take up the challenge of disambiguating dictionary definitions.

The eXtended WordNet [7] is used as a Core Knowledge Base for applications such as Question Answering, Information Retrieval, Information Extraction, Summarization, Natural Language Generation, Inferences, and other knowledge intensive applications. Its glosses contain a part of the world knowledge since they define the most common concepts of the English language. In this project, many open-class words in WordNet glosses have been hand-tagged and provide an excellent source of data.

The Senseval-3 Word-Sense Disambiguation of WordNet Glosses task is essentially identical to the Senseval-2 and Senseval-3 "all-words" tasks, except for the fact that neither the text has been tokenized nor the MWEs identified, and that there will be very little context and the gloss will not constitute a complete sentence. However, the placement of the synset with WordNet (and all its relations) can be used to assist in disambiguation.

2 WSD System Architecture

Our WSD system, as most in the literature, follows the *Sequential* model. Each level receives the output of the previous one and sends its output to the next. Thus, multiwords are solved before any syntactic or semantic analysis is done.

[2] http://www.senseval.org
[3] http://xwn.hlt.utdallas.edu/

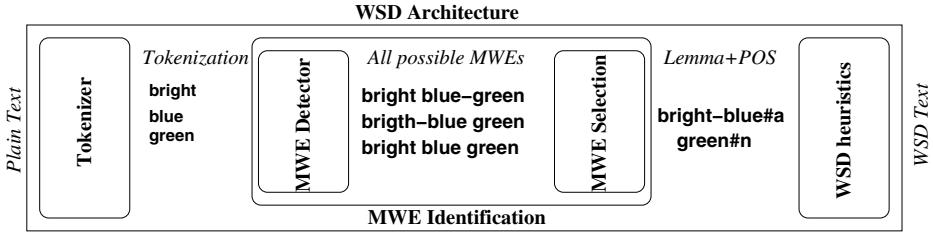


Fig. 1. System Architecture

In our system (see figure 1), the sentences are first tokenized and then passed on to the MWE Identification Module. Then, the output containing the multiwords is POS-tagged using Eric Brill's tagger [8]. Tagged words and MWEs are lemmatized using WordNet's API, which also provides all possible senses. Finally, a set of WSD heuristics are applied to choose the most reliable senses.

The MWE identification module identifies all the MWE occurrences in a sentence. The identification of MWEs can be divided into two tasks: First, there is the detection task, in charge of detecting all possible MWE occurrences in the sentence. Then, there is the selection task, which decides whether the word subsequence previously detected acts as a MWE or not.

3 MWE Detection

The MWE detection module examines the sequences of consecutive[4] tokens looking for occurrences of a closed list of MWEs. The closed list of possible multiword expressions is obtained from WordNet2.0, given that WordNet has become the de facto standard sense repository for WSD.

However, WordNet is not a perfect resource and there are a number of problems or unclear points that pop up when using it as gold standard for MWE detection. For instance, WordNet does not contain information about which elements of a MWE can vary (e.g., which can have morphological inflection), which may be very helpful for our detection task.

So far, our system only addresses the issue of morphological inflection in MWEs. Yet, morphological inflection is not the only variation that a MWE can have: for instance, the adjective synset 00433548a has the following 2 variants: *naked as the day one was born* and *naked as the day you were born*, but not *naked as the day I was born*. One could think that this indicates the possibility of some pronominal variation. However, it rather looks like some kind of inconsistency, allowing for two representations of the same variant, but not with all possible pronominal combinations.

Another important phenomenon is the use of spaces, hyphens, or even the consideration of a unique word, for MWEs. In the text of the test carried out in

[4] So far, we have not dealt with discontinous MWEs, e.g: *look the word up*.

this work, there are eight compound words (*life-time*, *strike breakers*, *work place*, *cross hairs*, *gold threats*, *school children*, *cut away*, *light weight*) that use either a hyphen or a blank space in between their elements but only appear as a unique word in WordNet2.0 (*lifetime*, *strikebreakers*, *workplace*, *crosshairs*, *goldthreats*, *schoolchildren*, *cutaway*, *lightweight*). To consider these cases as possible MWEs would be computationally hard and probably misleading, as we should consider all possible decompositions of any word in WordNet.

In our work, we only focus on the MWEs that can have morphological inflection. WN represents this kind of MWEs using the lemma of the word that can be morphologically inflected (e.g. *eat up*). Accordingly, the WordNet API will accept almost any morphological inflexion on each part of the multiword, such as *ate up* but also like *eat up***s**.

A few WSD systems have tried to improve the MWE information using Machine Readable Dictionaries (MRDs). For instance, Litkowski's WSD system at Senseval-2 [9] uses several MRDs to build regular expressions.

The morphological inflection of MWEs is an open issue. Here we will explore several ways of dealing with the morphological inflection inside a MWE, ranking from allowing no morphological inflection to free morphological inflection of all the elements in a MWE.

Basically, in order to allow morphological inflection, we lemmatize the input word sequence (using an English version of MACO [10]) and look up all the possible combinations of the lemmas and word forms in the WordNet MWE list. The sentence is POS-tagged *a priori* so as to allow lemmatizing according to the POS of the word. The different strategies studied are:

- **NONE**: Allows no morphological inflection. Only the concatenation of the word form that appears in the input will be looked up as a MWE. E.g. This strategy will identify *dig up* and *dead body* but not *digging up* or *dead bodies*.
- **PAT**: Allows some morphological inflection according to a set of pre-defined syntactic patterns. That is, if a possible MWE matches a syntactic pattern, the pattern establishes which words will be lemmatized according to their assigned POS. All possible combinations of the lemmas and the word forms will be looked up.
- **FORM**: Allows a MWE to vary exactly as seen in the SemCor [11] corpus. Although it can cover other phenomena it is more restrictive than PAT. E.g. it will recognise the MWE *looked after* as it appears in SemCor but not *looking after* as this form of the MWE is not present in SemCor.
- **ALL**: Allows all the elements of the MWE to have morphological inflection. E.g. *dig up*/*digs up*/*digging up*/*digged up* and *dead body*/*dead bodies*.

There are not many resources for the study of morphological inflection of the MWEs contained in WordNet. As specific MWE resources are being built with different criteria of what a multiword is in WordNet, we decided to use a WordNet-related resource such as the SemCor corpus as reference.

SemCor is a subset (about 250,000 words) of the Brown Corpus, consisting of texts that have been tagged with POS information and WordNet senses. SemCor consists of about 186 documents classified into 20 classes. This corpus was

semantically annotated with WordNet 1.6 senses, and actually, automatically mapped to WordNet 2.0.

Although SemCor is completely sense-tagged, in order to use it as a MWE resource we have to deal with Named Entities and multiwords not present in WordNet, new MWEs not present in WordNet1.6, or inconsistences (e.g. Sarah_Bernhardt is tagged as *person* instead of being tagged with its synset).

Other resources and techniques can be used to infer/learn patterns about morphological inflection (e.g. using ML techniques) but they are beyond the aim of this paper. The next subsection explains the **PAT** strategy in detail.

3.1 Linguistic Heuristics for MWE Detection

The main idea behind the design of these heuristics is to try and restrict which parts of the MWE can vary (inflect). That is, these heuristics should help us make a wiser inflection of the words involved in a MWE, so that we cover a wider range of possible MWEs (those MWEs that allow some kind of internal changes) and avoid overgenerating if we allow for all possible variations.

In order to manually extract some syntactic patterns for the design of the linguistic heuristics, all MWEs in the SemCor corpus were extracted so as to have a reference starting-point. These MWEs were then classified according to their morphological category and only those categories that, in principle, allowed some morphological variation were selected for revision and pattern extraction (i.e., nouns and verbs[5]). Only part of the noun and verb lists was used for this pattern extraction, so as to first test our hypothesis before doing a full check-up of the whole list of MWEs. Despite only checking part of the MWEs in SemCor, we could already foresee covering a large amount of the structures within the multiwords. These patterns contained a considerable number of combinations for the elements within the MWEs and, since we allowed for elements to be ommitted or repeated, we gained a lot on structural variety.

Once the MWEs in the SemCor corpus were extracted, all sentences where these occurred were POS-tagged with Brill's tagger. The aim of retagging the SemCor data was the achievement of consistency with the tagging of the input.

However, for all elements to be tagged and thus be allowed to play with the inflection of elements within a MWE, the MWEs were tokenized before the tagging step. Having thus at our disposal both the lists of MWEs and the POS-tagged sentences, the expert linguist checked these data and designed a list of syntactic patterns for the morphological inflection of MWEs, accordingly. Besides providing context to check the use of the candidate MWEs, the POS-tagged sentences provided us with information about the constituents of the MWEs themselves. This implies that we can:

- Design the syntactic patterns to constitute the MWEs (see Table 1).
- Consider POS variations inside the MWEs given that the tagging tool is not always consistent with certain categories, mostly with adverbs and closed-

[5] Adjectives, adverbs, conjunctions and prepositions were only allowed as part of a nominal or verbal MWE.

Table 1. Patterns of Morphological Inflection

Nominal Multiword Entities	
1 NN NN? NN? [NN]	*access road*
2 [NN] IN DT? NN	*pain in the neck*
3 NN POS [NN]	*arm's length*
4 JJ [NN]	*Analytical Cubism*
5 [NN] IN JJ [NN]	*balance of international payments*
6 IN [NN]	*anti-Catholicism*
7 [NN] CC [NN]	*bread and butter* or *nooks and crannies*
Verbal Multiword Entities	
8 [VB] IN (RB|IN)? (RB|IN)?	*allow for*
9 [VB] RB (RB|IN)? (RB|IN)?	*bear down on*
10 [VB] TO VB	*bring to bear*
11 [VB] JJ	*break loose*
12 [VB] IN? (DT|PRP$)? [NN]	*take a breath*

category elements such as conjunctions and prepositions. Thus, we allowed for variations among these categories, as it can be seen in patterns 8 and 9 in Table 1. When it comes to tagging verbs and their particles, these are sometimes mistagged. However, for the current experiments, no wrong combinations were accepted inside the patterns, even though a considerable number of wrong tags were obtained. For instance, preposition *to* is always labelled as infinitive *to* (*TO*), and a large number of nouns (e.g. *play, spell, start*) are usually tagged as verbs (with exceptions like *spelling out*, where *spell* is considered a noun).

- Check which elements within the MWE can inflect and thus determine which inflectional variations to allow for the components of our syntactic patterns.
- Compare structurally similar patterns and allow for elements to be optional.

Table 1 shows the set of syntactic patterns allowed for morphological inflection. Patterns are divided into nominal and verbal, which is the general category of the MWE, even if they can take other morphological categories in their components. The elements within the MWEs would follow the linear order presented in the patterns. The components that allow inflection are enclosed in square brackets, while those that are optional (that may occur building bigger structures) are followed by a **?**. The second column provides examples of the patterns.

4 MWE Selection

This module decides whether a possible MWE, which has been previously detected, acts as a multiword in the sentence or not. This could eventually involve choosing between different mutually excluding multiwords. For instance, in the case of *bright blue green*, the second element could be part of the following two overlapping MWEs in WordNet: *bright-blue* and *blue-green*.

The common, and almost only approach in WSD, has been that of taking the longest multiword as valid. This heuristic is based on the belief that the sequential occurrence of the words that a MWE contains always means that this sequence is a MWE (or, at least, that it is more probable).

We will use this heuristic as it is usually carried out, that is, from left to right. Apart from not taking into account the context in which the MWE appears, on the experiments here carried out, we have seen that in same cases, this is not the best way to carry out the heuristic (e.g. in: *pass out of sight* we wrongly identify **pass_out** as a MWE instead of **out_of_sight**) or that it is better to prioritize other factors (e.g. in *keep in-check*, the MWE **keep_in** is wrongly identified).

We will also experiment with another heuristic, similar to the most frequent sense. We have processed SemCor in order to see whether the words involved in a possible multiword co-occur most frequently as a MWE or as a sequence of isolated words. Then, taking this into account, we can remove (or not select) a possible MWE. As the set of possible MWEs recognized varies for each MWE detection strategy, we do this calculation for all the detection strategies.

5 WSD System

Once the glosses have been tokenized and all the MWEs identified, we can address the dissambiguation of their content words. Most of the success of the heuristics used in our own WSD system relies on a very accurate part of speech tagging and multiword identification. As the POS tagger seems to wrongly identify most of the MWEs as nouns, we replace each MWE by another word with the same POS category before carrying out the POS tagging. After concluding with the POS tagging step, we place back the MWEs with their POS.

The disambiguation of the content words consists in assigning to each open-class word the correct sense using its POS. Our main goal is to build a new WSD system based initially on the main heuristics of [7, 12, 13]. We plan to improve the current system's performance by considering the content of the MEANING Multilingual Central Repository (MCR) [14].

For each gloss, we use a set of heuristics. In this way, each heuristic provides zero, one, two or more votes for each word of the gloss. The program simply adds up the votes for each word and assigns the most voted sense. The main heuristics used in the disambiguation process are:

1. **Monosemous**: Applying a closed–world assumption, this heuristic identifies all the words appearing in the gloss with only one sense and marks them with sense #1. For instance, in the example below, this heuristic will vote for *uncomfortably#r#1* since adverb *uncomfortably* only has one sense:

   Synset: claustrophobic#a#1
   Gloss: uncomfortably closed or hemmed in
   MONOS: uncomfortably#r#1

2. **Most Frequent**: This heuristic uses the already existing WordNet 2.0 sense frequencies. The algorithm selects those word senses whose frequencies are

higher than the 85% of its most frequent sense. For instance, in the example below, the noun *rule* has 12 senses and the most frequent sense is the first one (16 occurrences). This heuristic votes for *rule* senses 1 and 2, which are the only ones whose frequencies (compared to the first sense) are higher than 85% (100% and 93%, respectivetly):

Synset: ancestral#a#1 **Gloss:** inherited or inheritable by established rules of descent **MOSTFRE:** rule#n#1 **MOSTFRE:** rule#n#2

3. **Hypernym**: This method follows the hypernym chain looking for words appearing in the gloss (e.g. the genus term). This is the case in the example below, where *fastening#n#2* is a direct ancestor of the synset *doweling#n#1*:

Synset: doweling#n#1 **Gloss:** fastening by dowels **HYPER:** fastening#n#2

4. **WordNet Relations**: This heuristic follows any relation of the synset, looking for words appearing in its gloss. The method does not use only direct relations. The process also performs a chaining search following all relations and stopping at distance five. For instance, in the example below following the chain relation: *vastly#r#1* **Derived from adj** $\longrightarrow$ *vast#a#1* **Synonyms** $\longrightarrow$ *large#a#1, big#a#1* **Synonyms** $\longrightarrow$ *great#a#1*, this heuristic selects the sense *great#a#1*:

Synset: immensely#r#1 vastly#r#1 **Gloss:** to an exceedingly great extent or degree **RELATIONS3:** great#a#1 **RELATIONS5:** great#a#6 **RELATIONS5:** extent#n#2 **RELATIONS5:** degree#n#1 **RELATIONS5:** degree#n#7

5. **MultiWordNet Domains** : Having a synset with a particular WN Domain label, this method selects those synsets from the words of the gloss having the same Domain label:

Synset: corinthian#a#1 (architecture) **Gloss:** most ornate of the three orders of classical Greek architecture **DOMAINS:** ornate#a#1 (architecture) **DOMAINS:** classical#a#1 (architecture) **DOMAINS:** architecture#n#3 (architecture) **DOMAINS:** architecture#n#2 (architecture)

6. **Patterns**: This method uses the "One sense per collocation" heuristic [15]. A set of collocational patterns were acquired directly from the eXtended WordNet [12]. For instance, in the example below, the pattern **IN * MANNER** will select the sense *manner#n#1*:

Synset: cerebrally#r#1
Gloss: in an intellectual manner
PATTERN: manner#n#1

7. **Lexical Parallelism**: This heuristic identifies the words with the same part of speech that are separated by comas or conjunctions and marks them, when possible, with senses that belong to the same hierarchy:

Synset: in-name#r#1
Gloss: by title or repute though not in fact
LEXPAR: repute#n#1 (→ honor#n#1 → standing#n#1 → status#n#1)
LEXPAR: title#n#4 (→ high_status#n#1 → status#n#1)

8. **SUMO** [16]: Having a synset with a particular SUMO label, this method selects those synsets from the words of the gloss having the same label:

Synset: open-sesame#n#1 SubjectiveAssesmentAttribute
Gloss: any very successful means of achieving a result
SUMO: very#r#1 SubjectiveAssesmentAttribute
SUMO: very#r#2 SubjectiveAssesmentAttribute
SUMO: successful#a#1 SubjectiveAssesmentAttribute
SUMO: means#n#3 SubjectiveAssesmentAttribute
SUMO: achieve#v#1 SubjectiveAssesmentAttribute

9. **Category**: Having a synset being connected to a particular WN CATEGORY, this method selects those synsets from the words of the gloss connected to the same CATEGORY:

Synset: accommodation#n#6 CATEGORY: Physiology#n#1
Gloss: (physiology) the automatic adjustment in focal length of the lens of the eye
CATEGORY: physiology#n#1 CATEGORY TERM: accommodation
CATEGORY: automatic#a#3 CATEGORY: Physiology#n#1

10. **Bigram**: This heuristic uses high-frequency wordsense pairs in SemCor:

Synset: pace#n#4
Gloss: a step in walking or running
BIGRAM: step#n#1 **BIGRAM:** walk#v#1

11. **Sense One**: Finally, this heuristic always assigns the first WN sense:

Synset: break#v#59
Gloss: weaken or destroy in spirit or body
SENSE1: weaken#v#1 **SENSE1:** destroy#v#1
SENSE1: spirit#n#1 **SENSE1:** body#n#1

6 Experiments and Results

A set of experiments has been designed to evaluate the impact of different MWE strategies on the disambiguation of glosses, allowing four different levels of MWE morphological inflection and two different strategies for MWE selection.

WordNet 2.0 contains a total number of 115,424 glosses but in order to build our evaluation test we need those glosses disambiguated. In the XWN, the glosses are syntactically parsed and content words are semantically disambiguated.

Table 2. Disambiguated Words in Each Category

POS	Words	Gold	Silver	Normal
noun	505,946	10,142	45,015	296,045
verb	48,200	2,212	5,193	30,813
adj	74,108	263	6,599	50,359
adv	8,998	1,829	385	4,920

To disambiguate these open-class words they used both manual and automatic annotation. The precision of annotation was classified as "gold" for manually checked words, "silver" for words automatically tagged with the same sense by their two disambiguation systems, and "normal" for the rest of the words automatically annotated by the XWN_WSD system. Table 2 presents the number of open-class words in each category for the glosses corresponding to each POS.

From the Extended WordNet 2.0, we extracted the 1,300 glosses that had all their elements tagged as gold. These glosses, which contain 397 multiwords, were processed using our system.

In order to evaluate the identification of the MWEs, we have adapted the MUC-3 evaluation metrics (Precision, Recall, F-measure) and established the following cases: **Correct** (COR) MWEs correctly assigned by the system, **Incorrect** (INC) MWEs incorrectly assigned by the system, **Missing** (MIS) MWEs that should have been assigned and **Spurious** (SPU) MWEs that should not have been assigned.

Table 4 shows the results[6] obtained using the different strategies for the morphological inflection (NONE, PAT, FORM, ALL) and the two different strategies for MWE selection (Longest Match and Based-on-SemCor).

As expected, the Based-on-SemCor MWE-selection heuristic increases precision (with a small decrease of recall) in all the methods. On the other hand, the best results (81% **P**recision, 82% **R**ecall given an F_1 0.81) were achieved using our simple set of patterns (**PAT**). It is surprising that **PAT** correctly identifies more MWEs than the **ALL** heuristics. This is because the **PAT** heuristic has few possibilities when lemmating MWEs (e.g. *ice skating* could be two MWEs **ice-skating** or **ice-skate**) or in the selection module when resolving overlapping

[6] The number of MWEs with a wrong POS appears between brackets.

Table 3. Overall Results for the WSD System

	NONE						PAT					
	COR	**INC**	**SPU**	**MIS**	**P**	**R**	**COR**	**INC**	**SPU**	**MIS**	**P**	**R**
Long. Match	284	13 (5)	74	100	77%	72%	334	18 (5)	90	45	76%	84%
SemCor	275	13 (5)	53	109	81%	69%	324	18 (5)	60	55	81%	82%
	FORM						**ALL**					
	COR	**INC**	**SPU**	**MIS**	**P**	**R**	**COR**	**INC**	**SPU**	**MIS**	**P**	**R**
Long. Match	290	14 (6)	99	93	72%	73%	331	22 (6)	110	44	71%	83%
SemCor	284	13 (5)	77	100	76%	72%	319	22 (6)	72	56	77%	80%

Table 4. WSD Results for MWE Identification

System	**Votes**	**COR**	**INC**	**SPU**	**Missing**	**Prec.**	**Recall**
PAT+SemCor	4,601	2,934	1,377	290	418	63.0%	63.8%
NONE+LM	4,652	2,893	1,368	390	468	62.2%	61.7%

conflicts. **PAT** also controls not only which lemmas morphologicaly inflect but also the right combination of elements (e.g., in the gloss of noun *ring_7* " *a/DT square/NN platformmark/NN off/IN by/IN ropes/NNS in/IN which/WDT contestants/NNS box/NN*" the **PAT** heuristic will prevent us from recognizing *rope-in* as a MWE, even if the lemma is *rope* because it has been tagged as a noun and there is no *NN–IN* pattern).

It is hard to evaluate the impact of MWEs in WSD, not just because wrongly identifying a MWE will generate spurious votes (for the spurious MWEs or for the words that contain the missing MWEs) but also because it can affect the POS assigned to the other words in the sentence. Comparing the figures on the initial strategy (**NONE+LM**) with the **PAT+SemCor**, our WSD system gains 0.8% in Precision and 2.1% in Recall (see table 3). Taking into account that the MWEs are fewer than the 10% of the content words to disambiguate, this increment is highly significant. However, as MWEs tend to be less ambiguous than other words we expect a higher impact of the MWEs on the precision of the WSD results of the overall system. Probably, the massive use of infinitives in the glosses with respect to free text has softened the results between the heuristics.

7 Conclusions and Future Work

The improvement carried out in the MWE modules has improved the overall performance of the WSD system. The system here described has participated in the Senseval-3 task achieving the third best results. Further, the same WSD system but using the gold tokenization of the solution (including MWEs) has obtained the best scores in the competition.

We aim to study the inclusion of MWE information in WN, e.g. associating each WordNet MWE to a class or pattern that establishes which parts can vary.

As future work, we also want to explore more sophisticated techniques to select MWEs, that is, to decide whether a sequence of words acts as a multiword by looking at the context of the concrete sentence (e.g syntax or semantics), or using other kinds of information that some extensions of Wordnet (e.g. the MCR [14]) could provide. We foresee the need for a new integrated approach [17] for the identification of Multiword Expressions (MWEs). We need to design a robust and flexible approach to identify MWEs, where several heuristic methods (linguistic and statistic), which may be centered on different knowledge types (syntactic and semantic), are fully integrated to identify MWEs.

References

1. Maynard, D., Ananiadou, S.: TRUCKS: a model for automatic multi-word term recognition. Journal of Natural Language Processing **8** (2000) 101–125
2. Schone, P., Jurafsky, D.: Is knowledge-free induction of multiword unit dictionary headwords a solved problem? In: EMNLP2001. (2001) 100–108
3. Litkowski, K.: Question-answering using semantic relation triples. In Voorhess, E., Harman, D., eds.: Information Technology: The Eighth Text REtrieval Conference (TREC-8). (2000) 349–356 NIST Special Publication 500-246.
4. Litkowski, K.: SENSEVAL: The CL Research Experience. Computer and the Humanities **34** (2001) 153–158
5. Villavicencio, A.: Verb-particle constructions in the world wide web. In: Proc. of the ACL-SIGSEM Workshop on the Linguistic Dimensions of Prepositions and their use in Computational Linguistics Formalisms and Applications. (2003)
6. Fellbaum, C.: WordNet. An Electronic Lexical Database. The MIT Press (1998)
7. Mihalcea, R., Moldovan, D.: eXtended WordNet: Progress Report. In: Proceedings of NAACL Workshop on WordNet and Other Lexical Resources. (2001)
8. Brill, E.: A Corpus-Based Approach to Language Learning. PhD thesis, Department of Computer and Information Science, University of Pennsylvania (1993)
9. Litkowsky, K.: Sense information for disambiguation: Confluence of supervised and unsupervised methods. In: SIGLEX/SENSEVAL Workshop on Word Sense Disambiguation: Recent Successes and Future Directions. (2002) 47–53
10. Atserias, J., Carmona, J., Castellón, I., Cervell, S., Civit, M., Márquez, L., Martí, M., Padró, L., Placer, R., Rodríguez, H., Taulé, M., Turmo, J.: Morphosyntactic analysis and parsing of unrestricted spanish text. In: First International Conference on Language Resources and Evaluation (LREC'98), Granada, Spain (1998)
11. Miller, G., Leacock, C., Tengi, R., Bunker, R.: A Semantic Concordance. In: Proceedings of the ARPA Workshop on Human Language Technology. (1993)
12. Novischi, A.: Accurate semantic annotations via pattern matching. In: Florida Artificial Intelligence Research Society (FLAIRS'02), Pensacola, Florida (2002)
13. Gangemi, A., Navigli, R., Velardi, P.: Axiomatizing wordnet glosses in the ontowordnet project. In: 2nd International Semantic Web Conference Workshop on Human Language Technology for the Semantic Web and Web Services. (2003)
14. Atserias, J., Villarejo, L., Rigau, G., Agirre, E., Carroll, J., Magnini, B., Vossen, P.: The MEANING multilingual central repository. In: Proceedings of the Second International Global WordNet Conference (GWC'04). (2004)

15. Yarowsky, D.: One sense per collocation. In: Proceedings, ARPA Human Language Technology Workshop, Princeton (1993)
16. Niles, I., Pease, A.: Towards a standard upper ontology. In: In Proceedings of the 2nd International Conference on Formal Ontology in Information Systems (FOIS-2001), Chris Welty and Barry Smith, eds (2001) 17–19
17. Mahesh, K.: A theory of interaction and independence in sentence understanding. Master's thesis, Georgia Institute of Technology (1993)

Context Expansion with Global Keywords for a Conceptual Density-Based WSD

Davide Buscaldi[1], Paolo Rosso[2], and Manuel Montes y Gómez[3,2]

[1] Dipartimento di Informatica e Scienze dell'Informazione (DISI),
Università di Genova, Italy
buscaldi@disi.unige.it
[2] Dpto. de Sistemas Informáticos y Computación (DSIC),
Universidad Politecnica de Valencia, Spain
{prosso, mmontes}@dsic.upv.es
[3] Lab. de Tecnologías del Lenguaje,
Instituto Nacional de Astrofísica, Optica y Electrónica, México
mmontes@inaoep.mx

Abstract. The resolution of the lexical ambiguity, which is commonly referred to as Word Sense Disambiguation, is still an open problem in the field of Natural Language Processing. An approach to Word Sense Disambiguation based on Conceptual Density, a measure of the correlation between concepts, obtained good results with small context windows. This paper presents a method to integrate global knowledge, expressed as global keywords, in this approach. Global keywords are extracted from documents using a model based on term frequency and distribution. Preliminary results show that a slight improvement in recall can be obtained over the base system.

1 Introduction

The resolution of lexical ambiguity that appears when a given word in a context has several different meanings is commonly referred as Word Sense Disambiguation (WSD). Supervised approaches to WSD usually perform better than unsupervised ones [4]. However, such approaches are afflicted by the lack of large, semantically annotated corpora. The unsupervised approach to WSD based on *Conceptual Density* and the frequency of WordNet senses [5] is an unsupervised approach which obtained good results, in terms of precision, for the disambiguation of nouns over SemCor (81.55% with a context window of only two nouns, compared with the MFU-baseline of 75.55%), and in the Senseval-3 all-words task (73.40%, compared with the MFU-baseline of 69.08%) as the CIAOSENSO-2 system [2].

Our approach obtained the above results with a context window of only two nouns, one before and one after the noun to disambiguate, exploiting the relationship existing between adjacent words. The obtained results [5] show that a larger context deteriorates the performance of the approach. We suppose that

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 263–266, 2005.

such decrease is due to the fact that distant words have little or no meaning for the disambiguation of a given word. The only relationship existings between two distant words in the same document is that they are related to the content of the document itself.

In order to introduce this information into our approach we needed to select the most representative words in a document, and adding them to the context of the word to disambiguate. The selected model for extracting document keywords was based on term frequency and distribution as presented in [3].

2 The CD-Based Approach

Conceptual Density *(CD)* is a measure of the correlation among the sense of a given word and its context. Our approach carries out the noun sense disambiguation by means of a formula [5], derived from the original Conceptual Density described in [1].

Due to the granularity of the version 2.0 of WordNet, we consider only the *relevant* part of the subhierarchy determined by the synset paths (from the synset at the top of subhierarchies to an ending node) of the senses of both the noun to be disambiguated and its context, and not the portion of subhierarchy constituted by the synsets that do not belong to the synset paths. In order to take into account also the information about frequency contained in WordNet the following fomula was introduced [5].

3 Extraction of Global Keywords

Document keywords appear usually in very different locations in the document. The Information Retrieval (IR) model proposed by [3] allows to use distribution characteristics of words to determine keywords, by computing their standard deviation. The standard deviation for the i-th word in document is computed as:

$$s_i^2 = \frac{1}{(f_i - 1)} \sum_j (l_{ij} - m_j)^2 \tag{1}$$

where f_i is the frequency of the i-th word, l_{ij} is the j-th position of the word in document, and m_j is the mean of relative location j. Thereafter, we can extract document keywords, having great frequency and standard deviation, that is, wide distribution over the text.

We applied this IR model to the three documents which are part of the Senseval-3 all-words corpus, obtaining the global keywords as shown in Table 1.

Document 1 is a part of a novel, document 2 is a newspaper article about presidential elections, while document 3 is a collection of excerpts from a bulletin board. It is noteworthy how representative are the global keywords extracted from document 2.

Table 1. Keywords extracted for each document in the Senseval-3 all-words corpus, sorted by standard deviation. Frequency is the total number of occurrences in the document, positions are the numbers identifying words' positions in the document, deviation is the standard deviation calculated over the document

Document	keywords	frequency	positions	deviation
doc1	guy	5	65,229,648,1658,1875	330.8
	course	4	124,990,1207,1994	332.9
	something	4	202,1011,1127,1907	302.1
	accident	4	776,1193,1969,1999	260.6
doc2	level	4	33,1271,1278,1344	274.2
	ticket	4	35,490,789,1258	222.6
	gop	5	6,126,431,951,1232	211.3
	pattern	4	155,498,891,1266	208.4
	election	5	51,113,510,666,1200	186.4
doc3	berkeley	3	278,356,1405	296.7
	bay	3	11,96,1105	286.9
	line	3	59,454,1214	276.7
	phone	3	58,723,1213	273.3
	book	3	301,663,1283	234.1
	room	3	306,662,1289	234.6
	night	3	5,128,887	225.2

4 Experimental Results

The Global Keywords (GK) extracted were added to the context of each word, taking them into account for the computation of Conceptual Density. Table 2 shows the obtained results, compared with those obtained with the CIAOSENSO-2 system at Senseval-3 [2] and the Most Frequent Sense (MFS) heuristic.

We obtained a slight improvement in Recall (1.7%) and Coverage ($\sim$ 3%), but there was a $\sim$ 1% loss in precision. In order to obtain better results, we decided to add to the context only two words for each document. The two words were selected on the basis of the following criteria:

1. Polysemy (i.e., those having fewer senses);
2. Depth in the WordNet hierarchy (i.e., the words whose synsets' average depth is the greatest);
3. Specificity (i.e., the words whose synsets' averaged number of hyponyms is smaller).

in Table 2 we show the characteristics of polisemy, depth and specificity of all the extracted global keywords.

5 Conclusions and Further Work

The number of the experiments and the size of the used corpus are too small to fully understand the impact of representative global information on WSD. How-

Table 2. Results obtained over the nouns in the Senseval-3 all-words corpus using context expanded with GK (*CD+GK*), the CD approach (*CIAOSENSO-2*), the MFS heuristic, and filtering GK by their characteristics - polisemy (CD+Less Polysemic), averaged depth of synsets (CD+Deepest), and averaged number of hyponyms (CD+Most Specific)

	Precision	Recall	Coverage
CIAOSENSO-2	0.743	0.497	66.9%
MFS	0.691	0.691	100%
CD+GK	0.734	0.508	69.2%
CD+Less Polysemic	0.729	0.506	69.3%
CD+Deepest	0.731	0507	69.3%
CD+Most Specific	0.730	0.507	69.4%

ever, it seems that a slight improvement in recall and coverage can be obtained without losing too much in precision. This has to be proofed over a larger corpus, such as SemCor. Filtering global keywords depending on their polisemy, depth and hyponyms features extracted from WordNet did not prove to be helpful in WSD, even if we suppose that this can be exploited in other applications, such as IR, to improve the model based on frequency and distribution of words.

Acknowledgments

We would like to thank R2D2 CICYT (TIC2003-07158-C04-03), CONACyT-Mexico (43990A-1, U39957-Y) and ICT EU-India (ALA/95/23/2003/077-054) projects, as well as the Secretaría de Estado de Educacin y Universidades de España, for partially supporting this work.

References

1. Agirre, E., Rigau, G.: A proposal for Word Sense Disambiguation using Conceptual Distance. Proc. of the Int. Conf. on Recent Advances in NLP (RANLP'95). 1995.
2. Buscaldi, D., Rosso, P., Masulli, F.: The upv-unige-CIAOSENSO WSD System. Senseval-3 Workshop, Association for Computational Linguistics (ACL-04). Barcelona, Spain (2004).
3. Lee, J., Baik, D..: A Model for Extracting Keywords of Document Using Term Frequency and Distribution Lecture Notes in Computer Science, Vol. 2588. Springer-Verlag (2004)
4. Mihalcea, R., Moldovan, D.I.: A Method for Word Sense Disambiguation of Unrestricted Text. In: Proc. of the 37th Annual Meeting of the Association for Computational Linguistics (ACL-99). Maryland, NY, U.S.A. (1999) NOTA: sostituire con Senseval-3
5. Rosso, P., Masulli, F., Buscaldi, D., Pla, F., Molina, A.: Automatic Noun Disambiguation. Lecture Notes in Computer Science, Vol. 2588. Springer-Verlag (2003) 273–276.

Two Web-Based Approaches for Noun Sense Disambiguation

Paolo Rosso[1], Manuel Montes-y-Gómez[2,1], Davide Buscaldi[3], Aarón Pancardo-Rodríguez[2], and Luis Villaseñor Pineda[2]

[1] Dpto. de Sistemas Informáticos y Computación (DSIC), Universidad Politécnica de Valencia, Spain
{prosso, mmontes}@dsic.upv.es
[2] Lab. de Tecnologías del Lenguaje
Instituto Nacional de Astrofísica, Optica y Electrónica, Mexico
{mmontesg, aaron_cyberman, villsen}@inaoep.mx
[3] Dipartimento di Informatica e Scienze dell'Informazione (DISI), Università di Genova, Italy
buscaldi@disi.unige.it

Abstract. The problem of the resolution of the lexical ambiguity seems to be stuck because of the knowledge acquisition bottleneck. Therefore, it is worthwhile to investigate the possibility of using the Web as a lexical resource. This paper explores two attempts of using Web counts collected through a search engine. The first approach calculates the hits of each possible synonym of the noun to disambiguate together with the nouns of the context. In the second approach the disambiguation of a noun uses a modifier adjective as supporting evidence. A better precision than the baseline was obtained using adjective-noun pairs, even if with a low recall. A comprehensive set of weighting formulae for combining Web counts was investigated in order to give a complete picture of what are the various possibilities, and what are the formulae that work best. The comparison across different search engines was also useful: Web counts, and consequently disambiguation results, were almost identical. Moreover, the Web seems to be more effective than the WordNet Domains lexical resource if integrated rather than stand-alone.

1 Introduction

The problem of the resolution of the lexical ambiguity that appears when a given word in a context has several different meanings is commonly referred as Word Sense Disambiguation (WSD). The state of the art of WSD [15] shows that the supervised paradigm is the most efficient. However, due to the lack of big sense tagged corpora (and the difficulty of manually creating them), the unsupervised paradigm tries to avoid, or at least to reduce, the knowledge acquisition problem the supervised methods have to deal with. In fact, unsupervised methods do not need any learning process and they use only a lexical resouce (e.g. WordNet) to carry out the word sense disambiguation task [1] [16] [17] [19].

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 267–279, 2005.

In order to tackle the problem of the knowledge acquisition bottleneck, it seems to make sense to investigate the possibility to use the Web as an extra lexical resource for WSD. The majority of methods which use the Web try to automatically generate sense tagged corpora [13] [2] [9] [18]. In this paper, we describe our first attempt to use the Web not for extracting training samples but for helping during the word sense disambiguation process. Our work was initially inspired by [14] in which noun-verb relationships are looked for in the Web. The Web as corpus for linguistic research [22] was alredy used with success in many Natural Language Processing areas: question answering [4], question classification [20], machine translation [10], anaphora resolution [6], Prepositional Phrase [8], PP attachment ambiguity [21], and ontology learning [3].

The two approaches we present are based on the idea of redundancy of information in the Web [5]. The first approach exploits the redundancy of the Web to calculate the probabilities associated to each sense of a certain noun according to its context. The second approach tries intuitively to disambiguate a noun on the basis of the adjective that modifies it. Because of the redundancy of the Web, the probability of finding an adjective-noun pair or a noun together with its noun-context increases. The two approaches are based on the hypothesis that a document has a thematic cohesion which is reflected in a semantic relantionship between its words. Therefore, the noun to disambiguate and its noun-context (in the second approach the noun and the adjective which goes before it) have a certain semantic relationship which should become apparent in the Web in a high co-occurrence of the noun-context with the synonyms and hypernyms of the noun to disambiguate (in the second approach, the adjective with the synonyms, the hypernyms and also the hyponyms of the noun).

In the following two sections we describe the two unsupervised context nouns and the adjective-noun pairs Web-based approaches. In the fourth section we present the results of the preliminary experiments, whereas in the fifth section we compare them when different search engines are used. In the last section we discuss the results we obtained when frequency of terms in SemCor was also taken into account. Finally, the conclusions are drawn and the further work is planned.

2 Description of the Noun-Context Approach

Given a noun w, with $|w|$ senses and within a context C of nouns, the function $F(w_k, C)$ indicates the thematic cohesion between the sense w_k of the noun and its context. The estimation of $F(w_k, C)$ is based on the Web. It is carried out by considering the n synonyms $\{s_{ik}, 0 < i \leq n\}$ of the k-th sense w_k of the noun, and the m words in the direct hypernym synset $\{h_{jk}, 0 < j \leq m\}$. The occurrence of these elements within the given context is computed by the fuction $f_S(x, y)$. This function returns the number of pages containing the pattern x AND y, obtained by searching the web with a search engine S. Besides, we name $f_S(x)$ the function returning the number of Web pages containing the string x according to the search engine S. If we assume that $F(w_k, C) \approx P_{Web}(w_k|C)$, i.e., that the

thematic cohesion between the k-th sense w_k of the noun and its context is proportional to the probability of finding an occurence of the noun w with sense w_k in a Web page containing the nouns from the context C, then $F(w_k, C)$ can be calculated using one of the two following formulae:

$$F_1(w_k, C) = \frac{1}{n+m}\left(\sum_{i=1}^{n} P(s_{ik}|C) + \sum_{j=1}^{m} P(h_{jk}|C)\right) \tag{1}$$

$$F_2(w_k, C) = \arg \max_{\substack{0<i\leq n,\\ 0<j\leq m}} (P(s_{ik}|C), P(h_{jk}|C)) \tag{2}$$

where $P(s_{ik}|C) = f_S(C, s_{ik})/f_S(C)$, and $P(h_{jk}|C) = f_S(C, h_{jk})/f_S(C)$.

Similarly, if we presume that $F(w_k, C) \approx P_{Web}(C|w_k)$, that is, that the thematic cohesion between the k-th sense w_k of the noun and its context is proportional to the probability of finding all the nouns of the context in a Web page containing an occurrence of the noun w with the k-th sense w_k, then the thematic cohesion can be computed using one of the two formulae below:

$$F_3(w_k, C) = \frac{1}{n+m}\left(\sum_{i=1}^{n} P(C|s_{ik}) + \sum_{j=1}^{m} C|P(h_{jk})\right) \tag{3}$$

$$F_4(w_k, C) = \arg \max_{\substack{0<i\leq n,\\ 0<j\leq m}} (P(C|s_{ik}), P(C|h_{jk})) \tag{4}$$

where $P(C|s_{ik}) = f_S(C, s_{ik})/f_S(s_{ik})$, and $P(C|h_{jk}) = f_S(C, h_{jk})/f_S(h_{jk})$.

The formulae F_1 and F_3 are based on the *average* weights of the probabilities, and they presume that all the synonyms and hypernyms of a current sense must be related to its context in order to distinguish a thematic cohesion. On the contrary, the formulae F_2 and F_4 are based on the *maximum* of the probabilities (i.e., it is enough to find one frequent synonym-hypernym of the current sense w_k of the noun in a given context in order to establish the thematic cohesion between them).

The algorithm to disambiguate a noun using its noun-context is basically divided into the following steps:

1. Select the set of nouns around the noun to disambiguate w using the sentence as window size (let us named this set C).
2. For each sense w_k of w, and for every synonym s_{ik} and direct hypernym h_{jk} of w_k, compute $f_S(C, s_{ik})$, $f_S(C, h_{jk})$, $f_S(s_{ik})$, and $f_S(h_{jk})$.
3. Assign to each sense w_k a weight depending on a function $F(w_k, C)$ which combines the results obtained in the step before.
4. Select the sense having the resulting highest weight.

3 Description of the Adjective-Noun Pairs Approach

The disambiguation of a noun w with $|w|$ senses is carried out by taking into account the adjective a referring to the noun itself, the n synonyms $\{s_{ik}, 0 \leq i < n\}$ of the k-th sense (w_k) of the noun, and the m words in the direct hypernym synset of w_k, $\{h_{jk}, 0 \leq j < m\}$ (or also in one of its hyponym synsets). We name $f_S(x, y)$ the function returning the number of pages containing the pair "x y", obtained by searching the Web with a search engine S, where x and y are strings. Moreover, we name $f_S(x)$ the function returning the number of pages containing the string x, by using a search engine S. The weights obtained for the k-th sense (w_k) of the noun to disambiguate w depend on a formula $F(w_k, a)$. The quite comprehensive set of weighting methodologies for combining Web counts used during the experiments is described below. This study give an interesting picture of what are the various possibilities for performing the disambiguation of nouns using modifier adjectives as supporting evidence.

- F_I: This formula is based on the *average* of weights and it takes into account the probabilities of each synonym s_{ik} and each hypernym h_{jk} of having the same sense of w_k. The probability, $p(x|w_k)$, was calculated over the SemCor corpus: even if probabilities vary with domain, in this approximation we assumed that they are the same over the SemCor and the Web.

$$F_I(w_k, a) = \frac{1}{2}\left(\frac{\sum_{i=0}^{n-1} f_S(a, s_{ik})p(s_{ik}|w_k)}{n} + \frac{\sum_{j=0}^{m-1} f_S(a, h_{jk})p(h_{jk}|w_k)}{m}\right) \quad (5)$$

  The motivation of taking into account this probability is that some words can appear in the Web with a different sense than the appropriate one, e.g. *air* as synonym of *melody* is rare, with a probability $p(\text{"}air\text{"}|6598312) = 0.0022$, where 6598312 is the WordNet 2.0 offset corresponding to the synset {*tune, melody, strain, air, line, melodic line, melodic phrase*}.
- F_{II}: This formula derives directly from F_I and it takes into account also the hyponyms of the sense w_k of the noun to disambiguate. The hyponyms can be seen as "use cases" of the sense of the word to disambiguate. The hyponym weights are computed in exactly the same way of the synonyms and hypernyms in the formula above, where 1/2 is replaced by 1/3.
- F_{III}: This formula calculates the *maximum* of weights instead of the average. It also takes into account the probabilities of synonyms, hypernyms and hyponyms.

$$F_{III}(w_k, a) = \max_{\substack{0 \leq i < n, \\ 0 \leq j < m}} (f_S(a, s_{ik})p(s_{ik}|w_k), f_S(a, h_{jk})p(h_{jk}|w_k)) \quad (6)$$

- F_{IV}: This formula is based on the *Mean Mutual Information*[23], or *Relative Entropy, similarity measure*. The formula measures how much information

is in the dependency of two successive words. It has been adapted to take into consideration information obtained both by synonyms and hypernyms:

$$F_{IV}(w_k, a) = \sum_{i=0}^{n-1} f_S(a, s_{ik}) \log \frac{f_S(a, s_{ik})}{f_S(a) f_S(s_{ik})} + \sum_{j=0}^{m-1} f_S(a, h_{jk}) \log \frac{f_S(a, h_{jk})}{f_S(a) f_S(h_{jk})} \quad (7)$$

The algorithm to disambiguate a noun using an adjective as modifier is basically divided into the following steps:

1. Select the adjective a immediately before the noun w;
2. for each sense w_k of w, and for every synonym s_{ik} and direct hypernym h_{jk} of w_k, compute $f_S(a, s_{ik})$ and $f_S(a, h_{jk})$ (in some formulae we used also the direct hyponyms of the noun);
3. assign to each sense w_k a weight depending on a formula F which combines the results obtained in the step before;
4. select the sense having the resulting highest weight.

For example, consider the following sentence, extracted from the Senseval-3 All-Words task corpus: *A faint crease appeared between the man's eyebrows.* Suppose we are disambiguating the word *crease*, having three senses, according to WordNet 2.0: $crease_1 : \{fold, crease, plication, flexure, crimp, bend\}$, $crease_2 : \{wrinkle, furrow, crease, crinkle, seam, line\}$ and $crease_3 : \{kris, creese, crease\}$. The direct hyperonyms are, for each sense: $h_1 =$*{angular shape, angularity}*, $h_2 =$*{depression, impression, imprint}* and $h_3 =$*{dagger, sticker}*. Then we search the web for the following pairs: (*faint fold*), (*faint plication*), (*faint flexure*), (*faint crimp*), (*faint bend*), (*faint angular shape*), (*faint angularity*) for the first sense, (*faint wrinkle*), (*faint furrow*), etc. for the second sense and so on.

4 Preliminary Experimental Results

The preliminary noun sense disambiguation experiments were carried out over 215 nouns of the Senseval-3[1] corpus for the English all words task [24]. Web counts were collected through the MSN Search[2]. In the first approach, frequencies are used to calculate the cohesion of the different noun senses with respect to its noun-context. Table 1 shows the poor results obtained for the noun-context approach, always below the Most Frequently Sense (MFS) baseline (0.689 over the all Senseval-3 English all-words task corpus). The two leftmost columns show the overall precision obtained by the four formulae when just the best answer was accepted, whereas the following two columns illustrate the precision when

[1] http://www.senseval.org
[2] http://search.msn.com

Table 1. Noun-context approach. S: number of senses allowed in the evaluation; F: Formula; P: overall Precision; columns 1-2: best answer, columns 3-4: best two answers, columns 5-6: best $\lceil n/2 \rceil$ answers (fuzzy WSD)

F	P	F	P	F	P
F_1	0.181	F_1	0.362	F_1	0.548
F_2	0.190	F_2	0.358	F_2	0.609
F_3	0.209	F_3	0.400	F_3	0.637
F_4	0.218	F_4	0.469	F_4	0.679

the first two answers were accepted. Finally, the two rightmost columns indicate that also for a fuzzy WSD in which $\lceil n/2 \rceil$ answers were accepted results were pretty poor. The aim of this part of the experiments was to understand if this approach could be useful at least for putting aside the not likely senses.

In all cases, formulae F_3 and F_4 behaved better than F_1 and F_2. This result is quite important because the hypothesis of thematic cohesion in the Web described in the second section shows that normalising with respect to $f_S(s_{ik})$ and $f_S(h_{ik})$ makes the calculation of probabilities less sensitive to very rare or general synonyms and hypernyms. Finally, we observed that also the formulae F_2 and F_4 which take into account the maximum probability between the context and one of the synonyms or the hypernyms of the noun to disambiguate, are also less sensitive to very rare or general synonyms and hypernyms (the calculation of the maximum seems to be less sensitive than the average).

The result analysis allowed us to understand that one of the reasons of the poor performance of the method could be that whereas a window of a sentence is used to select the set of nouns of the context, the pattern x AND y may be contained in the context of a document (the Web page returned by the search engine).

The approach failed especially for the disambiguation of highly polysemic nouns (with more than five senses on average), and in case of right answer, generally, the probability of the right sense was much higher than those of the other senses. Therefore, Web-based approaches like this could be more effective if integrated with other WSD methods rather than stand-alones.

Table 2. Adjective-noun pairs approach. F: formula; P: overall Precision; R: overall Recall; C: overall Coverage; P_{na}: Precision over the disambiguated nouns (i.e., nouns with an adjective before: "adjective noun")

F	P	R	C	P_{na}
MFS	0.689	0.689	100%	0.623
F_I	0.627	0.271	43.3%	0.318
F_{II}	0.661	0.286	43.3%	0.392
F_{III}	0.660	0.278	42.0%	0.373
F_{IV}	0.579	0.239	41.2%	0.179

With respect to the adjective-noun pairs approach, Table 2 shows the results of the preliminary experiments we carried out using the MSN search engine and the formulae described in the third section. For each formula we obtained worse results than the Most Frequently Sense baseline. An interesting result is that the hyponym information seems to be helpful when using the Web to perform WSD. For instance, the formula F_{II} obtained a better performance than the related formula F_I which used only synonyms and direct hypernyms.

5 Comparison of Search Engines

The results of the noun-context approach obtained using the MSN search engine were compared with those obtained when AltaVista[3] and Google[4] were used. Table 3 shows the results of this comparison. It is interesting to notice that the precision obtained with the three different search engines are almost identical (a difference of 0.03 on average).

Table 3. Noun-context approach: comparison of search engines. S: number of senses allowed in the evaluation; F: formula; P_{MSN}: overall Precision with MSN; P_{AV}: overall Precision with Altavista; P_G: overall Precision with Google; columns 1-4: best answer, columns 5-8: best two answers, columns 9-12: best $\lceil n/2 \rceil$ answers (fuzzy WSD)

F	P_{MSN}	P_{AV}	P_G	F	P_{MSN}	P_{AV}	P_G	F	P_{MSN}	P_{AV}	P_G
F_1	0.181	0.186	0.227	F_1	0.362	0.353	0.348	F_1	0.548	0.534	0.595
F_2	0.190	0.186	0.279	F_2	0.358	0.339	0.395	F_2	0.609	0.576	0.595
F_3	0.209	0.215	0.237	F_3	0.400	0.386	0.432	F_3	0.637	0.613	0.641
F_4	0.218	0.215	0.251	F_4	0.469	0.427	0.451	F_4	0.679	0.623	0.646

With respect to the adjective-noun pairs approach, the experimental results of the fourth section, in which with the MSN search engine was used, were compared to those obtained when AltaVista. In this comparison, we decided also to use the Lucene search engine[5], substituting the Web with the TREC-8 collection[6], in order to evaluate the effectiveness of the Web with respect to a large document collection. We calculated the precision, recall and coverage over the Senseval-3 AWT corpus, for every search engine, using the formulae described in the third section. The results obtained evidentiate that MSN and AltaVista are equivalent (even if we obtained slight differences in some experiments, on average results are almost the same for both engines). We decided to use MSN to carry out the experiments due to a lower response time for the queries (e.g. the duration of the most demanding experiment needed 237 minutes with AltaVista whereas

[3] http://www.altavista.com
[4] http://www.google.com
[5] http://jakarta.apache.org/lucene
[6] http://trec.nist.gov

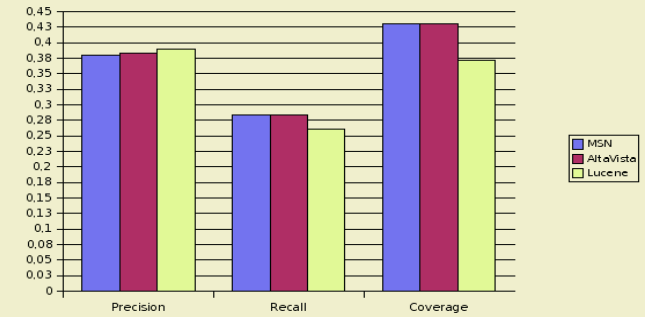


Fig. 1. Adjective-noun pairs approach: comparison of search engines; averaged Precision (polysemic words only), Recall and Coverage over the used formulae

171 minutes with MSN). A remarkable difference between the Web-based search engines and the Lucene is the 6% drop in coverage obtained when using the TREC-8 collection instead of the Web, confirming that the huge redundancy of data in the web allows to disambiguate a greater number of words. Moreover, the precision obtained with Lucene is only 1% higher than the precision obtained when using the web. We expected a higher precision, due to the lower quality of the information in the Web. However the improvement is not so evident to justify the use of a large text collection instead of the Web. Figure 1 shows the obtained results.

6 Experimental Results with Frequency Correction

A second tranche of experiments was carried out just for the adjective-noun pairs approach which obtained better results. In this second attempt, we decided to investigate the possibility of including a *frequency correction* (fc) factor in each of the formulae described in the third section. This frequency factor indicates whether the resulting weight for a sense w_k was multiplied by $p(w|w_k)$, that is, the probability for the word w of having sense w_k in the SemCor corpus, or not. Moreover, a new formula based on a different similarity measure was studied during these experiments. This formula resembles the *Similarity Theorem* [11], and the problem due to large denominators is reduced thanks to the use of logarithms.

$$F_V(w_k, a) = \max_{\substack{0 \le i < n, \\ 0 \le j < m}} \left(f_S(a, s_{ik}) \frac{\log f_S(a, s_{ik})}{\log f_S(s_{ik})}, f_S(a, h_{jk}) \frac{\log f_S(a, h_{jk})}{\log f_S(h_{jk})} \right) \qquad (8)$$

Table 4 shows that the frequency-corrected formulae outperform those without the frequency factor. Moreover, an average 4% gain is obtained in recall when frequency is taken into account. In one case (with F_V) we obtained better results than the MFS baseline. We believe that it could depend on the fact that this formula has the advantage of not taking into account only the relevance of the

Table 4. Adjective-noun pairs approach with and without frequency factor. F: Formula; fc: frequency correction; P: overall Precision; R: overall Recall; C: overall Coverage; P_{na}: Precision over the disambiguated nouns (i.e., nouns with an adjective before: "adjective noun"); P_{nd}: Precision over the nouns not disambiguated by the CD method

F	fc	P	R	C	P_{na}	P_{nd}
MFS		**0.689**	0.689	100%	**0.623**	**0.629**
CD		0.734	0.518	70.5%	0.625	0.000
$CD + WND$		0.653	0.584	89.3%	0.583	0.500
F_I	no	0.627	0.271	43.3%	0.318	0.328
F_I	yes	0.718	0.311	43.3%	0.511	0.478
$F_I I$	no	0.661	0.286	43.3%	0.392	0.367
$F_I I$	yes	0.759	0.329	43.3%	0.596	0.507
$F_I II$	no	0.660	0.278	42.0%	0.373	0.333
$F_I II$	yes	0.755	0.326	43.1%	0.586	0.500
F_{IV}	no	0.579	0.239	41.2%	0.179	0.152
F_{IV}	yes	0.720	0.259	42.1%	0.532	0.565
F_V	yes	**0.777**	0.337	43.3%	**0.634**	**0.666**

adjective but also the number of co-occurrences of the pair adjective-noun. We also compared the adjective-noun pairs approach with one based on Conceptual Density (CD) [7] and WordNet Domains (WND) [12]. The Web-based disambiguation provided better results, especially over the words not disambiguated by the standard CD method (16.6% with respect to the CD+WND formula). Therefore, the Web was more effective than the WordNet Domains. Moreover, the Web-based approach was more effective when integrated with another system rather than stand-alone.

We investigated the importance of polysemy of the adjective in the disambiguation of a noun. We calculated the averaged polysemy of the adjective when the referred word was disambiguated correctly and when the word was assigned the wrong sense. The results showed in Table 5 demonstrate that the less polysemic is the adjective, the higher will be the probability of selecting the right sense. However, frequency-corrected formulae tend to be less subject to the polysemy of the adjective, obtaining values for the polysemy of the adjective closer to the values obtained with the MFS heuristics.

Finally, we investigated also the possibility of using the same counter-intuitive approach to disambiguate an adjective based on the noun which goes after (i.e., using the Web to look for $f_S(a_{ik}, w)$, where a_{ik} is the i-th synonym of the k-th sense of adjective a). In fact, traditionally, this is done the other way around and evidence for the sense of an adjective is found by looking at the noun it modifies. In order to do so, we used an equivalent formula to F_V (the formula we obtained the best results for the disambiguation of nouns). Unfortunately, in this first attempt we obtained quite poor results (21.3% precision).

Table 5. Polysemy of adjectives. F: Formula; fc: frequency correction; $Right$: average polisemy of adjectives for correctly disambiguated nouns; $Wrong$: average polisemy of adjectives for incorrectly disambiguated nouns

F	fc	$Right$	$Wrong$
MFS		4.26	4.3
F_I	no	3.65	4.55
F_I	yes	4.18	4.40
F_{II}	no	3.28	4.86
F_{II}	yes	4.17	4.40
F_{III}	yes	4.17	4.41
F_{IV}	no	3.9	4.36
F_V	yes	4.87	5.54

7 Conclusions and Further Work

The paper explores the disambiguation of nouns using Noun-context or modifier adjectives as supporting evidence, and using Web counts collected through different search engines. The comparison we made across the different search engines should be useful, and we consider interesting that the Web counts (and consequently disambiguation results) obtained with different search engines are almost identical (despite the fact that the search engines considered in the experiments could cover different sections of the Web).

The main aim of the paper is to bring a contribution in terms of various weighting methodologies for Web counts, and in terms of insights into methodologies that work best for the purpose of word sense disambiguation. A noun-context approach was first investigated. In this approach the hypothesis of thematic cohesion in a document is made and the sense is chosen as a statistics of the co-occurrence in the Web of the context and the synonyms and hypernyms of the noun to disambiguate. Although the system obtained very poor results (only a precision of 28%, if only one sense is accepted, and of 68% in case of fuzzy WSD) it could be a promissing approach if more contextual information rather nouns is taken into account to increase the precision (especially for highly polysemic nouns). As further work we plan to carry out some experiments including hyponyms instead of hypernyms (we realised that hypernyms do not characterise very well the meaning of a particular noun sense with respect to the other senses of the same noun, whereas hyponyms usually do better when dealing with corpora). Moreover, it could be also interesting to take into account the probability in SemCor for each noun (as we did for the adjective-noun pairs approach).

The approach based on adjective-noun pairs obtained instead a better precision than the baseline, even if with a low recall. We believe that this depends on that the majority of pairs is still ambiguous. That is, the adjective is not enough to understand the meaning of the noun and a bigger context should be taken into account. Our study over the importance of the polisemy of the adjective in

the disambiguation seems to confirm our intuition. For example, the pair *cold fire* is ambiguous, since it can be assigned both the sense corresponding to *cold passion* or the sense corresponding to *cold fire*.

We detected some problems in the use of WordNet synonyms and hypernyms, since they are composed of multi-word expressions rarely found in the Web. Our further investigation directions will be to investigate the use of another ontology to overcome the multi-word expression issue, as well as to use shallow parsers in order to determine an unambiguous context for the word to disambiguate. It could be more intuitive to disambiguate a noun based on a syntactically related verb than on a modifier adjective. The aim is to understand whether it is most likely obtaining significantly better results using such noun-verb relationships.

We conclude remarking that preliminary results showed that it should be better using the Web as lexical resource for WSD if integrated with existing systems rather than using it stand-alone. Moreover, when we integrated the Web, instead of the WordNet Domains, with a unsupevised method based on conceptual density we obtained better results.

As further work for both approaches we plan to take into consideration the complete direct surroundings of each concept: hyponyms, part-of, derived, or even words from the corresponding gloss. Moreover, with respect to the comparison across search engines, it would also be interesting to make a study across results provided by the same search engine, at different points in time. This could make an interesting result that could validate (invalidate) claims made by some researchers that Web counts are not very stable over time.

Acknowledgments

We would like to thank R2D2 CICYT (TIC2003-07158-C04-03), CONACyT-Mexico (43990A-1, U39957-Y) and ICT EU-India (ALA/95/23/2003/077-054) projects, as well as the Secretaría de Estado de Educacin y Universidades de España, for partially supporting this work.

References

1. Agirre, E., Rigau, G.: A Proposal for Word Sense Disambiguation using Conceptual Distance. In: Proc. of the Int. Conf. on Recent Advances in NLP. RANLP'95. (1995)
2. Agirre, E., Martinez, D.: Exploring Automatic Word Sense Disambiguation with Decision Lists and the Web. In: Proc. of the the COLING-2000. (2000)
3. Agirre, E., Olatz, A., Hovy, Martinez, D.: Enriching Very Large Ontologies using the WWW. ECAI 2000, Workshop on Ontology Learning. Berlin. (2000)
4. Brill, E., Lin, J., Banko, M., Dumais, S., Ng, A.: Data-intensive Question Answering. In: Proc. of the Tenth Text REtrieval Conference TREC-2001. (2001).

5. Brill, E.: Processing Natural Language Processing without Natural Language Processing. Proc. Computational Linguistics and Intelligent Text Processing (CICLing-2003), Gelbukh Ed. Lecture Notes in Computer Science, Vol. 2588. Springer-Verlag (2003) 360–369.
6. Bunescu, R.: Associative Anaphora Resolution:A Web-Based Approach. In: Proc. of the EACL-2003 Workshop on the Computational Treatment of Anaphora, Budapest, Hungary, April. (2003).
7. Buscaldi, D., Rosso, P., Masulli, F.: The upv-unige-CIAOSENSO WSD System. Senseval-3 Workshop, Association for Computational Linguistics (ACL-04). Barcelona, Spain (2004).
8. Calvo, H., Gelbukh., A.: Improving Prepositional Phrase Attachment Disambiguation Using the Web as Corpus. Lecture Notes in Computer Science, Vol. 2905, Springer-Verlag (2003) 604–610.
9. Gonzalo, J., Verdejo, F., Chugar, I.: The Web as a Resource for WSD. In: 1st MEANING Workshop, Spain. (2003)
10. Grefenstette, G.: The World Wide Web as a resource for example-based Machine Translation Tasks. In: Proc. of Aslib Conference on Translating and the Computer. London. (1999)
11. Lin, D.: An Information-Theoretic Definition of Similarity. In: Proc. of the 15th Int. Conf. on Machine Learning. Toronto, Canada (2003)
12. Magnini, B. and Cavaglià, G.: Integrating Subject Field Codes into WordNet. In: Proc. of LREC-2000, 2nd Int. Conf. on Language Resources and Evaluation. (2000) 1413–1418.
13. Mihalcea, R., Moldovan, D.I.: An Automatic Method for Generating Sense Tagged Corpora. In: Proc. of the 16th National Conf. on Artificial Intelligence. AAAI Press. (1999)
14. Mihalcea, R., Moldovan, D.I.: A Method for Word Sense Disambiguation of Unrestricted Text. In: Proc. of the 37th Annual Meeting of the Association for Computational Linguistics (ACL-99). Maryland, NY, U.S.A. (1999)
15. Mihalcea, R., Edmonds, P. (Eds.): Proc. of Senseval-3: The 3rd Int. Workshop on the Evaluation of Systems for the Semantic Analysis of Text. Barcelona, Spain. (2004)
16. Montoyo, A: Método basado en Marcas de Especificidad para WSD. Procesamiento del Lenguaje Natural. In: Revista n 26, Septiembre. (2000).
17. Rosso, P., Masulli, F., Buscaldi, D., Pla, F., Molina, A.: Automatic Noun Disambiguation. Proc. Computational Linguistics and Intelligent Text Processing (CICLing-2003), Gelbukh Ed. Lecture Notes in Computer Science, Vol. 2588. Springer-Verlag (2003) 273–276.
18. Santamaria, C., Gonzalo, J., Verdejo, F.: Automatic Association of WWW Directories to Word Senses. Computational Linguistics (2003), Vol. 3, Issue 3 - Special Issue on the Web as Corpus, 485–502.
19. Sidorov, G., Gelbukh, A.: Word Sense Disambiguation in a Spanish Explanatory Dictionary. In: TALN-2001: Tratamiento Automtico de Lengauje Natural, France, 2001, pp 398-402; www.gelbukh.com/CV/Publications/2001/TALN-2001-WSD.htm.
20. Solorio, T., Pérez, M., Montes, M., Villaseñor, L., López, A.: A Language Independent Method for Question Classification. In: Proc. of the 20th Int. Conf. on Computational Linguistics (COLING-04). Geneva, Switzerland. (2004)

21. Volk, M.: Exploiting the WWW as a Corpus to Resolve PP Attachment Ambiguities. In: Proc. of Corpus Linguistics. Lancaster. (2001)
22. Volk, M.: Using the Web as Corpus for Linguistic Research. In: Catcher of the Meaning. Pajusalu, R., Hennoste, T. (Eds.). Dept. of General Linguistics 3, University of Tartu, Germany. (2002)
23. Wackerbauer, R., Witt, A., Atmanspacher, H., Kurths, J., Scheingraber, H.: A Comparative Classification of Complexity Measures. Chaos, Solitons and Fractals, 4: 133-173 (1994).
24. Zinder, B.,Palmer, M.: In: Proc. of Senseval-3: The 3rd Int. Workshop on the Evaluation of Systems for the Semantic Analysis of Text. Barcelona, Spain. (2004)

Finding Instance Names and Alternative Glosses on the Web: WordNet Reloaded

Marius Paşca

Google Inc., 1600 Amphitheatre Parkway,
Mountain View, California 94043
mars@google.com

Abstract. This paper presents an approach to extending existing lexical resources with instance names and alternative definitions acquired from textual documents. The experiments involve WordNet and approximately 300 million Web documents, but the method is more generally applicable. We leverage formally-structured, human-validated resources, on one hand, and data-driven instance names and definitions on the other, which opens the path to new applications of the reloaded resources.

1 Motivation and Goals

Large-scale lexical, hierarchical resources have a broad range of applications in computational linguistics, information extraction and information retrieval. When manually building such resources, the focus is justifiably on selecting and organizing words into hierarchies of conceptual entries, with manual selection of an ideal, single definition for each entry. For example, by grouping together English words with the same meaning (e.g., *lawyer* and *attorney*) into sets of synonyms (or *synsets*, such as {*lawyer, attorney*}) associated with a single definition (or *gloss*), WordNet [1] became a de-facto standard for lexical resources. Its uses span word sense disambiguation [2], information extraction [3] and machine translation [4], to name only a few.

Hierarchical resources organize noun synsets along *IsA*/*InstanceOf* relations. The conceptual coverage of WordNet is impressive, with more than 150,000 English words encoded in over 115,000 synset entries or lexical concepts - more than half of which are nouns. However, WordNet and other resources are not necessarily complete for obvious practical reasons. This particularly applies to the lower-level hierarchies, where the more specific concepts occur, in the form of both missing specialized concepts and missing instance names. WordNet does not contain *telecom company* or *meta search engine* under *company* and *search engine* respectively; similarly, there are no instance names such as *Google* under *search engine*, or *Ferrari* under *car company*. Only a fraction of the encoded concepts are accompanied by corresponding instances; the number of such instances embedded under a given concept is usually small. For instance, 600 instance names exist under *city*; comparatively, there are eight instance names under *lawyer* (including *Francis Scott Key* and *Abraham Lincoln*), one instance

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 280–292, 2005.

name under *skyscraper* (*World Trade Center*), and one instance name under *cavern* (*Carlsbad Caverns*). The first goal of this paper is to expand lower-level hierarchies with instance names acquired from textual Web documents.

In most resources, including WordNet, the lexical concept entries contain single rather than multiple strings as definitions. For example, *machine translation* is defined in WordNet as *"the use of computers to translate from one language to another"*. Since definitions are unique per word sense, higher-level applications that operate on them, e.g. lexical chains [5] or semantic similarity measures [6], can rely only on the particular sequence of words actually included in the definitions. However, usually there is more than one way to express the same definition. As an illustration, alternative definitions for *machine translation"* include *"process of translating documents from one language to another by computer"*; *"process by which a machine translates text from one language to another"*; and *"automatic translation of human language by computers"*. [1] The alternative definitions will capture various morphological, lexical and semantic variations, and make them available to the higher-level applications. This also represents a novel source for extracting paraphrases, which are useful in information extraction, document retrieval and question answering. The second goal of this paper is to use Web textual documents to extract alternative glosses or definitions for existing concepts situated in lower-level noun hierarchies.

The remainder of the paper is structured as follows. After an overview of the method in Section 2, Section 3 describes the identification and extraction of relevant text nuggets from unstructured text. The nuggets are the raw material for deriving alternative glosses and new instance names, as shown in Section 4. Section 5 describes experiments on approximately 300 million Web documents. After further discussion in Section 6, we conclude in Section 7.

2 Method at a Glance

The method and experiments described in this paper augment the concepts situated in the lower-level WordNet hierarchies with new information, namely instance names and alternative glosses. As an illustration, the top box (A) in Figure 1 contains part of the original WordNet hierarchy under the lexical concept of *computer program*. In contrast, the other boxes (B, C and D) in the figure contain some of the new information actually extracted from Web documents.

The main source of the newly acquired information are *text nuggets*, which are sentence fragments extracted uniformly from unstructured text. To identify relevant nuggets across the Web, we focus on textual content rather than structural clues. Shallow lexico-syntactic extraction patterns are applied to the unstructured text of Web documents. The extraction patterns are designed to be lightweight and simple to handle robustly the noise and diversity of Web documents. The conversion of the extracted text nuggets into higher-level information, namely instance names and glosses, also relies on minimal text processing

[1] These definitions were extracted from the Web with the method described herein.

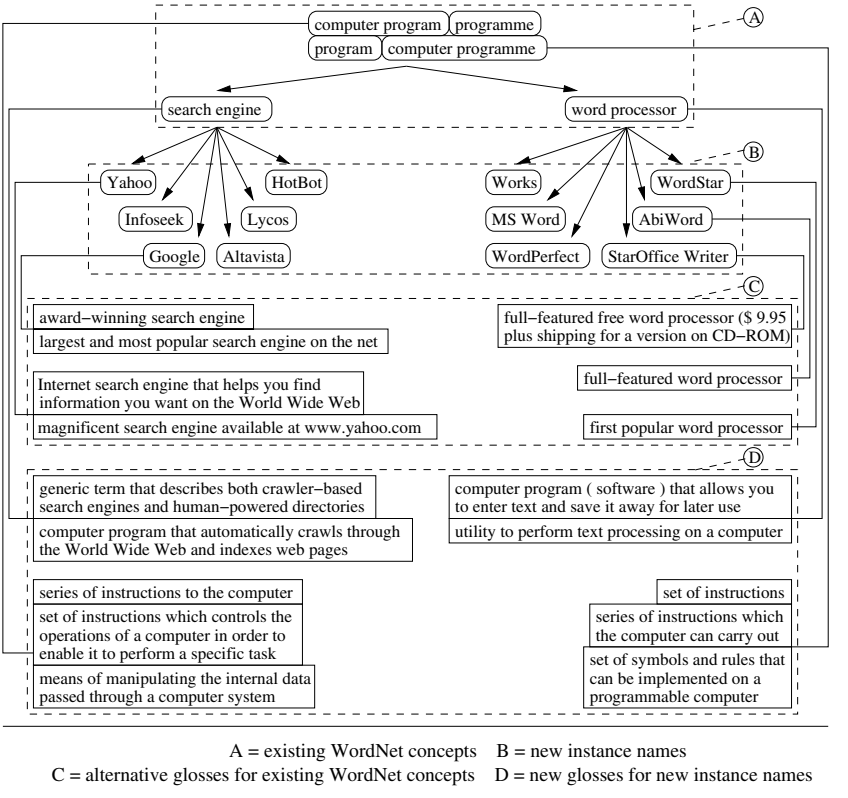


Fig. 1. Overview of augmenting WordNet with instance names and alternative glosses extracted from Web text nuggets

and robust heuristics. The overall method is data-driven, without any a-priori restrictions on the type of the targeted concepts and glosses. Thus, it harnesses some of the unstructured knowledge available across the Web.

3 The Web as a Source of Factual Text Nuggets

The goal of the Web is sharing information and knowledge. The use of complex text processing tools as a step towards accessing the knowledge within the text is impractical. Without attempting true text understanding, it is still possible to extract a small part of the available knowledge via shallow text processing. In this case, the knowledge is assumed to be encoded within text nuggets.

3.1 Text Nuggets

A text nugget captures a factual property of a lexical concept (phrase or word). Both the nugget and the lexical concept occur in text. The type of property

Table 1. Examples of Web sentences containing descriptive (*Desc*) and categorical (*Categ*) nuggets (W=lexical concept; X=text nugget)

Desc	[WordNet]^W is an [English lexical reference system based on current psycholinguistic theories of human lexical memory]^X.
Categ	[The world is less dominated by mega-cities (10 million or more people)]^X such as [Mexico City]^W, than many predicted only a few years ago [..]

encoded in the nugget determines its form and relation with the concept. The presence of potential concepts and simple domain-independent, lexico-syntactic patterns in sentences is a signal of a text nugget associated with the concept. A *descriptive* nugget introduces distinguishing properties (differentia) of the lexical concept W to which it is associated, by connecting it to a description X. As shown in Table 1, descriptive nuggets often occur in the form of appositives, linking verbs and subordinate clauses. The patterns are encoded as:

(1) ⟨W [,] [who|which] [is|was] [the|a|an] X [,|.]⟩
(2) ⟨[StartOfSent] W [,] [a|an|the|who|which] X [,]⟩
(3) ⟨[StartOfSent] [A|An|nil] W [is|was] [the|a|an] X [,|.]⟩

A *categorical* nugget, as shown in Table 1, is likely to provide the category (genus) of the associated concept. The set of patterns can be summarized as:
(4) ⟨[StartOfSent] X [such as|including] W [and|,|.]⟩.

3.2 Extraction of Text Nuggets

To ensure robustness on large collections, the extraction relies on lightweight tools and minimal resources. As a pre-requisite, the input documents are first pre-processed to filter out HTML tags. After tokenization and sentence-boundary detection, documents are part-of-speech tagged using the TnT tagger [7]. Each of the lexico-syntactic patterns is matched against document sentences, resulting in pairs (W,X) of a concept and an associated text nugget for each match. There are two modes of operation, depending on how the concepts W are detected:

1. The concepts are part of a *closed vocabulary* (e.g., WordNet nouns) which is given as input. In this case, their detection is equivalent to searching the current sentence for the longest matching vocabulary entries that are not preceded by noun modifiers (other nouns or adjectives), then checking for the presence of a pattern around them.
2. The concepts are part of an *open vocabulary*, which is not specified as part of the input. Since languages such as English tend to distinguish proper names from other nouns through capitalization, each sequence of capitalized terms in the sentence is marked as a potential concept. Non-capitalized sequences (complex noun phrases) are not considered due to increased difficulty in detecting their boundaries.

In both cases, potential concepts that are not associated with a text nugget via a pattern are discarded. The output is a set of concepts with their corresponding nuggets (descriptive or categorical) as derived from Web documents.

4 Derivation of Higher-Level Information

Text nuggets represent low-level information that is not directly suitable for existing resources. This section describes the processing of descriptive and categorical nuggets to derive alternative glosses and new instance names respectively.

4.1 Alternative Glosses

Given a lexical concept, its descriptive nuggets define a semantic space that is usually only partially overlapping with that defined by the corresponding WordNet gloss. The partial (vs. complete) semantic overlap is mainly due to two reasons. First, a nugget may reveal properties that are different (although useful) and therefore not directly comparable to those included in the manually-created WordNet gloss. Second, a nugget may include a "perfect" definition, but for a sense that is different from the one(s) in WordNet. Examples are *Jaguar*, which is found as an *animal* in WordNet but also as a *car company*, *operating system codename*, and *64-bit video game system* in the nuggets extracted from the Web; and *Metropolis*, which is a *city* or the *people living in a city* in WordNet, but also a *movie*, *algorithm*, *club*, *magazine* and *festival* in the extracted nuggets. Therefore, the main issue in converting the descriptive nuggets into alternative glosses is how to divide the semantic space defined by the set of nuggets.

The solution proposed here is to perform hierarchical agglomerative clustering [8] of the descriptive nuggets of a given lexical concept, based on pairwise nugget similarities. A practical metric for the similarity of two nuggets is the dot-product of their term vectors, after removal of stop words and vector Euclidean-length normalization. The initial weights are the term frequencies within the nugget. The first few (three, in this case) non-stop terms in the nugget are heuristically assigned higher weights (three times higher), following the intuition that they correspond frequently to the genus of the lexical concept. In terms of clustering method, our early experiments suggest that group-average clustering [8] is the best for our purpose. The method starts by placing each nugget into a separate cluster, and builds hierarchical clusters iteratively. All elements (nuggets) of intermediate clusters contribute to the computation of inter-cluster similarities, as a group-average of the pairwise element similarities. The clustering ends when all inter-cluster similarities are lower than a minimum threshold, which is experimentally set to 0.1. The gloss clusters are ranked in decreasing order according to their number of elements as illustrated in Table 2.

4.2 New Instance Names

The process of deriving instance names uses categorical nuggets, which are searched for the noun phrase that encodes the category of the associated lexi-

Table 2. Examples of top-ranked gloss clusters and some of their elements (R/S=rank/size of the cluster)

R/S	*Examples of glosses in the cluster*
Thomas Jefferson:	
1/21	third president of the United States of America
	third president of the US
	3rd president of the US and the author of the Declaration of Independence
2/17	author of the Declaration Of Independence was educated at William and Mary College
	key author of the Declaration of Independence
	principal author of the Declaration of Independence
3/10	best educated and the most original man of his day
	noted scientist and inventor himself
Joshua Tree:	
1/7	fascinating place to visit and photograph
	enchanting place to go for a hike
	great place to see the desert largely unspoiled by vehicles
2/5	big national park in the desert outside LA
	namesake of Joshua Tree National Park near Palm Springs
	home of Joshua Tree National Park and situated in the very heart of the Morongo Basin
3/2	is actually a type of yucca
	variety of yucca and a member of the Lilly family
search engine:	
1/36	computer program that searches the indexes of web sites using keywords
	computer program that automatically crawls through the World Wide Web and indexes web pages
	computer program that searches a database to find those objects that meet the search criteria you specify
2/29	web site that is linked to a database of web sites
	web site that is devoted to searching all the other web sites
	Web site that is like a catalog of the Web

Table 3. Samples of categories and their top instance names acquired from the Web

Category	*Top instance names*
color	Black, Red, White, Blue, Green, Yellow, Orange, Pink, Purple
rapper	Eminem, Jay-Z, Nas, Dmx, Snoop Dogg, Dr. Dre, Ja Rule
high-speed network	ATM, Gigabit Ethernet, B-ISDN, FDDI, Myrinet, Frame Relay
operating system	Linux, Windows, Windows NT, Unix, DOS, Solaris
car rental company	Hertz, Alamo, Budget, Avis, National, Dollar, Thrifty, Europcar

cal concept. This phrase is approximated by the rightmost non-recursive noun phrase whose last component is a plural-form noun, e.g. *mega-cities* for *Mexico City* in one of the entries of Table 1. Such a coarse approximation is more scal-

able to millions of Web documents. The acquisition of instance names with their categories from the Web is described in more detail in [9]. Table 3 illustrates instance names extracted in the open-vocabulary mode of operation. Note that WordNet may contain all new instances of a category (e.g., for *colors*), contain none of them (e.g., for *rappers*), contain only some of them (e.g., for *operating systems*), or not contain the category in any of its entries (e.g., *high-speed network* and *car rental company*).

4.3 Integration into Existing Resources

The insertion of bits of information extracted through shallow text processing from a decentralized, anonymized knowledge repository (the Web) into a high-quality hand-made resource (WordNet) is certainly challenging. Ideally, human intervention should not be needed for double-checking, correcting or guiding the integration process. But this comes at odds with the need to insure the correctness or at least graceful degradation of the resulting resource.

A conservative integration approach will embed new knowledge into WordNet while minimizing the chances of errors. In the case of alternative glosses, this translates into applying a set of restrictive filters to any gloss before linking it to an existing WordNet lexical concept. First, the gloss must belong to a relatively higher-ranked, that is, larger gloss cluster of that word as shown in Table 2. Second, the gloss must have a relatively high similarity with a WordNet gloss of that word. The metric for computing the similarity of a gloss to a reference WordNet gloss is the same as that used for clustering, namely the dot-product of the non-stop, length-normalized term vectors. Table 4 shows the most similar alternative glosses extracted for a subset of existing WordNet concepts.

New instance names correspond to a new node being linked to an existing node at the bottom of the hierarchies. For example, each of the instance names *Google* (in the category *search engine*), *Swiss National Bank* (in *central bank*) and *Joschka Fischer* (in *foreign minister*) generates a new leaf node inserted under the WordNet concepts *search engine*, *central bank* and *foreign minister* respectively. A different set of conservative restriction filters applies here. First, there should be only one possible insertion point, i.e. the category of the name matches exactly one WordNet concept, and the latter is a leaf node. If a category does not match any WordNet concept, its modifiers are discarded until a match is found. Thus, *high-level programming languages*, *Internet portals* and *science fiction writers* match the WordNet concepts *programing language*, *portal*, and *writer* respectively. It is useful to assign glosses to new instance names as well. In this case, an additional conservative restriction is that the gloss must contain the word to which it is linked. For example, if *Google* is inserted under *search engine*, its gloss must contain a lexicalization of *search engine*.

With the conservative approach discussed so far, the restriction filters remove possible errors due to spurious extraction or ambiguity, at the expense of discarding a lot of nuggets that might be otherwise useful. A more aggressive approach gradually removes restrictions. This increases the percentage of new knowledge that can be integrated into WordNet. For example, instance names with several

Table 4. Examples of reference WordNet glosses (*Ref*) with their most similar alternative glosses (*Alt*) extracted from the Web

Ref	(Apollo): Greek god of light; god of prophesy and poetry and music and healing; son of Zeus and Leto; twin brother of Artemis
Alt_1	Greek god of light and music
Alt_2	Greek god of death and pestilence as well as of the sun and medicine
Ref	(chemistry): the science of matter; the branch of the natural sciences dealing with the composition of substances and their properties and reactions
Alt_1	branch of science that deals with the composition and properties of matter
Alt_2	science dealing with the structure and composition of substances and the mechanisms by which changes in composition occur
Ref	(artificial intelligence): the branch of computer science that deal with writing computer programs that can solve problems creatively; "workers in AI hope to imitate or duplicate intelligence in computers and robots"
Alt_1	branch of computer science that involves writing computer programmes that solve problems creatively
Alt_2	branch of computer science concerned with making computers think
Ref	(fluoxetine): a selective-serotonin reuptake inhibitor commonly prescribed as an antidepressant (trade name Prozac)
Alt_1	Selective Serotonin Reuptake Inhibitor (SSRI)
Alt_2	selective serotonin reuptake inhibitor (SSRI) has been suggested in reducing irritability in PTSD patients
Ref	(emoticon): a representation of a facial expression (as a smile or frown) created by typing a sequence of characters in sending email; ":-(and :-) are emoticons"
Alt_1	face created out of keyboard characters
Alt_2	group of 3 or 4 punctuation characters which resemble a face turned sideways displaying a mood

possible points of insertion could be reviewed, rather than discarded. Similarly, a human reviewer might inspect gloss clusters that are not very similar to the reference WordNet gloss, yet describe a different, valid, relevant property of that concept; alternatively, such a gloss cluster could reveal a different word sense that is absent from WordNet. Human intervention could also refine the linking of an alternative gloss, by selecting the precise sense of the word to which the gloss applies. Moreover, one could explore the idea of creating missing intermediate concepts, rather than preserving the WordNet hierarchy structure. Intermediate concepts occur as categories of extracted instance names, e.g. *software company* and *high-level programming language*. They would fit under existing concepts (*company* and *programming language*), but are missing from WordNet.

5 Evaluation

5.1 Experimental Setting

The experiments are performed on approximately 300 million Web documents in English from a snapshot of the Google index from 2003. Two parallel runs

Table 5. Words with the highest number of descriptive nuggets

Word	*Count*	*Word*	*Count*	*Word*	*Count*	*Word*	*Count*
Trauma	26936	Jesus Christ	6791	Church	4055	John	2954
God	18561	Jesus	5148	New York	3883	Tigers	2934
Christ	11638	Internet	5110	Holy Spirit	3501	commission	2870
United States	9459	Spirit	4646	President	3406	President Bush	2851

use this data. Run_1 aims at extracting alternative glosses for existing WordNet concepts. After discarding words not starting in alphabetic characters, the closed vocabulary contains 114,487 WordNet noun entries. Run_2 collects new instance names and their glosses using an open vocabulary.

5.2 Results

The extraction method identifies one or more descriptive nuggets in the collection for 60% percent of the input words in Run_1. Among the 46,329 words without a descriptive nugget, 3,598 are compound nouns starting in *"genus"* (*genus icterus*, *genus iguana*), 1,136 start with *"family"* (*family adelgidae*, *family Erethizontidae*), and 324 start with *"order"* (*order anoplura*, *order batrachia*). Some of these words are synonyms to words which have extracted nuggets. Thus, *family Erethizontidae* is a synonym of *Erethizontidae*, whose extracted nugget *"New World porcupines"* is very similar to WordNet's *"New World arboreal porcupines"*.

The average number of descriptive nuggets across the words with at least one such nugget is 110, whereas the median is 11. Table 5 shows the words with the highest number of descriptive nuggets. We were baffled to see *Trauma* at such a high rank, so we checked a sample of its nuggets. All are due to adult-content spam, and have the form *"Trauma, which is a blow to* ⟨*SpamPhrase*⟩*, can occur under a variety of circumstances"*. In fact, the majority of the URLs from which they were extracted were available in 2003 but are no longer valid. Three words from Table 5 illustrate another undesirable phenomenon, namely nuggets that refer to particular instances of the same, more general concept (a certain *commission*, one *John*, a certain *president* etc.). This problem is related to the correct identification and disambiguation of names in text [10].

The quality of the descriptive nuggets in Run_1 is further investigated through manual evaluation of a set of 100 randomly selected WordNet nouns with exactly one descriptive nugget extracted from the Web. Nuggets are deemed as *partially* correct if they contain enough information to infer the genus but little else. Other nuggets, which are valid but apply to a different sense of the word than the senses in WordNet, are evaluated as a separate category, similarly to the evaluation in [11]. An example of a word with a different sense is *Leonberg*, whose WordNet definition captures the sense of a large dog breed rather than that of a city. Table 6 shows that most nuggets are correct or partially correct.

A complementary evaluation of Run_1 considers a different set of 100 randomly selected WordNet words, without any restriction on the number of their descriptive nuggets. For each word, the alternative glosses are compared against

Table 6. Accuracy on a random set of 100 words with only one alternative gloss extracted from the Web (C=correct; P=partially correct, i.e., correct but incomplete; O=other sense than WordNet; I=incorrect)

Type	*Pct.*	*Examples*	
		Word	*Alternative Gloss*
C	35%	Colubridae	advanced snakes and the largest snake family
		fire warden	voluntary officer responsible for safe rural fire management within the community
P	35%	Gentiana acaulis	stemless gentian
		Polygalaceae	family of plants
O	3%	Leonberg	small suburb outside Stuttgart
		Rumex	old Latin word for "lance"
I	27%	siege of Vicksburg	city's main claim to fame
		yobbo	backwards spelling of boy

the WordNet reference glosses of that word. The comparison is implemented through the dot-product similarity of the length-normalized term vectors, as discussed in Section 4.3. In the experiment, the alternative glosses whose similarity value is below 0.5 are discarded; the rest are manually checked. Out of the 100 test words, 41 have at least one extracted descriptive nugget above the threshold. The average number of such nuggets per tested word is 7, and the median of 4. The nuggets are classified into one of the judgment classes listed in Table 6. Among the 280 verified nuggets, 195 nuggets are deemed correct, 70 partially correct, 6 incorrect and 9 correct for a different word sense. Nuggets like *"branch of computer sciences"* (for *artificial intelligence*), *"German composer"* (for *Ludwig van Beethoven*) and *"acronym for: Light Amplification"* (for *laser*) are marked as partially correct rather than correct. Note that the decision on whether a nugget is only partially correct is highly subjective. Depending on the application, one could argue that the classification of *"German composer"* as partially correct is pessimistic, since the nugget contains both the genus (*composer*) and what may be relevant differentia (a composer from *Germany*).

To assess the impact of the instance names on WordNet, it is useful to look at the categories to which they are associated according to the data. There are almost 300,000 such raw, distinct lexicalized categories collected in Run_2 that are not isolated occurrences, i.e. are associated with at least 4 instance names. As a primary aggregated result, around 15,000 of these data-driven categories are WordNet nouns. Under ideal conditions (perfect extraction, non-ambiguity, sense matching the one present in WordNet etc.), all instance names associated to these categories potentially belong under existing WordNet concepts.

A secondary result of Run_2 is the acquisition of interesting categories which may be missing from WordNet (see Section 4.3). Intuitively, a category containing a larger number of instances is a better candidate to become a new concept in WordNet. A few of the larger extracted categories with the head *company* are already in WordNet, e.g. *insurance*, *pharmaceutical* and *oil (company)*; others, like

technology, *software*, *blue chip*, *media*, *high-tech*, *Internet*, *telecommunications*, *manufacturing*, *industrial* and *biotechnology (company)* are not in WordNet. Similarly, among the larger categories with the head *programming language*, the category *object-oriented (programming language)* is already in WordNet, whereas none of *web*, *high-level*, *logic*, *functional*, *Internet*, *procedural*, *imperative*, *general purpose* or *structured (programming language)* belong to its noun database.

6 Discussion and Previous Work

One of the early proposals for mining unstructured text with lightweight extraction patterns was considered precisely in the context of discovery of WordNet-style information, i.e. hyponyms [12]. Others successfully applied lightweight patterns to text collections for various applications, including summarization [13], information extraction [14] and question answering [15, 16]. As the availability of large text corpora increased, it became possible to collect large semantic lexicons and resources of instance names, either manually or automatically [17]. However, these experiments tend to organize the instance names into a fixed, coarse-grained set of categories. Comparatively, the new instance names collected in this paper are associated to a large set of data-driven categories. The usefulness of a larger and noisier resource, namely the Web, is indicated by experiments in finding domain-specific definitions [18] and encyclopedic term descriptions [11]. However, we are not aware of work that extends existing WordNet glosses with alternative glosses extracted from the Web, for existing concepts (closed vocabulary) and unspecified instance names (open vocabulary). Alternative glosses are also a possible source for paraphrases, whose acquisition is different from recent approaches focused specifically on collecting paraphrases [19].

7 Conclusion

The largest search engines provide access to more than 8 billion Web documents, as part of an unstructured, unreliable yet powerful knowledge resource that seems to be growing endlessly. Hidden inside documents on different topics, small text nuggets capture some information about the world in a form that is relatively easier to exploit automatically. This paper described a lightweight method to collect text nuggets from the Web and morph them into information that can be linked into existing lexical, hierarchical resources. The insertion of automatically derived glosses and instance names into a resource such as WordNet is certainly challenging. Yet leveraging formally-structured, human-validated resources, on one hand, and data-driven sets of instance names and definitions on the other, opens the path to new applications of the reloaded resources.

References

1. Fellbaum, C., ed.: WordNet: An Electronic Lexical Database and Some of its Applications. MIT Press (1998)

2. Agirre, E., Rigau, G.: Word sense disambiguation using conceptual density. In: Proceedings of the 16th International Conference on Computational Linguistics (COLING-96), Copenhagen, Denmark (1996) 16–22
3. Chai, J., Biermann, A.: The use of word sense disambiguation in an information extraction system. In: Proceedings of the 16th National Conference on Artificial Intelligence (AAAI-99), Menlo Park, California (1999) 850–855
4. Dorr, B., Katsova, M.: Lexical selection for cross-language applications: Combining LCS with WordNet. In: Proceedings of the 3rd Conference of the Association for Machine Translation in the Americas (AMTA-98), Langhorne, Pennsylvania (1998) 438–447
5. Green, S.: Automatically generating hypertext in newspaper articles by computing semantic relatedness. In: Proceedings of the 2nd Conference on Computational Language Learning (CoNLL-98), Sydney, Australia (1998) 101–110
6. Banerjee, S., Pedersen, T.: Extended gloss overlaps as a measure of semantic relatedness. In: Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI-03), Acapulco, Mexico (2003) 805–810
7. Brants, T.: TnT - a statistical part of speech tagger. In: Proceedings of the 6th Conference on Applied Natural Language Processing (ANLP-00), Seattle, Washington (2000) 224–231
8. Voorhees, E.: Implementing agglomerative hierarchic clustering algorithms for use in document retrieval. Information Processing and Management **22** (1986) 465–476
9. Paşca, M.: Acquisition of categorized named entities for Web search. In: Proceedings of the 13th ACM Conference on Information and Knowledge Management (CIKM-04), Washington, D.C. (2004)
10. Wacholder, N., Ravin, Y., Choi, M.: Disambiguation of proper names in text. In: Proceedings of the 5th Conference on Applied Natural Language Processing (ANLP-97), Washington, D.C. (1997) 202–208
11. Fujii, A., Ishikawa, T.: Summarizing encyclopedic term descriptions on the web. In: Proceedings of the 20th International Conference on Computational Linguistics (COLING-04), Geneva, Switzerland (2004) 645–651
12. Hearst, M.: Automatic acquisition of hyponyms from large text corpora. In: Proceedings of the 14th International Conference on Computational Linguistics (COLING-92), Nantes, France (1992) 539–545
13. Schiffman, B., Mani, I., Concepcion, C.: Producing biographical summaries: Combining linguistic knowledge with corpus statistics. In: Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL-01), Toulouse, France (2001) 450–457
14. Phillips, W., Riloff, E.: Exploiting strong syntactic heuristics and co-training to learn semantic lexicons. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-02), Philadelphia, Pennsylvania (2002) 125–132
15. Ravichandran, D., Hovy, E.: Learning surface text patterns for a question answering system. In: Proceedings of the 40th Annual Meeting of the Association of Computational Linguistics (ACL-02), Philadelphia, Pennsylvania (2002)
16. Solorio, T., Pérez, M., Montes, M., Villasenor, L., López, A.: A language independent method for question classification. In: Proceedings of the 20th International Conference on Computational Linguistics (COLING-04), Geneva, Switzerland (2004)

17. Cucerzan, S., Yarowsky, D.: Language independent named entity recognition combining morphological and contextual evidence. In: Proceedings of the 1999 Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-99), College Park, Maryland (1999) 90–99
18. Liu, B., Chin, C., Ng, H.: Mining topic-specific concepts and definitions on the web. In: Proceedings of the 12th International World Wide Web Conference (WWW-03), Budapest, Hungary (2003) 251–260
19. Dolan, W., Quirk, C., Brockett, C.: Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In: Proceedings of the 20th International Conference on Computational Linguistics (COLING-04), Geneva, Switzerland (2004)

Automatic Synonym Acquisition Based on Matching of Definition Sentences in Multiple Dictionaries

Masaki Murata[1], Toshiyuki Kanamaru[2], and Hitoshi Isahara[1]

[1] National Institute of Information and Communications Technology,
3-5 Hikaridai, Seika-cho, Soraku-gun, Kyoto, 619-0289, Japan
{murata, isahara}@nict.go.jp
http://www.nict.go.jp/jt/a132/members/murata/
[2] Kyoto University, Yoshida-nihonmatsu-cho,
Sakyo-ku, Kyoto, 606-8501, Japan
kanamaru@hi.h.kyoto-u.ac.jp

Abstract. Studies on paraphrasing are important with respect to various research topics such as sentence generation, summarization, and question-answering. We consider the automatic extraction of synonyms (which are a kind of paraphrase) through the matching of word definitions from two dictionaries, and describe a new method for extracting paraphrases. Higher precision was obtained than with a conventional frequency-based method. The new method provided a precision rate of 0.764 for the top 500 data pairs and 0.220 for 500 randomly extracted data pairs when only synonyms were considered a correct answer. It provided a precision rate of 0.974 for the top 500 data pairs and 0.722 for 500 randomly extracted data pairs when hypernyms and similar expressions were also considered correct answers. Our method should be useful for other studies on paraphrase extraction.

1 Introduction

Studies on paraphrasing [6, 2] have had important consequences in various domains such as sentence generation, summarization, and question-answering [3, 12]. Likewise, studies on paraphrase extraction are also important. In this paper, we discuss the automatic extraction of synonym expressions which can be considered a kind of paraphrase. We extract synonym expressions by matching definitions of the same word from two dictionaries. In this work, we studied the extraction of synonym expressions in the Japanese language.

For example, we examined the definition sentences for the word *abekobe* meaning "reverse". Two Japanese dictionaries gave the definitions shown in Figure 1 for the word. We expected to extract pairs of expressions having the same meaning when we compared the two definitions, since they both defined the same word and thus had the same meaning. We compared the two definition sentences and obtained the results shown in the figure. From the results, we deter-

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 293–304, 2005.

Definition of "reverse" in Dictionary A:

junjo , *ichi* *nado* *-no* *kankei* *-ga* *sakasama-ni* *irekawat*
(order) (,) (location) (etc.) (of) (relation) nom (upside-down) (change places)
-teiru
(-ing)
(The relationship of the order, location, and so on is changed upside-down.)

Definition of "reverse" in Dictionary B:

junjo , *ichi* , *kankei* *-ga* *hikkuri-kaet* *-teiru*
(order) (,) (location) (,) (relation) nom (be overturned) (-ing)
(The relationship of the order and location is overturned.)

Results of comparing the two definitions

junjo , *ichi*	*nado* *-no* (etc.) (of) , ,	*kankei* *-ga*	*sakasama-ni* (upside-down) *hikkuri-kaet* (be overturned)	*irekawat* (be changed)	*-teiru*

Fig. 1. Example of rule extraction for paraphrasing

mined that *nado-no* "etc." and "," were interchangeable, as well as *sakasama-ni irekawatte* "be changed upside-down" and *hikkuri-kaet* "be overturned". In short, our method for extracting synonym expressions is to extract synonym expressions by matching definition sentences from two dictionaries having the same content.

The advantages of our method can be summarized as follows.

- Although synonym expressions were extracted from text pairs having the same content in previous studies, there have been no studies where definition sentences in multiple dictionaries were considered text pairs having the same content and synonym expressions were extracted from them. This paper is useful in showing that many synonym expressions can be extracted from definition sentences in multiple dictionaries.
- In this paper, we propose a new method, which is useful for extracting synonym expressions. We show, based on our experiments, that this method is more effective than several comparable methods. This method can also be used for other studies on the extraction of synonym expressions.

2 Method of Extracting Synonym Expressions Based on Matching Two Dictionaries

In this study, we extracted synonym expressions by matching definitions of the same word from two dictionaries: the Iwanami Japanese dictionary and the Daijirin Japanese dictionary.

We first aligned definition sentences for the same word that were extracted from the two dictionaries. When a word had more than one definition sentence,

Table 1. Examples of the results from definition sentence matching

Degree of matching	Word	Compared definition sentences
0.69	*appuappu* (gasp for breathe)	*mizu-ni obore-kakete, mogaiteiru sama* (to struggle by reason of drowning) *mizu-ni obore-kakete kurushimu sama* (to suffer by reason of drowning)
0.20	akarasama (frank)	*kyu-na sama* (the state of suddenness) *tsutsumi-kakusanaide, hakkiri arawasu sama* (the state of expressing something plainly without concealing one's feelings)

we assumed one-to-one alignment and aligned the pair of definition sentences having the best degree of matching.

We separated each definition sentence into several words by using the JUMAN Japanese morphological analyzer[4] and arranged each word on each line. We detected matching parts and non-matching parts by using the UNIX diff command [8, 5, 9]. We defined the degree of matching as

$$\text{Degree of matching} = \frac{N_{match} \times 2}{N_{all}}, \tag{1}$$

where N_{match} is the number of characters in the matching part and N_{all} is the total number of characters in the two definition sentences. The degree of matching takes a value from "0" to "1" with the value being larger when the matching part is larger.

When we performed the above alignment and matching of definition sentences, we obtained 57,643 definition sentence pairs. Some examples of the results from definition sentence matching are shown in Table 1. In the table, parts that differ are underlined and these were extracted as candidate synonym pairs.

We found some good synonym pairs such as the pair of *mogaiteiru* "struggle" and *kurushimu* "suffer", but also found some pairs that were not synonyms such as the pair of *kyuu-na* "suddenness" and *tsutsumi-kakusanaide, hakkiri arawasu* "expressing something plainly without concealing one's feelings". These results were not particularly accurate and could not be used as synonyms as they were.

Therefore, we next extracted better synonym pairs from candidate synonyms. We based this extraction on the following characteristics.

- Differing parts that are surrounded by lower-frequency words are better synonym pairs.
- Differing parts that occur more frequently are better synonym pairs.

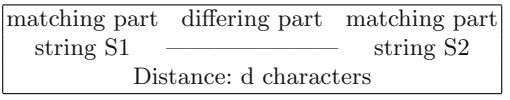


Fig. 2. Occurrence of differences

First, we considered the first characteristic, "differing parts that are surrounded by lower-frequency words are better synonym pairs." We assumed that a differing part would be surrounded by strings, $S1$ and $S2$, which were matching parts and the distance between $S1$ and $S2$ would be d characters[3] as shown in Figure 2. The probability, $P(S1)$ or $P(S2)$, of the occurrence of $S1$ or $S2$, respectively, in an inner region consisting of no more than d characters from $S2$ or $S1$ is approximately expressed as

$$P(S1) \simeq (d+1) \times \frac{Freq(S1)}{N} \tag{2}$$

$$P(S2) \simeq (d+1) \times \frac{Freq(S2)}{N}, \tag{3}$$

where $Freq(S1)$ and $Freq(S2)$ are the respective numbers of occurrences for strings $S1$ and $S2$, and N is the total number of characters in the database. If we assume the probability $P(dfp, S1, S2)$ that the differing part (dfp) is good equals the probability that strings $S1$ and $S2$ do not appear in the situation shown in Figure 2, $P(dfp, S1, S2)$ can be expressed as

$$P(dfp, S1, S2) \simeq (1 - P(S1))(1 - P(S2)), \tag{4}$$

where we assume that $S1$ and $S2$ are independent of each other.

Next, we considered the second characteristic, "differing parts that occur more frequently are better synonym pairs." We have only to combine the probabilities in multiple situations. We assumed that when at least one of the multiple situations was correct, we would extract the differing part as a correct one. Since the differing part being correct is the complement of the case where all the situations for the differing part are incorrect, the probability $P(dfp)$ that the differing part (dfp) is correct is expressed as

$$P(dfp) \simeq 1 - \prod_{S1,S2} (1 - P(dfp, S1, S2)), \tag{5}$$

where we assume that each situation for the differing parts is independent of one another.

The extraction of synonym pairs is done by sorting candidate differences according to the value of the above equation and extracting the one having a higher value. In this paper, we refer to this method as *our method*.

[3] In this study, we used a longer length of characters in the differences as d.

3 Comparison Method

In this section, we describe the comparison method used in the experiments to evaluate the effectiveness of our method.

- Frequency method
  Extracted differing parts are sorted by their respective frequencies. A difference having a higher frequency is judged to be a more plausible synonym pair.
- Katoh's method
  It is based on Katoh et al's study [3]. Differences that satisfy the following conditional equation are first extracted and these are then judged to be plausible synonym pairs based on the frequency method.

$$\frac{N_{S1} + N_{S2}}{d} > 1, \tag{6}$$

  where N_{S1} and N_{S2} are the numbers of characters of $S1$ and $S2$, respectively. When the summation of the lengths of $S1$ and $S2$ exceeds the length of the differing part, we judge that the difference is not an accidentally extracted one and represents a plausible synonym pair.
- Combined method
  This is a combination of Katoh's method and our method. Differences that satisfy Equation 6 are extracted and then are judged based on Equation 5 as to whether they are plausible synonym pairs.

4 Experiments

We used our method to obtain synonym pairs. Examples of extracted differences are shown in Table 2, and examples of good extracted synonym pairs are shown

Table 2. Examples of extracting differences

-log(1-P)	Frequency	Preceding contexts	Differing parts		Succeeding contexts
4975	786	*shinpai ga naku* (without trouble)		, (,)	*nonbiri shiteiru* (peaceful)
2266	301	*dankai* (grade)	*ga* (is)	*no* (is)	*hikui koto* (low)
1528	234	*kinzoku* (metal)		*no* (-lic)	*gen* (string)
208.8	60	*inkoku ni* (knife)	*tsukau* (used)	*mochiiru* (utilized)	*kogatana* (for linocut)
162.6	22	*tadashii kaitou* (true answer)	*matawa* (or)	*ya* (or)	*kaishaku* (interpretation)
105.8	22	*seizou* (method)	*suru* (performing)	*no* (of)	*houhou* (production)

Table 3. Examples of extracted synonym pairs

tsutsu (while)	*nagara* (while)
honyuu doubutsu (a mammal)	*honyuu rui* (the mammals)
tyuuto (halfway)	*totyuu* (on the way)
gyou (job, work)	*shoku* (job, work)
naru (become)	*kawaru* (change)
hedatari (gap)	*sa* (difference)
tsuku (get to)	*tyoutyaku suru* (arrive)
de tsukutta (made by)	*no* (of)
kachiku (a domestic animal)	*gyuuba nado* (horses and cows etc.)
ga umai (be good at)	*ni takumina* (be skillful at)
daiji ni (important)	*taisetsu ni* (precious)
tsutaeru (tell, report)	*dentatsu suru* (tell, report)
tameni (for)	*mokuteki de* (for the purpose of)
hazurete iru (be out of)	*awanai* (do not match)
ku (eat)	*taberu* (eat, have)
genshou suru (decrease)	*sukunaku naru* (become fewer)

in Table 3. We were able to extract many word-level synonym pairs and phrase-level synonym pairs such as *ga umai* "be good at" and *ni takumina* "be skillful at". We could also extract some functional word pairs such as *tsutsu* and *nagara*, which have the same meaning of "while".

Next, we compared our method to comparison methods. These results are shown in Tables 4 and 6. Here, we judged as correct the extracted pairs (differences) that have a context where they are judged to be synonym pairs. Table 4 shows the precision for the top X pairs for each method. Table 6 shows the precision and the number of extracted synonym pairs.

"Precision" in Table 6 means the precision for 500 randomly extracted pairs. The "Number of extracted pairs" is the total number of extracted pairs. The "Expected number of extracted synonym pairs" was obtained by multiplying

Table 4. Precision (Top 500 pairs)

	Our method	Frequency method	Katoh's method	Combined method
Top 50	0.900 (45/ 50)	0.580 (29/ 50)	0.680 (34/ 50)	0.900 (45/ 50)
Top 100	0.870 (87/100)	0.560 (56/100)	0.620 (62/100)	0.870 (87/100)
Top 200	0.820 (164/200)	0.580 (116/200)	0.645 (129/200)	0.825 (165/200)
Top 300	0.790 (237/300)	0.583 (175/300)	0.657 (197/300)	0.780 (234/300)
Top 400	0.767 (307/400)	0.590 (236/400)	0.642 (257/400)	0.767 (307/400)
Top 500	0.764 (382/500)	0.588 (294/500)	0.616 (308/500)	0.738 (369/500)

Table 5. Precision (Top 500 pairs excluding cases where a difference on one side is a null expression)

	Our method	Frequency method	Katoh's method	Combined method
Top 50	0.960 (48/ 50)	0.880 (44/ 50)	0.920 (46/ 50)	0.980 (49/ 50)
Top 100	0.960 (96/100)	0.900 (90/100)	0.930 (93/100)	0.960 (96/100)
Top 200	0.950 (190/200)	0.910 (182/200)	0.905 (181/200)	0.950 (190/200)
Top 300	0.933 (280/300)	0.903 (271/300)	0.907 (272/300)	0.927 (278/300)
Top 400	0.917 (367/400)	0.907 (363/400)	0.895 (358/400)	0.907 (363/400)
Top 500	0.904 (452/500)	0.910 (455/500)	0.878 (439/500)	0.876 (438/500)

Table 6. Precision and number of extracted synonyms

	Our method	Katoh's method
Precision	0.220 (110/500)	0.400 (200/500)
Number of extracted pairs	67851	17104
Expected number of extracted synonym pairs	14927	6841

Table 7. Occurrence rates for several relationships

	Synonym	Hypernym	Similar expression	No relation
random	0.220 (110/500)	0.454 (227/500)	0.048 (24/500)	0.278 (139/500)
Top 50	0.900 (45/ 50)	0.100 (5/ 50)	0.000 (0/ 50)	0.000 (0/ 50)
Top 100	0.870 (87/100)	0.120 (12/100)	0.000 (0/100)	0.010 (1/100)
Top 200	0.820 (164/200)	0.165 (33/200)	0.000 (0/200)	0.015 (3/200)
Top 300	0.790 (237/300)	0.190 (57/300)	0.000 (0/300)	0.020 (6/300)
Top 400	0.767 (307/400)	0.212 (85/400)	0.003 (1/400)	0.018 (7/400)
Top 500	0.764 (382/500)	0.206 (103/500)	0.004 (2/500)	0.026 (13/500)

"Precision" and "Number of extracted pairs", and is the expected number of synonym pairs that each method will be able to extract. Since Katoh's method uses elimination by Equation 6, the total number of extracted pairs in the method is smaller than in our method. Since the frequency method does not use elimination by Equation 6, the total number of extracted pairs is the same as in our method.

Table 8. Occurrence rates for several relationships (excluding cases where a difference on one side is a null expression)

	Synonym	Hypernym	Similar expression	No relation
random	0.313 (106/339)	0.274 (93/339)	0.071 (24/339)	0.342 (116/339)
Top 50	0.960 (48/ 50)	0.020 (1/ 50)	0.000 (0/ 50)	0.020 (1/ 50)
Top 100	0.960 (96/100)	0.010 (1/100)	0.000 (0/100)	0.030 (3/100)
Top 200	0.950 (190/200)	0.030 (6/200)	0.000 (0/200)	0.020 (4/200)
Top 300	0.933 (280/300)	0.033 (10/300)	0.007 (2/300)	0.027 (8/300)
Top 309	0.932 (288/309)	0.032 (10/309)	0.006 (2/309)	0.029 (9/309)

As shown in Table 4, the precision of our method and that of the combined method using Equation 5 were higher than with the other methods. We thus found that our proposed Equation 5 was effective.

Comparing the frequency method and Katoh's method, we found that Katoh's method provided higher precision. Thus, the deletion of candidates through Equation 6 in Katoh's method was effective.

Table 5 shows the results when we excluded cases where a difference on one side was a null expression. When a difference on one side is a null expression, the differences are not likely to be synonyms. Therefore, the results in Table 5 were better than those in Table 4.

In Table 6, the precision when 500 pairs were randomly extracted was 0.22 with our method and 0.40 with Katoh's method. This was because Katoh's method deleted unreliable candidate pairs through Equation 6. However, fewer synonyms were extracted with Katoh's method than with our method. Katoh's method has a shortcoming in that it fails to extract many synonyms. The estimated number of extracted synonyms with our method was 15000, and the precisions were 0.764 for the top 500 data (Table 4) and 0.220 for the 500 data that were extracted randomly (Table 6).

We next performed a more detailed examination using the results extracted by our method, which provided good results in the above experiments. The results are shown in Table 7. In the examinations, we counted the number of pairs having contexts that enabled them to be judged as either synonym pairs, hypernym pairs, similar to each other, or having no relationship. In the table, "random" indicates the results for 500 randomly extracted pairs and "Top X" indicates the results for the top X pairs. Table 8 shows the results from Table 7 when cases where a difference on one side was a null expression were excluded. Examples of expressions that were judged to be hypernyms or similar expressions are shown in Tables 9 and 10, respectively. This examination was done because most of the errors that were not judged to be synonyms were hypernyms or similar expressions.

When we considered the pairs that were hypernyms or similar expressions, as well as synonyms, to be correct, the precision rates became very high. For all the data, the precision rose to 0.722 ($= 1 - 0.278$). When we excluded cases where a difference on one side was a null expression, the precision became 0.658

Table 9. Examples of pairs having a hypernym relationship (a left expression meaningfully includes a right expression)

Preceding contexts	Differing parts		Succeeding contexts
me ya kuchi (open eyes and mouth)	*nado* (etc.)		*wo kyuuni hiraku sama* (suddenly)
nihiki	*ijou* (more than)		*no kaiko ga* (two silkworms)
houkou wo shimesu		*dai* (broad)	*houshin* (policy indicating directions)
takakkei no (polygon's)	*subete no* (all)	*kaku* (each)	*tyouten ga* (the vertex/vertexes)
hana ga (flowers)	*zenbu* (all)	*issei-ni* (all and simultaneously)	*saku* (be out)
	oya (parent)	*chichi* (father)	*nado* (etc.)
	hoso nagai (fine long)	*himojou no nagai* (corded long)	*sita de* (tongue)

Table 10. Examples of pairs having a similar meaning

Preceding contexts	Differing parts		Succeeding contexts
23 *do* (23 degrees)	27 (27)	26 (26)	*hun no isen* (minutes latitude)
	hori (a moat)	*ike* (a pond)	*ya* (or)
kaijou no (at sea)	*kokubou* (national defense)	*bouei, kougeki* (defense, attack)	*wo*
	ookina (big)	*hijouna* (extraordinary)	*rieki* (profit)

$(= 1 - 0.342)$. When we considered the pairs that were hypernyms or similar expressions to be correct also, the precision decreased if we excluded cases where a difference on one side was a null expression. We explain this as follows. Consider Table 9, which shows expressions having a hypernym relationship. When a difference on one side is a null expression and the difference on the other side is an expression having an effect of expanding the region of a meaning, such as *nado* "etc." or *ijou* "more than", the difference pair relationship is that the difference which is a null expression is a hyponym of the other difference. When a difference on one side is a null expression and the difference on the other side is an expression restricting or decreasing the region of a meaning, the difference pair relationship is that the difference which is a null expression is a hypernym of the other difference. Therefore, when a difference on one side is a null expression, the pair of differences is likely to have a hypernym relation and the

precision when including cases where a difference on one side is a null expression will increase.

When we considered pairs that were hypernyms or similar expressions to be correct also, the top X precision became extremely high. For example, the precision for the Top 500 of all data was 0.974 ($= 1 - 0.026$).

We can summarize our experimental results as follows.

- Our method using Equation 5 provided high precision for the Top X data. We also found that this method provided higher precision than the frequency method which is normally used. We can therefore effectively extract synonyms with high precision using our method.
- The number of extracted synonyms with our method was larger than that with Katoh's method. When we would like to extract more synonyms, we should not delete candidate synonym pairs through Katoh's Equation 6.
- In terms of synonym extraction, the precision was higher when we excluded cases where a difference on one side was a null expression than when we include these cases. However, in terms of also extracting hypernyms and similar expressions, the precision when we included cases where a difference on one side was a null expression was higher than when we excluded these cases.
- Our method, which extracts many synonyms, provides high precision for top X data. We obtained a precision rate of 0.764 for the top 500 data pairs and 0.220 for 500 randomly extracted data pairs when only synonyms were considered a correct answer. We obtained a precision rate of 0.974 for the top 500 data pairs and 0.722 for 500 randomly extracted data pairs when hypernyms and similar expressions were also considered correct answers.

5 Related Studies

Our approach was to extract synonyms by matching a pair of text sections sharing the same meaning. Other studies on automatic extraction using this approach include the following.

- Use of multiple sentences translated from the same original sentence
  Since sentences translated from the same original sentence should have the same meaning, synonyms can be extracted by matching those sentences. Barzilay and McKeown obtained synonyms by using this method [1]. Shimohata also obtained synonyms by using this method and then used the extracted synonyms to improve the performance of machine translation [10].
- Use of document pairs having the same content
  Documents from multiple newspaper publishing companies are gathered and pairs of documents having the same content are extracted. By matching these pairs of documents, synonyms are extracted. Shinyama et al. obtained synonyms using this method [11]. They extracted document pairs having the same content by using proper nouns appearing in the documents.
- Use of pairs consisting spoken data and corresponding written data

Murata et al. used presentations at academic conferences as spoken data and the corresponding papers as written data. They obtained synonyms and rewriting rules for paraphrasing between spoken language and written language by matching spoken data and the corresponding written data. They also performed paraphrasing between the spoken and written language [7].

- Use of parts having the same content in a document
  Murata et al. obtained rewriting rules and synonyms by matching a patent claim and its embodiment [9]. A patent claim and its embodiment share the same meaning, so such a pair can be used to obtain synonyms.
- Use of pairs consisting of an original sentence and a summarized version
  An original sentence and its summarized version share the same meaning, so we can extract synonyms by matching them. Katoh et al. used this method and obtained rewriting rules and synonyms for summarization [3].

As described above, many studies have been done on extracting synonyms by matching sentences or texts having the same meaning. Our method of using Equation 5 should be useful for such studies because it is convenient and effective.

6 Conclusion

Studies on both paraphrasing and paraphrase extraction are important in various research fields such as sentence generation, summarization, and question-answering. In our current work, we have studied methods of automatic paraphrase extraction based on matching definitions of the same word in two dictionaries.

Through our experiments, we confirmed that our proposed method using Equation 5 provided higher precision for the top pairs than other existing methods. This method should therefore be useful and effective, for application in other studies on automatic synonym extraction.

In our experiments, the estimated number of synonyms extracted with our method was 15,000, a much larger number than were extracted with Katoh's method. Although there have been studies on the extraction of synonym expressions from text pairs having the same content, no studies have been reported where definition sentences in multiple dictionaries were used as text pairs having the same content and synonym expressions were extracted from them. This paper has explained how many synonym expressions can be extracted from such definition sentences.

For synonym extraction, the precision was higher when we excluded cases where a difference on one side was a null expression. However, when we considered pairs that were hypernyms and similar expressions, as well as synonym pairs, to be correct, the precision was higher if we did not exclude cases where a difference on one side was a null expression.

With our method, we obtained precision rates of 0.764 for the top 500 data and 0.220 for 500 randomly extracted data when we considered only synonyms

to be a correct answer. We obtained corresponding precision rates of 0.974 and 0.722, though, when we considered hypernyms and similar expressions to also be correct answers.

References

1. Regina Barzilay and Kathleen R. McKeown. Extracting paraphrases from a parallel corpus. In *39th Annual Meeting of the Association of the Computational Linguistics*, pages 50–57, 2001.
2. IWPT committe. Nlprs'2001 workshop on automatic paraphrasing: Theories and applications. 2001.
3. Naoto Katoh and Noriyoshi Uratani. A new approach to acquiring linguistic knowledge for locally summarizing Japanese news sentences. *Journal of Natural Language Processing*, 6(7), 1999. (in Japanese).
4. Sadao Kurohashi and Makoto Nagao. *Japanese Morphological Analysis System JUMAN version 3.6*. Department of Informatics, Kyoto University, 1998. (in Japanese).
5. Masaki Murata. NLP using DIFF — use of convenient tool for detecting differences, MDIFF —. *Journal of Natural Language Processing*, 9(2), 2002. (in Japanese).
6. Masaki Murata and Hitoshi Isahara. Universal model for paraphrasing — using transformation based on a defined criteria —. In *NLPRS'2001 Workshop on Automatic Paraphrasing: Theories and Applications*, 2001.
7. Masaki Murata and Hitoshi Isahara. Automatic extraction of differences between spoken and written languages, and automatic translation from the written to the spoken language. In *LERC 2002*, 2002.
8. Masaki Murata and Hitoshi Isahara. Using the diff command for natural language processing. 2002. http://arxiv.org/abs/cs.CL/ 0208020.
9. Masaki Murata and Hitoshi Isahara. Using the diff command in patent documents. *Proceedings of the Third NTCIR Workshop (PATENT)*, 2002.
10. Mitsuo Shimohata. Acquiring paraphrases from corpora and its application to machine translation. *Doctor's Thesis*, 2004. NAIST-IS-DD0261014.
11. Yusuke Shinyama and Satoshi Sekine. Paraphrase acquisition for information extraction. In *The Second International Workshop on Paraphrasing: Paraphrase Acquisition and Applications*, 2003.
12. Tetsuro Takahashi, Kozo Nawata, Kentaro Inui, and Yuji Matsumoto. Effect of structural matching and paraphrasing in question answering. *IEICE Transactions on Information and Systems*, E86–D(9):1677–1685, 2003.

Enriching WordNet with Derivational Subnets

Karel Pala and Radek Sedláček

Faculty of Informatics, Masaryk University,
Botanicka 68a, 60200 Brno, Czech Republic
{pala, rsedlac}@fi.muni.cz

Abstract. In this paper, we deal with the derivational (word formation) relations as they are handled by the Czech morphological module Ajka. First, we show that they represent empirically well-based semantic relations forming small semantic networks, and then we solve the problem how to integrate them into lexical database such as (Czech) WordNet. In this respect we examine the relation between the derivational relations and semantic roles (deep cases) defined as Internal Language Relations in EuroWordNet. An attempt is made to match up the inventory of the semantic roles in EWN with the derivational (semantic) relations. We also use a tool called SAFT that can process a raw (corpus) text in such a way that it uses module Ajka to find links relating the WordNet senses to the noun and verbal lemmata obtained from the raw (corpus) text. This technique allows us to enrich Czech WordNet with the derivational subnets and represent them in a XML format. The result is a new kind of the semantic network, which consists of two layers, upper and lower. The result is a more **powerful** and efficient resource for applications like tools for WSD, web searching or information extraction.

1 Derivational Relations as Semantic Networks

For computer processing highly inflected language like Czech it is necessary to have a high quality morphological module that can perform lemmatization of a given word form and yield all the grammatical categories that are carried by the word form. Such a tool for Czech is a morphological analyzer and generator called Ajka developed in NLP Lab at FI MU (Sedláček, 2001, 2004). Other tools exist for Czech as well (Hajič, 2004) but we prefer Ajka for its properties—it is able to deal with derivational relations automatically.

Ajka is based on the system of the (approx.) 2000 inflectional paradigms, contains about 350 000 Czech stems and is able to generate about 5,7 million Czech word forms. Its coverage/recall for Czech is about 96 % (tested on the corpus All containing 640 mil. Czech word forms and implemented in the NLP Lab at FI MU). It is based on the 'paradigmatic' model of morphology and though it has been primarily devised for Czech its engine can work also with other synthetic languages (such as Slavonic, e.g. Slovak, Serbian, Russian) as well as with analytic ones – there are versions for English, German, French, Dutch, Spanish, Italian.

As we said the morphological module Ajka captures not only the inflectional relations but also the derivation ones (word formation relations). For Czech we know

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 305 – 311, 2005.

that approximately 67% of the word stock is obtained by means of the word formation and that the derivation of the new words is highly regular and can be described by the formal rules. The word formation rules have been recently integrated into Ajka (Sedláček, 2004) so that it is now able to generate and recognize word **derivational networks** (subnets) automatically. An example of such derivational nest is given in Fig. 1 (both for English and Czech, actual output from Ajka looks slightly different):

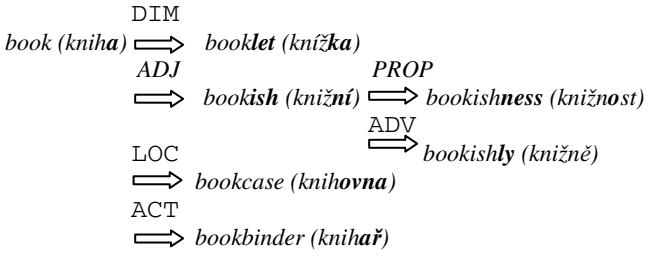


Fig. 1. Word derivation graph (subnet) for the root *book*

One can observe that the semantic relations between the above items are empirically well founded since they can be deduced from the given root or stem and the respective suffixes indicated in bold face (or prefixes as well). They represent a part of the language knowledge that speakers have. The derivational semantic network can be formally represented as a graph with one or more roots (see Figure 1). Its nodes represent the individual lemmata or word forms and edges can be labelled by the corresponding semantic relations, which follow from the relation between roots and derivational suffixes. The following example with *home* shows that derivational relations in English are reasonably rich, and in fact, follow almost the same principles as in Czech which, however, is more regular and productive (the data comes from BNC):

As a NOUN	In Real Estate domain:	As an ADJ	As an ADV	As a VERB
Home *homework* *homeland* *homecoming* *homeboy* *homestead* *homecare* *homebase* *homebrew* *(the) homeless* *homelessness* *homeliness*	*homeowner* *homebuyer* *homeloan* *homecover*	*homesick* *home-made* *home-based* *homegrown* *homeless* *homely*	*home* (e.g. *go home*, *I am at home*) *homeward* *homewards*	*home* (e.g. *If you want to home to a beacon…*) phrasal verb: *to home in on*

2 Are Derivational Relations Semantic?

In Czech linguistic works related to the word formation (Dokulil, 1962) the derivational relations are treated as a special group of the relations that express "semantic relations sui generis", i.e. they are understood as different from other „standard" semantic relations based on the sentence constituents. In our opinion, this differentiation can be empirically justified since the derivational relations are in fact morphological relations whereas "standard" semantic roles are viewed as the relations resting on sentence constituents.

However, if we have a look at the collection of the Czech derivational suffixes (67 in Ajka), we can distinguish various types of the derivational relations expressing the particular semantic relations as e.g. agentive (*to teach -.teacher*) or expressing property (*home – homelessness*). According to our intuition they can be seen as similar to other semantic relations usually characterized as "semantic roles" or "semantic cases" but they are obtained in a different way, i.e. derivationally (morphologically). While the semantic roles are typically associated with verbs and their arguments, the derivational relations hold between the four open parts of speech (in many languages), i.e. we have derivational pairs like *noun – adjective* or *adjective – noun*, *noun – verb* or *verb – noun*, *noun – noun*, *adjective – adverb*. Thus, formally they go across the individual parts of speech being XPOS relations. Though it requires more complete examination of the derivational data, the intuition is that basically there should not be an essential difference between "derivational" semantic relations and "sentence" semantic relations resting on predicate-argument structure of verbs. From the cognitive point of view the Occams Razor principle supports this intuition as well.

Below we are proposing a labelling (tagging) that can be used for the individual derivational relations and capture their semantic nature. It is tentative and it should be further refined when the empirical data becomes more complete. The labels in the list below are experimentally sub-classified (by numbers showing more detailed semantic differences) but it is not the only possible solution.

The tentative list of the derivational relations below is based on the rich Czech data but if the corresponding semantic relations can be considered rather universal (we think so), then we are convinced that they can be applied also in other languages such as English or German (not speaking about Slavonic ones).

- AG0, agent performing an action: *teach**er*** (*učit**el***) from *to teach* (*učit*),
- AG1, agent producing an object: *glass**maker*** (*skl**ář***) from *glass* (*sklo*),
- PROPB0, ownership of an object: *farmer* (*statk**ář***) from *farm* (*statek*),
- PROPB1, pertaining to an object: *villager* (*vesnič**an*** from *village* (*vesnice*),
- INS, means or instrument by which an action is performed: *excavat**or*** (*rypa**dlo***) from *to excavate* (*rýpat*),
- PAT, patient of an action: *prison**er*** (*vězeň*) from *imprison* (uvěznit),
- RES, result of an action: *printed **copy*** (***výt**isk*) from *to print* (*tisknout*),
- PROP (XPOS), property expressed by noun: *quick**ness*** (*rychl**ost***) from adjective *quick* (*rychlý*),
- PROP1, property, *dilig**ent*** (*pil**ný***) from *diligence* (*píle*),

- ACT (XPOS), action verb – noun: *the fall, falling* (*pád, padá**ní***) from *to fall* (*padat*),
- ACTPROP, property changing to an action: ***become*** *green* (*zelen**at***) from *green* (*zelený*),
- PROPMANN (XPOS), property of the action, i..e. manner: *quick**ly*** (*rychl**e***) from *quick* (*rychlý*),
- PROPDIM, diminutive relation: *book**let*** (kníž**ečka**) from *book* (*kniha*),
- PROPAUG, augmentative: ***big*** *oak* (*dub**isko***) from *oak* (*dub*),
- PROPGEN, shift of gender: ***female*** *teacher* (*učitel**ka***) from *teacher* (*učitel*), PROPYOUNG, young animal: *lion* ***cub*** (*lví**če***) from *lion* (*lev*),
- POSS, possessive, *father**'s*** (*otc**ův***) from *father* (*otec),*
- LOC, location: *battle**field*** (*boj**iště*** from *battle* (*boj*).

3 Adding Semantic (Derivational) Subnets into WordNet

As it follows from the above, we list above 17 derivational (semantic) relations that are morphologically well justified by the respective suffixes or morphemes (in English). The complete list will be a bit larger and the labelling is still tentative but the main point is that the indicated relations have a firm empirical (and formal) base. To prove the basis of the abovementioned intuition concerning the unity of the semantic relations we find it is useful to compare them with an inventory of semantic roles, particularly with the semantic roles that have been defined within the set of the Internal Language Relations introduced in EuroWordNet (Vossen, 1999). We find the following 15 roles there:

- ROLE_AGENT – INVOLVED_AGENT
- ROLE_PATIENT – INVOLVED_PATIENT
- ROLE_INSTRUMENT – INVOLVED_INSTRUMENT
- ROLE_LOCATION – INVOLVED_LOCATION
- ROLE_SOURCE_DIRECTION
- ROLE_TARGET_DIRECTION
- STATE_OF – BE_IN_STATE
- CAUSES – IS_CAUSED_BY
- HAS_SUBEVENT – IS_SUBEVENT_OF
- XPOS_NEAR_SYNONYM
- XPOS_NEAR_ANTONYM
- ROLE_RESULT – INVOLVED_RESULT
- IS_MANNER_FOR – IN_MANNER
- DERIVES – DERIVED FROM
- DERIVATIVE (defined in PWN v.2, not in EWN)

Obviously, roles like AGENT, PAT, INSTR, RES, LOC, MANN can be found in both lists but still, there is a question **how similar** they are. We have to be aware of the fact that ILRs are not always associated with the synsets while the derivational relations are always associated with the literals representing the individual items

within the synsets (being XPOS relations). In this way with derivational relations we obtain denser network containing not more relations but between more lexical items.

Some ILRs, e.g. DIRECTION, CAUSES, SUBEVENT, do not occur according to our knowledge (at least in Czech) as derivational so they can be kept and used in the same way as in EWN. The role SUBEVENT can be exploited to capture the aspect relations like Perfective – Imperfective – Iterative, which in Czech and other Slavonic languages are not treated as derivational but morphological, aspect is an obligatory grammatical category that has to be expressed by each Czech (Slavonic) verb. E.g. *přečíst* (*to read to the end, read through*) can be considered as a subevent of *číst* (*read*), but the category of the aspect is not so broad in Czech, so this is rather a tentative solution. In Czech WordNet we record aspect pairs (Perfective – Imperfective) associated with the individual verbs. The iterative verbs are obtained directly from Ajka through the respective derivational relation.

The special case is the relation DERIVED which was introduced into EWN to capture derivational relations existing in some EWN languages, however, according to our knowledge it was not elaborated in the way we do it here.

In fact, the role DERIVED was designed to cover any derivational relation that can occur between two synsets or literals, and in this respect, it is too general. However, it should be noted that in Princeton WordNet v.2 (PWN2) there is a relation DERIVATIVE which covers derivational relations between nouns and verbs (*teach – teacher*), adjectives and adverbs (*quick – quickly*) but not relations between nouns and adjectives like *stupidity – stupid*). Thanks to multilingual WordNets as in EuroWordNet or Balkanet the relation DERIVATIVE as it exists in PWN2 can be translated into other languages that are linked to English via Interlingual Index (ILI). But obviously, it can also work the other way around, i.e. if e.g. rich Czech derivational relations (together with their semantic labels) are integrated into Czech WordNet it is possible to exploit ILIs in another direction, i.e. from Czech to English and "derivational" semantic relation can be reflected in English as well.

Originally, the ILRs have been employed in the process of connecting hyperonyms with their respective hyponyms and holonyms with their meronyms, however the XML representation of the ILRs fundamentally allows us to capture any type of general relations.

The important result is: as we have indicated above – thanks to module Ajka we are able to work with the derivational relations automatically. Therefore, we can introduce them into Czech WordNet and exploit them in various ways there automatically as well. The derivational relations also can help considerably in a more reliable discrimination of the individual senses, which are sometimes too fine-grained (especially in PWN2).

4 Morphological Interface for Czech WordNet – Saft

WordNet synset literals are naturally stored as lemmata. That is why we cannot use plain text as an input stream for any kind of analysis. It is obvious that if we have large amount of data to be semantically tagged, we cannot use WordNet as it stands (at least in Czech).

The Ajka module mentioned above can be fully exploited as a bottom module for other applications. We want to exploit its ability to find a lemma for each word form in a text and associate it with its derivational subnet as we demonstrated above. The only problem is that Ajka as such is limited in one relevant respect: it can only process the words one by one as separate units.

For this purpose we have implemented a tool that employs Ajka and handles the multi-word expressions (collocations). It is named Mwe (Svoboda 2003) and uses Ajka as its bottom module. It recognizes multi-word expressions (MWE) that occur in Czech WordNet and many others. They are: collocations (*cumulative shot, diamond dust, dipterous insect*), proper, geographical and other names (*Albert Einstein, Lisabon, Kuril Islands, Matrix Reloaded*) and abbreviations (*NATO, colloq., A.D.*). Each recognized collocation is associated with its unique lemma. We can easily see from corpus texts or magazine articles that there is approximately one MWE in every other sentence.

In WordNet (both English and Czech) we find about 40 % collocations so it is obvious that if we want to semantically tag a sentence where the collocation *'cumulative shot*' occurs, we must recognize it as a whole. If simple analysis uncovers that '*cumulative*' is a lemma and '*shot*' is a lemma, and then we would manually look up for these two words, we will get plenty of false hits. It is likely that the desired synset will be among them but still the other synsets are unwanted when we process the data automatically. Collocations (or their lemmata) tend to display only one semantic unit, so if we recognize them as a whole, we practically recognize them unambiguously.

The implementation of the idea discussed above, i.e. interconnection of the functionalities of the Ajka and MWE tool with Czech WordNet can be found in the module called **Saft** (Čapek, 2004). It takes plain text as its input, parses it and recognizes the collocations, then looks them up in WordNet. Single-word expressions are processed by Ajka directly. Saft is now able to analyze and lemmatize any text in Czech and to associate relevant lemmata with the appropriate literals in Czech WordNet.

It should be noted that no attempt is made to disambiguate senses that may be associated with the individual literals. It is also possible to generate identification numbers of the synsets containing these literals and import them into VisDic (Smrž, Horák, 2004), which is the tool for storing, managing and editing WordNet lexical databases.

5 Conclusions

In the presented paper we offer the description of the selected derivational relations in Czech and their implementation in morphological analyzer Ajka, which is able to generate the derivational semantic networks). Then we show what semantic relations they capture and compare them briefly with the ILRs as they are defined in EuroWordNet. The comparison leads us to the conclusion that they are in many respects similar if not the same: ILRs are implicitly associated with the sentence constituents whereas the derivational relations (DR) rest on the morphological relations.

The derivational relations are labelled semantically and they are presented in the list containing 17 semantic relations. The list of ILRs from EuroWordNet contains 15 relations.

Then we show how the DR can be integrated into Czech WordNet. A tool called Saft is mentioned that makes it possible to process a free (corpus) text and search both for the individual synsets and literals linked with DRs. This does not mean that we do semantic disambiguation; the described processing is only a necessary first step that has to be done in any case.

As a result we obtain a Czech WordNet in which we have two levels of the semantic relations—the first one are ILRs and the second one are DRs. They are more subtle and detailed than ILRs, thus they yield more powerful resource for Information Extraction and Web searching. In this sense DRs represent a subnet that relates the individual literals above the standard synonymy/antonymy and hypero/hyponymy relations.

References

1. Baker, C. F., Fillmore, Ch. J., Lowe, J. B.: *FrameNet Project*, in: Proceedings of the Coling-ACL, Montreal, Canada (1998).
2. Dokulil, M. (1962) Tvoření slov v češtině 1 (Word-Formation in Czech 1). Nakladatelství ČSAV. Prague
3. Fellbaum, Ch.: (Ed.) WordNet: An Electronic Lexical Database, MIT Press (1998).
4. Horák, A., Smrž, P. New Features of WordNet editor VisDic, in: RomanianJournal of Information Science and Technology, Vol. 7, No 1-2, 2004, pp.201-214.
5. Klímová, J., Pala, K. (2000) Application of WordNet ILR in Czech Word-Formation. In Proceedings of LREC 2000. p. 987-992. ELRA.
6. Levin, B.: English Verb Classes and Alternations: a preliminary investigation, The University of Chicago Press, (1993).
7. Lopatková, M., Žabokrtský, Z.: Valency Dictionary of Czech Verbs, ELRA (2002).
8. Mráková-Žáčková, E.: Partial Parser DIS/VADIS (for Czech), Ph. D. Dissertation, Faculty of Informatics, Masaryk University, Brno (2002).
9. Pala, K., Rychlý, P., Smrž, P. (1997) DESAM—Annotated Corpus or Czech. In Proceedings of SOFSEM 97. Heidelberg: Springer Verlag. pp. 523--530.
10. Sedláček, R., Smrž, P.: A New Czech Morphological Analyser ajka, *Proceedings of TSD 2001*, Springer-Verlag, LNAI 2166, p.100-107.
11. Sedláček, R., Čapek, T., Svoboda, L.: Morphological Analysis and Czech WordNet, abstract of the non-published paper.
12. Smrž, P., Rychlý, P. (2001) Finding Semantically Related Words in Large Corpora. In Proceedings of TSD 2001. Berlin: Springer-Verlag, p. 108-115. LNAI 2166.
13. Vossen, P.: (Ed.), EuroWordNet: a multilingual database with lexical semantic networks for European languages, Kluwer Academic Publishers, (1999), Dordrecht

Customisable Semantic Analysis of Texts

Vivi Nastase and Stan Szpakowicz

School of Information Technology and Engineering,
University of Ottawa, Ottawa, Ontario, Canada
{vnastase, szpak}@site.uottawa.ca

Abstract. Our customisable semantic analysis system implements a form of knowledge acquisition. It automatically extracts syntactic units from a text and semi-automatically assigns semantic information to pairs of units. The user can select the type of units of interest and the list of semantic relations to be assigned. The system examines parse trees to decide if there is interaction between concepts that underlie syntactic units. Memory-based learning proposes the most likely semantic relation for each new pair of syntactic units that may be semantically linked. We experiment with several configurations, varying the syntactic analyzer and the list of semantic relations.

1 Introduction

Deep processing of natural language data often requires suitably annotated data. Recognition of semantic relations is such a task that benefits from the availability of annotated texts from which we can learn to analyze new data. Manual semantic annotation is a time-consuming activity, and it is seldom possible to capitalize on the annotation effort of other researchers. This is because they work with a different set of semantic phenomena, for example a different list of relations, or because they consider different types of texts or different domains. We present a customizable, domain-independent tool for certain style of semantic analysis. It relies on syntactic information usually supplied by parsing. When the tool achieves its full functionality, its user will be able to impose her own list of semantic relations, select the type of relations she is interested in (between eventsm between an event and an entity, and so on), and plug in her own parser.

Knowledge acquisition from texts spans the range between fully automatic and fully user-driven systems. Automation relies on manually built resources and on statistical or machine-learning methods that extract classifiers from annotated data. The shortcomings of such methods include high cost of annotation and low accuracy of such classifiers on new data. User-driven systems, with friendly interfaces that domain experts use to identify knowledge in texts, allow much higher accuracy (insofar as humans agree on semantic relations). On the other hand, they require time to train people with minimal AI or NLP background, and to encode knowledge.

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 312–323, 2005.

Our approach falls between these extremes. We rely on parsers for the grammatical structure of sentences, in which we identify concepts and pair up those that may interact. The user will associate the types of concepts of interest with syntactic units that the parser's grammar recognizes. For example, if entities are sought, the user will choose nouns and noun-phrases.

Our system extracts pairs of syntactic units from the text, which express concepts that according to syntactic indicators are semantically linked. Each pair is assigned a semantic relation that describes their interaction in the context in which they appear. While there is a default list of 47 semantic relations, the actual list may be user-defined, to acknowledge the fact that no set of semantic relations is appropriate for all NLP tasks. Semantic relations are assigned to pairs semi-automatically. The user can accept a unique suggestion made by the system, choose from a (usually short) list, enter the correct answer manually or reject the pair.

Barker et al. [1] presented and tested a similar idea. One of our innovations is to treat the input text uniformly, without separating syntactic levels (noun phrase, simple clause, compound clause, paragraph and so on). This emphasizes the fact that the same concept can surface in different syntactic forms. We let the user decide what structures are interesting, and focus on the concepts behind these structures. We use syntactic clues to decide which structures interact and to label the interaction. The user may specify the list of semantic relations that best fit the domain and the application.

This paper is organized as follows. Section 2 presents related work in semantic analysis and knowledge acquisition, Section 3 describes the semantic analysis process used by our system, the experiments performed are presented in Section 4, and their results are discussed in Section 5; Section 6 assesses the system's customisability, and the conclusions are presented in Section 7.

2 Related Work

One style of semantic analysis for knowledge acquisition uses predefined templates, filled with information from processed texts [2]. In other systems lexical resources are specifically tailored to meet the requirements of the domain [3] or of the system [4]. Such systems extract information from some types of syntactic units: clauses [5], [6], [7] and noun-phrases [7], [8]. Lists of semantic relations are designed to capture salient information from the domain.

An interesting approach has been tested in the Rapid Knowledge Formation project. The goal was to develop a system for domain experts to build complex knowledge bases by combining components: events, entities and modifiers [9]. The system's interface facilitates the expert's task of creating and manipulating structures representing domain concepts. Descriptions of relations between components come from a relation dictionary; it includes interaction between two events (e.g., causality), an event and the entities involved (e.g., agent), an entity and an event (e.g., capability), two entities (e.g., part), or an event or entity and their properties (e.g., duration or size) [10]. The relations cover three syntactic levels [11].

In purely statistical approaches that traverse corpora to establish connections between concepts based on word collocations, the incidence of errors is not negligible [12], [13], [14].

In our system, user feedback helps produce accurate results, and we will extract knowledge tailored to the user's interests. The knowledge acquisition systems that we have considered suggest that in some domains relations between entities are considered more important, e.g., in medicine [3]. In others it is important to see how entities are related to an event, e.g., in legal texts [2]. We are building a *customisable* system that will focus on the structures of interest to a particular domain. We also experiment with two different lists of relations, to test the flexibility of the semantic analysis module. The goal is to allow the user to plug in a list of relations that describes the input text best.

3 Semantic Analysis

To get the grammatical structure of the input sentence we need a parser, preferably one that has good coverage and produces detailed syntactic information. The parse trees give us syntactic units, from which we choose those of interest to the user, based on the information he provides (explained in detail in Section 3.1). To pair units up we use simple structural information: if a unit is directly embedded in another unit, we assume a subordinate relation between the two; if the two units are coordinate, we assume a coordinate relation. These assumptions are safe if the parse is correct: a modifier is subordinate to its head noun, an argument to its head verb, and a clause perhaps to the main clause in the sentence. If we conclude that two units should interact, we seek an appropriate semantic relation to describe this interaction.

3.1 Extracting Syntactic Units

The user can specify a list of syntactic structures of interest among those recognized by the parser's grammar. It will contain the relevant non-terminals. For example, if the user is interested in entities and their attributes, the list will contain non-terminals that describe nouns, noun phrases and their modifiers. If the user is interested in events and the way they interact, the list will contain non-terminals that describe clauses in the grammar. To simplify the interaction, we let the user choose the corresponding syntactic level (noun phrase, intra-clause or clause level). To allow finer-grained distinctions we will construct a tool that helps the user make a detailed unit selection.

Each syntactic unit will be represented by the uninflected form of its head word. For each unit we also extract the head word's **part of speech**, the **syntactic role** it plays in the sentence (subject, object, noun modifier, etc.), the **indicator** of the structure if one exists (the preposition for a prepositional complement, the subordinator for a subordinate clause, etc.), and additional information if available (tense, number, etc.).

3.2 Pairing Syntactic Units

After finding all syntactic structures of interest, we traverse each structure to extract pairs that are connected by a syntactic relation (modifier, argument, subordinate clause). This means testing whether one structure is embedded in another, or whether they are at the same level, linked by a connective.

3.3 Semi-automatic Assignment of Semantic Relations to Pairs of Syntactic Units

Automatically Finding Suggestions for Semantic Relations. Our system starts with a minimum of manually encoded knowledge, and accumulates information as it processes texts. This design principle was adopted from TANKA [1]. The manually precoded knowledge consists of a dictionary of markers (subordinators, coordinators, prepositions). These markers are closed-class words, so not much effort is required to build such a resource. The system has the option to run without these resources, in which case it will take longer to begin making good predictions.

We apply memory-based learning, so that in every semantic relation assignment the system uses every previously processed example. This allows us to find the best match [15].

Every stored example is a tuple with the structure:

$$(word_{x1}, attr_{x1}, word_{x2}, attr_{x2}, relation)$$

where $word_{xi}$ is the head-word in structure x_i, and $attr_{xi}$ is a vector containing the structure's attributes listed in Section 3.1:

$$attr_x = (POS_{wordx}, SyntRole_x, Indic_x, OtherInfo)$$

Figure 1 presents the distance metric between two examples, represented as tuples.

The first option in our metric applies when a pair containing the same words as the current pair has already been tagged. The same two words may be connected by different semantic relations, if their attributes differ.
(1) *When* **you look** *at a cloud in the sky ...*
(2) **Look** *at the sky above* **you**.

In sentence (1) **you** is the subject, while in sentence (2) it is the prepositional complement. The two **(look,you)** pairs should be assigned different relations (AGENT in (1) and DIRECTION in (2)).

If we constrain the system to match only pairs of structures with the same attributes, generalization to pairs from different syntactic levels will not occur. The pairs **(protest,student)** from the sentences:
(3) *The students protested against tuition fee increase.*
(4) *student protest against tuition fee increase*
should both be assigned the AGENT relation, even though their attributes are obviously different.

$$P_i = [w_{i1}, a_{i1}, w_{i2}, a_{i2}, \text{Rel}]$$

$$dist(P_1, P_2) = \begin{cases} 0 & : \quad w_{11} = w_{21}, w_{12} = w_{22} \\ 0 & : min(d(net(w11), net(w21))) = 0 \\ d(P_1, P_2): & \quad otherwise \end{cases}$$

$net(w) = \{[w_1, w_2] | [w_1, w_2]$ extracted from sentence S, $w \in \{w_1, w_2\}\}$

$d(net(w_1), net(w_2)) = \sum_k d(P_{1k}, P_{2k}); P_{ik} \in net(w_i)$
$d([w_{11}, a_{11}, w_{12}, a_{12}, \text{Rel}], [w_{21}, a_{21}, w_{22}, a_{22}, \text{Rel}]) = \sum_k d(a_{12k}, a_{22k});$
(a_{ik} is an element in vector $attr_i$ associated withw_i)

$$d(a_x, a_y) = \begin{cases} 0: a_x = a_y; \\ 1: a_x \neq a_y \end{cases}$$

Fig. 1. Distance metric used for memory-based learning

The first option in the metric shows that we choose to allow the system to match tuples that do not have the same attributes, in order to let it generalize. The downside is that occasionally the metric will give inaccurate predictions.

When the words in two tuples P_1 and P_2 differ, we consider the distance between P_1 and P_2 to be 0 if the networks centered on the heads of P_1 and P_2 match. This idea was adopted from Delisle *et al.* [16] who applied it to verbs. We extend it to nouns.

A network centered on w consists of:

- a central vertex which represents w. It also contains syntactic and morphological information about w,
- a set of vertices connected with the central one, which represent the syntactic units from a sentence S and their syntactic attributes, with which w is connected through syntactic relation. w may be either the main element in relation with these units, or the modifier. For the sentence
  (5) *Weathermen watch the clouds day and night.*
  the system builds the following network centered on the verb:

```
[watch, v, svo,
 [weatherman,(sent,nil),(subj,nil),_],
 [cloud,(sent, nil),(compl, nil),_],
 [day_and_night,(sent,nil),(compl,nil),_]]
```

  The underscore replaces the semantic relation on that particular edge, which has not been yet assigned.

$d(net(w_1), net(w_2))$ shows how we compute the distance between two networks. It is the sum of distances between pairs of edges. Two edges match if the attributes of the word in vertices have the same syntactic role, and the same indicators. When we match edges, the actual words in the corresponding vertices do not matter, only their attributes. The best match will give the minimum distance. We only attempt to match networks centered on words with the same part of speech.

After processing each example, we will store the networks of tuples centered on both words in the example.

To show how the networks are matched, let us consider sentence (5) from above, and the network centered on the verb *watch*. The system will extract, from previously stored networks, those centered around verbs[1]. If sentence (6):
(6) *Air pilots know that clouds can bring rain, hail, sleet and snow.*
were processed before sentence (5), the system would find the following matching pattern:

```
[know, v, svo,
 [pilot,(sent,nil),(subj,nil),AGENT],
 [bring,(sent,nil),(compl,nil),OBJECT]]
```

The two networks match, because the centers of the networks match - the words have the same part of speech, and the same subcategorization pattern, and the edges match because the attributes of the words in the vertices match.

Because the edges with vertices **(watch,weatherman)** and **(know,pilot)** match, the AGENT relation for the pair **(know,pilot)** is proposed as a possible relation for **(watch,weatherman)**.

In the case where no matching network is found, the distance between two examples is computed as the distance between the modifiers. The key information is in particular the syntactic role and the indicator (if it exists). Our system works with a dictionary of indicators (prepositions, subordinators, coordinators), which are semantic relation markers. One indicator usually signals more than one relation (e.g. *since* may indicate a causal or a temporal relation).

After processing each sentence, the networks of pairs around head words are compiled and stored in memory for use with new examples.

4 Experiments

We need to compare our system with other knowledge acquisition systems available. There are no measures of time or precision that show how an automatic or user-based system performs.

The system that is most similar to ours is the one that we have started from, TANKA [11]. In order to compare the systems we will use the same input data – a text on meteorological phenomena [17] – the same syntactic analyser and the same evaluation measures. An exact comparison is not possible, since the two systems have different working paradigms. We will discuss this in Section 5.

After running the system in a set-up that allows us to compare it with TANKA, we run three more experiments, designed to evaluate its performance with a different list of semantic relations, and with a different parser.

[1] If more detailed information is available, the system will choose only networks associated with verbs that have the same subcategorisation structure (svo,svoc, etc.).

4.1 Parsers

We compare the performance of the system when it uses different syntactic analysers. We first use DIPETT [18], a comprehensive English parser. After using the results obtained to compare the system with TANKA, we plug in different parsers.

We have looked at the Link Grammar Parser [19] and the Xerox Incremental Parser (XIP) [20]. While the Link Grammar Parser is quite robust – it produces a parse tree for every input – its parse trees are too coarse-grained for the type of analysis that our system does. For example, for the sentence:

(7) These tiny clouds are real clouds

LINK produces the following output:

```
[S But [NP these tiny clouds NP] [VP are [NP real clouds NP] VP] . S]
```

We cannot extract modifier-noun relations from this parse tree.

XIP on the other hand produces relatively detailed parse trees. As a bonus for us, it also has the option to extract dependencies, which reduces our task of processing the parse tree looking for pairs. For the sentence

(8) Clouds tell the story.

the parser extracts the following information (apart from the parse tree):

DETD(story,the)
VDOMAIN(tell,tell)
VDOMAIN(cloud,cloud)
OBJ_POST(tell,story)
MAIN(cloud)
HEAD(story,the story)

Since XIP gives as a result pairs of syntactic units, we adjust the system to work with this output, without processing the parse tree.

The original TANKA system analysed three syntactic levels: clause, intra-clause and noun-phrase. For a better comparison with the new system, the list of syntactic units will have to contain structures from all these levels. The new system does not distinguish syntactic levels, but treats all structures the user wants uniformly. Table 1 shows the list of syntactic units that we ask the system to extract. These non-terminals come from DIPETT's code.

In Table 2 we show the list of non-terminals that describe the possible roles that each of the structures plays in a sentence.

XIP uses a much simpler grammar than DIPETT. We use the dependencies it detects to extract the data. The dependencies of interest are presented in Table 3.

The dependency relations also give us information about the syntactic roles that the words play in a sentence. They are shown in Table 4.

[2] The asterisk can be the empty string, or a string containing other dependency information, for example _PRE or _POST (it refers to the position of the modifier relative to the head), _PROGRESS (progressive verb), etc.

Table 1. List of syntactic units from DIPETT

`adj`	adjectives
`n, proper_noun`	nouns (common, proper)
`advs, adv_clause, simple_adv_clause, pp_adv`	adverbial modifiers (simple adverbs, adverbial clauses, adverbial phrases)
`entity`	covers anything that can be conceived of as an entity
`predicate`	head of a verb phrase
`statement`	clause
`simple_sentence, complex_sentence`	sentence (simple or compound)
`subord_clauses, head_main_clause, next_main_clause`	types of clauses (subordinate, main or coordinate)

Table 2. Possible syntactic roles in DIPETT

`subj`	subject
`complement`	complement
`attrs`	attributes
`adverbial`	adverbial
`np_postmodifiers` `pre_modif` `post_modif`	modifiers of the noun phrase
`s_qualifier`	sentence qualifier
`rel_clause` `single_main_clause` `head_main_clause` `next_main_clause`	type of clause
`initial final medial`	type of subordinate clause
`ing_clause` `genitive_ing_clause` `to_infinitive_clause`	type of relative clause

Table 3. List of dependency relations from XIP

`MAIN`	main element in the sentence
`HEAD`	head of a phrase
`VDOMAIN*`[2]	head verb in a clause

Table 4. List of dependency relations that indicate syntactic roles from XIP

`NMOD*`	noun modifier
`SUBJ*`	subject
`OBJ*`	object
`*COMPL*`	complement
`VMOD*`	verb argument

4.2 Semantic Relations

The list of 47 semantic relations that we use combines three separate lists used in [1], one for each syntactic level that TANKA analysed. The semantic relations included are general, domain-independent. They are presented in [21].

Since the system is meant to be customisable, we experiment with plugging in a different list. This list contains 6 relations *causal, temporal, spatial, conjunctive, participant, quality.*

5 Results

The input text consisted of 513 sentences.

When DIPETT was plugged in, the experiment was performed by two judges (to make the assignment of semantic relations more objective) in 5 sessions of approximately 3 hours each. The overall time spent on semantic relation assignment was 6 hours, 42 minutes and 52 seconds. We have used the results collected from this run to automate the system when we changed the list of semantic relations, and when we changed the parser to XIP. Because the alternative list of

semantic relations we used is a generalised version of the original list, a simple mapping allowed us to change the results collected and the marker dictionary file.

Neither DIPETT nor XIP produced a correct parse for every sentence. When a complete parse (correct or incorrect) was not possible, DIPETT produced fragmentary parses. The semantic analyser extracted units even from tree fragments, although sometimes the fragments were too small to let us find pairs. XIP produces a parse tree for each input sentence, although not always a correct one.

Since XIP and DIPETT did not always parse correctly the same sentences, the pairs of concepts extracted by XIP were cross-referenced with the pairs tagged semi-automatically when DIPETT was plugged in, and then manually checked. Pairs obtained from XIP which were correctly identified were kept, even if the parse was erroneous (wrong part of speech, wrong phrase, etc.).

In the experiment with DIPETT, the semantic analyser extracted a total of 2020 pairs, 555 of which were discarded by the user in the dialogue step. An example of an erroneous pair comes from the sentence in example (9).

(9) *Tiny clouds drift across like feathers on parade.*

The semantic analyser produces the pair *(drift,parade)*, because of an erroneous parse tree, in which *parade* is parsed as a complement of *drift*, instead of a post-modifier for *feathers*. The correct pairing *(feather,parade)* will be missing, because it cannot be inferred from the parse tree.

XIP produced fewer correct parses than DIPETT. Its errors come mostly from mistagging words with part-of-speech information. For example, *clouds* is tagged mostly as a verb, even in structurally simple sentences. From the output produced, we extracted 1153 pairs, 445 of which were discarded.

Table 5 shows a summary of the results obtained, for the two parsers and the two lists of semantic relations (with 47 and 6 relations respectively), and the statistics of user actions (accept, choose, supply) during the semi-automatic memory-based semantic analysis step.

Table 5. Summary of results

Parser	nr. of rels	correct pairs	accept	choose	supply
DIPETT	47	1465	30.7% (450)	27.3% (401)	41.9% (614)
DIPETT	6	1465	49%% (718)	24.6% (360)	26.5% (388)
XIP	47	708	27.5% (195)	20.3% (144)	52.1% (369)
XIP	6	708	37% (262)	21.1% (150)	41.8% (296)

The results in Table 5 and the plots in Figures 2(1) and 2(2) show how the system behaves in 4 configurations: with two parsers (DIPETT and XIP), and two lists of relations (47 and 6 respectively). When the system works with the short list of relations it performs better for both parsers. Both lists make the system perform better with DIPETT than with XIP. This may be due to the amount of information that DIPETT provides, compared with XIP. Also, the system runs faster with DIPETT, when information about verb subcategorization allows it to filter out many networks before trying to match them. In

1. User action results for DIPETT, with 47 and 6 relations

2. User action results for XIP, with 47 and 6 relations

3. Comparison between case assignment in TANKA and our system

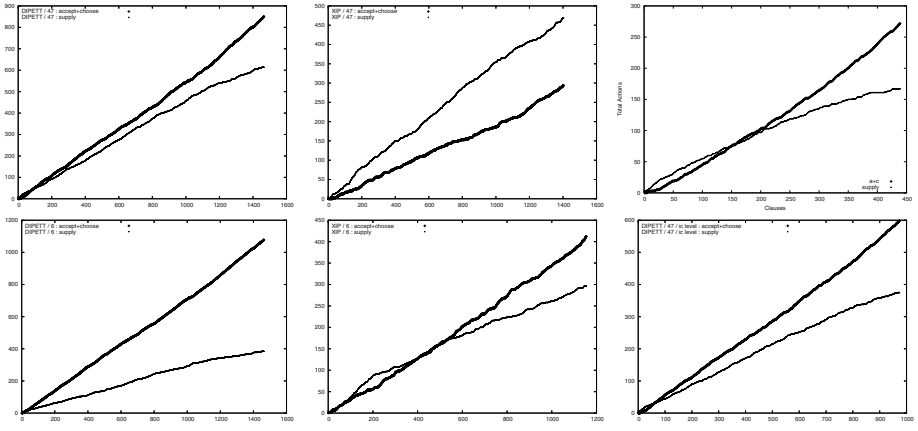


Fig. 2. Comparison of results with two parsers, and with TANKA

each figure the x axis shows the number of examples analysed, and the y axis shows the cummulative number of user actions (accept or choose versus supply). The plots show that as more examples are analysed, the system makes better suggestions.

For comparison of TANKA and our system, we present Figure 2(3). The first graph shows the user action results for the intra-clause level over the course of the experiment for the original TANKA system. Our system does not differentiate between syntactic levels, but based on the structures corresponding to each pair we can decide to which syntactic level it belongs. We have separated the results obtained for pairs from the intra-clause level, and present them for comparison in the second graph in Figure 2(3). The difference in the number of examples tagged comes from the fact that TANKA analyses the entire argument structure around the verb in one step, while our system tags each (argument,verb) pair separately.

We observe from these results that the new system starts learning much earlier. The original TANKA system processed half the input examples before the combined results of the *accept* and *choose* user actions surpassed *supply*; the new system obtains good results almost right away.

6 Evaluating the System's Customisability

The system's (Prolog) code is grouped in two modules: a module for extracting syntactic units and producing pairs, and a module for semantic analysis.

In order to allow the user to choose syntactic structures once a parser is plugged in, no modifications are necessary. During processing, the system automatically assigns a level label to the pair (*np* for noun-modifier pairs, *ic* for verb-arguments pairs, *cl* for pairs of clauses). The user can just set a parameter to *np, ic* or *cl* to choose the level she is interested in. For a more fine-grained selection, the user can access a detailed list of non-terminals used by the parser. When we do not have access to the parser's grammar, a list of nonterminals can be extracted from the parse trees produced. A tool that performs this task is part of future work.

Plugging in a new syntactic parser has various degrees of difficulty. If the structure of the output it produces matches the one obtained with DIPETT, no change is required in the code. Otherwise, the system must be provided with a description of the grammar for the new parser. In the case of XIP, the parser itself produces a list of dependencies, so the system was adjusted to bypass the tree processing stage, and its rules for finding syntactic roles and extract indicators were modified.

Plugging in a different list of semantic relations requires modifying one rule in the semantic analysis module (the rule simply lists the possible semantic relations to be assigned) and, optionally, modifying the dictionary containing 325 markers. While the system will function without this dictionary, its performance will drop since it needs either indicators or previously tagged examples to find semantic relations. In our experiments, we have used a list of 6 relations that generalize the original list of 47, so the dictionary change was automatic; we manually built a hash table to indicate the mapping between the two lists.

7 Conclusions

Having a human judge supervise the task of semantic analysis produces accurate results, but the time needed to spend on the task may be prohibitively long. Also, the type of knowledge that one wants to extract from a text, and the semantic relations to assign to it may vary. We propose a semi-automatic semantic analysis system, customisable to the task at hand. It can use different syntactic analysers, it will extract the syntactic units that the user is interested in, and will tag them with the semantic labels that are relevant to the domain of the input text.

We have compared our system with a similar endeavour. The results show that having a unified approach to analysing text leads to better results, in the form of faster learning. The learning that the system performs is memory-based, in which all examples previously analysed are used when processing a new one.

Part of future work is to deploy the system on the Web, so that it can be used for semantic analysis with various configurations. We also aim to refine and improve our system's learning part by using machine learning tools and lexical resources. We experimented with using other methods other than memory based learning, and lexical resources such as WordNet and Roget's Thesaurus. The experiments performed with base noun-phrases were promising [21], and we plan to incorporate these resources in our system.

References

1. Barker, K., Delisle, S., Szpakowicz, S.: Test-driving TANKA: Evaluating a semi-automatic system of text analysis for knowledge acquisition. In: Canadian AI, Vancouver, BC, Canada (1997) 60–71
2. Baker, C.F., Fillmore, C.J., Lowe, J.B.: The Berkeley FrameNet project. In: COLING-ACL, Montreal, Canada (1998) 86–90
3. Rosario, B., Hearst, M.: Classifying the semantic relations in noun-compounds via a domain specific hierarchy. In: EMNLP, Pittsburg, PA, USA (2001) 82–90
4. Gomez, F.: A representation of complex events and processes for the acquisition of knowledge from text. Kowledge-Based Systems **10** (1998) 237–251
5. Fillmore, C., Atkins, B.T.: FrameNet and lexicographic relevance. In: LREC, Granada, Spain (1998)
6. Gildea, D., Jurafsky, D.: Automatic labeling of semantic roles. Computational Linguistics **28** (2002) 245–288
7. Hull, R.D., Gomez, F.: Semantic interpretation of nominalizations. In: 13th National Conference on Artificial Intelligence, Portland, Oregon, USA (1996) 1062–1068
8. Rosario, B., Hearst, M., Fillmore, C.: The descent of hierarchy and selection in relational semantics. In: ACL, Philadelphia, PA, USA (2002)
9. Clark, P., Porter, B.: Building concept reprezentations from reusable components. In: AAAI, Providence, Rhode Island (1997) 367–376
10. Fan, J., Barker, K., Porter, B., Clark, P.: Representing roles and purpose. In: KCAP. (2001) 38–43
11. Barker, K.: Semi-Automatic Recognition of Semantic Relationships in English Technical Texts. PhD thesis, University of Ottawa, Department of Computer Science (1998) http://www.cs.utexas.edu/users/kbarker/thesis.
12. Kilgarriff, A., Tugwell, D.: WORD SKETCH: Extraction and display of significant collocations for lexicography. In: Workshop on Collocation: Computational Extraction, Analysis and Exploitation, 39th ACL & 10th EACL, Toulouse, France (2001) 32–38
13. Lin, D., Pantel, P.: Concept discovery from text. In: COLING, Taipei, Taiwan (2002) 577–583
14. Pantel, P., Lin, D.: Discovering word senses from text. In: SIGKDD, Edmonton, Canada (2002) 613–619
15. Daelemans, W., van den Bosch, A., Zavrel, J.: Forgetting exceptions is harmful in language learning. Machine Learning **34** (1999) 11–34
16. Delisle, S., Copeck, T., Szpakowicz, S., Barker, K.: Pattern matching for case analysis: A computational definition of closeness. In: ICCI, Sudbury, ON, Canada (1993) 310–315
17. Larrick, N.: Junior Science Book of Rain, Hail, Sleet and Snow. Garrard Publishing Company, Champaign, Illinois (1961)
18. Delisle, S., Szpakowicz, S.: Realistic parsing: Practical solutions of difficult problems. In: PACLING, Brisbane, Queensland, Australia (1995)
19. Temperley, D., Sleator, D., Lafferty, J.: The LINK parser (1998) http://www.link.cs.cmu.edu/link.
20. Chanod, J.P., Ait-Mokhtar, S., Roux, C.: Xerox Incremental Parser, ongoing research (2004) Xerox Research Centre Europe.
21. Nastase, V., Szpakowicz, S.: Exploring noun-modifier semantic relations. In: International Workshop on Computational Semantics, Tillburg, Netherlans (2003)

ITOLDU, a Web Service to Pool Technical Lexical Terms in a Learning Environment and Contribute to Multilingual Lexical Databases

Valérie Bellynck[1], Christian Boitet[2], and John Kenwright[3]

[1] équipe STG, LGP2
461 rue de la Papeterie - BP 65, 38402 Saint-Martin-d'Hères - France
Valerie.Bellynck@efpg.inpg.fr
[2] équipe GETA, laboratoire CLIPS,
385 rue de la bibliothèque - BP 53, 38041 Grenoble Cedex 9 - France
Christian.Boitet@imag.fr
[3] Cellule TICE, bureau 2.12
701 rue de la piscine - BP 81, 38402 Saint-Martin-d'Hères - France
John.Kenwright@inpg.fr

Abstract. The first stage of the ITOLDU project aims to facilitate technical English teaching, especially for vocabulary acquisition. We are pursuing two immediate goals: maximizing positive student contributions, even outside of the classroom, and minimizing teacher intervention. The resulting application is designed to support investigations on what can entice users to contribute collaboratively towards enriching a bilingual technical lexicon in a fertile teaching context. The second stage will be to investigate how to use ITOLDU and similar tools to elicit free (but not necessarily voluntary or even conscious) contributions to the research-oriented, linguistically very rich multilingual PAPILLON lexical database.

1 Introduction

The cost of building respectable bilingual or multilingual dictionaries specialized in a certain technical field is very high if one uses professional lexicographers and terminologists. Even if enough money is available, such professionals are quite difficult to find for many domains. Hence, several projects have been started to create such lexical resources via Internet, by setting up web sites requesting free contributions. However, it is difficult to entice web surfers to contribute without any kind of reward. This is the specific type of problem that the Papillon project (http://www.papillon-dictionary.org/) is encountering (Mangeot-Lerebours 2001, 2003).

One solution is to offer a service such as the Oki Electric web site (http://www.yakushite.net) where free access to the Pensée MT system is offered, in exchange for contributions to bilingual dictionaries (organized in a hierarchy corresponding to domains of interest an associated communities) (Murata 2003).

In our case, we would like to "populate" the Papillon database, by letting students in classes of computer science and English, in a French engineering school, contribute specialized terms and their translations (plus if possible definitions and references).

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 324 – 332, 2005.

Our proposal invites students to contribute dictionary items as part of their English course assignment. The idea, then, is not only to exchange contribution for grades, but more so to stimulate mutual aid, increase motivation, favour self-learning, attach importance to student implication in their education, create a lasting tool which can accompany them through their working life and finally, trigger a common interest and pride in their acquisition of a foreign language.

In the first section, we will explain the teaching and learning context in more detail (students, goals, resources). In the second section, we will explain how to merge access and contribution to the lexical database in this context. In the third section, we will describe the current version of our system and associated contribution-based web site, ITOLDU (Industrial Technical On Line Dictionary for Universities) – extranet version at http://www.pagesperso.laposte.net/kenwright/ITOLDU. In the last part, we will present some ideas on how to induce more contributions from users.

2 The Teaching and Learning Context

2.1 Size and Types of Classes with ITOLDU V1 Test

At the EFPG engineer school, we train each year about 200 students in 10 groups with 3 years of study for each class. We have to manage different initial English levels, some students having learned English as a second foreign language (LV2). This year, the ITOLDU web site is being used via the EFPG intranet with 200-250 students.

The technical specific fields of study cover pulp and paper science, fiber chemistry, packaging, rheology, digital printing, and colour management.

For preliminary experiments to evaluate usability, the ITOLDU web site (V1 test) was accessed via an extranet version by a class of 6 mature "sandwich course" students doing a technical degree.

The experiment took place between 15th May and 30th June 2004, and was composed of a total of 14 hours contact teaching and a final 2- hour written and oral exam. Lessons were held every 3 weeks and students had between 2 to 3 two-hours lessons per "contact week". (The class was composed of students who lived as far away as Paris).

It was interesting to test ITOLDU in this context due to the imposed spacing between lessons and the opportunity for students with varying levels of English to contribute to vocabulary acquisition and share findings with their "community". In the first test, the specific field was not technical but common, and centred on professional communication as it was a skill that all of the students in the class needed and could cope with given their varied level of command of the language. The results from this test would serve as a point of reflection for any modifications needed in the long-term and prepare teachers for any trouble-shooting before the generalisation of the tool in 2004-2005.

2.2 About the Vocabulary to be Learned

- Learning technical English is heavily sought after by French institutions.
- The most important direction is English – French: the tool should help remember English terms to express accurate technical concepts.

- The students do not yet know the technical terms in English and have only recently encountered them in French.
- There are probably 10,000-20,000 terms with which the teacher is not necessarily familiar (whether in French or in English).
- The basic part is to be learned by all students and represents about 10% (1000-2000 items). In reality they know how to use between 150-300 specific words/terms in English, associated with their technical field (paper science) by the time they leave in 3rd year.
- Each student should choose and learn a small fraction of the remaining 90%.

3 How to Merge Access and Contribution

Human manipulation of digital dictionaries helps users firstly to use new ways of accessing words, and secondly to take their actions into account as "unconscious contributions". The most important factors seem to be the tightness of integration of contribution of the contributing and learning environments, and the simplicity of both web interfaces.

3.1 Access and Contribution

In order to access words via a dictionary, people can start from synonyms they have in their head, look up their definitions, choose the one which seems the nearest, and then move again to words used in that definition. But one can also begin to read the dictionary from any page, trying to find some related idea ("linear" access).

In accessing words through a discussion with someone else, one can begin by expressing an idea, and then stop if that person can't find the word, ask people around to help find an expression or a word that could take the place of the sought after expression, and continue.

In accessing a digital dictionary, one is usually limited to entering a lemma (or wordform if there is a "lemmatize" option), and to filtering via a small number of constraints (part of speech/clause, domain, variety such a GB/US). The usual methods are already closed to the book access, but without its "linear" extension, which would be limited by the screen-window anyway. Extending access to more "human" ways, there are two problems.

Firstly, how to express the request (how to specify the word looked for)? Secondly, how to solve this problem and transcribe the request via digital access? A proposal for a few access modalities of access has been presented in a paper on "Sensillons for the Papillon project" (Bellynck, 2002).

Another point is to find how to transform the passive use of a digital dictionary into an active contribution to its creation. Use-friendliness is but one of the key factors.

Generating the will to participate in this community is also primordial, along with minimising the time required to add contributions. The notion of "reward" seemed necessary from the start. Thus, students were informed that not only would their participation count towards their final mark but also that the quality of their translations and their implication in "voting" for their peers' contributions would generate a bonus/minus mark at the end of the term.

3.2 Teaching and Learning Context

The context of English learning allows us to use the same experimental contexts for variants of experiments. Basic vocabulary needs are covered as well as specific technical ones shared by different communities. The teaching-learning context leads us to divide the vocabulary into domains of use (business, basic, or technical English for different specialities). The teacher has the option of adding or deleting categories according to the needs of his/her class(es).

Asking students to look for the French translation(s) of an English technical term may reveal the need for a strategy which is different from that used in the case of basic English, particularly in our case, where French students don't yet know the technical terms in their own language well enough. The current version could be used with other languages, but our learning context concerns only translations from French to English.

In order to investigate the modalities of access, we need voluntary and motivated users. In a learning context, the teacher can simply motivate students to contribute precise translations with specific bonuses. But in reality, due to time constraints, the teacher often can't spend a lot of time checking up on each contribution of every student: the work would be in addition to normal working hours. Our solution from the outset was to let the community take up this function in the full knowledge that the teacher would check a certain number of contributions per students through the year and that wrong translations or rushed voting would lead to low marks and minus scores.

4 The ITOLDU System

4.1 Functions of the First Version

In this first stage, we are pursuing two goals: maximize student positive contributions, even out of courses, and minimize teacher intervention.

The idea is simple: through the English courses and between two courses, each student has to collect or create the lexical data for her/his own digital dictionary based on texts or other sources given by the teacher. The student can also add other words or findings s/he comes across in her/his own pursuit of language acquisition. S/he can choose from existing propositions that s/he finds and correct or create her/his own proposition. Contributing a translation or selecting an existing proposition generates a vote for the student who has created it.

The resulting application should help us to investigate on what can help users to contribute to collaborative lexical technical thesaurus in the fertile teaching environment. In the larger project, we want to take advantage of convergent ideas that all entice to favour lexical-user contributions.

4.2 Teacher Side of ITOLDU

The resulting application should help us to investigate on what can help users to contribute to collaborative lexical technical thesaurus in the fertile teaching environment. In the larger project, we want to take advantage of convergent ideas that all entice to favour lexical-user contributions.

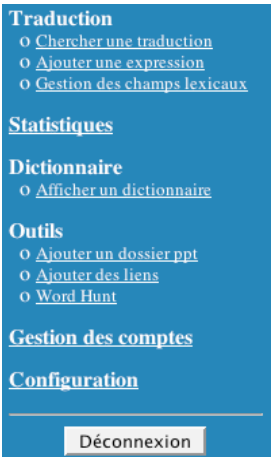


Fig. 1. Teachers' Summary

ITOLDU offers teachers the possibility of supervising student groups, encouraging involvement thanks to bonus marks, and livening up vocabulary via playful word hunts.

Figure 1 shows the summary of a teachers' session. He can customize general properties (title of the site, language), broadcast learning activities, contribute to the digital dictionary's construction (search for a translation, add a new expression and create new technical domains – called "categories"), manage student groups ("Gestion des comptes"), and look at each student or class-room contribution shown in Figure 4. ("Statistiques", "Afficher un dictionnaire"). A heaven-send is that teachers never have to look inside the source of a html page (or worse in code!).

4.3 Student Side of ITOLDU

ITOLDU allows students to gather words or expressions, and to contribute consciously with a proposition of translation or unconsciously with a selection of someone else's translation.

When a student connects to her/his own digital dictionary, s/he finds a summary (Figure 2) to access the digital dictionary (search translation and add a new expression), use the teachers' prepared "to-do" tools ("Outils": CV, application letter, word hunt…), look at her/his ows statistics, and this measure their own implication or knowledge against fellow classmates, or print the current digital dictionary (Figure 5).

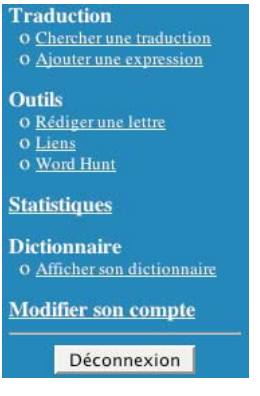


Fig. 2. Students' summary

4.4 Scenario

Let us imagine that a teacher prepares her/his course for a classroom and create groups and logins. S/he then gives the students some technical English text to study, which includes unknown technical words and expressions.

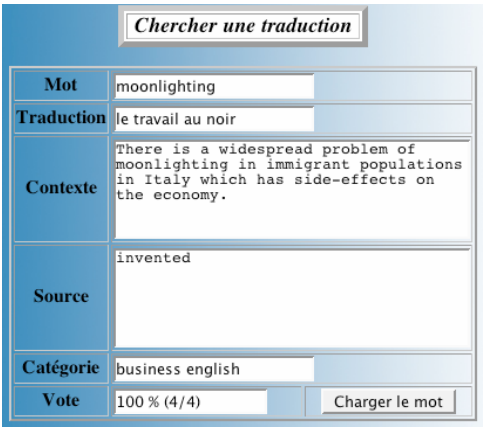


Fig. 3. Basic search access form

Students will be shown how the ITOLDU tool works, how contributions affect part of their final grade and the concept of sharing knowledge and mutual aid.

The teacher can also include an initial "word hunt" (list of targeted vocabulary) to set the ball rolling and encourage users to regularly check the site so as not to be the last to find a word.

When reading a text, a student can be confronted with an unknown word, s/he uses the ITOLDU search tool (Figure 3).

In this first version of the application, the access form is minimal: one can only enter an expression or the first letters of an expression in the first input field. But this form has been designed to be easily replaced or combined with richer ones later.

If there is no entry for the word or expression, the student can enter a translation proposal, with an example of use, the context where s/he has found it, and its bibliographical reference. Each voluntary contribution is cumulated for the statistics and the grades of each student.

If there are one or more entries for the targeted word or expression, the student can select the one seems to be the best and add it to her/his own dictionary. This action results in an involuntary or unconscious contribution: a vote for the student who suggested this translation (the author).

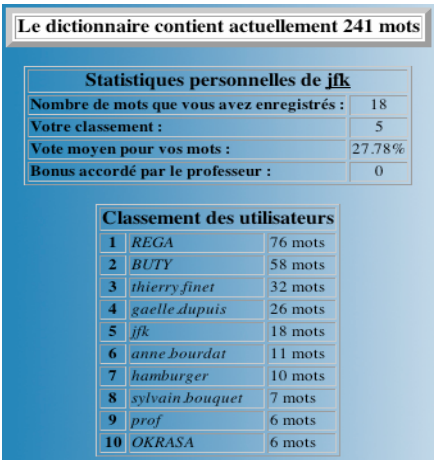
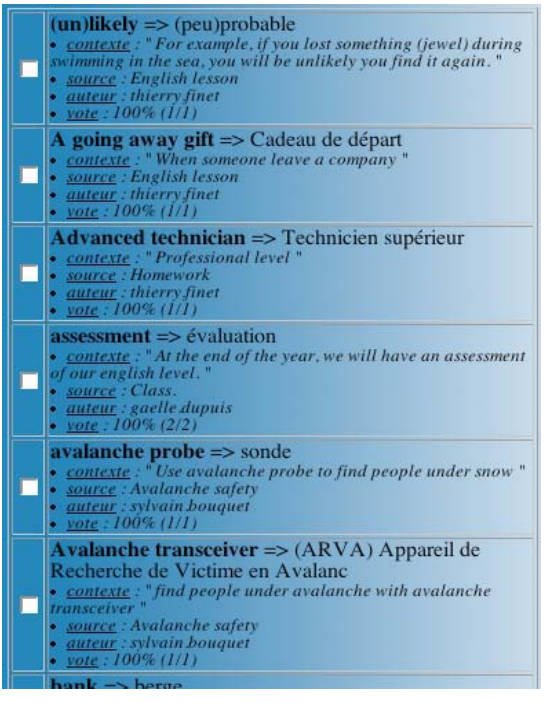


Fig. 4. Resource pooling statistics

Each vote is cumulated in the statistics of the author (Figure 4). Here, "jfk" is the name of a student, used for testing.

The method of using selections as implicit votes, and even more so as "unconscious contributions", is the kernel of the system.



As a matter of fact, it will replace teacher mediation. Students can't enter wrong definitions on purpose, because they would be incorporated in their own dictionaries (Figure 5), and teachers can trace contributions.

For word hunts (shown in Figure 6), the student who finds the word first "wins the game" and has her/his score published on a score board – just like in a computer game.

Initial experiments were beginning at the same time as this paper was being written, so comprehensive findings cannot be published at this stage. However, in the first test this proved to be a healthy instigator of competitiveness between competing classmates.

Fig. 5. Taking over dictionary

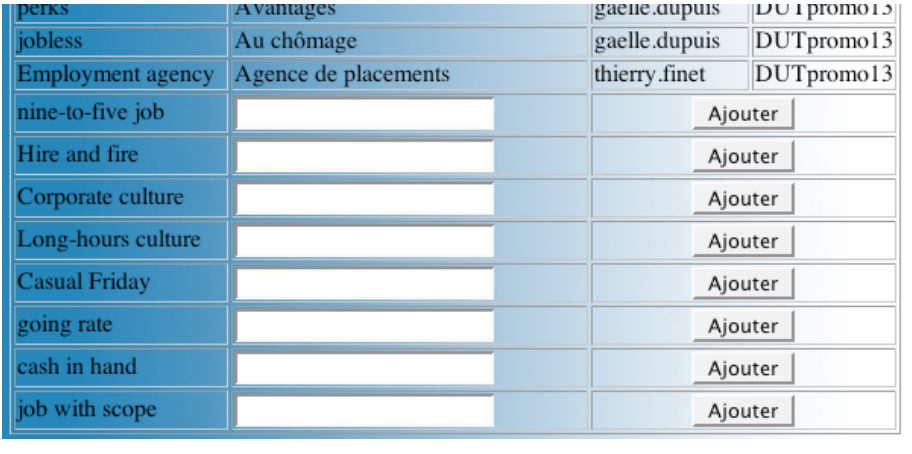


Fig. 6. Word hunt

4 How to Induce More Contributions

Other possible ways to induce more contributions are to:

- generalize the "scoreboard" idea so that credits can be shown for each part of the data.
- introduce personalization facilities (automatic or semi-automatic user profiling), so that the system can propose personalized lists of "things-to-do" or new contributions in the user's domain of interest. For example, the system could remember that a certain user likes to contribute definitions, and propose her/him to complete missing definitions. In Papillon, there are many other types of information to enter, such as pronunciation, examples of use, etc., and every user-contributor may have a specific mix of interest in them.
- allow users to self-organize in groups and groups of groups, each group having certain access rights and a profile.
- give users access to tools which can extract potential translation pairs from comparable corpora (texts on the same domain in two or more languages, usually not parallel).
- let users contribute directly through an "active reading" interface (where translated words or idioms appear in annotations of text read).
- make the importing environment accessible to users wanting to upload bunches of translation pairs from any format (Excel, Word, FileMaker, XML, etc.). At this moment, to import a dictionary into Papillon, the contributor must put it in XML (with his own tags), and the database manager has to adapt a PERL script to convert it into the CXM (Common Dictionary Format) DTD, convert it, and add it to the dictionary collection.
- as the ultimate objective, integrate the lexical contribution function as an add-on (plug-in) in as many applications as possible, used by the general public (text and document processors, spreadsheets, presentation tools, mailers, etc.).

5 Conclusion

We have presented the context use and functions of the ITOLDU system, a web site to help technical English teaching by student resource pooling via lexical access. That context is favourable for gearing users to contribute new terms to the dictionary. In order to ensure quality without asking for teacher's mediation, we have implemented a voting mechanism.

We have first used ITOLDU with a small class of economics students. The web site is used for technical English teaching in a French engineer school during this academic year with 200-250 students. In parallel, we will add more functions to help and entice users, not necessarily students, to contribute lexical data, and conduct experimentations.

With this kind of tool, we are implementing one of the two possible solutions to the nagging problem of enticing users to contribute. As it is in practice not possible to get voluntary and free contributions, one can try to get either voluntary and rewarded contributions, or free and unvoluntary (or even forced!) contributions. The first solution is that implemented in yakushite.net, where users contribute because they use a

free translation support tool where their lexical contributions become quicky active. The second solution is implemented in ITOLDU and can be generalized to any situation where students are learning information of interest, and where teachers are looking for a tool to alleviate their work.

Acknowledgements. Our thanks go to Cédric Sintes and Sébastien Duvillard-Charvaix for their contribution to the first developments of the project through a student project, and to Mathieu Mangeot and Gilles Sérasset for designing and implementing PAPILLON.

References

1. V. Bellynck (2002). *Bases lexicales multilingues et objets pédagogiques interactifs : Sensillon pour Papillon.* In "Proceedings of Papillon 2002 Seminar", NII, 13 p., Tokyo, July 2002.
2. M. Mangeot-Lerebours (2001). *Environnements centralisés et distribués pour lexicographes et lexicologues en contexte multilingue.* PhD in Computer Science, Université Joseph Fourier, Grenoble I, 280 p., Grenoble, France.
3. M. Mangeot-Lerebours. G. Sérasset. M. Lafourcade (2003). *Construction collaborative d'une base lexicale multilingue, le projet Papillon*. TAL, 44/2, pp. 151-176.
4. T. Murata, M. Kitamura, T. Fukui, T. Sukehiro (2003). *Implementation of Collaborative Translation Environment `Yakushite Net'*. Proceedings of MT Summit VIII. New Orleans, Sept. 2003.
5. Naoyuki Tokuda and Liang Chen (2001). *An Online Tutoring System for Language Translation*. IEEE Transactions on Multimedia, vol. 8, no. 3, pp. 46-55, July-September 2001.

Building a Situation-Based Language Knowledge Base

Qiang Zhou and Zushun Chen

State Key Laboratory of Intelligent Technology and Systems,
Dept. of Computer Science and Technology,
Tsinghua University, Beijing 100084, P. R. China
zq-lxd@tsinghua.edu.cn

Abstract. Language resources are very important for natural language processing research and applications. This paper will introduce our ongoing research work to build a situation-based language knowledge base for the Chinese language, based on two basic language resources: three Chinese semantic lexicons and a large scale Chinese treebank. We developed a supporting platform to make full use of the abundant information contained in current Chinese semantic lexicons so as to gradually summarize the complete situation descriptions, organize them as situation network and build corresponding descriptive definition dictionary for different concepts. We explored an efficient algorithm to link from syntax to semantics so as to introduce suitable semantic explanations into current Chinese treebank and gradually build a situation-based semantically-annotated corpus. All these research work will lay a good foundation for the computational infrastructure in Chinese natural language processing.

1 Introduction

Language resources are very important for natural language processing research and applications. In recent years many researchers have devoted themselves to the construction of large-scale language resources. Nowadays, there are two types of commonly used language resources. One is syntactically annotated corpora. Some typical examples include the Penn Treebank for English [9], the Prague Dependency Treebank for Czech [6] and the TIGER treebank for German [3]. The other is the semantic lexicons. Most of them are manually compiled by linguists or lexicographers. Some typical examples are the WordNet [10] and Levin's English verb classes [8]. The key issue is how to integrate these two types of language resource so as to build the linking bridge between syntax and semantics. The Proposition Bank (PropBank) [7] and FrameNet [1] projects have made some tentative explorations in these respects.

Unlike these research projects, we propose a new situation-based language knowledge description framework. In this framework, we use situation as a mathematical model to describe a cognition scheme and try to define a concept under its generating situation. Therefore, the situation theory can serve as a unified theoretical framework for constructing lexical semantics and the natural language knowledge infrastructure built upon it. This paper introduces our ongoing research work to build a situation-based language knowledge base for the Chinese language based on two basic language resources: three Chinese semantic lexicons and a large scale Chinese treebank.

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 333 – 336, 2005.

2 The Situation-Based Knowledge Framework

In our opinion, a concept is generated in a peculiar cognition scheme, which will be called its generating scheme. The new concept absorbs and condenses the new knowledge contained in the scheme to gradually form a relatively stable individual that can be independently quoted in furthermore cognitive activities. At that moment, we can coin a new word (or phrase) to name the concept so that it can be easily used or quoted in common communication and conceptual thought. So the word becomes the symbolic embodiment of a concept. The basic and the most important attributes of the concept are issued in its generating scheme. We cannot describe and define the concept clearly unless we put it into its generating scheme.

We proposed to use the situation [2] as a mathematical model to describe a cognition scheme. Therefore, the situation theory [2] can serve as a unified theoretical framework for constructing the lexical semantics and the natural language knowledge infrastructure built upon it. Under this framework, many new issues should be explored, including: (1) how to use a situation to express a scheme and use a situation to describe a concept; (2) how to formulate the situation algebra for describing the relations, transformations, and operations among situations so as to simulate conceptual thinking by means of algebraic calculation; (3) how to construct a situation network to implement a scheme structure and conceptual structure, where the key point is the constitution and organization of a semantic dictionary. Chen and Zhou (2002) discussed more detailed questions about them.

3 Supporting Platform for Situation Development

To build a large-scale situation-based language knowledge base for the Chinese language, we proposed a two-stage approach method.

At the first stage, we manually summarize some commonly-used typical cognition schemes under intuitional thought on several semantic lexicons. Then, we construct rough situations to express these cognition schemes so as to reflect the key information among them. These situations can be organized into an initial situation network, based on the basic ontological classifications under four main domains: physical world, mental world, symbolic world and human world.

At the second stage, we search and extract a group of word entries or concepts with coherent relations for a special situation and try to define or describe these conceptual meanings by using this situation. In the process of defining concept, the situation description can be refined to reflect more detailed cognitive contents, and a new semantic dictionary can be built. In the dictionary each word entry can be assigned a suitable situation-based definition for its conceptual meaning. So it is an interdependent and interaction process for constructing both situation descriptions and semantic dictionary.

To make the above construction method more feasible, we developed a supporting platform, under which three Chinese semantic lexicons were merged to form the basic semantic resources and many useful tools were developed to make full use of the semantic knowledge defined among them.

Based on the platform, we have summarized about 50 situations now. Some of them are basic situations for further descriptions, such as 'existence', 'maintain', 'transfer', 'destroy', etc. Some are detailed situations for organization name identification, such as 'transaction', 'transport', 'manufacture', etc. All of them are related with hundreds of words and concepts. We hope to build a small size situation network based on them at the end of this year.

4 Bridge the Gap Between Syntax and Semantics

Apart from the above semantic lexicons, another useful language resource for situation development is the large-scale annotated corpus, where different kinds of language performance phenomena will bring in the paraphrase problems that a semantic content may be expressed by a variety of ways. If we can anchor the parameter of a situation to the suitable referring expressions in real world sentences, we will obtain a new viewpoint to study the implementation of a concept in the procedure that contrasts, restores, and refers to its generating situation in a special contextual environment.

Nowadays, we have built a large-scale syntactically-annotated Chinese corpus: the Tsinghua Chinese Treebank (TCT) [11], which contains about 1,000,000 Chinese words of texts drawn from a balanced collection of texts published in 1990s. Here the key issue is how to bridge the gap between syntax and semantics so as to introduce situation-based description information in current TCT.

Our current strategy is to select a suitable semantic representation similar with the predicate-argument structure used in PropBank project, and focus on the research of syntax and semantics linking to bridge the largest gap between current syntactic annotations in TCT and the new semantic representation. Here, the function of syntax and semantics linking is twofold. Through syntax to semantics linking, we can assign suitable semantic explanation for each syntactic template in current treebank. The combination of syntactic and semantic representations will give us enough information for deep conceptual understanding. Through semantic to syntax linking, we can assign useful syntactic distributions for different sense descriptions. They are very important for natural language generation.

Now, we have developed an efficient Chinese 'syntax→semantic' linking algorithm [5], whose accuracy is about 83%. Based on it, we can extract and build a large scale Chinese verb knowledge base from current TCT, where each verb entry is related with its syntactic templates, semantic role frames and detailed annotated examples. It will bring strong supports to refine current situation network.

5 Conclusions

We take situation as a suitable framework for organizing and positioning lexical semantic knowledge. This paper introduced our current research work to build a situation-based Chinese language knowledge base, whose basic language resources are a large scale Chinese treebank and three Chinese semantic lexicons. The research of syntax and semantics linking algorithm build the bridge between these two basic language resources through the assignation of suitable semantic explanations for syn-

tactic constructions in current treebank. And the development of a language resource supporting platform makes full use of the abundant syntactic and semantic knowledge to build a situation-based Chinese computational infrastructure.

In the future research work, we hope to refine the language knowledge base in the following respects: (1) Improve the syntax and semantics linking algorithm to obtain more accuracy linking results; (2) Develop a semi-automatic tool to summarize rough situations based on current syntactic and semantic knowledge.

This work was supported by the Chinese National Science Foundation (Grant No. 60173008), National 973 Foundation (Grant No. 1998030507) and National 863 plan (Grant No. 2001AA114040).

References

1. Baker, C.F., Fillmore, C.J., and Lowe, J.B. (1998). The Berkeley FrameNet project. In *Proceedings of the COLING-ACL'98, Montreal, Canada*, p86-90.
2. Barwise, J.; Perry, J. (1983) *Situations & Attitude*, MIT Press. Re-issued by CSLI Publications, 1999.
3. Brants, S., & Hansen, S. (2002). Developments in the TIGER annotation scheme and their realization in the corpus. In *Proc. of the Third Conference on Language Resources and Evaluation LREC-02,* Las Palmas, Spain. p.1643-1649.
4. Chen Zushun and Zhou Qiang (2002) Situation – A suitable framework to organize and position lexical semantic knowledge. *Computational Linguistics and Chinese Language Processing, 7(2)*, p1-36
5. Dang Zhengfa, Zhou Qiang (2004). Link Syntactic Tags with Semantic Roles. *Technical report 04-04, State Key Lab. of Intelligence Tech. and Systems, Tsinghua University.*
6. Hajic, J. (1999). Building a syntactically annotated corpus: The Prague Dependency Treebank. In E. Hajicova (Ed.), *Issues of valency and meaning. Studies in honour of Jarmila Panevova.* Prague, Czech Republic: Charles University Press.
7. Kingsbury, P.; Martha Palmer, and Marcus, M. (2002). Adding Semantic Annotation to the Penn TreeBank. In *Proceedings of the Human Language Technology Conference, San Diego, California*.
8. Levin, B. (1993). English Verb Classes and Alternations A Preliminary Investigation. MIT Press.
9. Marcus, M.P., Marcinkiewicz, M.A. and Santorini, B. (1993). Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics,* 19(2), 313-330
10. Miller, George A., & Fellbaum, C. (1991). "Semantic Network of English", In *Beth Levin and Steven Pinker (Eds.) Lexical & Conceptual Semantics*. Elsevier Science Publishers, B. V., Amsterdam, The Netherlands.
11. Zhou Qiang (2003) Build a Large-Scale Syntactically Annotated Chinese Corpus. In *Proc. of 6th International Conference of Text, Speech and Dialogue (TSD2003)*, Czech Republic, Sept. 9 –12. Springer LNAI 2807. p106-113.

Unsupervised Learning of P NP P Word Combinations*

Sofía N. Galicia-Haro[1] and Alexander Gelbukh[2]

[1] Faculty of Sciences UNAM Universitary City, Mexico City, Mexico
sngh@fciencias.unam.mx
[2] Center for Computing Research, National Polytechnic Institute, Mexico
gelbukh@cic.ipn.mx
www.Gelbukh.com

Abstract. We evaluate the possibility to learn, in an unsupervised manner, a list of idiomatic word combinations of the type preposition + noun phrase + preposition (P NP P), namely, such groups with three or more simple forms that behave as a whole lexical unit and have semantic and syntactic properties not deducible from the corresponding properties of each simple form, e.g., *by means of*, *in order to*, *in front of*. We show that idiomatic P NP P combinations have some statistical properties distinct from those of usual idiomatic collocations. In particular, we found that most frequent P NP P trigrams tend to be idiomatic. Of other statistical measures, log-likelihood performs almost as good as frequency for detecting idiomatic expressions of this type, while chi-square and point-wise mutual information perform very poor. We experiment on Spanish material.

1 Introduction

Our goal is to compile, in an unsupervised manner, a list of word combinations of the type preposition + noun phrase + preposition (P NP P) constituted by three or more simple forms (the noun phrase or even a preposition can consist of more than one word) that behave as one lexical unit, with non-compositional semantics. Specifically, such combinations are frequently equivalent to prepositions, i.e., they can be considered as one multiword preposition: e.g., *in order to* is equivalent to *for* (or *to*) and has no relation with *order*; other examples: *in front of* '*before*', *by means of* 'by', etc. Apart from semantic analysis, such a dictionary can be useful in syntactic disambiguation, namely, prepositional phrase attachment: given a compound preposition *in_order_to* is present in the dictionary, the *to* in *John bought flowers in order to please Mary* would not be attached to *bought*.

We experimented with Spanish material. There is no complete dictionary of such word combinations for Spanish. Only a limited number of such combinations are included in common dictionaries, which in addition do not give their variants such as *por vía de* 'by' ('by way of') / *por la vía de* 'by' (literally 'by the way of'), etc.

In this work, we investigate unsupervised corpus-based methods to learn the word combinations of the considered type (P NP P that behave as a single lexical unit; see case 1 in the example below) and the ways to differentiate such idiomatic collocations

* Work partially supported by Mexican Government (CONACyT, SNI, CGPI-IPN, PIFI-IPN).

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 337–340, 2005.

from literal combinations (case 2), on the one hand, and from larger idioms (in which such combinations very often participate; case 3), on the other hand:

1. Idiomatic expression: *a fin de obtener un ascenso* 'to obtain a promotion' (literally 'at end of'),
2. Free combination: *a fin de año obtendrá un ascenso* 'at the end of the year she will be promoted',
3. Part of a larger idiom: *a fin de cuentas* 'finally' (literally 'at end of accounts').

In Section 2 we outline our unsupervised method and present the experimental results which we discuss. Finally, in Section 3 we draw some conclusions.

2 Methodology and Results

We used a texts collection obtained from Internet, corresponding to four different Mexican newspapers that daily publish online a considerable part of their complete edition. The texts correspond to diverse sections: economy, politics, sport, etc., from 1998 to 2002. The collection has approximately 60 million tokens; see details in [4].

We extracted all word strings corresponding to PNPP in the following grammar:

PNPP → P NP P
NP → N | D N | V-Inf | D V-Inf

where P stands for preposition, N for noun, D for determinant, and V-inf for infinitive verb (in Spanish, infinitives can be modified by a determinant: *el fumar está prohibido*, literally 'the to-smoke is prohibited').

We found 2,590,753 such PNPP string tokens, or 372,074 types, of which 103,009 had frequency higher than 2. Not all of them were idiomatic: e.g., from the 20 more frequent items, the strings *en la ciudad de* 'in the city of', *del gobierno de* 'of the Government of', *del estado de* 'of the State of' are free (literal) combinations.

Then we computed for each extracted item various statistical measures as described below, in order to evaluate whether these characteristics correlate with idiomatic (as a unit, e.g., *in order to*) combinations and discriminate them from free (literal, e.g., *in the head of Mary*) ones.

To compare performance of these measures, we used a list of known Spanish idiomatic word combinations [7], containing 256 cases of the considered type (P NP P). For each particular measure, we ordered the list by this measure and plotted the number of the combinations present in [7] found among the top k elements of our extracted list ordered by this particular measure (proportional to recall at top k items).

2.1 Statistical Measures Considered

Various statistical measures to identify lexical associations between words from a corpus have been suggested in literature [3]. We applied the NSP statistical package [1] to obtain the following measures:

- Frequency (Freq),
- Point-wise mutual information (PMI),
- Log-likelihood (LL),
- Pearson measure (χ^2, or Chi-2).

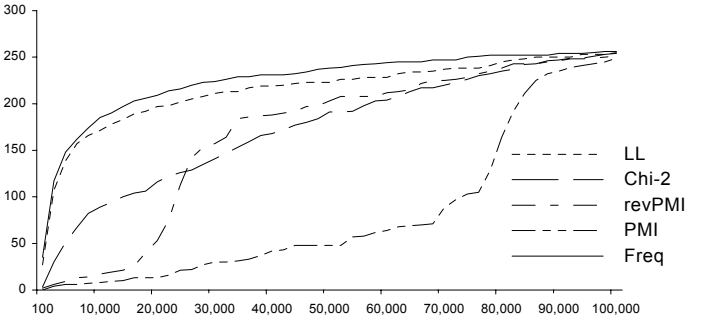


Fig. 1. Results for groups appearing more than 2 times

Since there is no generally accepted definition of the latter three measures for three elements, we applied pair-wise measures to [P NP] and [P] of the whole combination P NP P: e.g., the PMI assigned to *a fin de* was the PMI of the two strings (1) *a fin* and (2) *de*. Evaluation of other possible ways of calculating the dependency between the three elements was left for our future work.

We only considered the strings with frequency greater than 2, since statistical measures of this type are unreliable on sparse data. The results are shown in Figure 1.

2.2 Discussion

The best measure proved to be simple frequency, and log-likelihood shows nearly the same performance. On the first 100 items the precision obtained with frequency measure was 50% (which is 20% recall on the list [7]).

However, manual inspection of the results revealed among the top elements new idiomatic collocations, such as *por la vía de* 'by way of' (literally 'by the way of'), while [7] only contains *por vía de* 'by way of'. Thus, real precision of our method is better than that measured by comparison with the list [7], and the method allows detection of new combinations.

PMI gives very poor results, which is in accordance with the general opinion that PMI is not a good measure of dependency (though it is a good measure of independency) [6]. What is more, PMI seems to have inverse effect: it tends to group the idiomatic examples nearer the end of the list. With this, ordering the list in *reverse* order by PMI (revPMI in Figure 1) gives better results. However, such order is much worse than Freq and LL orders: it groups most of the combinations in question around the positions from 25,000 to 35,000 in the list of 100,000. Pearson measure also shows much worse performance than LL. For a detailed comparison of the log-likelihood and chi-squared statistics, see [8].

A possible explanation for the fact that simple frequency performs in our case better than statistical dependency measures is as follows. A usual collocation is often a new term formed out of existing words, so it is more specific, and of more restricted use, then each of the two words separately. However, in our case the idiomatic P NP P combinations are equivalent to functional words—prepositions—which are of much more frequent use than the corresponding NP used in its literal meaning. Also, [5] suggests that frequent word chains of some specific POSs (such as N P N) tend to be terminological; our study can be considered as a particular case of this method.

On the other hand, [2] reports that idiomatic expressions combine with a more restricted number of neighboring words than free combinations. However, we observed the opposite effect: e.g., *a fin de* (literally 'at end of') in the sense 'in order to' combines with nearly any verb, while in its literal sense nearly only with nouns with semantics of time: *a fin de semana* 'at the end of the week'. The explanation for this fact can be the same as the one suggested in the previous paragraph.

3 Conclusions

Idiomatic word combinations of P NP P type, usually functioning as compound prepositions, have statistical properties distinct from those of usual idiomatic collocations. In particular, they combine with a greater number of words than usual idioms. For their unsupervised learning from a corpus, a simple frequency measure performs better than other statistical dependence measures. In particular, among most frequent P NP P word chains, about 50% are idiomatic. Inspection of the most frequent chains of this type permits to detect idiomatic combinations not present in existing dictionaries.

References

1. Banerjee, S., and Pedersen, T. The Design, Implementation, and Use of the Ngram Statistics Package. In: A. Gelbukh (ed.). Computational Linguistics and Intelligent Text Processing (CICLing-2003), Lecture Notes in Computer Science, N 2588, Springer, 2003; see http://www.d.umn.edu/~tpederse/nsp.html.
2. Degand, L., and Bestgen, Y. Towards automatic retrieval of idioms in French newspaper corpora. *Literary and Linguistic Computing* 18 (3), (2003) 249–259.
3. Evert, S., and Krenn, B. Methods for the Qualitative Evaluation of Lexical Association In *Proc. ACL-2001*, pp 188-195.
4. Galicia-Haro, S. N. Using Electronic Texts for an Annotated Corpus Building. In: *4th Mexican International Conference on Computer Science, ENC-2003*, Mexico, pp. 26–33.
5. Justeson, J. S., S. M. Katz. Technical Terminology: Some Linguistic properties and an algorithm for identification in text. Natural Language Engineering 1:9–27, 1995.
6. Manning, C. D. & Schutze, H. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts, 1999.
7. Nañez Fernández, E. *Diccionario de construcciones sintácticas del español. Preposiciones*. Editorial de la Universidad Autónoma de Madrid, 1995.
8. Rayson, P., Berridge, D., Francis, B. Extending the Cochran rule for the comparison of word frequencies between corpora. In Vol. II of Purnelle G. *et al*. (eds.) *Le poids des mots*: Proc. of 7th International Conf. on Statistical analysis of textual data (JADT 2004), Belgium, 2004, Presses universitaires de Louvain, pp. 926–936.

Evaluating Evaluation Methods for Generation in the Presence of Variation

Amanda Stent, Matthew Marge, and Mohit Singhai

Stony Brook University, Stony Brook, NY 11794, USA
{stent, mmarge, mohit}@cs.sunysb.edu

Abstract. Recent years have seen increasing interest in automatic metrics for the evaluation of generation systems. When a system can generate syntactic variation, automatic evaluation becomes more difficult. In this paper, we compare the performance of several automatic evaluation metrics using a corpus of automatically generated paraphrases. We show that these evaluation metrics can at least partially measure adequacy (similarity in meaning), but are not good measures of fluency (syntactic correctness). We make several proposals for improving the evaluation of generation systems that produce variation.

1 Introduction

The task of surface realization is to select, inflect and order words to communicate the input meaning as completely, clearly and fluently as possible in context. Traditional grammar-based surface realizers, (*e.g.* [1]) focus on the production of at least one high quality output sentence for each input semantic form. By contrast, two-stage surface realizers (e.g. [2, 3]) produce many possible sentences for each input semantic form, but select only one for output. Comparatively little research has been performed on rule-based approaches to the generation of variation (but see [4, 5]). However, recently there has been increasing interest on corpus-based approaches to the generation of paraphrases, or text-to-text generation (e.g. [6, 7, 8, 9, 10]).

Variation in surface realization takes two basic forms: *word choice variation*, and *word order variation*. Example 1 shows both types of variation. Word order variation may entail word choice variation, as in example (1b).

Example 1
(a) I bought tickets for the show on Tuesday.
(b) It was the show on Tuesday for which I bought tickets.
(c) I got tickets for the show on Tuesday.
(d) I bought tickets for the Tuesday show.
(e) On Tuesday I bought tickets for the show.
(f) For the show on Tuesday tickets I bought.

Variation is widely used by humans both in text and dialog. However, not all variations are meaning-preserving. A variation may add meaning possibilities that were not there before, remove meaning possibilities (compare example (1a) with (1d) and (1e)), or

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 341–351, 2005.

otherwise change the meaning of part of a sentence. As example (1f) shows, a variation may also be unclear or syntactically incorrect.

In this paper, we will say that a *valid variation* of a sentence must meet three criteria: *adequacy*, or meaning equivalence; *fluency*, or syntactic correctness; and *readability*, or efficacy in context[1]. A sentence that is ambiguous, that does not express all the input meaning, or that communicates meaning not contained in the input, is not adequate. Even if a sentence is adequate, if it is not syntactically correct or idiomatic it is not fluent. Sentences that are both adequate and fluent may still not be adequate or fluent in a particular context; they are not readable in that context. Pronouns and discourse cues are two constructs that may affect the readability of a sentence.

It may seem odd to separate readability from adequacy and fluency, as context can affect both adequacy and fluency. For example, context can be used for disambiguation. However, very few surface realizers and no automatic evaluation metrics take context into account. The influence of discourse context in surface realization remains an important but poorly understood topic.

Most automatic evaluation metrics for generation and machine translation do not directly evaluate adequacy or fluency; rather, they indirectly evaluate these criteria by comparing the generated, or *candidate*, sentence to one or more human-created *reference* sentences [11, 12, 13, 14]. Where a metric permits the comparison of a candidate sentence to multiple reference sentences [11, 12, 13], only one reference sentence is typically used in evaluation of generation quality ([15, 16], c.f. [14]). Tree-based metrics incorporating the notion of constituency, *e.g.* [14], are not widely used because they require correct parse trees for reference and candidate sentences. No existing automatic evaluation metric evaluates readability.

An interesting question is the extent to which automatic evaluation metrics for text generation systems can be used to evaluate the output quality of generation systems that produce variation. A good automatic evaluation metric could be useful not only for evaluation and comparison of generation systems, but also for distinguishing valid from invalid variations in the output of a two-stage surface realizer. This would permit greater flexibility and efficiency in two-stage surface realization.

Because existing automatic evaluation metrics for generation evaluate by comparison to one or more reference sentences rather than evaluating adequacy and fluency directly, they will punish both word choice and word order variation. However, it may be that they can still distinguish to some extent between valid and invalid sentences, *i.e.* that the noise introduced by variation is not sufficient to drown out the signal of validity. The question we address in this paper is whether existing automatic evaluation metrics for text generation can accurately evaluate the adequacy and fluency of generated sentences when variation is permitted. Sections 2 and 3 describe the data and metrics we used. Section 4 describes an experiment we conducted comparing the performance of these metrics to human judgments of adequacy and fluency. Section 5 discusses the implications of our results and our proposals for evaluation of generation systems that permit variation. We conclude with some ideas for future work.

[1] These are very similar to criteria used in machine translation evaluations, *e.g.* [11].

2 Data

The data we used for this study consists of a set of 118 automatically-generated paraphrase sentences made available by Barzilay and Lee[2]. Barzilay and Lee employ a corpus-based approach to paraphrase generation [6]. Sentences in a corpus are grouped by similarity, and then the *multiple sequence alignment* of each group of sentences is computed. The multiple sequence alignment of a group of sentences is a word lattice capturing places where the sentences are the same and places where they differ; it is a compact representation of possible variations of a sentence. A paraphrase is generated for a new input sentence by aligning the input sentence with one of the word lattices and then choosing an alternative path through that lattice.

The data we used includes sentences produced by Barzilay and Lee's baseline system (50%) and sentences produced by Barzilay and Lee's multiple sequence alignment based (MSA) system (50%). The baseline system simply replaces words in a sentence with one of their WordNet synonyms, at a rate proportional to the word replacement rate of the MSA system for that sentence. Therefore, the baseline system includes word choice variation only (example 2), whereas the MSA system includes both word choice and word order variation (example 3).

Example 2
(a) Another person was also seriously wounded in the attack.
(b) Another individual was also seriously wounded in the attack.

Example 3
(a) A suicide bomber blew himself up at a bus stop east of Tel Aviv on Thursday, killing himself and wounding five bystanders, one of them seriously, police and paramedics said.
(b) A suicide bomber killed himself and wounded five, when he blew himself up at a bus stop east of Tel Aviv on Thursday.

The variations produced using multiple sequence alignment are of very high quality and are typically highly fluent. However, because there is no explicit representation of the meaning of the input sentence, words chosen may occasionally carry connotations not carried by the words they replace, and sometimes words are included or removed that alter the meaning of the sentence (*e.g.* example 3).

3 Evaluation Metrics

We used five evaluation metrics for this study: NIST simple string accuracy (SSA) [14], the BLEU and NIST n-gram co-occurrence metrics [12, 11], Melamed's F measure [13], and latent semantic analysis (LSA) [17]. Only SSA and BLEU have previously been used to evaluate the output of generation systems; SSA, BLEU, NIST and the F measure are designed for the evaluation of machine translation output. As Table 1 shows, all these metrics evaluate the fluency and adequacy of generated candidate sentences indirectly

[2] http://www.cs.cornell.edu/Info/Projects/NLP/statpar.html

Table 1. Evaluation metrics

Metric	SSA	NIST n-gram, BLEU	F measure	LSA
Means of measuring fluency	Comparison to reference sentence	Comparison to reference sentences – matching n-grams	Comparison to reference sentences – longest matching substrings	None
Means of measuring adequacy	Comparison to reference sentence	Comparison to reference sentences	Comparison to reference sentences	Comparison using word co-occurrence frequencies learned from corpus
Means of measuring readability	Comparison to reference sentence(s) from same context*	Comparison to reference sentences from same context*	Comparison to reference sentences from same context*	None
Punishes length differences?	Yes (punishes deletions, insertions)	Yes (weights)	Yes (weights)	Not explicitly

by comparison with one or more reference sentences. Table 1 also shows how one might use these metrics to evaluate readability, although we are not aware of any research that uses this approach.

Simple String Accuracy. The NIST simple string accuracy (SSA) metric scores a candidate sentence by tallying the number of substitutions, insertions, and deletions necessary to convert the reference sentence to the candidate sentence and dividing by the length of the candidate sentence. SSA has been used to evaluate the output of SURGE [16] and FERGUS [14].

BLEU. IBM's BLEU metric, designed for evaluating machine translation quality, scores candidate sentences by counting the number of n-gram matches between candidate and reference sentences. It also punishes differences in length between candidate and reference sentences. The BLEU evaluation metric has been shown to correlate highly with human judgments [12]. The BLEU metric has been used to evaluate the output of HALogen [15].

NIST. The NIST n-gram based evaluation metric, also designed for evaluating machine translation quality, differs from the BLEU metric in three ways. First, The arithmetic mean of co-occurrences is used instead of the geometric mean. Second, n-grams that occur less frequently are weighted more highly than those that occur more frequently. Third, there is a slightly different length penalty. These differences have been shown to lead to a higher correlation with human judgments than BLEU has [11]. Unlike the other metrics, NIST n-gram scores are not in the range $[0, 1]$. We include it primarily for comparison with BLEU.

F Measure. This metric was developed by Melamed et. al. for evaluating machine translation quality [13]. It is designed to eliminate the "double counting" done by n-gram based metrics such as the NIST and BLEU n-gram based metrics (which penalize

the same word insertion, deletion or movement as it occurs in a unigram, a bigram, etc.). It uses two scores, precision and recall, computed separately for each candidate sentence. Both precision and recall are defined in terms of the maximum match size, which is the weighted sum of the lengths of the longest matching text blocks between candidate and reference sentences. Precision is the maximum match size divided by the length of the candidate sentence; recall is the maximum match size divided by the length of the reference sentence. The maximum match size can be adjusted to weight longer matches more or less heavily by using a different exponent; for this data, 1 was the best exponent. Studies by Melamed et. al. show a high correlation between this metric and human judgments of translation quality [13]. This metric punishes variation in sentence length less than BLEU and NIST, so we hypothesized that it would be more closely correlated with human judgments for variation generation.

Latent Semantic Analysis. Latent semantic analysis (LSA) computes the semantic similarity of two texts by measuring the semantic similarities of the words they contain [17]. Semantic similarity is computed by means of word co-occurrence counts obtained from a large corpus. LSA differs from the other metrics we used in two ways. First, it treats each sentence as a bag of words (compares sentences without regard to word order). Second, it uses word co-occurrence statistics learned from a large corpus to compute the semantic similarity of words. Therefore, we hypothesized that LSA would be good at evaluating adequacy in the presence of variation, although obviously it cannot serve as a measure of fluency.

4 Procedure

As Table 1 shows, most automatic evaluation metrics compare generated sentences to one or more reference sentences. This means that automatic evaluation metrics will tend to punish word choice and word order variation. The questions addressed in this experiment are: a) Are automatic evaluation metrics sufficiently robust to variation to distinguish between sentences that are valid (adequate and fluent) and those that are not?; and b) What is the relative impact of word choice and word order variation on the performance of these metrics? Our procedure was to compare human judgments of adequacy and fluency to the scores of the five selected evaluation metrics for the two sets of paraphrases provided by Barzilay and Lee.

We had three human judges evaluate the paraphrase pairs provided by Barzilay and Lee. In the following discussion, the reference sentence is the original (human-created) sentence, and the candidate sentence is an output from one of the two systems used by Barzilay and Lee.

Each judge answered two questions for each reference/candidate sentence pair, one pertaining to adequacy and one to fluency. For each sentence pair, the reference sentence is sentence A and the candidate sentence is sentence B. In our evaluation judges did not see the sentences in a larger discourse context, since the evaluation metrics do not consider discourse context, so there is no evaluation of readability. The questions the judges were asked are:

1. How much of the meaning expressed in Sentence A is also expressed in Sentence B?
__All __Most __Half __Some __None
2. How do you judge the fluency of Sentence B? It is
__Flawless __Good __Adequate __Poor __Incomprehensible

The paraphrases were rated very highly in general. In the experiment reported below, the judges' ratings are averaged and normalized to the range [0,1]. The mean rating for adequacy was 4 (st. dev. 0.66, min. 2, max. 5), and for fluency was 4.13 (st. dev. 0.72, min. 2.33, max. 5).

All paraphrases were also evaluated using the five automatic evaluation metrics described in the previous section. We then computed the correlation between the human evaluations of adequacy and fluency and the scores for each evaluation metric. We used the Spearman rank coefficient of correlation, which is a measure of the strength of the linear relationship between two variables. We used Spearman rather than the Pearson coefficient because this data is not normally distributed.

5 Results

Our comparison of these evaluation metrics is shown in Table 2. All correlations are significant at $p < .01$ unless italicized. Correlations greater than 0.67 indicate strong relationships, while correlations between 0.34 and 0.66 indicate some relationship.

As one would expect, the automatic evaluation metrics are highly positively correlated with each other. Also as one would expect, the automatic evaluation metrics are positively correlated with the length of the candidate sentence.

There is no significant correlation between human judgments of adequacy and human judgments of fluency, indicating that the judges considered these two dimensions separately. Because the judges could always see both the candidate and the reference sentences, they may have tended to make slightly higher judgments of adequacy than they would have otherwise. On the other hand, the paraphrases are of very high quality in general, even when meaning is not completely preserved. It should be noted that the median difference in sentence length between source and target sentences was 2 words;

Table 2. Correlation between human judgments of meaning preservation and syntactic accuracy and automatic evaluation metrics

	BLEU	**NIST**	**SSA**	**F**	**LSA**	**Adequacy**	**Fluency**
BLEU	1.00						
NIST	0.910	1.00					
SSA	0.894	0.863	1.00				
F	0.927	0.900	0.955	1.00			
LSA	0.725	0.727	0.742	0.795	1.00		
Adequacy	0.388	0.421	0.412	0.457	0.467	1.00	
Fluency	-0.492	-0.563	-0.400	-0.412	-0.290	*-0.032*	1.00
Length candidate	0.540	0.722	0.426	0.467	0.421	*0.169*	-0.374

Table 3. Baseline vs. Multiple Sequence Alignment

System	BLEU	NIST	SSA	F	LSA	Adequacy	Fluency
Baseline	.753	4.156	.864	.888	.954	*.833*	.756
MSA	.290	1.945	.423	.530	.845	*.770*	.897

i.e. generated sentences were not usually summaries of the input sentences, so typically most of the meaning was preserved. There were cases where information was added, but it was typically attribution information (*e.g. police said*).

Adequacy. There are positive, but not strong, correlations between the scores of the automatic evaluation metrics and human judgments of adequacy. We conclude that *these automatic evaluation metrics are adequate, but not good, evaluators of adequacy.*

Fluency. There are negative correlations between the scores of the automatic evaluation metrics and human judgments of fluency. This is weakest in the case of LSA, which does not consider word order. We conclude that (at least in the presence of variation) *these automatic evaluation metrics are poor evaluators of fluency.*

Impact of Word Order Variation. Recall that Barzilay and Lee's baseline system performs word choice variation only, while their MSA system performs both word choice and word order variation. Furthermore, the frequency of word choice variation was held constant across both systems. Therefore, this data set is useful for evaluating the relative impact of word order variation on automatic evaluation scores.

The means of the scores for each system are shown in Table 3. A paired t-test showed that these differences are significant at $p < .01$ for all except human adequacy judgments. The automatic evaluation metrics all scored the baseline system *higher* than the MSA system. In contrast, the human judges rated the fluency of the MSA system output higher than that of the baseline system. Mostly, this is because the MSA system can make decisions about word choice variation based on the context in which the word appears while the baseline system cannot; however, sometimes the MSA system produced paraphrases that were clearly more readable than the input sentence. The human judges rated the output of both systems highly for adequacy. We conclude that, because these evaluation metrics punish word order (and word choice) variation in ways that do not distinguish between valid and invalid variations, *these automatic evaluation metrics are not adequate for the task of evaluating variation generation.*

6 Discussion

The results of the experiment in the previous section demonstrate that existing automatic evaluation metrics are inadequate for evaluating the output of generation systems that produce variation. In this section, we discuss four proposals for improving the automatic evaluation of generation systems that produce variation.

Proposal 1: Multiple Reference Sentences. Several of the metrics used in this paper (*e.g.* [11, 12, 13]) permit multiple reference sentences. Where it is possible to find multiple

reference sentences covering the range of possible variants on a sentence, these metrics might prove more closely correlated with human judgments of adequacy and fluency. We therefore recommend that *automatic evaluations of the quality of surface realizers should be conducted using multiple reference sentences*.

This recommendation comes with two caveats. First, it can be time-consuming to find multiple reference sentences for each sentence in a test set for a particular domain, and an out-of-domain test set may not provide an honest accounting of the quality of the surface realizer output. Second, even if multiple reference sentences are provided, two problems remain: a) it is highly likely that some valid variations will not be included, and b) it is possible for two variations of parts of a sentence to be fluent and adequate separately, but not in combination, as example 4 shows:

Example 4

(a) She killed her with a gunshot to the head.
(b) She shot her in the head.
→ (c) She shot her to the head with a gunshot.

Proposal 2: Shallow Models of Constituency. As Callaway points out in [16], most automatic evaluation metrics for generation do not contain models of syntactic constituency. This is a serious drawback when it comes to evaluating generation systems that permit variation. In particular, the lack of a model of constituency means that automatic evaluation metrics cannot distinguish between valid movement such as that in example 1b and invalid movement (for example, *I bought tickets on Tuesday the show for.*).

There are three possible solutions to this problem: use a parser to evaluate the fluency of generated sentences, use a grammar checker to evaluate fluency, or use tree-based evaluation metrics. Unfortunately, since parsers are descriptive rather than prescriptive models of language, they are not suitable for evaluation purposes. We tried parsing a set of fluent and disfluent permutations of the words in example (1a) using the Collins and Charniak parsers, and obtained parses for all of them. Furthermore, the probabilities assigned to some of the very disfluent parses were higher than those assigned to some of the less disfluent ones.

Similarly, grammar checkers do not currently make the sort of fine-grained syntactic and semantic judgments needed for automatic evaluation of generation systems. We ran a set of permutations of the words in example (1a) through a number of grammar checkers, including the Microsoft Word, Grammar Expert Plus, Conexor True Styler, Grammar Station, Grammar Slammer, WGrammar and Grammatica grammar checkers. None of the errors in the sentences were identified.

Tree-based evaluation metrics (*e.g.* [14]), while not encoding an explicit model of constituency, can be indirect models of constituency. The task then becomes one of annotating reference and candidate sentences with syntax trees. For evaluation of general-purpose surface realizers, a treebank can be used; for evaluation of domain-specific surface realizers or selecting a variation from a two-stage surface realizer at run time, this is not currently possible.

The output of a chunker is a shallow model of constituency, is easier to obtain than a full parse tree, and may serve as an approximation to a parse tree for evaluation purposes. To test this, we chunked the sentences in our data using the ILK chunker

[18], which chunks noun phrases and prepositional phrases. We used a chunk-based version of simple string accuracy to evaluate the paraphrases. This metric is somewhat correlated with human judgments of adequacy (0.461) and negatively correlated with human judgments of fluency (-0.383). The disagreements are due to two factors. The most frequent is word choice variation; there are also some generated sentences that are much shorter than the original. Performance could perhaps be improved with the inclusion of automatic semantic role labeling.

This method gives the second highest correlation with human judgments of adequacy that we have observed. We therefore recommend that *tree- or chunk-based metrics should be preferred over string-based ones for evaluating adequacy*. However, these metrics do not show promise for evaluating fluency in the absence of a model of word choice variation, or at least the use of multiple reference sentences.

Proposal 3: Models of Semantic Similarity. Existing automatic evaluation methods for generation do not incorporate any measure of semantic similarity other than string equality on words. This affects the evaluation of systems that permit word choice variation, and also those that permit word order variation, since some word order variations (e.g. the use of passive voice) affect word choice.

We have explored two possible solutions to this problem. One can extend existing automatic evaluation metrics like Melamed's F measure using a resource like WordNet, so that the replacement of a word with one of its synonyms is not penalized. This method could not be used for the baseline system data which was created using WordNet. However, we applied this method to the MSA sentences; it performed worst of all the automatic evaluation metrics because it was far too forgiving.

The problem of word choice variation also motivated our decision to include LSA in our experiment. LSA performed well compared to other evaluation metrics. We therefore recommend that *a measure of semantic similarity (e.g. LSA) should be incorporated in automatic evaluation metrics for systems that permit word choice variation*. However, LSA works best when the items being compared are of similar length and are not too short; a single phrase or even a sentence may be too short. We are currently exploring ways to combine semantic similarity and chunk-based metrics.

Word choice variation presents significant difficulty for automatic evaluation of surface realizers, and requires considerable further research.

Proposal 4: Separating Different Features. Recall that our definition of a valid sentence is one that is fluent, adequate and readable. As the experiment in this paper shows, evaluation metrics that are adequate for evaluating adequacy may fail at evaluating fluency and readability.

We propose that the evaluation of surface realization quality should involve more careful analysis than has been previously used, particularly if the surface realizer permits word choice or word order variation. In particular, we recommend that *researchers should evaluate the adequacy, fluency and readability of generator output separately* until there is a metric that can evaluate all three together with high accuracy. Existing string- or tree-based metrics can be used to evaluate adequacy. We recommend the use of multiple reference sentences and tree- or chunk-based metrics where possible. Existing metrics cannot be used to evaluate fluency (at least where there is only one reference sentence), and there is no existing automatic metric that can evaluate readability.

Of course, evaluation of the quality of surface realization output should usually be combined with evaluation of coverage (as in [16]).

7 Conclusions

In this paper, we compared several automatic evaluation metrics, some of which have not previously been used to evaluate the quality of generation system output. We looked at the particular question of whether these automatic evaluation metrics are useful for evaluating the adequacy and fluency of the output of surface realizers that permit variation. We found that these automatic evaluation metrics are not adequate for the task of evaluating fluency, and are only barely adequate for evaluating adequacy, in the context of variation generation. We made several proposals for overcoming this problem, including: use multiple reference sentences, use tree- or chunk-based metrics that give better models of constituent movement, and evaluate adequacy, fluency and readability separately.

This experiment shows that, when selecting an evaluation metric, it is important to consider whether the metric can evaluate the phenomena that the system was designed to handle. There is no single evaluation metric that will work for all surface realizers, across domain, task and discourse type. This makes it harder to compare different surface realizers, but if they perform different tasks it is not clear what use a comparison would be in any case. It is crucial to have a clear understanding of the focus of both surface realizer and evaluation metric before evaluation.

In future work, we plan to explore whether it is possible to use automatic clustering approaches such as those used by [6], together with a Web search engine, to automatically locate multiple reference sentences given a single reference sentence. We also plan to explore other means for automatically evaluating the fluency and readability of generated sentences.

Acknowledgments

We thank Dr. Lee and Dr. Barzilay for sharing their data, our judges for their help, and NIST and Dr. Melamed for providing evaluation software. We would like to thank the anonymous reviewers of this paper for their comments. This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA), through the Department of the Interior, NBC, Acquisition Services Division, under Contract No. NBCHD030010. The content of the information in this publication does not necessarily reflect the position or the policy of the US Government, and no official endorsement should be inferred.

References

1. Elhadad, M., Robin, J.: Controlling content realization with functional unification grammar. In: Proceedings of the 6th International Workshop on Natural Language Generation. (1992)
2. Bangalore, S., Rambow, O.: Exploiting a probabilistic hierarchical model for generation. In: Proceedings of COLING 2000. (2000)

3. Langkilde, I.: Forest-based statistical sentence generation. In: Proceedings of ANLP 2000. (2000)
4. McKeown, K.: Paraphrasing using given and new information in a question-answer system. In: Proceedings of ACL 1979. (1979)
5. Murata, M., Isahara, H.: Universal model for paraphrasing – using transformation based on a defined criteria. In: Proccedings of the NLPRS 2001 workshop on Automatic Paraphrasing: Theories and Applications. (2001)
6. Barzilay, R., Lee, L.: Learning to paraphrase: An unsupervised approach using multiple-sequence alignment. In: Proceedings of HLT-NAACL 2003. (2003)
7. Barzilay, R., K.McKeown: Extracting paraphrases from a parallel corpus. In: Proceedings of ACL/EACL 2001. (2001)
8. Ibrahim, A., Katz, B., Lin, J.: Extracting structural paraphrases from aligned corpora. In: Proceedings of the 2nd International Workshop on Paraphrasing. (2003)
9. Pang, B., Knight, K., Marcu, D.: Syntax-based alignment of multiple translations: Extracting paraphrases and generating new sentences. In: Proceedings of HLT-NAACL 2003. (2003)
10. Shinyama, Y., Sekine, S., Sudo, K., Grishman, R.: Automatic paraphrase acquisition from news articles. In: Proceedings of HLT-NAACL 2002. (2002)
11. NIST: Automatic evaluation of machine translation quality using n-gram co-occurrence statistics (2002)
12. Papenini, K., Roukos, S., Ward, T., Zhu, W.: BLEU: A method for automatic evaluation of machine translation. Technical Report RC22176 (W0109-022), Thomas J. Watson Research Center, IBM Research Division (2001)
13. Turian, J., Shen, L., Melamed, I.D.: Evalaution of machine translation and its evaluation. In: Proceedings of MT Summit IX. (2003)
14. Bangalore, S., Rambow, O., Whittaker, S.: Evaluation metrics for generation. In: Proceedings of INLG 2000. (2000)
15. Langkilde, I.: An empirical verification of coverage and correctness for a general-purpose sentence generator. In: Proceedings of INLG 2002. (2002)
16. Callaway, C.: Evaluating coverage for large symbolic NLG grammars. In: Proceedings of IJCAI 2003. (2003)
17. Deerwester, S., Dumais, S., Furnas, G., Landauer, T., Harshman, R.: Indexing by latent semantic analysis. Journal of the American Society for Information Science **41** (1990)
18. Daelemans, W., Buchholz, S., Veenstra, J.: Memory-based shallow parsing. In: Proceedings of CoNLL-99. (1999)

Reconciling Parameterization, Configurability and Optimality in Natural Language Generation via Multiparadigm Programming

Jorge Marques Pelizzoni and Maria das Graças Volpe Nunes

Universidade de São Paulo, Instituto de Ciências Matemáticas e de Computação
Av. do Trabalhador São Carlense, 400. CEP 13560-970. São Carlos – SP – Brasil
{jorgemp, gracan}@icmc.usp.br

Abstract. This paper focuses on how multiparadigm – namely, constraint, object-oriented and higher-order – programming can be drawn upon not only to specify multiparameterized linguistic realization engines but also and above all to rationalize their configuration into full-fledged generation modules for specific language-application pairs. We describe Manati, one such engine whose instantiations render linguistic form to conceptual/semantic directed hypergraphs, and point out how its constraint-based concurrent architecture entails collaboration and interleaving so as to allow the definition and optimization of global quality measures.

1 Introduction

Natural Language Generation (NLG) refers to rendering linguistic form to input in a non-linguistic representation. As pointed out by e.g. Reiter & Dale [13], Cahill & Reape [2], Paiva [11], this can be a very complex task involving processing both linguistic (e.g. lexicalization, aggregation and referring expression generation) and otherwise (content selection and layout planning). In this paper, we are exclusively concerned with the linguistic aspect of generation, herein referred to as *linguistic realization*.

A range of linguistic realization work has been reported on so far in the literature varying in scope and depth. Nonetheless, it is a rare work that focuses on configurability issues, especially in a multiparameterization scenario. By *configurability* we mean ease of configuration, or rather, instantiation of required parameters in a disciplined, manageable, friendly manner. *Parameter*, in turn, refers to any blank whatsoever that should be filled in so as to make a generic solution into a full-fledged linguistic realization component. Possible parameters are grammars, lexicons, strategies, heuristics, etc.

In fact, there is far more usual to be material either detailing an isolated solution (an instantiation of a single parameter, e.g. Eddy [7]) or sketching a complex solution (i.e. a multiparameterized one, e.g. Stone & Doran [16]) – in either case, usually with not much regard to the discipline of instantiation. This communication takes a complementary path and attempts to focus on (i) the case of abstracting away a reusable NLG solution by (ii) (multi)parameterizing it in (iii) a hopefully highly configurable

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 352 – 356, 2005.

fashion on the basis of (iv) multiparadigm – namely, constraint, object-oriented and higher-order – programming. An additional concern is to give evidence that the concurrent constraint-based architecture thus obtained is highly collaborative and prone to yield globally optimal results by enabling interparameter synergy.

Most of this paper is dedicated to demonstrating how the above principles guided the design of Manati, a linguistic realization engine that has originally and so far been developed as a hopefully reusable component in a Portuguese-Brazilian Sign Language (LIBRAS) interlingua-based semiautomatic translation project.

The paper proceeds as follows. Section 2 motivates our linguistic realization effort by describing the requirements of the translation application that gives rise to Manati. Section 3 sketches Manati proper and shows how it addresses the three-way tug-of-war between parameterization, configurability and optimality and meets the requirements in Section 2. Finally, Section 4 draws conclusions and hints at future enhancements.

2 Context: Interlingua-Based Semiautomatic Cross-Modal Translation

This paper reports on partial results of a comprehensive, currently ongoing project in machine translation named PUL∅ (Portuguese-UNL-LIST deOralizer), whose aim is to reduce the cost and turnaround of translating written Portuguese into "spoken"[1] LIBRAS, the Brazilian Sign Language. PUL∅ is not intended to produce actual LIBRAS speech, but a script thereof – LIST (LIBRAS Script for Translation) – to feed an eventual speech synthesizer.

It might be assumed that **cross-modal translation** – i.e. bridging the gap between oral and sign languages – should have no special status *in principle*, were one to avail of sufficient formal descriptions of the languages involved. Even so, it is often the case that:

- sign languages are much less understood, thus lacking in description;
- one cannot rely so much on isomorphism as in intra-modal translation. For a start, one-to-one mappings between sentences is much less likely in cross-modal translation. Second, different text planning strategies arise with mode-specific referential devices [1][15]. Third, the iconic/mime substratum of sign languages usually pulverizes semantic fields, spoiling lots of usually valid direct translations;
- sign languages are usually not written, in spite of the occasional availability of a writing system. This entails serious further difficulties, as a translation project can hardly be validated without a speech synthesis/recognition module.

Finally, PUL∅ follows the semantic transfer approach to machine translation (Hutchings & Sommers [8]), using the UNL (Universal Networking Language [10][17]) as an **interlingua** or s**emantic-content representation formalism**. The UNL attempts to capture sentence meaning by means of directed hypergraphs, where (i) basic nodes refer to instances of basic "universal concepts", or Universal Words

[1] The words *spoken, speech,* etc. are employed here especially as opposed to *written*, *writing*, etc. More specifically, those words should not be regarded as necessarily implying orality.

(UWs), (ii) labeled directed edges state binary relations between node referents, (iii) hypernodes provide for recursion and nesting, or simply "complex concepts" and (iv) node attributes allow for concept "modalization".

3 Case Study: Manati

Manati is the linguistic realization engine all UNL-LIST conversion in PUL∅ is based on. In other words, PUL∅ includes a configuration of Manati, i.e. a module obtained by fixing Manati's parameters. The engine is fully implemented in Oz (http://www.mozart-oz.org [14][18]) and heavily draws upon the expressiveness and elegant, seamless multiparadigm integration of this language to meet its requirements. Like Koller & Striegnitz's generation work [9], it builds upon work by Duchier [3][4] & Debusmann [6], but is fundamentally distinct from the former, which strictly focuses on taming flat semantics, a non-issue here. For space reasons, a very shallow description follows; for further information, please refer to [12].

3.1 Parameters

Manati currently allows the rationalized configuration of ten *orthogonal* parameters in that independently and modularly defined, namely:

a) **input formalism**, which, even though restricted to hypergraph types, is free to accept any open set of UWs (node labels) and closed set of relations (edge labels) and attributes;
b) **morphosyntax:** each part of speech (POS) in the target language must be defined as a record with arity *{avm, constr}*, where feature *avm* is an attribute-value matrix (AVM) type, and *constr*, a constraint on instances of *avm*;
c) **syntactic mapping:** a specific mapper class hierarchy must be provided in order exclusively to specify the mapping of UNL (hyper)graphs onto syntactic dependency trees [3] in the target language. Roughly speaking, mappers simply convert (i) semantic nodes into lexemes (classes of lexical items) and (ii) semantic relations into syntactic roles; or, in NLG jargon, they are responsible for *lexical choice* and *aggregation*. It is worth noticing that mappers are not interested either in morphosyntatic constraints, such as agreement, or in final linear ordering of morphemes;
d) **mapping preconditions:** in order to optimize resource usage during search, part of if not all precondition checking in mappers can optionally be delegated to a specific class hierarchy. Such so-called *precond* classes are associated with mappers by lexicon data;
e) **governor-governee constraints:** a specific class hierarchy must be provided in order exclusively to tell morphosyntactic constraints on each pair of syntactically related target nodes (i.e. words or morphemes). Such so-called *gamma* classes are associated with mappers by lexicon data and have methods of the signature *Role(Parent.feats Child.feats)* invoked for each syntactic relation *Role* their corresponding mappers establish between any target nodes *Parent* (governor) and *Child* (governee);

f) **linear precedence:** a specific class hierarchy must be provided in order exclusively to determine the final ordering of target nodes and carry out whatever further tasks that might occasionally be required on mapper completion, when all direct child nodes are accessible – though not as yet fully determined – for e.g. telling further constraints. Such so-called *finishUp* classes tackle linear precedence by telling constraints relating target nodes to each of their children and children to each other;
g) any number of **oracles** – e.g. user prompts, knowledge bases, etc. – to resort to at virtually any generation stage;
h) **lexicon:** Manati's lexicon is more of a **transfer rule base**, each of whose entries is a tuple *(UW, TransList, POS, Precond, Mapper, Gamma, FinishUp)*, where *UW* is a source node label; *TransList*, a character string list of possible target language translations; *POS*, the part of speech of the elements of *TransList*; and *Precond*, *Mapper, Gamma* and *FinishUp*, classes of the homonymous types;
i) **output formalism,** i.e. how the resulting syntactic trees are to be printed out. This is highly configurable ranging smoothly from raw lists of target language words to fully structured trees by means of user-defined bracketing. Words and bracketed groups may be associated with arbitrary *Output AVMs* (OAVMs) created by *FinishUp* classes. OAVMs may be useful to add syntactic and prosodic annotations (as required by PUL∅) or even to output morphologic features, leaving full inflection of words to dedicated modules and thus downsizing the lexicon.

4 Conclusions and Future Work

We have presented Manati, a linguistic realization engine that attempts to equate the tension between parameterization, configurability and optimality. Manati is currently being configured to generate the Brazilian Sign Language and shall be evaluated against other generation engines in the near future. Scheduled further work on Manati includes full coverage of generation tasks – e.g. content selection and referring expression generation – and support to configurability of additional or alternative optimality measures goals and optimum search strategies.

References

1. Brito, L. F. Por uma Gramática de Línguas de Sinais. Tempo Brasileiro Ed., Departamento de Lingüística e Filologia, Universidade Federal do Rio de Janeiro, 1995.
2. Cahill, L. and Reape, M. *Component tasks in applied NLG systems*. Technical Report ITRI-99-05, Information Technology Research Institute (ITRI), University of Brighton, 1998. http://www.itri.brighton.ac.uk/projects/rags.
3. Duchier, D. Configuration of labeled trees under lexicalized constraints and principles, *Journal of Language and Computation*, 2002.
4. Duchier, D. Axiomatizing dependency parsing using set constraints. In *Proceedings of the 6th Meeting on the Mathematics of Language*, USA, 1999.
5. Duchier, D., Gardent C., and Niehren J. *Concurrent Constraint Programming in Oz for Natural Language Generation*. http://www.ps.uni-sb.de/~niehren/Web/Vorlesungen/Oz-NL-SS01/vorlesung (accessed on Aug. 2004).

6. Duchier, D. and Debusmann, R. Topological dependency trees: A constraint-based account of linear precedence. In *Proceedings of the Association for Computational Linguistics (ACL)*, France, 2001.
7. Eddy, B. Toward balancing conciseness, readability and salience: an integrated architecture. In *Proceedings of the International Natural Language Generation Conference (INLG'02)*, 2002.
8. Hutchins, W. J. and Somers, H. L. An introduction to Machine Translation, Academic Press, San Diego (CA), 1992.
9. Koller, A. and Striegnitz, K. Generation as Dependency Parsing. In *Proceedings of the Association for Computational Linguistics (ACL)*, 2002, 17-24.
10. Martins, R. T. *A Língua Nova do Imperador*. Ph.D. Thesis, Instituto de Estudos da Linguagem, Universidade Estadual de Campinas (UNICAMP), 2004.
11. Paiva, D. *A survey of applied natural language generation systems*. Technical Report ITRI-98-03, Information Technology Research Institute (ITRI), University of Brighton, 1998. http://www.itri.brighton.ac.uk/techreports.
12. Pelizzoni, J. M. and Nunes, M. G. V. Flexibility, Configurability and Optimality in UNL Deconversion via Multiparadigm Programming. In: I. Boguslavsky, J. Cardeñosa, A. Gelbukh. *UNL, other Interlinguas and their Applications*. Research on Computer Science, 2004, to appear. http://www.CICLing.org/2005/UNL.htm.
13. Reiter, E. and Dale, R. *Building Natural Language Generation Systems*. Cambridge University Press, 2000.
14. Schulte, C. *Programming Constraint Services: High-Level Programming of Standard and New Constraint Services*, Lecture Notes in Computer Science Series, Springer-Verlag, 2002.
15. Speers, d'A. L. *Representation of American Sign Language for Machine Translation.* PhD. dissertation, Graduate School of Arts and Sciences, Georgetown University, 2001.
16. Stone, M. and Doran, C. Sentence planning as description using tree-adjoining grammar. In *Proceedings of the Association for Computational Linguistics*, 1997, 198-205
17. Uchida, H., Zhu, M., and Della Santa, T. *UNL: A Gift for a Millennium.* Institute of Advanced Studies, University of the United Nations, 1999. http://www.undl.org/publications.
18. Van Roy, P. and Haridi, S. *Concepts, Techniques, and Models of Computer Programming*, MIT Press, 2004.

Message Automata for Messages with Variants, and Methods for Their Translation

Christian Boitet

GETA, laboratoire CLIPS,
385 rue de la bibliothèque - BP 53, 38041 Grenoble Cedex 9 - France
Christian.Boitet@

Abstract. In messages with variables and variants, such as "the imag.fr $n-[st|nd|rd|th] trial was successful, and the $p file[|s] found [is|are] satisfactory.", variable types are specific (cardinal, ordinal, politeness…) and induce different "variant cases" in each language. Controlled loop-free FSAs, called here "message automata" (MAs), are proposed to model such messages. To translate a MA, one generates an instance of it for each possible variant in the target language. After translation, the values used in the instances are discarded and a target language MA is built by factorization (not classical minimization), using an original dynamic programming algorithm. A library for handling catalogues of MAs, GetAMsg, has been implemented in C, and can be used from many usual programming languages. A still speculative idea is to use a UNL graph conform to the official specifications, but with some special conventions, to represent a message with variables, and generate the language-specific MAs from it.

1 Introduction

Software of all kinds needs to be localized in dozens of languages because of the increasing availability of PCs and multilinguality of the web. Elements to be translated are often dynamic, because they depend on the values of some variables. We will call them "variables with variables and variants" (MW). This includes not only "classical" short messages, such as "$n files have been processed", which give rise to singular/plural variants in English and other languages, but also more personalized and often longer messages, such as paragraphs in commercial offers, or short texts in games, or sentences from online documentation, where several variables and different kinds of variants (direct/polite, masculine/feminine/neutral, calendar…) can appear.

MWs are linguistically interesting, because variable types are specific (cardinal, ordinal, hour of day…) and induce different "variable cases" (classical, not grammatical cases) in each language. For instance, there are 3 cases in Russian for cardinals (1 год, 2 года, 5 лет, 21 год…), and 3 different cases (singular, dual, plural) in Arabic.

We will call *format* a string pattern like the example above, which may contain variables such as $n, and generate a potentially infinite set of message instances after variable substitution. A format may also contain formatting commands such as "%3i" (3 character place holder for an integer) or "\t" (tab) in C.

We will say that a message has *variants* if it corresponds to different formats depending on the values of some variables, which may appear or not appear in these

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 357 – 376, 2005.

formats. In general, a message is a *generator* of formats, which in turn generate actual strings. In almost all programming environments, it is impossible to write messages with variants without writing language-specific code such as printf("%d file%s been processed", n, ((n>1)?"s have":" has")).

Different kinds of variants may be caused by different types of variables, such as gender, politeness, ordinals, hour, etc., in ways differing from language to language. More details are given below. Here is an example in English, French and Russian, with 2 integer variables, the first used as an ordinal and the second as a cardinal.

```
"The $n-[st|nd|rd|th] trial was successful, and the $p
file[|s] found [is|are] satisfactory."
"Le $n-[er|ième] essai a réussi, et le[|s $p] fichier[|s]
trouvé[|s] [est|sont] satisfaisant[|s]."
"$n-ая попытка удалась, и найденны[й|е $p] файл[|а|ов]
удовлетворител[ен|ьны]."[1]
```

It is possible that 2 variables do not occur in the same order in two languages, and/or not with the same number of occurrences[2]. Also, a variable such as a family name can occur in a language and not in the other ("Good morning $title", "Guten Tag $title $name"). Finally, variants due to some variable (such as $politeness) may appear even if the value of the variable does not appear in any variant.

MWs pose two interesting computational problems: (1) considering that variants are not handled well by current message-related libraries even in English, how to model all types of variants, for all languages? and (2) how to translate them?

We propose to use controlled loop-free FSAs, called here "message automata" (MA), to model messages (with or without variants). To translate a message, one generates an instance of it for each possible variant in the target language. After translation, the particular values used in the instances are discarded and the resulting formats are factorized into a target language MA. As this factorization cannot be obtained by a classical minimization, we use a new dynamic programming algorithm. A library for handling catalogues of MAs representing MWs, GetAMsg, has been implemented in C by VO-TRUNG Hung, and can be used from many usual programming languages (C, C++, Perl, PHP, Pascal, etc.).

In the following, we will briefly review the state of the art in handling multilingual messages, present the syntax, semantics and implementation of MAs in more detail, and propose a method to translate MAs without asking translators (human or automatic) to directly handle automata.

2 State of the Art

2.1 Catgets (Sun) and Gettext (GNU)

Catgets has been proposed by Sun Microsystems as a standard for multilingual programming (X/Open Portability Guide, Volume 3, XIF Supplementary Definitions et

[1] Russian nouns have 2 forms for the plural in the nominative case, 1 for the singular.

[2] "Do you want to keep or discard the $n files?", becomes in Japanese "Do you want to keep the $n files?" Do you want to discard the $n files?

Single Unix Specification) [Wheeler, 2003]. A typical command to print a message is: printf(catgets(cat_id, set_num, msg_num, def_fmt), variables) ; where cat_id is a catalog identifier, set_num a subset number, msg_num a message number, and def_string a default format used if msg_num in set_num of cat_id cannot be accessed.

Messages are usual C formats with variables. A catalog contains sets of messages, and each set contains a part for each language. The language used is set by a global variable (NL_CAT_LOCALE). The genmsg utility extracts messages from a program and stores them in a .msg file, and gencat is used to build catalogs (.cat) from one or more .msg files.

GNU gettext (www.gnu.org/software/gettext/) is built on very similar ideas [Drepper et al., 2002]. Calls are simpler than with catgets, as default formats are used as message identifiers. Gettext can be used from many programming languages[3], and offers more functions to handle message files (msgcat, msgconv, msgmerge).

There is also a way to handle messages with 2 variants depending on an integer variable. For example, instead of writing

```
printf(gettext("%d file(s) [was|were] compiled"), n) ;
```

one can write

```
printf(ngettext("%d file was compiled", "%d files were compiled", n), n) ;
```

and the second variant will be produced (in the current interface language) if n>1. This facility is mostly used for singular/plural, but it can be used for other binary variants, as any integer expression can be used as third argument of ngettext.

2.2 AG5MSG and "Message Networks" in Ariane-G5

In the Ariane-G5 environment for building MT systems, developed at GETA from earlier Ariane-78 and Ariane-85 versions, we have integrated a more powerful idea. A MW is represented as a chart where branching nodes can have any number of outgoing arcs. The DECODMSG function is called with a message identifier, a list of values for variables, and a list of "choice numbers". It returns a final string and not a format. The multilingual message environment AG5MSG [Guillaume, 2002] contains utilities similar to those of gettext. It can be used from the programming languages used to build Ariane-G5 (ASM370, PL360, Pascal/VS, PL/I), but has not been adapted to others.

Here is an example, with the exact syntax used in messsage files.

```
MSG01: 'The &01-'(st':nd':rd':th') trial was successful, and the &02 file '(':s') found
'(is':are') satisfactory.'
```

If &01 = 13 and &02 = 5, the call will be: DECODMSG((13, 5), (3, 2, 2)).

The 30,000 messages of Ariane-G5 exist in French and English, which are quite similar in word order and have the same kind of variants for cardinals. However, more words vary in French than in English. Because French was handled first, and most

[3] C, C++, Objective C, sh (Shell Script), bash (Bourne-Again Shell Script), Python, GNU clisp (Common Lisp), GNU clisp (C sources), Emacs Lisp, librep, GNU Smalltalk, Java, GNU awk, Pascal (Free Pascal Compiler), wxWindows library, YCP (YaST2 scripting language), Tcl (Tk's scripting language), Perl, PHP Hypertext Preprocessor, Pike.

messages have only one cardinal variable, it was not necessary to change the calls: either the choice list prepared for French was longer, which causes no problem, as in

```
MSG02: '&01 fichier(':s') '(a':ont') été compilé(':s').'
MSG02: '&01 file(':s') '(has':have') been compiled.'
```

or one added "dummy choices" in English to parallel the French choices:

```
MSG03: '&01 fichier(':s') compilé(':s'), '(il fait':ils font') &02 octet(':s') au total.'
MSG03: '&01 file(':s') compiled(':'), '(it takes':they take') &02 byte(':s') in total.'
```

However, that kind of technique cannot be generalized to more complex cases, and to languages like Japanese in which the order of phrases in sentences and hence of variables in formats is different.

In conclusion, currently used techniques such as catgets and gettext are multilingual but allow no variants, and the AG5MSG approach can handle all kinds of variants, but is not truly multilingual.

3 Modelling Messages with Variables and Variants

We model MWs in two steps. First, we define the types of message variables, predefined global variables of these types, and special type-defining forms of variable identifiers. Then we define the syntax and semantics of message automata.

3.1 Variable Types, Global Variables and Language-Specific "Cases"

In GetAMsg, all variable identifiers begin with a $. If a variable begins with a predefined prefix such as $g_ it is of the corresponding type: $g_MFN will automatically be of type t_gm_genre. Otherwise, its type must be declared, e.g. $MFN: t_gm_genre. See Tables 1 and 2 below.

Table 1. Types of message variables

Type	Prefix	Extension	Description	Def
t_gm_langue	$l_	enum (fra, eng…)	ISO-639-2 language symbols	eng
t_gm_cardinal	$n_	int	Integer used as cardinal	1
t_gm_ordinal	$o_	int	Integer used as ordinal	1
t_gm_string	$s_	string	Character string	""
t_gm_real	$r_	Real	Real number	0.5
t_gm_genre	$g_	enum (m, f, n)	Gender	m
t_gm_politesse	$p_	[1..4]	Politeness level	3
t_gm_titre	$t_	[0..3]	Level of title (Mr, Dr…)	2
t_gm_age	$a_	[0..120]	Age	30
t_gm_calendrier	$c_	enum (JUL, GRE...)	Calendar type	GRE
t_gm_heure	$h_	[0..24]	Hour of day	12
…	…	Int	Date, time, day, month…	0

Table 2. Global variables

Name	Description
Variables concerning the system	
$s_system_name	Name of system (Mac OX 10.3.5)
$s_system_nickname	Nickname (Darwin)
$l_system_lang	Current interface language
$p_system_politeness	Current politeness level
$c_system_calendar	Usually gregorian
$s_catalogue	Message catalog name
Variables concerning the user	
$s_user_nickname	Pat
$s_user_family_name	Johnson
$s_user_first_name	Patrick
$s_user_second_name	Georges
$n_user_age	User age
$t_user_civile	Civil prefix (Mr, Mrs, Ms)
$g_user_genre	User gender
Variables concerning getamsg	
$l_gm_language	User preferred interface language
$p_gm_politeness	Politeness level
$t_gm_title	Civil title used

At this moment, the set of types is fixed, but we plan to make it modifiable by users.

Next, in Table 3, we define the "cases" for each variable type, and associate identifiers to them to abbreviate the writing of conditions in message automata. Note that the cases must be exclusive and cover all possibilities.

Remark. According to A. Gelbukh (see also [Sidorov & *al.* 1999, 2000], age in Russian does not correspond well to English, so the Russian cardinal cases should be defined differently from those of English in the table below. "Подросток" and "юноша" cover English teen, but none completely. They rather correspond to Spanish "adolecente" and "joven". "Подросток" is up to, say, 16, not 19, and cannot be married. Users of GetAMsg will be able to modify this table.

3.2 Message Automata

A MA is a loop-free FSA controlled at branching nodes to factorize variants of a message in a best way. Let us take a small formal example first. Suppose we want to represent the set of strings {a, abc, adce}. We can use the finite state automaton shown in Figure 1.

To produce a string, it is sufficient to give the corresponding final state. For example, 3 for abc, or 6 for adce, if this structure is implemented with backward links. But this does not allow to factorize as much as one would like (not even to minimize by merging states 3 and 6). We would like to have the graph shown in Figure 2.

Table 3. Variable "cases"

Var	Conditions			
	English	French	Russian	..
$g_	HE: $g_=m SHE: $g_=f IT: $g_=n	IL: $g_=m ELLE: $g_=f ÇA: $g_=n	ОН: $g_=m ОНА: $g_=f ЭТО: $g_=n	..
$p_	(none)	TU: $p_ ≤ 1 VOUS: $p_ ≥ 2	ТЫ: $p_ = 1 ВЫ: $p_ ≥ 2	..
$t_	NONE: $t_=0 MR_MRS: $t_=1 FUNCTION: $t_=2 RANK: $t_=3	SANS: $t_=0 M_MME: $t_=1 FONCTION: $t_=2 GRADE: $t_=3	БЕЗ: $t_=0 (no title) ОБРАЩ: $t_=1 (gospodin) ДОЛЖН: $t_=2 (function) ЗВАНИЕ: $t_=3 (Dr...)	..
$n_	SINGULAR: $n_ ≤ 1 PLURAL: otherwise	SINGULIER: $n_≤1 PLURIEL: otherwise	ОДИН (two): ($n_==1)‖($n_%10=1) &&($n_%100>20) ДВА (two): (1<$n_%10<5) && ($n_%100>20) ‖ (1<$n_<5) ПЯТЬ (five): otherwise	..
$a_	BABY: $a_≤3 CHILD: 3<$a_≤13 TEEN: 13≤$a_<20 ADULT: 20≤$a_≤60 SENIOR: 60<$a_≤80 OLD: $a_>80	BEBE: $a_≤3 ENFANT: 3<$a_≤13 ADO: 13≤$a_<20 ADULTE: 20≤$a_≤60 SENIOR: 60<$a_≤80 VIEUX: $a_>80	МЛАДЕНЕЦ: $a_ ≤ 2 РЕБЁНОК: 2 < $a_ ≤ 13 ЮНОША: 13 ≤ $a_ < 20 ВЗРОСЛЫЙ: 20 ≤ $a_ ≤ 60 ПОЖИЛОЙ: 60 < $a_ ≤ 80 СТАРИК: $a_ > 80	..
$c_	JULIAN: $c_=1 GREGORIAN: $c_=2 JAPANESE: $c_=3 BUDDHIST: $c_=4 ISLAM: $c_=5 ...	JULIEN: $c_=1 GREGORIEN: $c_=2 JAPONAIS: $c_=3 BOUDDHISTE: $c_=4 ISLAM : $c_=5 ...	ЮЛИАНСКИЙ: $c_=1 ГРИГОРИАНСКИЙ: $c_=2 ЯПОНСКИЙ: $c_=3 БУДДИЙСКИЙ: $c_=4 ИСЛАМСКИЙ: $c_=5 ...	..
$h_	AM: $h_ ≤ 12 PM: 12 < $h_ ≤ 24	AM: $h_ ≤ 12 PM: 12 < $h_ ≤ 24	БЕЗ (ч)	..
$o_	FIRST: $o_%10=1 && $o!=11 SECOND: $o_%10=2 && $o!=12 THIRD: $o_%10=3 && $o!=13 NTH: otherwise	IER: $o_ = 1 IEME: otherwise	БЕЗ (-й) or 2 cases: ЫЙ 1|3-5|9-20...100...-ый ОЙ 0|2|6-8|22...-ой	..
...	...	...	...	..

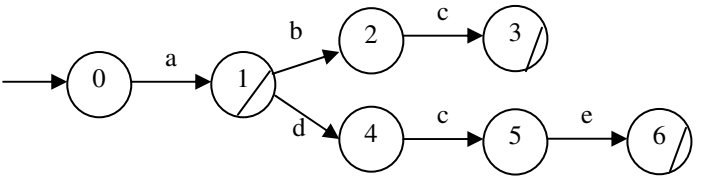


Fig. 1. Prefix tree (trie) FSA

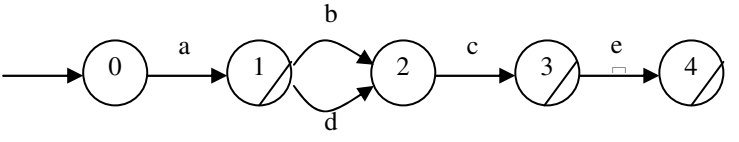


Fig. 2. Factorizing non-controlled FSA

But that automaton generates too many strings: a(ε +(b+d)c(e+ε))={a, abc, abce, adc, adce}. Hence the idea to add controls depending on the values of one or more variables. Here, the input variable is &n, with values in {1, 2, 3}, corresponding to {a, abc, adce}, and one introduces controls &c1 and &c3 on states 1 and 3:

$$\&c1 = \begin{cases} 1 & \text{if } \&n = 2 \quad \text{(abc)} \\ 2 & \text{if } \&n = 3 \quad \text{(adce)} \\ 3 & \text{otherwise} \quad \text{(a)} \end{cases}$$

$$\&c3 = \begin{cases} 1 & \text{if } \&n = 3 \quad \text{(adce)} \\ 2 & \text{otherwise} \quad \text{(stop after abc)} \end{cases}$$

If the value of a control is k, the k-th arc should be traversed if it exists, otherwise one should stop (cases &c1=3 et &c3=2 here). The GetAMsg syntax is simple: `EXAMPLE($n_) {[&c1:$n_==2,$n_==3], [&c3: $n_==2]} = "a[&c1:bld]c[&c3:e]".`

To call it from C, we write:

```
printf(getamsg("EXAMPLE n_", 2));
```

where `n_` indicates that there is one variable, of type `t_gm_cardinal`. This is necessary because we must know the number and types of variables in order to implement a function with prototype `f(arg, ...)`, and that has to be given by the value of `arg`.

Here is our initial example again, in GetAMsg syntax. We may use full expressions or abbreviations defined in the table of cases, such as `[&c1: FIRST($n), SECOND($n), THIRD($n), NTH($n)]`. In the future, we may allow more abbreviations, such as `[&c1($n): FIRST, SECOND, THIRD, NTH]`. See Figure 3.

The semantics are simple. A control &c is defined by a list of k conditions. Its value is the first i such that the i-th condition is true, if it exists, and k+1 otherwise. A branching node with control &c must have k+1 or k+1 outgoing arcs. If it has k outgoing arcs and &c = k+1, traversal stops.

```
MSG01($n_, $n_p)
{[&c1:FIRST($n_),SECOND($n_),THIRD($n_)],[&c2:$n_p<2]}=
"The $n_-[&c1:st|nd|rd|th] trial was successful, and the $n_p
file[&c2:|s] found [&c2:is|are] satisfactory."

MSG01($n_, $n_p){[&c1: ($n_==1)], [&c2: $n_p<2]}=
"Le $n_p-[&c1:er|ième] essai a réussi, et le[&c2:|s $p]
fichier[&c2:|s] trouvé[&c2:|s] [&c2:est|sont] satisfai-
sant[&c2:|s]."

MSG01($n_, $n_p)
{[&c1: ОДИН($n_p)], [&c2: ОДИН($n_p), ДВА($n_p),
ПЯТЬ($n_p)]}=
"$n_-ая попытка удалась, и найденны[&c1:й|е $n_p]
файл[&c2:|а|ов] удовлетворител[ен|ьны]."
```

Fig. 3. Example of a MA in 3 languages

We have used the same variable names here in each language, but that is not necessary. It is also possible that a message has more variables in some languages than in others. We could require that calls transmit enough variables for the shortest variable list across languages, but that is also not necessary: if there are not enough variables transmitted, we complete by the values of system global variables for the corresponding types, if any, or else by default values found in the type definition table.

```
R003($n_r, $n_n) {[&c1: $n_r==1], [&c2: $n_n==1]}=
"You reserved $n_r [&c1:room|rooms] for $n_n [&c2:night|nights]."
R003($n_k, $n_n, $g_, $p_) {[&c1:$n_p==1], [&c2: ($n_p==1)&&ОН($g_),
($n_p==1)&&ОНА($g_)], [&c3: ОДИН($n_k)], [&c4: ОДИН($n_n)]}=
"[&c1:Ты|Вы] зарезервировал[&c2:|а|и] $n_k комнат[&c3:у|] на $n_n
ноч[&c4:ь|ей]."
```

A call to this message might have 2, 3 or 4 variables:

```
getamsg("R003 n_ n_",          n1, n2);        or
getamsg("R003 n_ n_ p_",       n1, n2, pol);       or
getamsg("R003 n_ n_ p_ g_",    n1, n2, pol, gnr);
```

4 Translation of MAs

Human or machine translators cannot be expected to handle directly message automata, which are actually not normal sentences, but generators of sentences. The proposed method is to

- generate in the source language an instance of the message for each possible variant according to the variable cases of the target language,
- translate these instances, by any method,
- factorize the resulting set of formats into a MA, and compute adequate controls.

In order to be able to get variable names in the target MA at their proper place, the instances are produced by substitutions of the form $var → $var=value. After translation, the =value strings are removed.

4.1 Generation of Instances of Variants According to Target Language

Let us first take the example above:

```
R003($n_r, $n_n) {[&c1: $n_r==1], [&c2: $n_n==1]}=
"You reserved $n_r [&c1:room|rooms] for $n_n [&c2:night|nights]."
```

and suppose we want to translate it into Russian. First, we determine that there should be one more variable in Russian to handle politeness-related variants, so that the AM heading should be: `R003($n_r, $n_n, $p_)`. Then we choose an instance of each variable case in Russian, for instance `$p_=(1, 2)`, or (Ты, Вы), `$n_r=(1, 2, 5)`, `$n_n=(1, 2, 5)`. That gives 18 sentences:

N°		Source sentences to be translated
1	[Ты($p_)]	You reserved $n_r=1 room for $n_n=1 night.
2	[Ты($p_)]	You reserved $n_r=1 room for $n_n=2 nights.
3	[Ты($p_)]	You reserved $n_r=1 room for $n_n=5 nights.
4	[Ты($p_)]	You reserved $n_r=2 rooms for $n_n=1 night.
5	[Ты($p_)]	You reserved $n_r=2 rooms for $n_n=2 nights.
6	[Ты($p_)]	You reserved $n_r=2 rooms for $n_n=5 nights.
7	[Ты($p_)]	You reserved $n_r=5 rooms for $n_n=1 night.
8	[Ты($p_)]	You reserved $n_r=5 rooms for $n_n=2 nights.
9	[Ты($p_)]	You reserved $n_r=5 rooms for $n_n=5 nights.
10	[Вы($p_)]	You reserved $n_r=1 room for $n_n=1 night.
11	[Вы($p_)]	You reserved $n_r=1 room for $n_n=2 nights.
12	[Вы($p_)]	You reserved $n_r=1 room for $n_n=5 nights.
13	[Вы($p_)]	You reserved $n_r=2 rooms for $n_n=1 night.
14	[Вы($p_)]	You reserved $n_r=2 rooms for $n_n=2 nights.
15	[Вы($p_)]	You reserved $n_r=2 rooms for $n_n=5 nights.
16	[Вы($p_)]	You reserved $n_r=5 rooms for $n_n=1 night.
17	[Вы($p_)]	You reserved $n_r=5 rooms for $n_n=2 nights.
18	[Вы($p_)]	You reserved $n_r=5 rooms for $n_n=5 nights.

For the moment, we use the first value in each variable case, but the user might as well fix some typical values. Note that, if there are 2 variables or more, erasing the `$var=` strings would lead to possible confusions when cases have only 1 value.

The generation of the MA instances is simple: we enumerate all possible combinations, in lexicographic order, putting supplementary variables first, and prefixing the formats with the indication of cases for those variables. For each combination, we call the MA, with an appropriate global parameter to produce the `$var=value` forms. Remark that, if we had allowed that 2 cases overlap, it would not be possible in general to generate distinguishing instances.

4.2 Translation of Instances

As seen in this example, the translation work appears to be multiplied by the number of instances generated. But the repetition rate among these instances is very high. Hence, the work is only marginally more important if one uses a translation support system with a translation memory. If machine translation is used, the supplementary machine time is negligible, but the case is the same as a human must postedit the results, or, more realistically, use them as suggestions for producing good translations. With this example, then, we may expect to get the following 18 translations.

N°	Target sentences after translation
1	[ТЫ($p_)] Ты резервировал $n_r=1 комнату на $n_n=1 ночь.
2	[ТЫ($p_)] Ты резервировал $n_r=1 комнату на $n_n=2 ночи.
3	[ТЫ($p_)] Ты резервировал $n_r=1 комнату на $n_n=5 ночей.
4	[ТЫ($p_)] Ты резервировал $n_r=2 комнаты на $n_n=1 ночь.
5	[ТЫ($p_)] Ты резервировал $n_r=2 комнаты на $n_n=2 ночи.
6	[ТЫ($p_)] Ты резервировал $n_r=2 комнаты на $n_n=5 ночей.
7	[ТЫ($p_)] Ты резервировал $n_r=5 комнат на $n_n=1 ночь.
8	[ТЫ($p_)] Ты резервировал $n_r=5 комнат на $n_n=2 ночи.
9	[ТЫ($p_)] Ты резервировал $n_r=5 комнат на $n_n=5 ночей.
10	[ВЫ($p_)] Вы резервировали $n_r=1 комнату на $n_n=1 ночь.
11	[ВЫ($p_)] Вы резервировали $n_r=1 комнату на $n_n=2 ночи.
12	[ВЫ($p_)] Вы резервировали $n_r=1 комнату на $n_n=5 ночей.
13	[ВЫ($p_)] Вы резервировали $n_r=2 комнаты на $n_n=1 ночь.
14	[ВЫ($p_)] Вы резервировали $n_r=2 комнаты на $n_n=2 ночи.
15	[ВЫ($p_)] Вы резервировали $n_r=2 комнаты на $n_n=5 ночей.
16	[ВЫ($p_)] Вы резервировали $n_r=5 комнат на $n_n=1 ночь.
17	[ВЫ($p_)] Вы резервировали $n_r=5 комнат на $n_n=2 ночи.
18	[ВЫ($p_)] Вы резервировали $n_r=5 комнат на $n_n=5 ночей.

4.3 Factorization into a Target MA

At this point, we erase the =value strings and get a list of target formats, together with the corresponding case combinations. With the previous example again:

N°	Target sentences after translation	Cases
1	[ТЫ($p_)] Ты резервировал $n_r комнату на $n_n ночь.	1 1 1
2	[ТЫ($p_)] Ты резервировал $n_r комнату на $n_n ночей.	1 2 1
3	[ТЫ($p_)] Ты резервировал $n_r комнату на $n_n ночей.	1 3 1
4	[ТЫ($p_)] Ты резервировал $n_r комнаты на $n_n ночь.	2 1 1
…		
12	[ВЫ($p_)] Вы резервировали $n_r комнату на $n_n ночей.	1 3 2
13	[ВЫ($p_)] Вы резервировали $n_r комнаты на $n_n ночь.	2 1 2
…		
17	[ВЫ($p_)] Вы резервировали $n_r комнаты на $n_n ночей.	3 2 2
18	[ВЫ($p_)] Вы резервировали $n_r комнаты на $n_n ночей.	3 3 2

Such a list can be trivially converted into a MA with 2 states, 18 arcs, and a control on the input state with 18 conditions corresponding to the 18 combinations. The remaining problem is to minimize this MA. We did not find a convenient "divide and conquer" algorithm. Perhaps one could be based on a convenient extension of the notion of distance between 2 FSAs (used at AT&T by Mohri) to the case of controlled FSAs, but we did not pursue that approach further.

The algorithm we propose here is simple, but certainly not optimal in general, although it gives satisfactory results. First, we create the graph of the "target" MA as a FSA which factorizes the variants and where each arc is annotated by a set Traj containing the identifiers (numbers) of the variants ("trajectories") which traverse it. Then we build the controls using that information and the conditions defining the variants.

Creation of the MA graph. We renumber the variants by putting first one of the longest. With this w_1, we create an initial MA, with one arc per symbol, and Traj={1} on each arc. Then we iteratively "merge" each remaining variant w_t ($2 \le t \le p$) into the MA at minimal cost. Because we begin with a longest variant, we will never have to create a new state. The only possible operations are the insertion of an empty arc or the insertion of an arc with a symbol.

We may choose that the symbols on the arcs of the MA are characters or words, giving rise to a character- or word-based factorization. Hence, the length is the number of characters or of words. If we choose to use words, the string is converted to a sequence of integers, in the range [1..nw] if there are nw different words in the set of formats. In the example after Fig. 5 below, a symbol is a character.

The algorithm to merge a string $a_1 \ldots a_n$ into a MA having a path longer or equal to n, hence with m+1 (m>n) states, is a dynamic programming algorithm inspired by that of [Wagner & Fischer, 1974] to compute the edit distance between 2 strings. The cost of adding an arc labelled by a symbol may be the same for all symbols, or may be related to the length of the symbol if symbols are words.

Let the states of the MA be [0..m] and C(k, i, w) be the minimal cost to add a path labeled by w between states k and i (0 if such a path exists in the MA). We have:

$$\begin{cases} C(0,i,a_1) & = \min\limits_k \{C(0,k,\varepsilon) + C(k,i,a_1), \quad C(0,k,a_1) + C(k,i,\varepsilon)\} \\ C(0,i,a_1 \ldots a_{j+1}) & = \min\limits_k \{C(0,k,a_1 \ldots a_j) + C(k,i,a_{j+1}), \quad C(0,k,a_1 \ldots a_{j+1}) + C(k,i,\varepsilon)\} \end{cases}$$

We build 3 matrices C, M and P of dimension [0..m, 0..n] where C[i, j] contains C(0, i, $a_1 \ldots a_j$), M[i, j] the action chosen, and P[i, j] the preceding state k chosen when last setting C[i, j]. The action is null if an existing arc of the MA has been used (that arc bears ε if HAD_ARC_EPS or a symbol if HAD_ARC_ATOM), otherwise it is an insertion (INS_ARC_EPS or INS_ARC_ATOM). At the end of this step, the minimal cost of merging w in the current MA is stored in C[m, n]. We then go backward and execute the actions corresponding to the sequence stored in M. See Figure 4.

Computing controls in the target MA. Suppose the MA has p trajectories (one for each variant w_t), m+1 states 0, …i, …m, and q arcs 1, …a, …q, numbered so that arcs are sorted according to their origins (those going out of state 0, then those going out of state 1, etc.). Suppose also there are n variables, V_1, …V_j, …V_n.

We build the following data structures:

- a "case matrix" CM[1..p][1..n] where CM[t, i]=W_{ti}, the number of the case of variable V_i on trajectory t. For each variable, we add to its normal cases (numbered from 1) a "0" case which will be assigned if the variable is not needed in the conditions defining the control on a certain state.
- a boolean matrix AT[1..q][1..p] linking arcs and trajectories, where AT[a, t] = 1 (true) if trajectory t traverses arc a, 0 (false) otherwise.
- a 3-column table States_Arcs[1..q] associating to each state i its first arc First_a[i], its number of outgoing arcs Nb_a[i], and the number of trajectories traversing it, Nb_t[i].

```
Merge (MA, w, t)
Input: the current MA, a string w=a1...an=w[1]...w[n], and its number, t.
Method: see above
begin                          -- Initialization
MAXCOST := n*COST_INS_ARC_ATOM;
C[0,0] := 0 ;   -- null cost for ε (denoted eps below)
n := length (w); m:= nbStates(MA) - 1;
for i from 1 to m do   -- 0-th column (empty string)
    if arc(0,i,eps)  then C[i,0] := 0 else C[i,0] := COST_INS_ARC_EPS; endif;
endfor;                    -- Main loop
for j:=1 to n do                              -- with this method, i  j is necessary:
    for i := j to m do                        -- no need to touch upper right triangle.
        C[i,j] := MAX_COST ;
            -- Priority is given to arcs with w[j] coming from near states, and
            -- otherwise to empty arcs coming from near states.
        for k from j-1 to i-1 do              -- for the same reason, k>j is necessary.
            if ¬arc(k, i, eps) then C := C[k, j] +  COST_INS_ARC_EPS; endif;
            if c<= C[i,j] then -- We have a new minimum
                C[i, j] := c; P[i, j] := k;
                M[i, j] :=(arc(k, i, eps) ? HAD_ARC_EPS : INS_ARC_EPS);
            endif;
        endfor;
        for k from 0 to i-1 do
            if ¬arc(k, i, w[j-1]) then c := C[k, j-1] + COUT_INS_ARC_ATOM; endif;
            if c<= C[i, j] then -- We have a new minimum (the last will be kept)
                C[i, j] := c; P[i, j] := k;
                M[i, j] := (arc(k , j-1, w[j-1]) ? HAD_ARC_ATOM : INS_ARC_ATOM);
            endif;
        endfor;
    endfor;
endfor;
        -- The minimum cost is in C[m,n]. Second step: we go back and modify MA by
        -- executing the actions stored in the traversed M[i,j], from M[m,n] to M[0,0].
finished := false; i :=m ; j :=n ;
while ¬finished do
    k:= P[i,j]; action := M[i,j];
    switch on action
        HAD_ARC_EPS     => arc:=(k, i, epsilon);
        INS_ARC_EPS     => insert_arc(k, i, epsilon); arc:=(k, i, epsilon);
        HAD_ARC_ATOM  => j:=j-1; arc:=(k, j, w[j]);
        INS_ARC_ATOM   => j:=j-1; insert_arc(k, i, w[j]); arc:=(k, j, w[j]);
    endswitch;
    i:=k; add(t, Traj(arc)) ; -- Recall t is the number of the variant (or "trajectory").
    if (i==0) && (j==0) then finished := true; endif;
endwhile;
end.
```

Fig. 4. Algorithm to merge a MA with a variant

– for each arc a, a matrix $D_a[1..Nb\_t[Origin(a)]][1..n]$ where $D_a[t, j] = W_{sj}$, the number of the case of variable V_j in the t-th trajectory traversing a (i.e., of trajectory s=First_a[Origin(a)]+t-1, as trajectories are renumbered and only those traversing a are considered in D_a).
– a boolean vector Active_Var[1..n] used to determine which variables are necessary to distinguish between 2 outgoing arcs.

Then we compute the text of the control on each node. Note that a node may have a non-empty control even if it has only one outgoing arc, because some trajectory may stop at it. An algorithm for computing the controls is given in Figure 5 below.

Remarks. In this algorithm, we write D[a] instead of D_a for legibility. Also, the action Internal_Reduction(D_a), not yet implemented, should reduce the complexity of the disjunctive normal form corresponding to D_a using boolean identities such as $W_{j1} \vee W_{j2} \vee \ldots \vee W_{jNb\_case(Vj)} \equiv$ true, $X \wedge$ true $\equiv X$, etc. This should augment the list of variables inactive on the considered arc. Further improvements are certainly possible.

To finalize the construction after computing the controls, we further collect the texts of the controls, eliminate copies so that all controls have different texts, reassign names to controls and control names to branching nodes, put the control definitions after the list of variables of the target MA, and then the linearized form of the graph.

Example. With our running example, we will not get the form as written above,

```
R003($n_k, $n_n, $p_) {[&c1:$n_p==1], [&c2: ODIN($n_k)], [&c3:
ODIN($n_n)]}=
"[&c1:Ты|Вы] резервировал[&c1:|и] $n_k комнат[&c2:у|ы] на $n_n
ноч[&c3:ь|и]."
```

but a standardized form such as (neglecting the gender)

```
R003($n_r,$n_n,$p_)
{[&c1:БЛИЗКИЙ($n_p)], [&c2: ОДИН($n_r)], [&c3:ОДИН($n_n)]}=
"[&c1:Т|В]ы резервировал[&c1:|и] $n_r комнат[&c2:у|ы] на $n_n
ноч[&c3:ь|и]."
```

if symbols are characters, or the longer following form, if symbols are words.

```
R003($n_r,$n_n,$p_)
{[&c1:БЛИЗКИЙ($n_p)], [&c2: ОДИН($n_r)], [&c3:ОДИН($n_n)]}=
"[&c1:Ты|Вы] [&c1:резервировал|резервировали] $n_r
[&c2:комнату|комнаты] на $n_n [&c3:ночь|ночи]."
```

Changing a variable name ($n_r to $n_k) must of course be done manually.

5 Discussion and Perspectives

5.1 Implementation Issues

Formatting variables in messages. As usual, variables can be followed by formatting commands. For example $n_1%3d tells GetAMsg to format the integer $n_1 on 3 digits rather than to use the default formatting. As all GetAMsg types are subtypes of C types which can appear in C formats, we allow all formatting commands of C. However, that is not enough, and we need formatting commands specific to some GetAMsg types.

```
Compute_controls(MA, variants)
        -- Initialisation
build CM[1..p][1..n], AT[1..q][1..p], and States_Arcs[1..q] as defined above.
        -- Body:  process each state in turn
for i from 0 to m do                        -- i = current state
    r:=Nb_a[i];                                 -- r = number of arcs leaving state i
    for a from 1 to r do                    -- build the matrices of the outgoing arcs
        build D[a][1..Nb_t[a]][1..n]; -- (discardable  after handling the state)
    endfor;
    for j from 1 to n do Active_Var[j]:=true; endfor;
    if r=1 then Internal_reduction(D[a]); endif;
    if r>1 then
        for j from 1 to n do            -- test whether Vj is useful
            Active_Var[j]:=false; useful_var:=false; a1 := 1;
            while (useful_var) && (a1 < r) do
                a2 :=  a1 + 1;
                while (useful_var) && (a2 <= r) do
                    -- Vj is useful if there are 2 identical rows in the D matrices of
                    -- two arcs a1 and a2, modulo the inactive variables
                    useful_var:= search_eq_modulo_inactive_var(Da1, Da2);
                    a2 := a2 + 1;
                endwhile; a1 := a1 + 1;
            endwhile; if useful_var then Active_Var[j]:=true); endif;
        endfor;             -- Useless variables are the Vj such that Active_Var[j]=false.
        buf_write:=C[i];                    -- C[i] is the text on state i, write there.
        write( "[&c" + i + ":" );           -- begin with "[&ci:"
        for a from 1 to r do                -- r = number of arcs leaving state i
                -- Create condition a for C[i] in text form.
                -- For this, use the active variables in D[a], line by line.
            t := 1;                                 -- t = number of trajectory on a.
            while t <= Nb_t[a] do               -- && has priority over ||, hence
                if t >= 2 then write( “||” ); endif; -- no parentheses are needed here.
                one_conjunct := false;
                for j from 1 to n do
                    if Active_Var[j] then
                        if one_conjunct then write( “ && ”); endif;
                        write(Case(D[a][t,j], Target_Lg) + “(” + Name_Var[j]+ ")" );
                        one_conjunct := true;-- example of conjunct: "DVA(&n_r)"
                    endif;
                endfor;
                t := t + 1;
            endwhile; write( “] ”);
        endfor;
    endif;
endfor;
```

Fig. 5. Computation of controls on branching nodes in target MA

Dates present a classical but complex case. Operating systems like Mac OS or applications like Excel have a large variety of predefined formats, localized to many languages, such as 10/11/04 or 10 novembre 2004 (France), 11/10/04 or November, 10, 2004 or Nov. 10th (US), etc. Applications such as Excel even allow users to define their own formats for a predefined list of types. This is necessary in our case as we want to be able to handle all languages. It will also be necessary to include the possibility to define functions in order to *change the values of variables.* For example, "19h" should become 7 pm in English, "7 heures du soir" in French (not "pm"), and "หนึ่งทุ่ม" ("neung thum" or "1 evening") in Thai.

At this moment, dates are not yet handled by GetAMsg (the type is not yet defined, and dates are passed as strings prepared elsewhere in the calling program). In the future, we plan to develop a simple syntax to define formats, for all types of GetAMsg variables. For example, %'dd/mm/yy', %'d mmm yyyy', %'mm/dd/yy h:mn:ss', etc. Functions transforming variables sould be defined together with the types, globals and cases, and called with a syntax like: %'f_function($variable)[%format], for example: "ตอนนี้%'f_thaiHour($h_)%'hh:mn'(ชั่วโมง)" ("Now %'f_thaiHour($h_)%'hh:mn' (by Thai hour counting)".).

5.2 Messages with Variables and Variants as Restricted Sublanguages

We have seen that message variables have richer types than usually assumed. On the other hand, *message variables cannot correspond to verbs and common nouns.*

Indeed, if we allow a variable like $verb in "Would you like to $verb me?", we have to restrict $verb to the transitive verbs, or a subset of them. But transitivity is not invariant across languages, and that is not predictable. For instance, "Would you like to help me" is translated in German as "Könnten Sie mir helfen?" (mir is dative, mich accusative). Also, verb variables should be "split" because of the possibility of "separable particles" ("he gives back the money" / "he gives the money back").

The problem with nouns is that many, if not all, languages have lexically bounded numerical specifiers. This is well known for Asian languages, but is also observed in western languages. For example, "$n_1 meuble[&c:ls]" should be translated as "$n1_ piece[&c:ls] of furniture" and not as "$n1_ furniture[&c:ls]". That poses no translation problem because the noun is known. But we cannot write "$n_1 [&c1:l$x_ of] $noun[&c2:ls]", because not only does &c1 depend on the value of $noun (&c1=1 if $noun has no specifier, and &c1=2 otherwise), but $x_ (the specifier, if any) cannot be passed as a parameter, it has to be retrieved from a dictionary.

In other words, what distinguishes the sublanguage of computer messages from other well studied sublanguages like weather bulletins, maintenance instructions or stock market flash reports, is that they must be generated with very simple means (variable substitution), and not by the full power of natural language generation.

5.3 Possibility to Use UNL Graphs to Represent and Translate MWs

UNL. An interesting and still more speculative idea is to represent a MW by a unique object, independently of any natural language. Interlingual representations of natural language utterances have long been studied, and used in the context of MT (machine translation) systems. There are different kinds of such interlingua, which we don't

want to discuss here. For various reasons, we promote the use of UNL (Universal Networking Language).

The UNL language is an "anglosemantic" interlingua. There are two reasons to use this term (anglosemantic). First, given an utterance in some language L, a UNL graph (or rather hypergraph) representing that sentence is an abstract structure of a semantically equivalent English sentence. Second, the symbols used in UNL for denoting semantic relations and attributes, and for building the UNL lexemes (UWs, for "Universal Words", or "Unité de Vocabulaire Virtuel"), are English or derived from English. That makes it possible for all developers around the world to understand UNL with far less effort than if it were based on any other language, or on artificial symbols. We refer to [Uchida 2001], [Boitet 2002] and the documents on the UNL web site (www.undl.org) for more details and linguistic examples.

Representation of a MW by a UNL Graph. A natural idea, then, is to represent a MW as a UNL graph with variables. From it, it should be possible to generate an MA for each language for which there is a deconverter[4].

That would be a distinct advantage, because there are deconverters for languages for which there are no MT systems, not even with English, it is far less costly to develop a deconverter into a language than a MT system from English into it, and there are groups building free deconverters for various languages. Another advantage is that source messages could be in any language, which would ease the life of developers who are not native speakers of English.

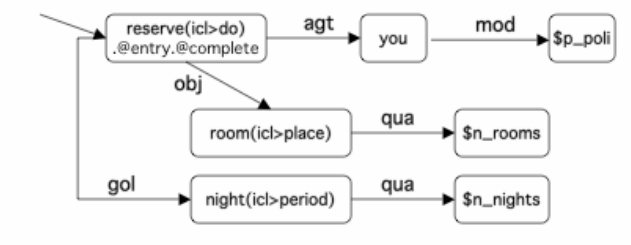
But there are no variables in a UNL graph. We could propose to extend UNL by introducing that new kind of object in its specification. In the UNL project, however, we try to introduce few modifications in the UNL specifications, and only if they prove to be really necessary, in order to keep the major quality of UNL, which is to be a "common format", enabling communication between languages.

What we propose here, then, is to represent a MW by a UNL graph conform to the official specifications, but with a special and simple convention: to represent a message variables, use an UW beginning with $, as in GetAMsg, and having a restriction indicating its type, if necessary.

For example,

"You reserved $n_rooms [&c2:room|rooms] for $n_nights [&c4:night|nights]."

can be represented by the following UNL graph:



[4] This term, introduced in 1996 by H. Uchida, is more appropriate than "generation", because the passage from UNL to a natural language involves a lexical transfer step, as opposed to a generation, which uses more or less abstract lexical symbols from the same lexical set.

```
<unl>
    agt(reserve(icl>do).@entry.@complete, you)
    mod(you, $p_poli)
    obj(reserve(icl>do).@entry.@complete, room(icl>place))
    qua(room(icl>place), $n_rooms)
    gol(reserve(icl>do).@entry.@complete, night(icl>period))
    qua(night(icl>period), $n_nights)
</unl>
```

To represent the full MA, we may reuse the technique proposed by M. Tomokiyo [Tomokiyo et Chollet, 2003] for VoiceUNL, another extension of the use of UNL which does not change its specification in any way. There is no need for controls here.

```
<msg id_msg=message identifier>
    <var_list> List of variables </var_list>
    <unl> UNL graph </unl>
</id_msg>
```

For example, to represent

```
R003($n_rooms,$n_nights, $p_poli) {[&c1:$n_poli==0],
[&c2:$n_rooms==1], [&c3:$n_nights==1]}
="You reserved $n_rooms [&c2:room|rooms] for $n_nights
[&c3:night|nights]."
```

we can write as in Figure 6:

```
< msg  id_msg= “R003”>
    <var_list> $n_rooms,$n_nights, $p_poli </var_list>
    <unl>
        agt(reserve(icl>do).@entry.@complete, you)
        mod(you, $p_poli)
        obj(reserve(icl>do).@entry.@complete, room(icl>place))
        qua(room(icl>place), $n_rooms)
        gol(reserve(icl>do).@entry.@complete, night(icl>period))
        qua(night(icl>period), $n_nights)
    </unl>
</id_msg>
```

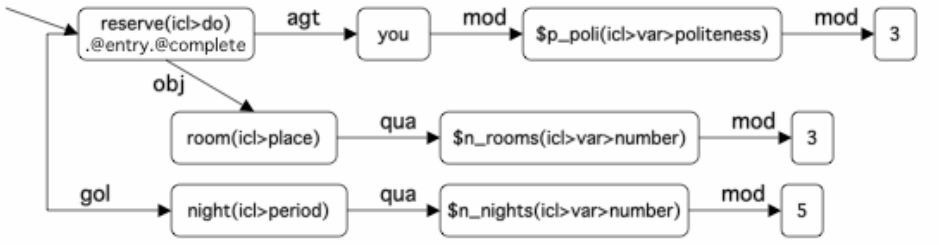
Fig. 6. UNL graph with variables to represent a multilingual MW

Localization. Our proposal is quite straightforward. To generate the MA corresponding to a UNL graph,

- generate an "extended instance" of the UNL graph for each possible variant in that language,
- deconvert these UNL graphs,

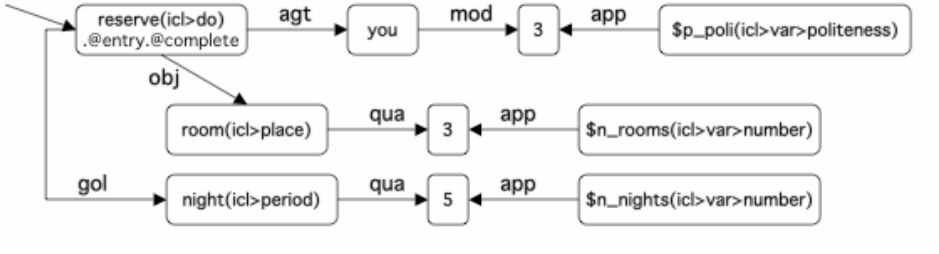
then continue as with normal translation:

- postedit the results
- factorize the results into a target MA.




```
<unl>
    agt(reserve(icl>do).@entry.@complete, you)
    mod(you, $p_poli(icl>var>politeness))
    mod($p_poli(icl>var>politeness), 3)
    obj(reserve(icl>do).@entry.@complete,
room(icl>place))
    qua(room(icl>place), $n_rooms(icl>var>number))
    mod($n_rooms(icl>var>number), 3)
    gol(reserve(icl>do).@entry.@complete,
night(icl>period))
    qua(night(icl>period), $n_nights(icl>var>number))
    mod($n_nights(icl>var>number), 5)
</unl>
```

Fig. 7. Instance of a variant of a UNL graph with variables representing a MW




```
<unl>
    agt(reserve(icl>do).@entry.@complete, you)
    mod(you, 3)
    app($p_poli(icl>var>politeness), 3)
    obj(reserve(icl>do).@entry.@complete, room(icl>place))
    qua(room(icl>place), 3)
    app($n_rooms(icl>var>number), 3)
    gol(reserve(icl>do).@entry.@complete,
night(icl>period))
    qua(night(icl>period), 5)
    app($n_nights(icl>var>number), 5)
</unl>
```

Fig. 8. "Prototype" UNL graph representing a MW, with values as in usual graphs

We say "extended instance" because the geometry of the graph has to change if we want to put the names and values of variables in the instances. For instance, the extended instance of the above graph with $n_rooms=3, $n_nights=5, and $p_poli=3, the extended instance could be as in Figure 7 below.

Another idea would be not to modify the geometry of the "message graph", but simply to change the values of variables in variant generation. That is, the "message graph" would be a *prototype* of all "variant graphs". The graph of Fig. 7 above, or any other graph obtained by changing the values of the variables in a way compatible with their GetAMsg type, could be such a prototype.

We have to further study which form would be best in practice. For example, developers of deconverters might want to start with usual graphs, and attach variable names to values in prototypes by inverse apposition relations (app), and not attach values to variables by modification (mod) relations as done above. The above example would become as in Figure 8:

6 Conclusion

Messages with variables and variants (MWs) are interesting linguistically, because variable types are specific (cardinal, ordinal, politeness…) and induce different "variant cases" in each language, and because, as a sublanguage, they are severely restricted: as no full-fledged NL generator can be used, but only variable substitution, there can be no variables for verbs and common nouns.

They are also interesting computationnally, because they are not yet handled in a satisfactory way, so that messages with variants are quite awkward in many languages, and because the problem of their translation (localization) into many languages is quite difficult.

Controlled loop-free FSAs, called here "message automata" (MAs), have been proposed to model such messages. To translate a MA, one generates an instance of it for each possible variant in the target language. After translation, the values used in the instances are discarded and a target language MA is built by factorization (not classical minimization), using an original dynamic programming algorithm. A library for handling catalogues of MAs, GetAMsg, has been implemented in C, and can be used from many usual programming languages.

A still speculative idea is to use a UNL graph conform to the official specifications, but with some special conventions, to represent a message with variables, and generate the language-specific MAs from it.

Acknowledgements

Pierre Guillaume elaborated on a first specification by the author in 1985, and single-handedly programmed the compiler, the interpreter and all utilities of the AG5MSG library, in assembler 370 and EXEC2, under VM/CMS. He also put all 30,000 messages of Ariane-G5 in "message network" form, and translated them into English. Hung Vo-Trung recently worked with the author to refine the controlled FSA approach, and implemented the GETAMSG library and all utilities. Last but not least,

many thanks to A. Gelbukh, who carefully read the manuscripts, corrected lots of mistakes in the Russian examples, and suggested several motivated and interesting changes and additions.

References

[Boitet 1982] Christian Boitet: *"Le point sur ARIANE-78 début 1982"*, 3 volumes, Partie 1, GETA-CHAMPOLLION, CAP SOGETI-France, avril 1982.
[Boitet, 1990] Christian Boitet: *"1980 – 90 : TAO du réviseur et TAO du traducteur",* in La TAO à Grenoble en 1990, école d'été à Lannion, www-clips.imag.fr/-geta/christian.boitet/pages_personnelles/ (in English in Proc. ROCling-90, Taipeh)
[Boitet, 2002] Christian Boitet: *"A rationale for using UNL as an interlingua and more in various domains",* Proceedings "First International Workshop on UNL, other Interlanguages and their Applications", LREC2002, Las Palmas, Spain, May 2002.
[Boitet, 2003] Christian Boitet: *"Messages avec variantes, automates finis contrôlés, et multilinguisme",* document interne, GETA, laboratoire CLIPS, IMAG, février 2003.
[Drepper et al., 2002] U. Drepper, J. Meyering, F. Pinard, B. Haible : *"GNU gettext tools, version 0.11.2"*, Published by the Free Software Foundation, April 2002.
[Guillaume, 2002] Pierre Guillaume : *"L'interface utilisateur multilingue en Ariane-G5"*, Rapport de recherche, Groupe d'Etude pour la Traduction Automatique (GETA), CLIPS, IMAG, avril 2002.
[Sidorov & al. 1999] G. Sidorov, I. Bolshakov, P. Cassidy, S. Galicia-Haro, A. Gelbukh: *"Non-adult' semantic field: comparative analysis for English, Spanish, and Russian",* Proc. 3rd Tbilisi Symposium on Language, Logic, and Computation. Batumi, Georgia, Sept. 12–16, 1999; www.gelbukh.com/CV/Publications/1999/Georgia-Non-Adult.htm.
[Sidorov & al. 2000] G. O. Sidorov, I. A. Bolshakov, P. Cassidy, S. Galicia-Haro, A. F. Gelbukh: *"A Comparative analysis of the semantic field "non-adult" in Russian, English, and Spanish"* (in Russian). Proc. Annual Int. Conf. on Applied Linguistics Dialogue-2000, Moscow, Russia; www.gelbukh.com/CV/Publications/2000/Dialogue-2000-Non-Adult.htm.
[Sun, 2000] Sun Microsystems Inc.: *"Building International Applications"*, Sun product documentation http://docs.sun.com/db/doc/806-6663-01.
[Tomokiyo & Chollet, 2003] Mutsuko Tomokiyo, Gérard Chollet : "*Voice UNL: a proposal to represent speech control mechanisms within the Universal Networking Digital Language",* Proc. Convergences 2003, Alexandria, Egypt, Dec. 2003.
[Tsai, 2001] Wang-Ju Tsai : *"SWIIVRE - a Web Site for the Initiation, Information, Validation, Research and Experimentation on UNL (Universal Networking Language)",* First International UNL Open Conference, Suzhou, China, Nov. 2001.
[Tsai, 2002] Wang-Ju Tsai: *"A Platform for Experimenting UNL (Universal Networking Language)",* Workshop "First International Workshop on UNL, other Interlinguas and their Applications", Proc. LREC2002, Las Palmas, Spain, May 2002.
[Tsai, 2004] Wang-Ju Tsai: *"La coédition langue-UNL pour partager la révision entre langues d'un document multilingue",* thèse d'informatique, Université Joseph Fourier, juillet 2004.
[Uchida, 2001] Uchida Hiroshi, "*The Universal Networking Language beyond Machine Translation",* International symposium on language in cyberspace, Sept. 2001, Seoul, South Korea.
[Wheeler, 2003] David A. Wheeler: *"Secure Programming for Linux and Linux HOWTO",* http://www.dwheeler.com/secure-programs/, March 2003.

The UNL Initiative: An Overview

I. Boguslavsky, J. Cardeñosa, C. Gallardo, and L. Iraola

Artificial Intelligence Department,
Universidad Politécnica de Madrid, Spain
{igor, carde, carolina, luis}@opera.dia.fi.upm.es

Abstract. We present here a description of the UNL initiative based on the Universal Networking Language (UNL). This language was conceived to support multilingual communication on the Internet across linguistic barriers. This initiative was launched by the Institute of Advanced Studies of the United Nations University in 1996. The initial consortium was formed to support 15 languages. Eight years later, this initial consortium changed, many components and resources were developed, and the UNL language itself evolved to support different types of applications, from multilingual generation to "knowledge repositories" or cross- lingual information retrieval applications. We describe the main features of this UNL Language, making a comparison with some similar approaches, such as interlinguas. We also describe some organizational and managerial aspects of the UNL according to criteria of quality and maturity, placing emphasis on the fact that the initiative is open to any interested group or researcher.

1 Background

The UNL project has an ambitious goal: to break down or at least to drastically lower the language barrier for Internet users. With time and space limitations already overcome, the Internet community is still separated by language boundaries. Theoretically, this seems to be the only major obstacle to international and interpersonal communication in the information society. This is why the problem of the language barrier on the Internet is perceived as one of the global problems of mankind, and a project aimed at solving this problem has been initiated under the auspices of the UN, by the Institute of Advanced Studies of the United Nations University. Launched in November 1996, the project embraced 14 groups from different countries representing a wide range of languages: Arabic, Chinese, German, French, Japanese, Hindi, Indonesian, Italian, Mongolian, Portuguese, Russian, Spanish and Thai. Later on, Latvian and Korean were also included.

2 General Description of UNL

The idea of the project is as follows. A meaning representation language has been designed which has sufficient expressive power to represent the informational content conveyed by natural languages. This language, called the Universal Networking

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 377 – 387, 2005.

Language (UNL), was proposed by Dr. Hiroshi Uchida at the Institute for Advanced Studies of the United Nations University [1]. One of the major applications of UNL is to serve as an interlingua between different natural languages. In addition, UNL can also be used for other applications such as information retrieval, text summarization and the like.

For each natural language, two systems should be developed: a "deconverter", capable of translating texts from UNL to this NL, and an "enconverter" ,which converts NL texts into UNL. A deconverter and an enconverter for a language form a Language Server residing in the Internet. It is not necessary for all Language Servers to be based on the same linguistic framework or to use the same software. The only thing they have to share is the UNL. All language servers will be connected to the UNL network. They will allow any Internet user to deconvert a UNL document found on the web into his/her native language, as well as to produce UNL representations of the texts he/she wishes to make available to a multiethnic public.

2.1 What Is a UNL Expression?

Formally, a UNL expression is an oriented hypergraph that corresponds to a natural language sentence with respect to the amount of information conveyed. The arcs of the graph are interpreted as semantic relations of the type agent, object, time, reason, etc. The nodes of the graph can be simple or compound. Simple nodes are special units, known as Universal Words (UWs), which denote a concept or a set of concepts. A compound node (hypernode) consists of several simple or compound nodes connected by semantic relations.

In addition to propositional content (of the type "who did what to whom"), UNL expressions are intended to capture pragmatic information such as focus, reference, speaker's attitudes and intentions, speech acts and other information. This information is rendered by means of attributes attached to the nodes.

UWs, relations and attributes are the three building blocks of the UNL. We will not describe them in detail; an interested reader can find full specification of the UNL at [1]. Instead, we will give a general idea of what a UNL representation looks like by means of an example of average complexity and comments on some of the UNL features it demonstrates.

(1) *When people turn grey, they often look back to the past.*

(2) agt(look back(icl>do).@entry, people(icl>person).@generic)
obj(look back(icl>do).@entry, past(icl>abstract thing))
man(look back(icl>do).@entry, often)
tim(look back(icl>do).@entry, turn(icl>occur, equ>become))
obj(turn(icl>occur, equ>become), people(icl>person).@generic)
gol(turn(icl>occur, equ>become), grey(icl>color, mod<hair)

A UNL representation of a sentence is a set of relations. Each relation connects two UWs. In (2), the following relations are used: **agt** (agent), **obj** (object), **man** (verb modifier), **tim** (time), and **gol** (final state of a change). In total, UNL makes use of 41 relations of this type. Most of the UWs in (2) are supplied with restrictions (expressions in brackets following the headword). Restrictions mostly serve to indicate the semantic class of the UW and to restrict the meaning of the headword (we

shall say more about the restrictions below, in section 3). For example, restriction **(icl>do)** of *look back* shows that it is an action, as opposed to *turn*, which is a process (marked by restriction **(icl>occur)**). Restriction **(icl>person)** of *people* identifies the relevant meaning of this word and differentiates it from other possible interpretations, as, for example, in *peoples of South America*, where **(icl>nation)** is more appropriate. *Grey* has two restrictions: **(icl>colour)** says that *grey* is a kind of colour and **(mod<hair)** shows that it characterizes the colour of hair. Attribute **@entry** marks the "main" element of the structure (called "entry node") which normally corresponds to the syntactic top node of the corresponding part of the sentence. Attribute **@generic** ascribed to *people* shows that it is used in the generic sense.

Graph (2) is not a tree (cf. [2] where meaning is represented by means of a tree structure): UW **people(icl>person)** is the end node of two relations, **agt** and **obj**. This mostly happens when various predicates share the same argument (*people look back* and *people turn grey*). This is how the problem of anaphoric relations within the sentence boundaries is tackled. Instead of introducing anaphoric pronouns (*he, she, it, they*) for recurrent occurrences of the same word, UNL allows only one occurrence at the expense of the loss of tree structure.

Graph (2) does not contain compound nodes. A phrase should be represented by a compound node if its link to some element of the outer context is not semantically equivalent to the link of its entry node. In the sentence *The old man looked back to the past*, there is no need to introduce a compound node, because linking the phrase *the old man* to the verb *look back* is semantically equivalent to linking the noun *the man* to this verb: 'the old man looked back to the past' = 'the man looked back to the past; this man is old'. The situation is different in the sentence *Old men and women often look back to the past.* This sentence is ambiguous, with two possible interpretations: 'old men and old women' and 'old men and (some) women'. According to the first of these interpretations, the word *old* is linked to the whole phrase *men and women*, while according to the second, it is only linked to *men.* Thus, to assure proper understanding of the sentence, one has to introduce a compound node, *men and women*, for the first interpretation and leave single nodes, *men* and *women*, in the second.

2.2 Internal Organisation of UNL Documents

Information encoded in UNL is organised into UNL documents. A UNL document is divided into UNL paragraphs and sentences by means of HTML-like tags. UNL paragraphs consist of UNL sentences, which in turn are divided into sections. Each UNL sentence is divided into the following sections:

- The original sentence, i.e. the information that has been encoded.
- The UNL expression corresponding to the original sentence.
- For each language for which a UNL Generator has been developed, the automatically generated text of the UNL code into that language. Generation results are then cached in the document and available to the reader without delay.

Besides these elements, a UNL header contains information and meta-information about the document as a whole.

3 Some Salient Features of UNL as Compared to Other Interlinguas

After this brief and general presentation of UNL, we shall focus on some aspects of the UNL approach that we consider to be of primary importance.

However, before that, a preliminary remark is in order. UNL is a language which has a single author. On the other hand, several groups from various parts of the world have been working with and on this language for several years. They were working to a large extent autonomously, and for this reason they could and in fact did form somewhat different notions of UNL, the more so since there exists no "canonical" and detailed presentation of UNL by the author. Therefore, it should be emphasized that what follows is our personal view of UNL, which may not be shared by other partners, including the author of the UNL design. In our exposition, we shall try to answer some of the questions on the UNL approach that were asked at conferences and in private discussions (cf., in particular, [3], [4]).

3.1 UNL Versus Machine Translation

The first distinction between UNL and existing interlingua representations developed for MT is the fact that, from the very beginning, MT was not considered to be the only application for UNL. Conceived as a language for meaning representation, UNL is supposed to serve various NLP purposes. UNL representations can be created and stored regardless of subsequent generation into particular languages. UNL documents can be processed by indexing, retrieval and knowledge extraction tools without being converted to natural languages. Generation will only be needed when the document is going to be read by the human user.

Another important difference from conventional MT is that the procedure of producing a UNL text (= enconversion) is not supposed to be fully automatic. It should be an interactive process, with the labour divided between the computer and a human expert ("writer") in UNL. One way of doing this is described in [5]. Due to the interactive enconversion, the UNL expression, which serves as input for generation, can be as precise as one wishes to make it. The UNL writer will edit the rough result proposed by the enconverter, correct its errors, and eliminate the remaining ambiguities. He/she can run a deconverter of his own language to test the validity of the UNL expression obtained, and then refine it again until one is fully satisfied with the final result. Besides that, UNL experts from other language communities can further co-edit the UNL document, adding those components of meaning that were not represented in the NL source text (as, for example, information on definiteness in Slavic languages) [6].

3.2 UNL Versus English

An important idea behind UNL is (roughly) that it can be considered to a large extent to be "disambiguated English". A UNL representation of an utterance of any language can be regarded as an abstract structure of the English sentence conveying more or less the same meaning. This representation may be underspecified with respect to some elements of meaning that are not expressed in the source language.

Underspecification may concern meanings of both a grammatical nature (such as determination in Slavic languages, aspect in French or number in Japanese) and a lexical one (an example will be given below).

3.3 Dictionary of Universal Words Versus Ontology

UWs are organized in the Knowledge Base (KB), which is a manually constructed network of UWs connected mostly by relations of hyponymy/hyperonymy, synonymy and meronymy (part/whole). Nevertheless, it is not an ontology in the strict sense of the word. The crucial difference between the UNL KB and existing ontologies is that it is often a matter of principle for ontology developers to make it as language-independent as possible [2, 7, 9]. Elements of the ontology are concepts that are abstracted from the meanings of concrete, natural language words. In our opinion, UNL does not have this ambition. UNL KB is language-neutral only at the upper levels of the hierarchy. Such labels as "thing" (standing for any nominal entity), "abstract thing", "living thing", "do" (standing for any verbal concept denoting an action or an activity), "occur" (standing for any verbal concept denoting a process) or "be" (standing for any verbal concept denoting a state or a property) can hopefully subsume concepts of any language equally well. However, as far as the terminal leaves of the hierarchy are concerned, UNL KB **it does not attempt to be language-neutral**. Instead, **the UWs situated at the lower levels of the KB hierarchy are a collection of word senses of all working languages.** Each lower level UW corresponds to a word sense of some working language (or a union of such word senses). This feature distinguishes UNL not only from the existing ontologies mentioned above, but also from various meaning representation languages in which lexical meanings are decomposed by means of a small set of semantic primitives [10, 11, 12, 13].

An approach similar to ours is that used in EuroWordNet [14]. On the one hand, it collects and links syntsets of several working languages which are obviously language-specific. On the other hand, all these syntsets are linked to the Top Ontology via the Inter-Lingual-Index. The links between the Top Ontology and the elements of the Inter-Lingual-Index provide some language-independent structuring of the latter. The difference between the EuroWordNet approach and the one adopted in the UNL project is that in UNL, semantic links are not established between the word senses of individual languages but only between the UWs. This means for example that the Spanish words *manzana* 'apple' and *fruta* 'fruit' are not linked to one another directly but only by means of the UWs to which they correspond: **apple(icl>fruit)** vs. **fruit(icl>plant)**.

To show how it works, let us consider another example. In Russian, there is no neutral equivalent of the English non-causative verb *to marry* as represented in sentences such as *John married Ann in June.* The expression that exactly corresponds to this English verb – *vstupat' v brak* ('to contract a marriage') – is an official term and is not used in everyday life. Instead, Russian speakers make use of two different expressions: *zhenit'sja*, if the agent of the action is a male, and *vyxodit' zamuzh*, if it is a female. Since the English and the Russian words differ in their meaning, they generate different UWs. The UW for English *to marry* looks like (1), while Russian expressions have UNL equivalents with a narrower meaning – (2) and (3), respectively (for simplicity's sake, only the relevant fragments of the UWs are given):

(1) marry(agt>human)
(2) marry(agt>male)
(3) marry(agt>female)

Suppose that the generators of various languages received, as input, a UNL that originated in Russian and contained UW (2). If the target language has an equivalent word sense (as, for example, Polish or Ukrainian), the generator will have no problem at all. It will make the exact match. If we have to generate the text in English (or French, German, Spanish), where this exact concept is not represented, the generator will search in the KB for the nearest concept with a more general meaning for which the target language has a direct equivalent, that is (1). Cf. a similar mechanism for searching for an alternative translation in the Interlingua lexicon in [13].

If English is the source language, the UNL expression contains UW (1), and exact matches will be found by French, German or Spanish generators. If the target language is Russian, Polish or Ukrainian, we shall have to make a difficult choice between two different equivalents. This is the problem of lexical underspecification that faces any translator, human or machine, from English into these languages. Sometimes the immediate context provides sufficient clues to make a decision (as in *John married Ann in June*). In other cases, this is more difficult and not even an ontology can help (as in *If one wishes to marry, one has to take it seriously*). Interactive disambiguation seems to be the only feasible solution here.

We would like to single out three distinctive features of the UNL dictionary organization.

1. **Flexibility.** There is no fixed set of semantic units. There is only a basic semantic vocabulary that serves as building material for the free construction of derivative lexical units with the help of restrictions. This makes it possible to balance, to some extent, the non-isomorphic nature of lexical meanings in different languages.
2. **Bottom-up approach.** The UNL dictionary consisting of Universal Words is not constructed a priori, top-down. Since it should contain lexical meanings specific to different languages, it grows in an inductive way. It receives contributions from all working languages. Due to this, one can expect the linguistic and cultural specificity of different languages to be represented more fully and more adequately than would be possible following the top-down approach.
3. **Knowledge base.** As the UNL dictionary comprises unique semantic complexes lexicalized in different natural languages, we face the task of bridging the gap between them. It can be done by means of the Knowledge Base – a network of UNL lexical units connected by different semantic relations. Special navigation routines will help to find the closest analogue to a lexical meaning not represented in the given language.

3.3 Types of Information Available for UWs

In a general case, UWs are supplied with several types of information, though it is far from complete (cf. a more elaborate and richer knowledge structure in the Mikrokosmos ontology [15], [16]). Most of this information is represented by restrictions.

- Hyponymy/hyperonymy, synonymy and meronymy relations which locate UW in the KB ("KB-restrictions"). Examples: **September(icl>month)** ["September" is a hyponym of "month"], **month(pof>year)** ["month" is part of "year"], **wood(equ>forest)** ["wood" is synonymous with "forest"].
- Information on the argument frame and selectional restrictions ("argument restrictions"). Example: **buy(agt>volitional thing, obj>thing, src>thing, cob>money)** ["buy" has an agent ("who buys?"), an object ("what is being bought?"), a source ("from whom?") and a co-object (second object; "for how much?")].
- Information on how the meaning of the headword is restricted ("semantic restrictions"). Example: UW **land(icl>do, plt>shore)** describes the act of landing on the shore. It is necessary to differentiate this type of landing from the landing of an aircraft, if only because in Russian these situations require different verbs. This information is obligatory only if KB and argument restrictions are insufficient to identify the meaning. For example, *leg* translates differently into Russian if it is the leg of a human or of some other animal (*noga*), the leg of an insect (*lapka*), part of the furniture (*nozhka*) or of a journey (*etap*). The UWs for these four concepts are clearly opposed by the KB meronymy restrictions (**leg(pof>living thing)**, **leg(pof>insect)**, **leg(pof>furniture)**, **leg(pof>journey)**) and therefore do not require any semantic restrictions. The same is true for the UWs given under (1) – (3) above.
- Examples and/or comments that are introduced when the UW is not self-evident. Examples: **leg(pof>living thing)**: *He broke his leg*, **leg(pof>insect)**: *A fly has three pairs of legs,* **leg(pof>furniture)**: *the leg of a table*, **leg(pof>journey)**: *the first leg of a round-the-world flight.*

When used in UNL representations, UWs do not bear all these restrictions. Each UW has a short form which unambiguously refers to the full description in the KB.

3.4 Universal Words Versus NL Words

If a NL word has several meanings, they should correspond to different UWs. In this sense, UNL is lexically disambiguated. Disambiguation is done by means of restrictions which have to be constructed in such a way as to clearly differentiate between different meanings. Examples: **leg(pof>living thing)**, **leg(pof>journey)**.

UNL does not strive to reduce quasi-synonyms to a single UW. If, however, the semantic difference between two words can be naturally grasped by means of restrictions or attributes, one can make use of the same headword, but the UWs should still be different. For example, it is acceptable to describe *jungle* and *taiga* as local varieties of a forest: *jungle* = **forest(icl>plant, plc>tropics)** ['forest situated in the tropics'], *taiga* = **forest(icl>plant, plc>Siberia)** ['forest situated in Siberia']. Another possibility is to treat each of the quasi-synonyms as a headword: **jungle(icl>forest, plc>tropics)**, **taiga(icl>forest, plc>Siberia)**.

3.5 Universal Words Versus English Words

The fact that UNL can be regarded as disambiguated English does not mean that all UWs correspond to English word senses. UNL has a certain capability to construct

concepts absent in English. To be more precise, when a lexicographer of language *L* seeks to produce a UW for a word sense W_L, he/she has the following choices:

1. If there exists an English word sense W_E which is to a reasonable extent synonymous with W_L, then the UW for W_E is accepted for W_L;
2. If English has no direct equivalent of W_L, then the lexicographer should try to construct a concept by means of semantic restrictions. Namely, one has to find the closest English word sense with a more general meaning and narrow it down. Example: in Spanish there are different words for fish when considered as a living being or as food; *pez* – **fish(icl>living thing)**, *pescado* – **fish(icl>food)**. This is a powerful way of overcoming mismatches between the languages, although it is obviously not universal.

It should be emphasized that semantic restrictions are not always required to describe lexical meaning in detail. Their role is twofold. First, they should help lexicographers identify the appropriate sense of the English headword in order to find an equivalent in their language. To fulfil this role, restrictions should only be sufficient to separate one English word sense from the other, e.g. **handicraft(icl>concrete object)** vs. **handicraft(icl>activity)**. All UNL lexicographers are expected to have a proficient command of English, and since a word sense is unambiguously separated from other senses of the same headword, its meaning should be completely clear.

Second, semantic restrictions are needed to construct a new concept absent in English. This function is more demanding. The more completely a concept is characterized by restrictions, the more precise will be its translations to other languages. In practice, however, UWs are seldom supplied with more than two or three semantic restrictions. In all cases in which the meaning of UW is not transparent, it should be supplemented by an example and/or comment.

4 Current State

After seven years of UNL of technical and institutional development, the results have been heterogeneous. Several parameters can help us to visualize the current situation.

Dissemination. UNL has been presented at several international conferences and workshops including COLING 2000, NAACL-ANLP 2000, LREC 2002, COLING 2004 and some others. Currently, we have begun compiling all kinds of written resources accumulated so far (technical reports, papers, UNL corpora, etc.) with a view to their open dissemination.

Institutional support. From the very beginning, the UNL project has been institutionally supported by the UN, first through the UN University (Tokyo) and then through the UNDL Foundation. The UN is the assignee of the UNL patent, a fact that guarantees the open character and free use of UNL (http://www.undl.org/).

Technical developments. At the moment, UNL deconverters exist for many of the working languages, although their quality varies. Prototypes of enconverters are being built for some of the languages (Spanish, Russian). Several useful tools for UNL

developers have been developed, such as the UNL Workbench, constructed by the Spanish group and the tool for compiling the UNL dictionary, constructed by the Italian group.

Real Experiences. UNL modules and tools for several languages (Spanish, Italian, Russian, French) have been tested in two experiences dealing with texts belonging to the domain of cultural heritage [17].

Marketable solutions. Right now the project is defining the market opportunities for UNL technology to be competitively applied. The most promising solution at the moment seems to be providing multilingual translation services, which may be profitable if the number of working languages is five or more.

Consortium. Over the last few years, the activities of the initial consortium of 14 language groups have slowed down. Some groups have suspended their participation due to financial and organizational problems. At this moment (fall, 2004) an attempt is being made to revitalize these activities with a view to providing multilingual services within two years for a group of languages (Spanish, Russian, Italian, Hindi, Portuguese, French, English and maybe some others). This initiative was taken by the Spanish and Russian Language Centers and supported by most of the groups.

4.2 UNL Marketable Applications and Usefulness

UNL applications go beyond the supporting of multilingual translation services. UNL could also be applied to cross-lingual information retrieval, multilingual transactional systems or even what are known as "knowledge repositories", where the knowledge is stored in the form of conceptual graphs without original/target language dependency. More information about marketable applications of UNL can be found in [18] and [19].

However, the market implementation of a technology requires much more than the technology itself. An important aspect of any technology is its maturity [20]. Maturity criteria can be divided into two groups: intrinsic and organizational factors. The first group of factors is connected to technological aspects and has been commented upon above. The second group comprises five factors, which we shall briefly characterize below.

1. **Organizational maturity**. There is an international organization – the UNDL Foundation – for the coordination of technological development and exploitation and technology transfer.
2. **Management**. There are two levels of management: the UNDL Foundation, at the top, and Language Centres responsible for all activity involving local languages coordinated by the Technical and Quality Committees.
3. **Openness.** The UNL initiative is completely open. Anybody can participate and have full access to all UNL documentation and materials. All details can be found at http://www.undl.org.
4. **Products and services offered.** At this moment the organisation is making an inventory of the UNL resources available and defining a business plan in order to offer multilingual services in 2006.
5. **Active organization.** Since 1996, research and development on UNL has been carried out by various Language Centres. Results of the R&D activities are

reported at the annual UNL conferences and at other congresses. Some Language Centres have opened UNL deconverters for experimental exploitation on the Internet (cf., e.g., www.unl.fi.upm.es, www.unl.ru, http://unl.ilc.pi.cnr.it).

5 Conclusions

UNL is a general purpose, meaning representation language that can be used in a variety of multilingual applications. One of the major applications is multilingual MT or generation in which UNL can serve as an interlingua. UNL generators (deconverters) and supporting tools have been created for a number of languages. UNL activities include R&D, technology transfer, dissemination, training and promotion. These activities are carried out by Language Centres associated to form a world-wide infrastructure.

The UNL initiative is open to constructive criticism, new ideas, and, above all, new languages and new partners.

References

1. Uchida, H. *The UNL Specifications*, v.3.2. (2003) http://www.undl.org
2. Busemann, S.: "Issues in generating text from interlingua representations". In *Proceedings of the 1st International Workshop on UNL, other Interlinguas and their Applications.* Las Palmas, Spain (2002)
3. http://crl.nmsu.edu/Events/FWOI/ThirdWorkshop/FinalPapers/martins.dorr.html
4. http://crl.nmsu.edu/Events/FWOI/ThirdWorkshop/FinalPapers/martins.hovy.html
5. Boguslavsky, I., Iomdin, L, Sizov, V. "Interactive enconversion by means of the ETAP-3 system". *Proceedings of the International Conference on the Convergence of Knowledge, Culture, Language and Information Technologies, Convergences'03*. Alexandria (2003)
6. Boitet, C. A. "Rationale for using UNL as an Interlingua". *Proceedings of the First International Workshop on UNL, other Interlinguas and their Applications*. International Conference on Language Resources and Evaluation (2002)
7. Beale, S., S. Nirenburg & K. Mahesh. "Semantic Analysis in the Mikrokosmos Machine Translation Project". In *Proceedings of the Second Symposium on Natural Language Processing*. Bangkok, Tailand (1995)
8. J.A. Bateman. "Upper Modeling: Organizing Knowledge for Natural Language Processing". In *5th International Workshop on Natural Language Generation*, Pittsburgh, PA., (1990)
9. Knight, K. and Luk, S. "Building a Large-Scale Knowledge Base for Machine Translation"., In *Proceedings of the 12th National Conference on Artificial Intelligence (AAAI'94)*, Seattle, USA (1994)
10. Schank, R.C. "Conceptual Dependency: A Theory of Natural Language Understanding", *Cognitive Psychology*, Vol 3 (1972) 532-631
11. Wilks, Y.: "Preference Semantics". In E. Keenan, (ed.) *The Formal Semantics of Natural Language*. Cambridge: Cambridge U. P. (1973)
12. Dorr, Bonnie J.: *Machine Translation: A View from the Lexicon*. MIT Press, Cambridge, MA (1993)

13. Dorr, B.J. and Voss, C.R.: "A Multi-Level Approach to Interlingual MT: Defining the Interface between Representational Languages". In Iwanska, L. and Shapiro, S. (eds.): *Natural Language Processing and Knowledge Representation: Language for Knowledge and Knowledge for Language*. AAAI Press/MIT Press (2000)
14. Vossen, P., Bloksma, L., Rodriguez, H., Climent, S., Calzolari, N. et al.: "The EuroWordNet Base Concepts and Top Ontology". *Deliverable D017, D034, D036, WP5. EuroWordNet, LE2-4003*, Amsterdam. (1998)
15. Mahesh, K. "Ontology Development for Machine Translation: Ideology and Methodology". NMSU. Computing Research Laboratory. *Technical Report MCCS-965-292*. New Mexico. (1996)
16. Mahesh, K. & S. Nirenburg "A Situated Ontology for Practical NLP", in *Proceedings of the Workshop on Basic Ontological Issues in Knowledge Sharing. International Joint Conference on Artificial Intelligence (IJCAI-95)*, Montreal, Canada (1995).
17. European Commission. "HEREIN Project (IST-2000-29355)". *Final Report* (2003)
18. Cardeñosa, J., Gallardo C., Iraola L., Tovar E.: "A multilingual approach for web applications: The UNL system". In *Proceedings of the 8th World Multi-Conference on Systemics, Cybernetics and Informatics*. Orlando. Florida (2004).
19. Cardeñosa, J., Iraola, L., Tovar, E.: "UNL: a Language to Support Multilinguality in Internet". In *Proceedings of the International Conference on Electronic Commerce. ICEC2001*.Vienna, Austria (2001)
20. Cardeñosa, J., Tovar, E.: "A Descriptive Structure to Assess the Maturity of a Standard: Application to the UNL System". In *Proceedings of the 2nd IEEE Conference on Standardization and Innovation in Information Technology*. Boulder. Colorado. USA (2001)

Interactive Resolution of Intrinsic and Translational Ambiguity in a Machine Translation System*

Igor M. Boguslavsky[1,2], Leonid L. Iomdin[1],
Alexander V. Lazursky[1], Leonid G. Mityushin[1], Victor G. Sizov[1],
Leonid G. Kreydlin[1], and Alexander S. Berdichevsky[3]

[1] Institute for Information Transmission Problems, Russian Academy of Sciences
Bolshoj Karetnyj pereulok, GSP-4,
Moscow 129994, Russia
{iomdin, lazur, mit, sizov, lenya}@iitp.ru
http://proling.iitp.ru
[2] Madrid Polytechnic University, Faculty of Informatics
Campus de Montegancedo S/N
28660 Boadilla del Monte, Madrid, Spain
igor@opera.dia.fi.upm.es
[3] Moscow State University, Moscow, GSP-2, Moscow 119992, Russia
alexberd1983@mail.ru

Abstract. The paper presents the module of interactive word sense disambiguation and syntactic ambiguity resolution used within a machine translation system, ETAP-3. The method applied consists in asking the user to identify a word sense, or a syntactic interpretation, whenever the system lacks reliable data to make the choice automatically. In lexical disambiguation, part of man-machine dialogue refers to the analysis phase, while the other part is activated during transfer. For this purpose, entries of the working dictionaries of the system are supplemented with clear diagnostic comments and illustrations that enable the user to choose the most appropriate option and in this way channel the course of system operation.

1 Introductory Remarks. ETAP-3 Overview

ETAP-3 is a full-scale rule-based machine translation system that serves Russian-English and English-Russian pairs and has a number of small prototype modules for Russian-German, French-Russian, Russian-Korean, Russian-Spanish and Arabic-English translation. The MT system is developed as part of a multipurpose linguistic processor at the Laboratory of computational linguistics, Institute for Information Transmission Problems in Moscow [1-4]. Other modules of the processor include a parsing tool for deep syntactic tagging of text corpora, a UNL enconverter and deconverter tool, and several smaller-scale components (a module of synonymous paraphrasing of sentences, syntax checker, and a computer-assisted language learning tool).

* This work was supported by a grant (No. 02-06-80085) from the Russian Foundation of Basic Research, whose assistance is gratefully acknowledged.

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 388 – 399, 2005.

ETAP-3 is based on the general linguistic framework of the Meaning ⇔ Text theory (MTT), proposed by Igor Mel'čuk e.g.[5], complemented by the concept of systematic lexicography and integrated description of language proposed by Jurij Apresjan [6]. However, the classic MTT was somewhat reduced and modified for ETAP-3. In particular, instead of the surface and deep levels of syntactic representation, the system uses a level largely corresponding to surface MTT syntax and a level of normalized syntactic structures in which syntactic relations remain the same but much of national specificity of the source language is removed (see below 1.2 for details).

1.1 Morphological Analysis

ETAP-3 processes written texts and translates them sentence by sentence. Every source language sentence is first morphologically analyzed, which means that every word is assigned a deep morphological representation, i.e. the lemma furnished with inflectional characteristics. If a word is morphologically and/or lexically ambiguous, it is assigned a set of morphological representations. Morphological analysis does not take into account any word context, so no lexical or morphological ambiguity is resolved at this stage. The sequence of all morphological representations of the words of a sentence is its morphological structure (MorphS).

The morphological module uses vast morphological dictionaries (the Russian dictionary counts 130,000 lemmas amounting to several million word forms, and the English dictionary counts 85,000 lemmas), and a computationally efficient finite-state software engine. The morphological analyzer is able to parse compound words absent in the dictionary, like English *bioterrorism* or *quasielastic* and Russian *odinnadcatimetrovyj* 'eleven-meter' or *neftepererabotka* 'oil processing'.

1.2 Parsing

The MorphS of the source sentence is processed by a small pre-syntactic module, which partially resolves lexical and morphological ambiguity using information on close linear context. To give a simple example, if the ambiguous word *lead* is preceded by an article, its verbal interpretation is excluded from further consideration. The partially disambiguated MorphS of the source sentence is sent to the parser – the system's most important and sophisticated part.

The parser transforms the MorphS of the sentence into a dependency tree structure. The tree nodes correspond to the words of the sentence processed, whilst the directed arcs are labeled with names of syntactic relations, or SR. The parsing algorithm creates a dependency tree from the linear MorphS using s y n t a g m s , or rules that produce minimal subtrees consisting of two nodes linked by a labeled directed arc. The set of syntagms comprises several hundred rules for each of the two main working languages, written in a specially designed formalism, FORET. Normally, every syntagm describes a specific binary syntactic construction (e.g. nominal subject plus verbal predicate as in *war stinks*, noun plus adjectival modifier, as in *fair play*, numeral plus noun, as in *seven seas*, etc.).

Parser operation consists of several phases. Syntagms create for the given MorphS all possible syntactic links, using all kinds of linguistic and contextual information available. At subsequent phases of parsing extraneous links are eliminated with the help of several filtering mechanisms.

To optimize the parsing process, syntagms are arranged into three types: general syntagms operating on each sentence processed, template syntagms referred to in dictionary entries of restricted word classes, and dictionary syntagms located directly in the entries of syntactically salient words (auxiliaries, conjunctions etc.). This type of rule arrangement is applied in most ETAP-3 phases and modules.

If a sentence is lexically and/or syntactically ambiguous, the parser is able to produce several syntactic structures (SyntS) corresponding to different readings.

An important innovation introduced to parsing theory and practice by ETAP-3 is a mechanism of the detection of the top node that resorts to empirical preference rules. The mechanism consists of several rules (applied after all hypothetical links have been formed) that assign weights to all words of the sentence depending on their likelihood to be the tree top, which is estimated by part-of-speech attribution of each word, its syntactic categorization, presence of sentential markers like conjunctions or connective words, and types of links established. Normally, this mechanism channels the course of parsing in the correct direction and improves the overall performance of the system.

Other recent innovations in the ETAP-3 parser include a system of empirical weights dynamically assigned to the elements of the dependency tree at earlier stages of the parsing process [7] and a module of preference rules based on statistics learned from syntactically annotated corpora [8].

The ready SyntS is sent to the SyntS normalization module that is used to strip the SyntS structure of some of the specific features of the source language. Typical normalization rules merge into single nodes verbal expressions formed with auxiliaries, remove from SyntS strongly governed prepositions and conjunctions, etc. The output of the normalization module is called Normalized Syntactic Structure, or NormS.

1.3 Transfer

The transfer proper is performed at the level of NormS, which provides sufficient control of sentence semantics as many of the SSRs are semantically motivated and the nodes carry semantic data inherited from the combinatorial dictionaries of the source language. As a result of the transfer phase operation, the NormS of the source language is replaced by a NormS of the target language, in which all nodes represent the words of the target language and the arcs are labeled with target SSR names. In a way, the NormS is a sort of tradeoff between the two level syntax of MTT and the complexity of the system.

The target NormS is sent to a refinement module which fulfils operations inverse to the ones performed by the normalization module. In particular, it generates analytical verb forms, introduces strongly governed prepositions and conjunctions and ensures the correct word order of the target sentence. The resulting expanded target SyntS is almost ready for the next-to-last phase of translation – syntactic synthesis, which produces the lacking morphological features (as required by agreement or government rules) and prepares ground for the final phase of translation – morphological generation that uses the target morphological dictionary to generate real word forms and produce the target sentence.

1.4 Combinatorial Dictionaries

Combinatorial dictionaries are slightly reduced (they provide no lexicographic definitions) but fully formalized versions of explanatory combinatorial dictionaries (ECD) of Mel'čuk's Meaning ⇔ Text theory. The dictionaries are highly reusable; in particular, the Russian combinatorial dictionary is used as the source dictionary in the Russian-to-English translation and as the target dictionary in the opposite direction of translation. For the English combinatorial dictionary, the reverse is true. Currently, each of the two dictionaries contains about 85,000 lexical entries.

An entry of the combinatorial dictionary contains, in addition to the lemma name, information on syntactic and semantic features of the word, its subcategorization frame, a default translation, rules of various types, and values of lexical functions for which the lemma is the keyword. The word's syntactic features characterize its ability (or inability) to participate in specific syntactic constructions. A word can have several syntactic features selected from a total of more than 200 items. Semantic features are needed to check the semantic agreement between the words in a sentence.

2 Ambiguity: Persistent Problem

Despite many innovations and rapid advances in both rule-based and statistics-based NLP systems, the disambiguation problem remains a stumbling block for such systems, especially those in which identification of meaning is essential. Recently, much effort has been expended to solve the problem with purely automatic means.

On the one hand, disambiguation techniques have been resorting to more and more sophisticated data supplied in lexical and grammar resources of NLP systems, such as fine-grained constraints on using specific word senses or special rules targeted towards selecting the correct word sense or syntactic interpretation in clearly stated contextual environments. This is what most systems of machine translation are doing, and ETAP-3 has been no exception. Obviously enough, such efforts have their natural limits as they require immense amount of time and labor. Besides, many ambiguous cases cannot in principle be resolved in this way, as they require extralinguistic knowledge far beyond the scope of what can be extracted from texts alone.

On the other hand, significant progress has been achieved in the development of statistical methods designed to disambiguate word senses and trained on large text corpora. This has recently been confirmed by the contributions to the Senseval-3 Workshop held within the framework of the Annual ACL Meeting in Barcelona, as well as papers presented to Coling 2004. Such an approach seems to be more promising; still, characteristically enough, even the most sophisticated statistical techniques (see e.g. [9]) show that the maximum degree of word sense disambiguation achieved on parallel corpora do not exceed 75% – which is impressive but still far from sufficient.

It seems that fully automatic procedures, including the most efficient ones, cannot ensure reliable resolution of linguistic ambiguity.

3 Interactive Disambiguation: A Promising Solution

In both approaches listed above, human participation in text processing is confined to preliminary stages (pre-editing) and final stages thereof (post-editing). The approach we have been developing lately takes a different perspective. The human is expected to intervene into text processing in the very heart of the interpretation stage. In machine translation, such a human must know the source language, whereas the command of the target language is not necessary (though of course it can do no harm). In a particular case, such a human may well be the author interested in translating his text into a language with which he is not familiar.

This idea was first put forward some 25 years ago: as evidenced by W. Hutchins [10], American MT systems, ALPS and Weidner in Provo, Utah, used interactive disambiguation of English in early 1980s. Maruyama *et al.* [11] reported the use of the technique for Japanese. The idea was then elaborated in detail by Christian Boitet and Hervé Balchon in Grenoble [12-13]. Similar ideas can also found in [14]. Since then, it has been promoted by a number of research groups in a variety of NLP systems, including 1) the LIDIA dialogue-based machine translation system by the GETA group in Grenoble; 2) the multilanguage MT system SYSTRAN; 3) the ALT-J/E system by NTT Communication Science Laboratories of Japan; 4) the UMIST MT system in Manchester, 5) a system of spoken and written translation by the Spoken Translation group in the USA, 6) a system of multilingual search and Internet navigation by DFKI and the University of Saarland in Germany, and many others.

The first system of full-scale interactive disambiguation in NLP for Russian and English was started by the ETAP group in 2002 and has been rapidly progressing since then.

3.1 Lexical Disambiguation

The main idea of the project has been to provide the human expert operating the MT system with clear and simple diagnostic descriptions of ambiguous lexical units that could be viewed at certain phases of text processing. The analysis algorithm has been modified in such a way as to take into account the choices made by the expert and, accordingly, suppress other options that contradict these choices – (possibly, temporarily, in case the choices are incorrect and lead to system failure).

Several points in the algorithm have been specified at which the computer expert opinion is expected: (1) immediately before the parser starts the top node selection; (2) immediately after all syntactic hypotheses generated by syntagms have been checked; (3) immediately before translation options are to be chosen.

As of today, almost 15,000 Russian ambiguous lexical units that share their lemma names (or wordforms) with other lexical units were selected and supplied with diagnostic comments and examples. Information used in these diagnostic comments includes 1) an analytical definition of the word sense, or its important fragment; 2) part of speech tags, which can in case of need be supplemented by simple syntactic features; 3) reference to the word's synonyms and/or opposites. Examples are chosen in such a way as to maximally facilitate word sense identification by the expert.

Optionally, English translation equivalents are supplied for more advanced system users or experts. All information is presented in the respective entries of the Russian combinatorial dictionary. With the help of these comments, many types of lexical ambiguity can be resolved.

At present, the ETAP team is starting a new phase of the project that envisages a similar treatment for the English dictionary. A list of 20,000 ambiguous lexical units of English has been prepared, for which diagnostic comments and examples are being developed.

Importantly, this technique is implemented in a system that strives to obtain all possible analyses of each sentence, rather than choosing just one, even if it is the most probable of all. This approach is motivated by the fact that the system is viewed as a testing ground for a specific theoretical model of language and, insofar as it is possible, must take into account all interpretations that the language allows.

Naturally enough, such an approach narrows the scope of statistical methods applicable in disambiguation. Even though the system has a host of techniques, which can be used to suppress less probable interpretations, we are wary to use them. To be more exact, we want the system to operate in two modes: (a) automatic mode that makes maximum use of probabilistic considerations and discards less probable interpretations at early stages and (b) comprehensive interactive mode, in which the objective is to find any adequate interpretation. In this latter mode of operation, statistic considerations are not discarded altogether but are downplayed a bit.

Another important aspect of our approach is the fact that we treat diferently intrinsic ambiguity of the source language and translational ambiguity. This distinction may be disregarded in a system that only serves one pair of languages but it gains in importance in a multilanguage environment. Indeed, some cases of ambiguity must be dealt with regardless of the target language: thes can be exemplified by ambiguous English sentences like *he made a general remark,* see below, or Russian sentences like *muzhu izmenjat' nelzja* ('A husband must not be unfaithful' vs. 'One must not be unfaithful to one's husband'). Other ambiguities only arise when we translate something into a particular language. To give a simple example, we should not distinguish between *fish* as animal and *fish* as food when translating from English into Russian but we must do so when translating into Spanish, where *pez* is an animal fish and *pescado* is fish eaten as food. Similarly, we activate ambiguity resolution when translating the Russian adjective *razlichnyj* into English (*different* vs. *various*) and do not activate it when translating into German (*verschieden*).

Since these types of ambiguity are of different character, they are dealt with at different stages of sentence processing: intrinsic ambiguity is treated during analysis whilst translation ambiguity is resolved in transfer. If the distinction is neglected and both types are treated simultaneously, we will have to burden the description of the source language with all ambiguities of all working languages, which is tedious and highly unnatural. Conversely, if, we postpone intrinsic disambiguation until transfer, we will miss a good opportunity to discard wrong readings in other parts of the text under treatment.

To the best of our knowledge, ETAP-3 is the only system that clearly distinguishes between these types of ambiguity.

We will now give a couple of examples to illustrate the interactive mode of MT system operation in both directions of translation.

We will start with a short Russian sentence,

(1) *Soldat tochno vypolnjal prikazy komandira.*
Soldier carried out orders commander

This sentence is highly ambiguous due to the fact that the second word, *tochno*, has no less than four distinctly different senses: TOCHNO1 (an adverb meaning 'precisely'), TOCHNO2 (an adverb meaning 'definitely', 'by all means'), TOCHNO3 (a comparative conjunction meaning 'as though') and TOCHNO4 (a comparative particle meaning 'like'). All four lexical units are supplied with succinct comments and examples in the combinatorial dictionary. At least three of these senses (TOCHNO1, TOCHNO2 and TOCHNO4) may be in place in (1). If the interactive module of lexical disambiguation of ETAP-3 is on, the human expert will be offered a dialogue window (see Figure 1) and in this way given the chance to choose a word sense which he considers appropriate.

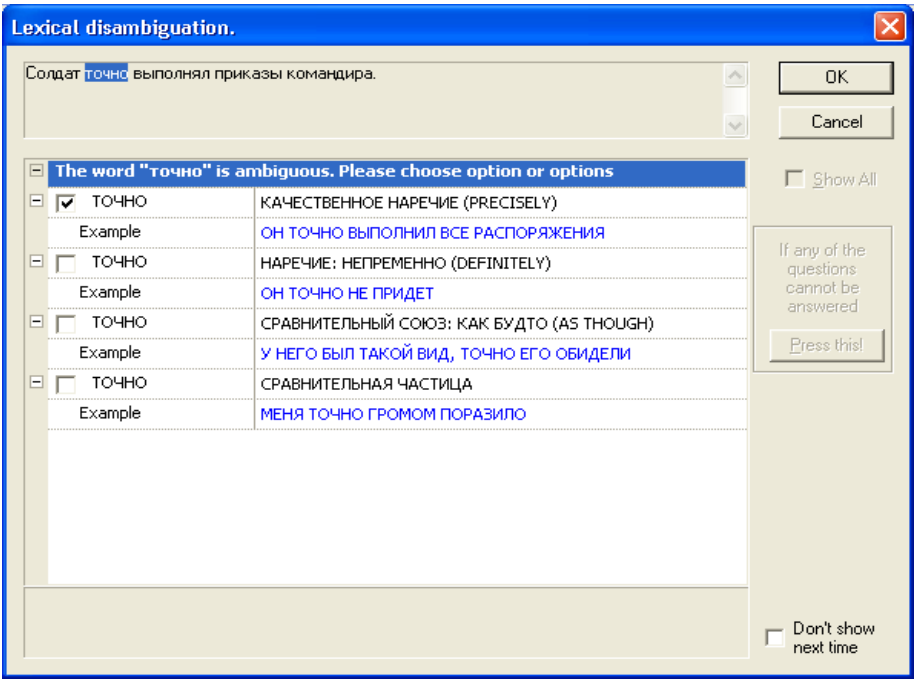


Fig. 1. Dialogue window enabling word sense disambiguation in ETAP-3 for Russian

Using the comments and examples as guidelines, the expert is able to make a reasonable choice and in this way direct the subsequent operation of the system, which eventually lead to the adequate translation. In case of sentence (1), should the first option be chosen, the generated structure will correspond to the interpretation (1a)

The soldier carried out the commander's orders precisely. If option 4 is chosen, the obtained syntactic structure will trigger a nontrivial transformation during the transfer phase, which will yield the translation (1b) *It looked like the soldier carried out the commander's orders*. Finally, if the expert highlights option 2, the system will generate the structure corresponding to the interpretation (1c) *The soldier definitely carried out the commander's orders.*

The remaining option 3 (the word *tochno* as a comparative conjunction) is impossible and will be discarded by the system automatically.

As it happens, the fact that the first reading of *tochno* 'precisely' is more frequent in sentences like (1) does not validate the rejection of other readings – because of the principle of multiple interpretation stated above.

Let us now consider an example of lexical disambiguation in English. It is easy to see that sentence

(2) *He made some general remark that everything was fine*,

is ambiguous between (at least) two interpretations: (2a) 'he made some (army) general say that everything was fine' and (2b) 'he made some general observation that everything was fine'. In fully automatic operation, the ETAP-3 MT system yields for sentence (2) two different SyntS (see Fig: 2 and 3), which correspond to the two interpretations.

In Figure 2, *general* is a noun and *remark* is a verb. In Fig. 3, *general* is an adjective and *remark* is a noun. Accordingly, in Fig. 2 the noun *general* serves as the first complement of the verb *make* while the verb *remark* is its second complement (thus creating a complex object construction); cf. labels **1-compl** and **2-compl** on the corresponding links. In Fig. 3, the adjective *general* modifies the noun *remark* (as shown by the **modif** label on the link that connects *general* to *remark*).

Even though ETAP-3 is able to identify this ambiguity, it cannot in the general case automatically decide which of the options is appropriate in a particular context.

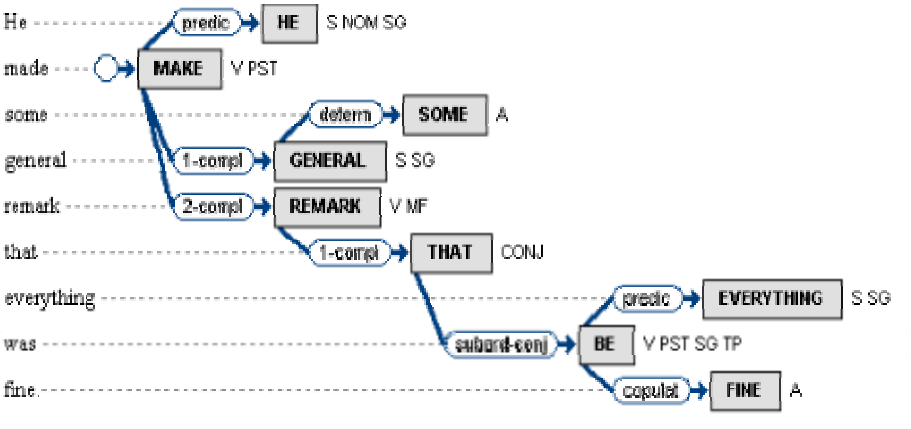


Fig. 2. SyntS for the first reading of sentence (2)

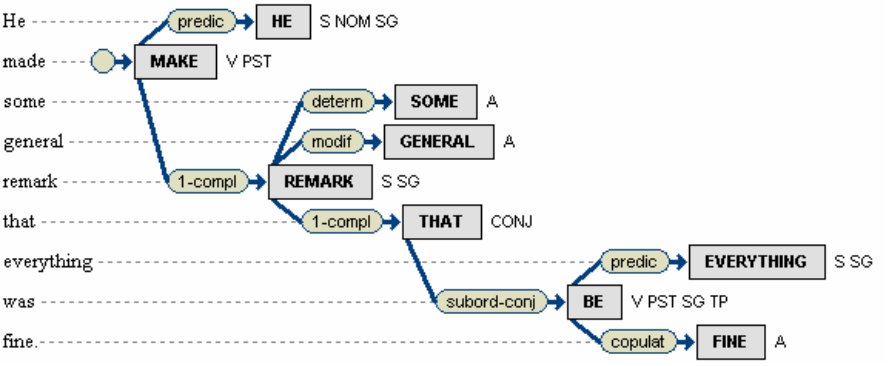


Fig. 3. SyntS for the second reading of sentence (2)

Let us now see what happens if we resort to interactive disambiguation. As in the previous case, the expert will be offered a dialogue window in which he or she has to choose between lexical readings (Fig. 4). If the adjectival reading of *general* and the noun reading of *remark* is chosen, the parser will build the SyntS presented in Fig. 3. Subsequently, the transfer phase will generate the corresponding translation.

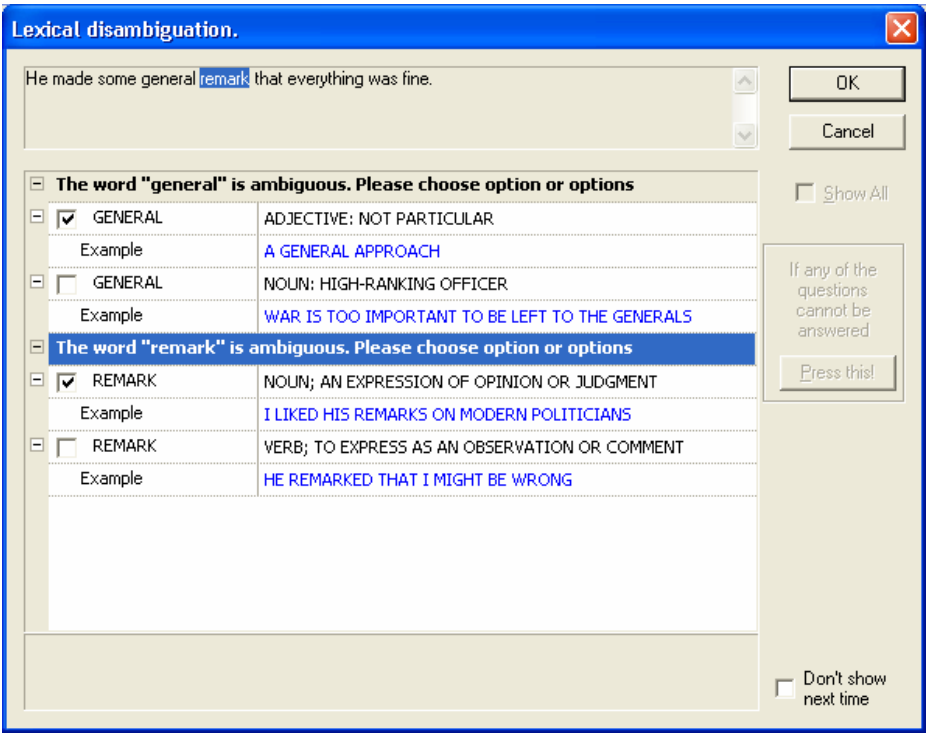


Fig. 4. Dialogue window enabling word sense disambiguation in ETAP-3 for English

Importantly, the module of **lexical** disambiguation helped to solve a rather complicated **syntactic** ambiguity in (2) without actually asking the expert anything about the syntactic structure. Such side effects contribute to broadening the scope of WSD potentials.

It must be emphasized that the lexical disambiguation module can help in far less trivial situations. In the practice of ETAP-3 operation, the system had to translate the following subheading from a recent article on the BBC website:

(3) *AIDS threatens economic collapse*.

For a human, the meaning of sentence (3) is perfectly clear: it says that 'AIDS endangers (probably, some country) with economic collapse'. In the meantime, the MT system is very likely to understand (3) in an entirely wrong way as 'AIDS poses a threat to economic collapse', and, consequently, yield a wrong translation, for the simple reason that the system lacks the resources needed to distinguish the syntactic structure of (3) from that of the sentence

(4) *AIDS threatens economic prosperity.*

Indeed, in order to make sure that (3) is parsed correctly, the system must know that the noun *collapse* instantiates the **instrumental** valency slot of the verb *to threaten* (whatever its sense) and not the **object** slot as in (4). However, to provide adequate word lists for different slots of particular verbs is virtually impossible because, among other things, such lists will inevitably intersect. Cf. ambiguous phrases like *threaten changes*, *threaten a revolution*, or *threaten tax reforms*: unlike *economic collapse,* which is universally viewed as an undesirable event or *economic prosperity,* which is definitely positive, *changes, revolutions,* or *tax reforms* may be viewed both positively and negatively. Such an assessment is exactly what a human expert familiar with the text can easily do if asked for a prompt. We do not believe that such cases of intrinsic ambiguity may be successfully solved by statistical methods at all, because this would require collection of data virtually unavailable in any type of linguistic resources (dictionaries or corpora).

3.2 Syntactic Disambiguation

It goes without saying that, for the interactive disambiguation system to be really powerful it must also be able to provide the human experts with prompts regarding syntax. This is a difficult task, because average users may readily disambiguate word senses but are normally unprepared to answer questions about the syntax.

The ETAP team is currently investigating possible approaches to the solution of interactive syntactic disambiguation problem. In the meantime, even now the system provides such a module for **an insider,** i.e. a specialist who is well familiar with the particular syntactic module of the ETAP environment. (On a broader scale, this disambiguator module can be used by experts who are specially trained in the system and use it professionally).

Originally, the syntactic disambiguator only offered the user a chance to channel the processing of a sentence by defining whether this was a full verbal sentence or a nominal phrase. This is especially relevant in certain types of English sentences like *Structure changes* ≈ 'changes of the structure' vs. 'arrange the changes' vs. 'the

structure is changing' or *Cleaning mechanisms* ≈ 'mechanisms for cleaning' vs. 'how to clean mechanisms': such sentences may, with comparable probabilities, be parsed as noun groups or verbal phrases. By determining the type of sentence the user triggered the choice of the top node detection rules.

Currently the module offers a dialogue that enables the human user to choose among syntactic hypotheses, expressed in terms of labeled binary subtrees. The method is especially effective if lexical disambiguation and syntactic disambiguation are applied to a sentence simultaneously.

We will confine ourselves to giving one example, which however is illustrative enough to enable the assessment of the scope and effect of the module as well as the amount of effort needed to use it. The Russian sentence

(5) *Odna iz samyx perspektivnyx oblastej nauchnyx issledovanij – nanotexnologii (texnologii, operirujushchie velichinami porjadka nanometra – nichtozhno maloj velichiny, sopostavimoj s razmerami atoma) – naxoditsja v Rossii poka v zachatochnom sostojanii,*

which has a significant amount of lexical ambiguity and syntactic homonymy, could be processed by ETAP-3 in a fully automatic mode. However, the translation it yielded had some errors and required over 7 minutes of computer time (on a 2 Ghz Pentium 4 computer having 512 Mb RAM).

In striking contrast to that, the use of the lexical and syntactic disambiguation module required one minute of time of an experienced linguist who had to answer about 20 questions, took 1.22 minutes (of which about 1 minute was taken by the linguist), and yielded a far better translation.

(5a) *One of the most promising domains of the scientific investigations - nanotechnologies (technologies, operating with the values of the order of the nanometre - the negligibly small value, comparable to the sizes of the atom), - is in the rudimentary state in Russia so far.*

The results obtained by the development of the two disambiguation modules within the ETAP-3 system so far are encouraging. Future directions of research in this area will focus on the elaboration of means enabling people untrained in the system to use the module of syntactic disambiguation.

References

1. Apresjan Ju.D., Boguslavskij I.M., Iomdin L.L., Lazurskij A.V. Pertsov, N.V., Sannikov, V.Z., Cinman, L.L. Lingvističeskoe obespečenie sistemy ETAP-2. (The linguistics of the ETAP-2 MT system.) Moskva, Nauka (1989) 295 p. (In Russian.)
2. Apresjan Ju.D., Boguslavskij, I.M., Iomdin, L.L., Lazurskij, A.V., Mitjushin, L.G., Sannikov, V.Z., Cinman, L.L. Lingvisticheskij processor dlja slozhnyx informacionnyx sistem. [A linguistic processor for advanced information systems.] Moskva, Nauka (1992) 256 p. (In Russian).
3. Apresjan, Ju.D., Boguslavskij, I.M., Iomdin, L.L., Lazurskij, A.V., Sannikov, V.Z. and Tsinman L.L. Système de traduction automatique {ETAP}. La Traductique. P.Bouillon and A.Clas (eds). Montréal, Les Presses de l'Université de Montréal. (1993).

4. Apresian, Ju.D., Boguslavsky, I.M., Iomdin, L.L., Lazursky, A.V., Sannikov, V.Z, Sizov, V.G., Tsinman, L.L. ETAP-3 Linguistic Processor: a Full-Fledged NLP Implementation of the MTT. // MTT 2003, First International Conference on Meaning – Text Theory. Paris, École Normale Supérieure, Paris, June 16–18, 2003, pp. 279-288.
5. Mel'čuk I.A. Opyt teorii lingvisticheskix modelej klassa "Smysl - Tekst". [The theory of linguistic models of the Meaning – Text Type]. Moscow, Nauka. (1974) (In Russian).
6. Apresjan, Ju. D. Systematic Lexicography. Oxford University Press (2000) XVIII p., 304 p.
7. Iomdin, L.L, Sizov, V.Z, Tsinman, L.L Utilisation des poids empiriques dans l'analyse syntaxique: une application en Traduction Automatique. META, 47. (3). (2002) 351-358
8. Boguslavskij I.M., Iomdin L.L., Sizov, V.G., Chardin, I.S. Ispolzovanie razmechennogo korpusa tekstov pri avtomaticheskom sintaksicheskom analize. (Using a syntactically tagged corpus of texts in automatic syntactic analysis).// Proceedings of the International Conference "Cognitive Modeling in Linguistics 2003. Varna (2003). 39-48.
9. Tufis D., Ion, R., Ide, N. Fine-Grained Word Sense Disambiguation Based on Parallel Corpora, Word Alignment, Word Clustering and Aligned Wordnets. // Proceedings of the 20th International Conference on Computational Linguistics, Geneva, August 23–27, 2004, pp. 1312-1318.
10. Hutchins W. Machine translation: past, present, future. Ellis Horwood, Chichester (1986).
11. Maruyama H., Watanabe, H., and Ogino, S. An interactive Japanese parser for machine translation. // Karlgren, H., ed. Proceedings of the 13th International Conference on Computational Linguistics, v. 2. Helsinki. (1990). 257-62
12. Boitet, C. & Blanchon, H. Multilingual Dialogue-Based MT for monolingual authors: the LIDIA project and a first mockup. // Machine Translation, 9/2 (1994), 99-132.
13. Blanchon, H. An Interactive Disambiguation Module for English Natural Language Utterances. // Proceedings of NLPRS'95. (Seoul, Dec 4-7, 1995), vol. 2/2: 550-555.
14. Goodman, K. and Nirenburg, S. (ed.). The KBMT Project: A case study in knowledge-based machine translation. Morgan Kaufmann Publishers. San Mateo, California. (1991). 330 p.

Chinese-Japanese Clause Alignment

Xiaojie Wang [1] and Fuji Ren [2]

[1]School of Information Engineering,
Beijing University of Posts and Telecommunications,
Beijing, China, 100876
[2]Department of Information Science & Intelligent Systems,
Tokushima University,
Tokushima, Japan
xjwang@bupt.edu.cn, ren@is.tokushima-u.ac.jp

Abstract. Bi-text alignment is useful to many Natural Language Processing tasks such as machine translation, bilingual lexicography and word sense disambiguation. This paper presents a Chinese-Japanese alignment at the level of clause. After describing some characteristics in Chinese-Japanese bilingual texts, we first investigate some statistical properties of Chinese-Japanese bilingual corpus, including the correlation test of text lengths between two languages and the distribution test of length ratio data. We then pay more attention to *n*-*m*(*n*>1 or *m*>1) alignment modes which are prone to mismatch. We propose a similarity measure based on Hanzi characters information for these kinds of alignment modes. By using dynamic programming, we combine statistical information and Hanzi character information to find the overall least cost in aligning. Experiments show our algorithm can achieve good alignment accuracy.

1 Introduction

Text alignment is an important task in Natural Language Processing (NLP). It can be used to support many other NLP tasks. For example, it can be utilized to construct statistical translation models (Brown et al. 1991), and to acquire translation examples for example-based machine translation (Kaji et al. 1992). It can be helpful in bilingual lexicography (Tiedemann 2003). It is also used to improve monolingual word sense disambiguation (Diab and Resnik 2002).

The approaches to text alignment can be classified into two types: statistical-based and lexical-based. The statistical-based approaches rely on non-lexical information (such as sentence length, co-occurrence frequency, etc.) to achieve an alignment task. As illustrated in the research of Gale and Church (1991) for the sentence-level alignment, they start from the fact that the length of a source text sentence is highly correlated with the length of its target text translation.

The method proposed in Kay and Roscheisen (1993) is based on the assumption that in order for the sentences in a translation to correspond, the words in them must also correspond. Their method makes use of lexical anchor points to lead an alignment at the sentence level.

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 400 – 412, 2005.

It has been shown that different language pairs are in favor of different information in alignment. For example, Wu (1994) found that the sentence-length correlation between English and Chinese is not as good as between English and French. Also, there is less cognate information between Chinese-English pair than that in English-French pair, while the alignment of Chinese-Japanese pair can make use of information of Hanzi commonly appearing in both languages (Tan and Nagao 1995). Currently, most methods rely on either or both of above two ideas (Veronis 2000). The approaches combining both length and lexical information, such as Melamed (2000), seem to represent the state of the art.

Standing on text alignment at sentence-level, structure-based alignment has been paid more and more attentions (Matsumoto et al. 1993, Wu 1997, Ding and Palmer 2004). Since bi-trees can bring more information and thus more helpful to machine translation. However, due to the limitation of parsers, especially for those non-English languages, current structure-based alignment cannot deal with complex or compound sentences well. Average length of Chinese sentences in test is about 12 Chinese words (Ding and Palmer 2004). Comparing with our bi-text corpora where there are nearly 25 words per Chinese sentence and 10 words per Chinese clause, we can reasonably believe that an alignment at the level of clause is more manageable for current parsers rather than sentence.

Some works have been done at the level of clause (Kit et al.2004) and phrase (Venugopal et al. 2003). As noted in Kit et al. (2004), a full Chinese sentence is often a compound, including several propositions. While in clause level, units are usually well-formed clauses (including only one proposition) and phrases or even words. It is thus more manageable for current parsers to do further structure-based alignment.

In this paper, we present our current work on Chinese-Japanese bilingual alignment at the level of clause. For Chinese-Japanese pairs, as noted in Tan and Nagao (1995), because of the different sentential structure, alignment at the level of sentence could sometimes result in pairing of quite a number of sentences en masse in both texts. This exacerbates burden for parsers. We will show that this problem is less severe in the level of clause.

Combination of both length-based information and lexicon-based information is proved to be the state of art approach to bi-text alignment (Veronis 2000). So we will also combine these two kinds of information in our alignment algorithm. In Chinese and Japanese pairs, up to now, there is no systematical investigation on length correlation between two languages. On the lexical information, Hanzi characters, which commonly occur in both languages, have been used for improving sentence alignment. But there are very different conclusions on the affects brought from Hanzi information. Tan and Nakao (1995) achieve a great improvement by combining Hanzi information with length information, and the accuracy is raised from 78% to 96% for their less than 500 Chinese and Japanese test sentences which are mostly from texts. But in Zaiman et al. (2001), authors draw a very different conclusion for the affect of Hanzi information. We will also dress these questions in this paper.

The remainder of paper is organized as follows: section 2 describes some characteristics in Chinese-Japanese bilingual corpora, and show what we can get in a clause level alignment. Section 3 describes several information sources we use in our approach, and how to combine these information sources for improving alignment. Where, we investigate the length correlation between two languages and the distribution of length ratio. We especially concern about the alignment modes of *n-m* (*n*>1 or *m*>1), which are well known to be prone to mismatch. We construct a deliberate measure to deal with these kinds of alignment. We then implement several experiments and evaluate different effects brought from different information sources in section 4. Finally we draw some conclusions.

2 Some Characteristics Chinese-Japanese Bi-texts

As noted in Tan and Nagao (1995), Chinese and Japanese have different sentential structures. One of these differences is the use of periods (including question marks and exclamation marks) to end sentence. If we use periods as the end of sentence, there will be quite a number of sentences en masse between both texts at sentence-level alignment. That is, there will be some alignments involving *n*>1 Chinese sentences or *m*>1 Japanese sentences. These kinds of alignment modes are usually more difficult for an algorithm to manage and fallible. They thus propose to allow a sentence in one text to be matched with a part of the sentence in the other text if possible in order to create a finer correspondence pair. But in their works, they only allow a sentence ended with periods to be matched with the sequence of clause and/or phrase ended with break points in the other text, rather than clause to clause alignment. It does reduce the *n-m*(*n*>1 or *m*>1) alignment at the level of sentence, but, on the other hand, as we will see in following example in Table 1, it will cause that *n*>1 Chinese clauses are aligned with one Japanese sentence or *m*>1 Japanese clauses are aligned with one Chinese sentence.

Table 1. An example of Chinese-Japanese bi-text: one Chinese sentence is aligned with two Japanese sentences

Chinese text	Japanese text
幸好，刀子小，拇指的骨头硬，所以直到今天指头还连在手掌上，不过那伤痕到死也不会消失。	幸ナイフが小さいのと、親指の骨が堅かったので、今だに親指は手に付いている。然し創痕は死ぬまで消えぬ。

In Table 1, one Chinese sentence is aligned with two Japanese sentences. If we allow Japanese sentence to be aligned with clauses in Chinese, then we can get some new alignments as listed in Table 2.

Table 2. When a Japanese sentence can be aligned with clauses in Chinese

Chinese text	Japanese text
幸好，刀子小，拇指的骨头硬，所以直到今天指头还连在手掌上，	幸ナイフが小さいのと、親指の骨が堅かったので、今だに親指は手に付いている。
不过那伤痕到死也不会消失。	然し創痕は死ぬまで消えぬ。

Since the aligned units in Chinese are clauses which marked by either commas or periods, while the units in Japanese is marked by periods, the first alignment in Table 2 is that 4 Chinese units are aligned with one Japanese unit. Comparing with alignment in Table 1 which includes at most two units in an alignment, this alignment unites more units in Chinese, thus more easily to expose to mismatch. Beside of that, texts on both sides are still a compound, including several propositions.

If we allow clause to clause alignment, where both languages can use break points as the end of a unit, we will have alignments as in Table 3 for the same bi-text. Where, one alignment is 2-1 and others are 1-1. The biggest lumped unit includes 2 clauses, which is same as that in sentence-level, but here we get several alignments in finer grain.

Table 3. Clause to Clause alignment

幸好，刀子小，	幸ナイフが小さいのと、
拇指的骨头硬，	親指の骨が堅かったので、
所以直到今天指头还连在手掌上，	今だに親指は手に付いている。
不过那伤痕到死也不会消失。	然し創痕は死ぬまで消えぬ。

The above example is not a singular situation. A statistics from manually aligned 251 bi-text paragraphs in our corpus shows that although there are more than 4 percents pairs including more than 2 units in either side in both sentence-level and clause-level bi-text, there are only 0.4 percents of pairs including more than 3 units at either side of clause-level alignment, while the percent is nearly 2 in sentence-level alignment. As often noted, *n-m* alignment modes (n>3 or m>3) are easily exposed to mismatch in alignment. Less such pairs, thus, make it more easily to be matched.

Also, as we have noticed, the clause-level alignment is more manageable for a parser to do further processing. We have some statistics in Table 4 to show a contrast between sentences and clauses in our Chinese-Japanese corpus.

Table 4. Average numbers of characters in sentences and clauses

	Chinese characters per unit	Japanese characters per unit
Sentence	25.5	35
Clause	10	16

Based on above two reason, that is, it is easier to deal with in alignment itself and easier for further processing (such as parsing). We thus think it is more useful and hopeful to do alignment at level of clause.

There is another characteristic we will utilize in our Chinese-Japanese alignment. For some kinds of corpus, such as novels which we currently work on, there are lots of dialogs quoted directly. There are often several clauses even sentences in these direct quotations. To align them at clause-level, we should break the quotations, but in Chinese-Japanese pair, it often involves re-orders of clauses. For example, let we consider the bi-text in Table 5.

If we do not break the quotation marks for texts in Table 5, they can be aligned as in Table 6 including several clauses or sentences in each of the alignments.

Table 5. An example of bi-text includes direct quotation

Chinese Text	Japanese Text
我看她依然带着奇怪的表情，就问：“我买点什么土产回来送你呢？你要什么来着？”她说：“想吃越后的竹叶糖。”	それでも妙な顔をしているから「何かみやげに買って来てやろう、何が欲しい」と聞いてみたら「越後の笹飴が食べたい」と云った。

When we break the quotations, we should re-order the clauses in order to get correct alignments as in Table 7. We have a special pre-process to manage these re-orders in

Table 6. An alignment when quotations are not broken

Chinese text	Japanese text
我看她依然带着奇怪的表情，就问：“我买点什么土产回来送你呢？你要什么来着？”	それでも妙な顔をしているから「何かみやげに買って来てやろう、何が欲しい」と聞いてみたら
她说：“想吃越后的竹叶糖。”	「越後の笹飴が食べたい」と云った。

Table 7. An alignment when quotations are broken

Chinese text	Japanese text
我看她依然带着奇怪的表情，	それでも妙な顔をしているから
就问：	と聞いてみたら
“我买点什么土产回来送你呢？	「何かみやげに買って来てやろう、
你要什么来着？”	何が欲しい」
她说：	と云った。
“想吃越后的竹叶糖。”	「越後の笹飴が食べたい」

3 The Approach to Clause Alignment

We use dynamic programming to find overall optimal alignment paragraph by paragraph. We combine both length-based information and Hanzi-based information to measure the cost for each possible alignment between Chinese and Japanese strings.

Before we give the measure function, we first define a few notations. Let s_i denote the i th clause in Chinese, $| s_i |$ denote the number of character it includes, s_{ij} denote the string from the i th clause to the j th clause in Chinese text, t_u denote the u th clause in Japanese, t_{uv} denote the string from the u th clause to the v th clause in Japanese text. $j < i$ means that s_{ij} is an empty string, and so does t_{uv} . Let $d(i,u;j,v)$ be the function that computes the cost of aligning s_{ij} with t_{uv} . We divide $d(i,u;j,v)$ into three parts, as shown in (1), to reflect that we utilize three information sources to compute the cost.

$$d(i,u;j,v) = L(i,u;j,v) + M(i,u;j,v) - \alpha H(i,u;j,v) \quad (1)$$

Where $L(i,u;j,v)$ depends on length ratio of the paired two strings s_{ij} and t_{uv}, different length ratios cause different $L(i,u;j,v)$; $M(i,u;j,v)$ depends on the alignment mode in the pair, that is, depending on how many clauses involved in this pair at both sides, different *n-m* modes cause different $M(i,u;j,v)$. $H(i,u;j,v)$ is the contribution from Hanzi characters common in both strings.

We first describe how to use length-based information, and then Hanzi-based information. We give a deliberate measure on multiple alignments which are prone to mismatch.

3.1 Length and Mode

To utilize the length information in Chinese-Japanese alignment, we make an investigation on length correlation between these two languages. We also check if the normal hypothesis of the length ratio is appropriate for this language pair. Following Gale and Church (1991), we use alignment data at paragraph-level to do the correlation and normal test. 4893 paragraphs in our Chinese-Japanese corpus are randomly chosen for these tests.

The correlation test is to examine if the lengths of Chinese texts are related to their Japanese translations. That is, if longer texts in Chinese tend to be translated into longer texts in Japanese, and if shorter texts tend to be translated into shorter texts. Figure 1 shows that the lengths (in characters) of Chinese and Japanese paragraphs are highly correlated.

In Gale and Church (1991), they based their distance measure on an assumption that each character in one language gives rise to a random number of characters in the other language. They assume those random variables are independent and identically distributed with a normal distribution. They check the English-French pair for this assumption, and estimate parameters for the normal distribution. However, this assumption is not checked in Chinese-Japanese pair. For making use of the same kind

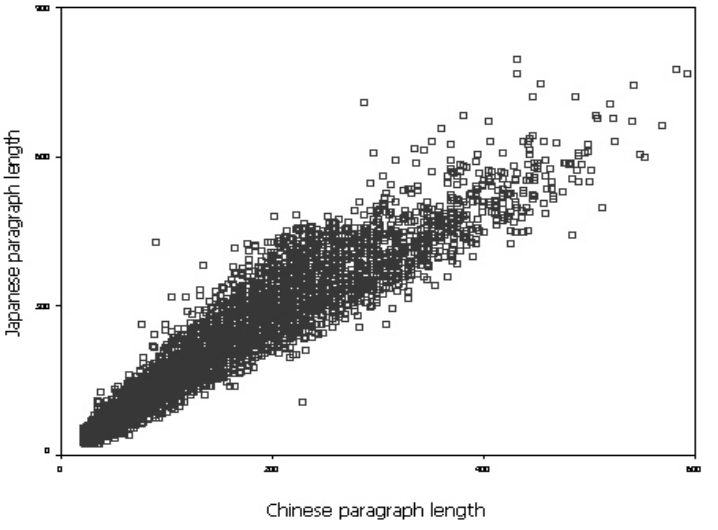


Fig. 1. Paragraph lengths are highly correlated in Chinese-Japanese bi-texts. The horizontal axis shows the length of Chinese paragraphs, while vertical scale shows the lengths of the corresponding Japanese paragraphs. The Pearson correlation (.948) is significant at the 0.01 level (2-tailed)

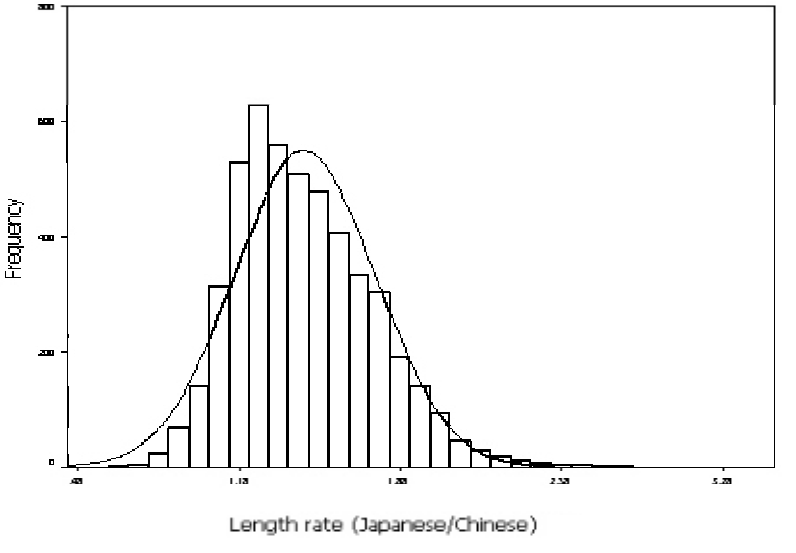


Fig. 2. The length-rate data of Chinese-Japanese pair is approximately normal. The horizontal axis shows the length-ratio (Japanese/Chinese), while vertical scale shows the frequency of each ratio. The histogram fits normal distribution approximately with mean=1.47, Std. Deviation=0.31, kurtosis=3.85, skewness=0.99

of length-based information in alignment, we thus give an investigation on if this assumption can be held in Chinese-Japanese pair. We also use the paragraph-based data to do the test. We use 4893 length ratios from 4893 Chinese-Japanese paragraphs as samples. Figure 2 is the histogram for these ratios.

We can find that length ratio data is approximately normal with mean μ =1.47 and Std. Deviation σ =0.31. Then following Gale and Church (1991), we compute $L(i,u;j,v)$ using equation (2).

$$L(i,u;j,v) = -100 * \log(2 * (1 - \Pr(|(|t_{uv}|/|s_{ij}| - \mu)/\sigma|))) \quad (2)$$

In (2), $L(i,u;j,v)$ is set to 0 when both strings s_{ij} and t_{uv} are empty, $L(i,u;j,v)$ is set to a given maximum value when only one of these two strings is empty. The maximum value is set to 2000 in our experiments.

To compute the cost for different alignment mode, we aligned 928 pairs at the level of clause manually. Then we estimate the probabilities of different alignment modes using frequencies of these modes occurring in data. The result is listed in Table 8.

Table 8. The frequencies of different alignment modes

Category (*n-m*)	Frequency f(*n-m*)	Probability Pr(*n-m*)
0-1 or 1-0	1	0.001
1-1	623	0.671
1-2	74	0.080
2-1	162	0.175
2-2	30	0.032
Others	38	0.041
Total	928	1.000

Suppose that $j - i = n$ means the string s_{ij} including n Chinese clauses, specially, we use $n = -1$ to denote that there is no clause in s_{ij}. Similarly, $v - u = m$ means the string t_{uv} including m Japanese clauses and $m = -1$ means that there is no clause in t_{uv}, the cost for mode $n - m$ can be computed by (3).

$$M(i,u;j,v) = -100 * \log(\Pr(n - m)) \quad (3)$$

3.2 Hanzi Information

Another source of information is Hanzi occurring commonly in both Chinese and Japanese. We include totally 2626 Chinese-Japanese Hanzi character pairs in a dictionary. We construct different contribution measures as in equation (4) and (5) for different alignment mode. The equation (5) is for 2-2 mode, and the equation (4) is for other modes. By including some length-related information in these measures, we can also control the alignment length.

$$H(i,u;j,v)=\begin{cases}0 & j<i \mid v>u & 0-1 or 1-0\\ h_1(i,0;u,0) & j=i \,\&\, v=u & 1-1\\ h_1(i,0;u,0)+h_2(i,j;u,0) & j=i+1 \,\&\, v=u & 2-1\\ h_1(i,0;u,0)+h_3(i,0;u,v) & j=i \,\&\, v=u+1 & 1-2\end{cases} \quad (4)$$

Where $h_1(i,0,u,0)=\mid s_i \cap t_u \mid / \min\{s_i, t_u\}$, here we use the relative number of commonly occurring Hanzi to measure the similarity between two strings, $h_2(i,j,u,0)=(\mid s_j \cap t_u \mid - \mid s_i \cap s_j \cap t_u \mid)/s_j$, which is the relative number of commonly occurring Hanzi only in s_j and t_u but not in s_i, similarly, we use $h_3(i,0,u,v)$ to denote $(\mid s_i \cap t_v \mid - \mid s_i \cap t_u \cap t_v \mid)/t_v$, which is the relative number of commonly occurring Hanzi only in s_i and t_v but not in t_u. For 2-2 mode, we use (5) as a measure.

$$\begin{aligned} H(i,u;j,v)=\min\{ \quad & h_1(i,0;u,0)+h_2(i,j;u,0)+h_1(0,j;0,v),\\ & h_1(i,0;u,0)+h_3(i,0;u,v)+h_1(0,j;0,v),\\ & h_1(i,0;0,v)+h_1(0,j;u;0)\} \end{aligned} \quad (5)$$

The three items in the right of equation (5) are the measures for three different ways of merging s_i and s_j and matching with t_u and t_v in Chinese-Japanese alignment. Comparing with previous measures, such as those in Tan and Nagao (1995), we think these measures are more reasonable. Also, by using these measures, we can not only manage different alignment modes such as contraction, expansion and merger, but also take length information into consideration.

This completes the description of distance measure. We finally use dynamic programming to find the overall least cost in matching. Let $D(i,u)$ be total distance aligning clauses from first one to i th in Chinese and clauses from first one to u th in Japanese, the recurrence then can be described in (6).

$$D(j,v)=\min\{D(i-1,u-1)+d(i,j;u,v)\} \quad (6)$$

4 Experiments and Evaluations

We use 251 Chinese-Japanese paragraph pairs in our experiments, which include total 2760 Chinese clauses and 2482 Japanese clauses respectively. If all the bilingual clauses are aligned correctly, it should produce 2161 pairs of translation examples at the level of clause. 100 paragraphs are originally in Chinese, and others are originally in Japanese. We have got all the parameters in last section except coefficient α in distance measure. To estimate this parameter, we align another 41 paragraphs which include 427 Chinese clauses and 381 Japanese clauses. We use it as training data in our experiments when α is needed.

All language data used in this paper are drawn from a Chinese-Japanese bilingual corpus which includes more than 3 millions of Chinese characters and more than 4 millions of Japanese characters. There are 31 contemporary novels written by either Chinese or Japanese authors, including totally more than 70,000 paragraph-based alignments.

We first implement some experiments to investigate the affects brought from different information sources and different combinations of these sources. The results are listed in Table 9.

There are 86.02% of pairs aligned correctly when all information is used. As we have mentioned, if all the bilingual clauses are aligned correctly, there are more than 4% of pairs including more than 2 units in either side of bi-text. Considering that our current program cannot manage these pairs, the accuracy of 86.02 percents is reasonably good. Also, 8.39% of pairs are partly aligned correctly, while only 5.59% of pairs are completely wrong.

From Table 9, we can find Hanzi information (H) is the most helpful one for alignment. When only one information source is used, H achieves the best accuracy, which at least gains 25% more than other two kinds of information. Under the same conditions, H always achieves more improvements than others. Combining H with mode information(M) improves M's initial accuracy from 34.52 to 78.84, while length information(L) raises it to 71.81. Combining H with L improves L's initial accuracy from 40.36 to 83.18, while M raises it to 71.81. When two of the three kinds of source are +, by combining H, we get an improvement from 71.81 to 86.02, more than 10 points, while combination of L improves the accuracy from 78.84 to 86.02, combination of M only contributes less than 3%, from 83.18 to 86.02.

Table 9. How different factors effect the accuracy (+: used, −: not used)

H	L	M	Accuracy(%)
+	-	-	66.65
+	+	-	83.18
+	-	+	78.84
+	+	+	**86.02**
-	+	-	40.36
-	-	+	34.52
-	+	+	71.81

Table 10. A Chinese-Japanese pair aligned by using a normal measure

Chinese text	Japanese text
当时，勘太郎无路可逃， 拼命扑过来。	その時勘太郎は逃げ路を失って、 一生懸命に飛びかかって来た。

Table 11. A Chinese-Japanese pair aligned correctly by using our measures

Chinese text	Japanese text
当时，勘太郎无路可逃，	その時勘太郎は逃げ路を失って、
拼命扑过来。	一生懸命に飛びかかって来た。

We also implement an experiment where $H(i,u;j,v)$ is just the number of Hanzi characters commonly occurring in both sides of bi-text under consideration. We name this $H(i,u;j,v)$ a normal measure for Hanzi information. When other information sources are same, the best accuracy for normal measure is 76.77%. There is nearly 10% improvement achieved by our $H(i,u;j,v)$. Our proposal shows its advantage in leveraging Hanzi information for different alignment modes. A normal measure tends to lump small units into the long string which may include more common Hanzi characters. For example, bi-text in Table 10 is aligned by using a normal measure. It can be aligned correctly using measure (5) as in Table 11.

Finally, we find that the accuracy is in fact sensitive to parameters of length and mode, but can be improved significantly by adjusting coefficient of Hanzi information in the cost function (1).

Some works have been done for Chinese-Japanese Sentence alignment. Tan and Nagao (1995) achieved 96% accuracy on less than 500 sentences which are mostly selected from text. They also included 20% (20 out of 100 messages) sentences used for parameter estimation in their test data. While in Zaima et al. (2001), the accuracy is less than 60% at the level of sentence. No previous work is reported on clause-level alignment. For comparative, we also implement a sentence-level alignment for the same 251 paragraphs. We achieve 87.28% accuracy on 1189 Chinese sentence and 1158 Japanese sentences. Clause-level alignment based on results of sentence-based alignment has an accuracy of less than 75%.

5 Conclusions

This paper describes a Chinese-Japanese alignment at the level of clause by combining both length information and Hanzi character information. We give a detail description on some characteristics in Chinese-Japanese bilingual texts. We check the correlation between the lengths of Chinese text and Japanese text, and find that length ratio data fits normal hypothesis approximately. We pay a special attention on *n-m* alignment where *n* or *m* is greater than 1, and propose a similarity measure based on Hanzi information to get better accuracy than normal one. Experiments show our proposal is very helpful. We believe that the proposed similarity measures can be also helpful for the alignment of other language pairs by using other lexical information instead of Hanzi information in Chinese-Japanese pairs.

We will extend our program to manage *n-m* alignments where *n* or *m* is bigger than 2 in our future works. By analyzing the new alignment modes such as 1-3 , 2-3 or 3-3 etc., we will build Hanzi information based measures to achieve higher accuracy for Chinese-Japanese alignment.

Acknowledgements. This work has been partly supported by the Education Ministry of Japan under Grant-in-Aid for Scientific Research B (No. 14380166). Part of this work was done while the first author was at Nara Institute of Science and Technology (NAIST). He appreciates Professor Yuji Matsumoto and the Matsumoto laboratory in NAIST very much for lots of help he had received, including permitting him to use the Chinese-Japanese corpus.

References

1. Peter F. Brown, Jennifer C. Lai and Robert L. Mercer. Aligning Sentences in Parallel Corpora. In *Proc. of 29th Conf. of Assoc. for Comput. Linguistics*, ACL-1991. pp.169-176.
2. Mona Diab and Philip Resnik. An Unsupervised Method for Word Sense Tagging using Parallel Corpora. In *Proceedings of the 40th Annual Conference of the Association for Computational Linguistics,* ACL-2002, pp.255-262.
3. Yuan Ding and Martha Palmer Automatic Learning of Parallel Dependency Treelet Pairs, In *proceedings of the First International Joint Conference on Natural Language Processing* ,IJCNLP-2004, pp.30-37.
4. William A. Gale and Kenneth W. Church. A Program for Aligning Sentences in Bilingual Corpora. In *Proc. of 29th Conf. of Assoc. for Comput. Linguistics*, ACL-1991, pp.177-184.
5. Hiroyuki Kaji, Yuuko Kida and Yasutusgu. Morimoto. Learning Translation Templates from Bilingual Text. In *Proceedings of the fifteenth International Conference on Computational Linguistics,* COLING-1992, Nantes, France, pp.672-678.
6. Martin Kay and Martin Roscheisen. Text-Translation Alignment. *Computational Linguistics.* Vol. 19, No. 1, pp.121-142, 1993.
7. Chew Lim Tan and Makoto Nagao. Automatic Alignment of Japanese-Chinese Bilingual Texts. *IEICE Trans. Information and System*. Vol, E78-D, No. 1, Jan., 1995.
8. Yuji Matsumoto, Hiroyuki Ishimoto, Takehito Utsuro: Sructural Matching of Parallel Texts. In *Proc. of 31st Conf. of the Association for Computational Linguistics*. ACL-1993 pp.23-30.
9. Chunyu Kit, Jonathan J. Webster, King Kui Sin, Haihua Pan and Heng Li. Clause alignment for Hong Kong legal text. *Intern. Journal of Corpus Linguistics* Vol. 9, No.1, 2004, pp.29-51.
10. I. Dan Melamed. Pattern recognition for mapping bitext correspondence. In *Parallel Text Processing*, J. Veronis (eds.). pp.25-47. 2000, Kluwer Academic Publishers.
11. Jörg Tiedemann. Recycling Translations - Extraction of Lexical Data from Parallel Corpora and their Application in Natural Language Processing, Doctoral Thesis, *Studia Linguistica Upsaliensia 1*, ISSN 1652-1366, ISBN 91-554-5815-7.
12. Ashish Venugopal , Stephan Vogel and Alex Waibel . Effective Phrase Translation Extraction from alignment model. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, ACL-2003, pp.319-326.
13. Jean Veronis. From the Rosetta stone to the information society—A survey of parallel text processing. In *Parallel Text Processing*. J. Veronis (ed.). 25-47. 2000, Kluwer Academic Publishers.

14. Wu Dekai. Aligning a Parallel English-Chinese Corpus Statistically with Lexical Criteria. In *Proc. of the 32st Meeting of Association for Comput. Linguistics*, ACL-1994. pp.80-87.
15. Wu Dekai. Stochastic inversion transduction grammars and bilingual parsing of parallel corpara. *Computational Linguistics*, Vol.23, No.3, pp.377-404.
16. Yasumichi Zaiman, Ryoko Yasukawa, Fuji Ren and Teruaki Aizawa. Text alignment using statistical technique and the language feature. *Technical Report of IEICE* TL2000-40, NLC2000-75(2001-03), pp.1-8.

Direct Combination of Spelling and Pronunciation Information for Robust Back-Transliteration

Slaven Bilac and Hozumi Tanaka

Tokyo Institute of Technology,
Ookayama 2-12-1, Meguro, 152-8552 Tokyo, Japan
{sbilac, tanaka}@cl.cs.titech.ac.jp

Abstract. Transliterating words and names from one language to another is a frequent and highly productive phenomenon. For example, English word *cache* is transliterated in Japanese as キャッシュ "kyasshu". Transliteration is information losing since important distinctions are not always preserved in the process. Hence, automatically converting transliterated words back into their original form is a real challenge. Nonetheless, due to its wide applicability in MT and CLIR, it is an interesting problem from a practical point of view.

In this paper, we demonstrate that back-transliteration accuracy can be improved by directly combining grapheme-based (i.e. spelling) and phoneme-based (i.e. pronunciation) information. Rather than producing back-transliterations based on grapheme and phoneme model independently and then interpolating the results, we propose a method of first combining the sets of allowed rewrites (i.e. edits) and then calculating the back-transliterations using the combined set. Evaluation on both Japanese and Chinese transliterations shows that direct combination increases robustness and positively affects back-transliteration accuracy.

1 Introduction

With the advent of technology and increased flow of goods and services, it has become quite common to integrate new words from one language to another. Whenever a word is adopted into a new language, pronunciation is adjusted to suit the phonetic inventory of the language. Furthermore, the orthographic form of the word is modified to allow representation in the target language script. For example, English word *cache* is transliterated in Japanese as キャッシュ "kyasshu".[1] In similar fashion, a proper noun Duncan is transliterated as 桔刃

[1] We use *italics* to transcribe the English words, while Japanese transliterations (e.g. キャッシュ) are given with romaji (i.e. roman alphabet) in "typewriter" font (e.g. "kyasshu"). The romanization used follows [1], thus closely reflecting English–like pronunciation with long vowels transcribed as "aa" rather than "ā".

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 413–424, 2005.

"deng4ken3" in Chinese.[2] This process of acquisition and assimilation of a new word into an existing writing system is referred to as transliteration [1].

Since integration of new words is a very productive process, it often happens that they are not recorded in machine or human dictionaries. Therefore, it is impossible to rely on dictionary lookup to find the transliteration pairs. Inability to find a target language equivalent represents a major problem in Machine Translation (MT) since it can cause translation failures. Furthermore, transliteration represents a serious problem in the area of Cross-Language Information Retrieval (CLIR) where new technical terms are frequently used in the queries and thus greatly affect performance [2, 3].

Back-transliteration is the transliteration back into the original language. It is generally more difficult than transliteration. Increase in difficulty results from the fact that various distinctions, present in the source language, are not always preserved when the word is transliterated into the target language. For example, Japanese has only five basic vowels and no /θ/ or /ð/[3] sounds. Non-existent sounds are replaced with their closest equivalents. Consequently, the following three English words: *bass*,*bath* and *bus* are transliterated as バス "basu".[4] A system trying to back-transliterate バス has therefore three valid choices which cannot be disambiguated in the absence of additional contextual information.

Transliterated words are normally written in katakana, one of three major character types making up the Japanese writing system. While other vocabulary (i.e. animal names or onomatopoeic expressions) can also be written in katakana, the fact that something is written in katakana is a good hint that it might be a transliterated foreign word or a name. Thus, unlike Arabic, Chinese or Korean, where a big part of the back-transliteration problem is identifying candidate transliterations [4, 5, 6], in Japanese back-transliteration can be directly applied to any katakana strings absent from the bilingual dictionary. In the evaluation on the Chinese data set, we avoid the problem of identifying the transliteration candidates since we train and evaluate on already extracted transliteration pairs. However, a real back-transliteration application would have to address this issue.

Previously, [7] proposed a hybrid back-transliteration method combining grapheme-based (i.e. spelling) and phoneme-based (i.e. pronunciation) information and demonstrated significant improvements over methods relying only on a single information source. In this paper, we show that the back-transliteration accuracy can be further improved by altering the manner in which grapheme-based and phoneme-based information is combined in the transliteration process and show that our method can easily be applied to Chinese (and, implicitly, other languages). Rather than producing back-transliterations based on grapheme and phoneme model independently and then interpolating the results, we propose a method of first combining the sets of allowed rewrites (i.e. edits) and then calcu-

[2] Chinese transliterations are given with the PinYin romanization with numerical tone codes.

[3] All phonemes given in // are written in IPA symbols.

[4] Here /θ/ is replaced with /s/, and /æ/ is replaced with /a/.

lating back-transliterations using the combined set. Evaluation on both Japanese and Chinese transliterations shows that the manner in which grapheme-based and phoneme-based information are combined can significantly affect the system performance.

The reminder of this paper is organized as follows: in Sect. 2 we review previous research. Sect. 3 outlines the proposed transliteration model and the system implementation. Finally, Sect. 4 gives an evaluation and a discussion of the results obtained, whereas Sect. 5 gives the conclusion.

2 Previous Research

Transliteration has received significant attention from researchers in recent years. Commonly, the source language considered is English and the target language is an Asian language void of an alphabetic writing system (e.g. Japanese, Korean, Chinese). Based on the underlying model, previous approaches to (back-) transliteration can be roughly divided into grapheme- and phoneme-based.

In the grapheme-based (or direct) modeling framework, the English string is not converted into a phonemic representation before its alignment with the transliterated string. Several different methods have been proposed within this framework: a decision tree based transliteration model [8], a maximum entropy based model [9], etc. Recently, [10] proposed a joint source-channel model that simultaneously models both source and channel context. In this model, after initial alignment, the preceding context (bigram, trigram) of both English and Chinese aligned units is considered simultaneously. Its main advantage is that it can be used for both transliteration and back-transliteration.

For back-transliteration of Japanese, [11] propose a noisy channel model allowing for non-atomics edits [12]. The input string is broken down into arbitrary substrings, each of which is output independently (and possibly incorrectly). The best back-transliteration is chosen using a modified edit distance algorithm [13, 14]. Since the best transliteration is determined through comparison with dictionary entries, this method does not handle phrases directly. This is a significant shortcoming since a large percent of transliterated strings are phrases.

In the phoneme-based (or pivot) modeling approach the pronunciation, rather than the spelling of the original string, is considered as a basis for transliteration. Among several approaches proposed are: an HMM based transliteration model [5], a rule based model [15] and a machine-learned phonetic similarity model [3].

For Japanese, [1] employ a compositional noisy-channel model combining romaji-to-phoneme, phoneme-to-English and English word probability models. The combined structure is treated as a graph, and the top ranking strings are found using the *k-best* path algorithm [16]. However, in this model there is no consideration of context information although context is crucial in determining the correct pronunciation of a given phoneme.

Recently, [7] proposed a back-transliteration model combining statistical string segmentation with a hybrid grapheme-phoneme transliteration model. The segmentation of the input string decreases the calculation complexity and allows

for correct back-transliteration even of strings containing abbreviations. Furthermore, by taking both the spelling and pronunciation of the original into account when modeling transliteration, the system is able to achieve higher accuracy. However, the manner of combination is suboptimal since back-transliterations are produced by each model independently and then the results are interpolated to obtain the final back-transliterations. In this paper we propose a different method of combining the grapheme- and phoneme-based models. The unit alignment sets (collections of allowed edits in the noisy channel model) are combined into one set and then, back-transliterations are produced based on this set. Since alignment sets obtained by each base model are different, their combination results in a more comprehensive alignment set that can successfully handle a larger number of transliterations.

3 Transliteration Model

As mentioned above, in Japanese transliterated words are normally written in katakana. However, we implement katakana to romaji conversion as a preprocessing module and view back-transliteration as a process starting with a romanized string.[5]

Given some string in romaji J_a, the goal is to find the English word (phrase) E_a that maximizes the probability $P(E_a|J_a)$. Applying the Bayes' rule and dropping the constant denominator we get $P(J_a|E_a) \times P(E_a)$, where $P(E_a)$ is the source model and $P(J_a|E_a)$ is the noisy channel. For the source model, we use a word-based model which can output words from a list of valid tokens E_a with a certain probability distribution $P(E_a)$. For the channel model we train several different models (Sec. 3.1), and then invert each to enable handling of the romaji input. The source model and each inverted channel model are then combined to obtain the back-transliterations.

3.1 Base Models

Grapheme-Based Model. In the grapheme-based model (GM) the English word is directly rewritten as a Japanese romaji string with the probability $P_g(J_a|E_a)$. Rewriting can be viewed as a sequential process where the first stage is the partitioning of the input string into sub-word units which are then rewritten according to some mapping rule (1). Rather than trying to estimate the probability of breaking up the string in a certain way, we allow all valid segmentations given the segments in our learned mapping set with equal probability. Thus, the resulting equation is simplified as (2). Under the assumption of segment independence, (2) can be further simplified so the resulting probability of outputting J_a can be rewritten as in (3).

[5] Katakana is a syllabary, and each character corresponds to one or more alphabet letters (e.g. ア "a", マ "ma" or シ "shi", etc.). By romanizing the input, we allow the model to capture the similarities on the letter level (e.g. that *m* in English often maps to "m" in Japanese) and reduce the data sparseness problem.

$$P_g(J_a|E_a) = P(E_{a_1}, E_{a_2} \dots E_{a_n}|E_a) \times P_g(J_{a_1}, J_{a_2} \dots J_{a_n}|E_{a_1}, E_{a_2} \dots E_{a_n}) \ . \quad (1)$$

$$P_g(J_a|E_a) \cong P_g(J_{a_1}, J_{a_2} \dots J_{a_n}|E_{a_1}, E_{a_2} \dots E_{a_n}) \ . \quad (2)$$

$$P_g(J_a|E_a) \cong \prod_{i=1}^{n} P_g(J_{a_i}|E_{a_i}) \ . \quad (3)$$

In Sect. 3.2 we describe how we get the set of allowed edits $(E_{a_i} \rightarrow J_{a_i})$ and how we go about assigning the probability to each $P_g(J_{a_i}|E_{a_i})$.

Phoneme-Based Model. In this model the channel is broken up into two stages: a) conversion of the English alphabet into English phonemes with some probability $P(E_p|E_a)$ and b) conversion of English phonemes into romaji with some probability $P(J_a|E_p)$. Hence, $P_p(J_a|E_a)$ can be rewritten as Eq. (4). Rather than manipulating these two distributions separately, we compute their composition to obtain a unique probability distribution $P_p(J_{a_i}|E_{a_i})$. The composition produces a set of edit pairs $(E_{a_i} \rightarrow J_{a_i})$ such that $(E_{a_i} \rightarrow E_{p_i})$ is a member of the first set and $(E_{p_i} \rightarrow J_{a_i})$ is a member of the second set for some E_{p_i} [19].[6]

$$P_p(J_a|E_a) \cong \prod_{i=1}^{n} P(J_{a_i}|E_{p_i}) \times \prod_{i=1}^{n} P(E_{p_i}|E_{a_i}) \ . \quad (4)$$

Since obtained mapping set does not contain any English phoneme units, all English alphabet strings can be rewritten directly into romaji without requiring their conversion into intermediate phoneme representation. This removes the requirement of having a pronunciation dictionary for the back-transliteration.[7] Note also that both GM and PM are dealing with the same types of edits (i.e. English alphabet to romaji $(E_{a_i} \rightarrow J_{a_i})$) and can be directly combined.

3.2 Training

The training consists of learning possible edits and their probabilities. We train the GM and PM sets independently and then combine them.

For the grapheme-based model, romanized Japanese strings are aligned with English strings directly using the non-weighted Levenshtein distance [13, 14]. After initial alignment, each atomic edit (i.e. one aligning unit pair) is expanded through combination with adjacent edits. By doing so, we add contextual information to each edit, which helps reduce the ambiguity associated with atomic edits [17]. For example, for the pair (*vinyl*, `biniru`) we get the following alignment:

$$v \rightarrow \texttt{b} \quad i \rightarrow \texttt{i} \quad n \rightarrow \texttt{n} \quad y \rightarrow \texttt{i} \quad l \rightarrow \texttt{r} \quad \_ \rightarrow \texttt{u}$$

[6] This is a slight simplification since E_{p_i} can actually be a concatenation of several $E_{p_{n..m}}$ with respective mappings $E_{p_{n..m}} \rightarrow J_{a_{m..n}}$.

[7] However, the pronunciation dictionary is still necessary for the training since the mapping set is obtained via intermediate use of English phoneme representations.

For $N = 1$, the following edits are then also added to the set:

$$vi \rightarrow \texttt{bi}, in \rightarrow \texttt{in}, vin \rightarrow \texttt{bin}, ny \rightarrow \texttt{ni}, iny \rightarrow \texttt{ini},$$
$$yl \rightarrow \texttt{ir}, nyl \rightarrow \texttt{nir}, yl \rightarrow \texttt{iru}, l \rightarrow \texttt{ru}$$

We collect a complete set of edits $\alpha_g \rightarrow \beta_g$ in the training set and assign the probability to each according to (5). Throughout, we distinguish edits that appear at the beginning or the end of the word or neither.

$$P(\alpha \rightarrow \beta) = \frac{count(\alpha \rightarrow \beta)}{count(\alpha)} \quad . \tag{5}$$

For the PM, we obtain the optimal romaji to English phoneme alignment using the Estimation Maximization (EM) algorithm [18]. After EM selects the optimal alignment, we proceed to expand the set of individual alignments with N adjacent units as above to obtain a set of possible rewrites $\alpha_{e_p} \rightarrow \beta_{j_a}$. This process is repeated to obtain the set of all possible mappings of English alphabet strings into phoneme strings $\alpha_{e_a} \rightarrow \beta_{e_p}$. Each input α_{e_a} with all its mappings β_{e_p} and corresponding probabilities $P(\beta_{e_p}|\alpha_{e_a})$ is converted into a Weighted Finite State Transducer (WFST) T_p with α_{e_a} as inputs, β_{e_p} as outputs [19, 1] and transition costs as negative logs of probabilities. WFST T_p is then composed with a WFST T_c encoding the complete set of mappings $\alpha_{e_p} \rightarrow \beta_{j_a}$ to obtain the set of all possible rewrites of English alphabet strings α_p into romaji strings β_p based on the PM.

Directly Combined Grapheme-Phoneme Model. By following the above procedure, we can extract two sets of English alphabet to romaji edits and their corresponding probabilities. One set is based on direct alignment of Japanese string with English spelling (6) and the other one on the indirect alignment via the English pronunciation as a pivot (7). Although the edits in each set are trained from the same data, the sets are different in the sense that the first set better reflects influence of English spelling on transliteration and the second set better reflects the influence of English pronunciation. Since we would like to obtain a set of edits which reflects the influence both of spelling and pronunciation we combine the two as given in (8). We designate this set as S_{gp} and the corresponding model as GPM.

$$S_g = (\alpha_1, \beta_{11}, P_g(\beta_{11}|\alpha_1)), (\alpha_1, \beta_{12}, P_g(\beta_{12}|\alpha_1)) \ldots (\alpha_n, \beta_{nm}, P_g(\beta_{nm}|\alpha_n)) \tag{6}$$
$$S_p = (\alpha_1, \beta_{11}, P_p(\beta_{11}|\alpha_1)), (\alpha_1, \beta_{12}, P_p(\beta_{12}|\alpha_1)) \ldots (\alpha_p, \beta_{pq}, P_p(\beta_{pq}|\alpha_p)) \tag{7}$$
$$S_{gp} = S_g \cup S_p \qquad \text{s.t. } P_{gp}(\beta_{ij}|\alpha_i) = \gamma P_g(\beta_{ij}|\alpha_i) + \delta P_p(\beta_{ij}|\alpha_i)$$
$$\text{and} \quad \gamma + \delta = 1 \tag{8}$$

Some statistics of learned mapping sets for $N = 2$ are given in Table 1. We can see that the GM and PM sets differ significantly. Hence, their combination (GPM) results in a more comprehensive set of edits which can handle a wider variety of transliterations.

Table 1. Statistics of learned mapping sets. Statistics given are for romanized Japanese strings

	GM	PM	GPM
Unique English strings	15 120	8 662	18139
Avg. length of English strings (char.)	4.18	4.23	4.34
Max. English string length (char.)	7	11	11
Unique Japanese strings	13 933	7 392	16665
Avg. length of Japanese strings (char.)	4.63	4.66	4.80
Max. Japanese string length (char.)	9	10	10
Japanese strings per English string (avg.)	1.95	4.47	3.35

3.3 Model Implementation

Here, we describe how we calculate back-transliterations for a given input string in romaji. First, the string is segmented as described in [7]. Next, the resulting string is encoded as a Finite State Acceptor (FSA) I in which any path from the start to accept state represents a possible breakup of the string into substring units based on the extracted mapping set S_σ.[8] We add a transition for each substring α_i that appears in the learned mapping set together with special beginning-of-string and end-of-string marks.

Next, we rewrite S_σ as a WFST T with α_i as inputs, β_{ij} as outputs and transition costs as negative logs of probabilities. This WFST T is then inverted and composed with the FSA I to obtain a WFST $I \circ T_I$ which represents all the possible ways to rewrite the input string into English given the mappings learned in the training process.

We also compile the source model into a WFST O so that all word tokens are added as valid outputs. A null input, word delimiter output transition is also added, allowing for multiple words to be output for a single string input. Hence, phrases can also be handled. Note that the valid word tokens need not come from any bilingual/English dictionary but can be any list of words/phrases we would like to make the target of back-transliteration. Probabilities assigned to each token can be either reflecting corpus trained frequencies or uniformly distributed. Thus, the source model can easily be adjusted to the given domain or application.

Finally, WFST $I \circ T_I$ is composed with the WFST O and the resulting WFST $I \circ T_I \circ O$ is searched for k-best transliterations using the k-best path algorithm. A probability $P(E_a|J_a)$ is associated with each path obtained. In cases several paths correspond to the same word (phrase), their probabilities are summed up. Thanks to the cascading WFST composition we are able to search all possible back-transliterations based on the available mappings and select the optimal solution.

[8] Here, $\sigma \in \{g, p, gp\}$.

Table 2. Examples of the NTCIR-2 data

Katakana	Romanized	Segmented	English
ウィンドウサイズ	`uindousaizu`	`uindou#saizu`	*window size*
エキシマレーザ	`ekishimareeza`	`ekishima#reeza`	*excimer laser*
ガスレーザ	`gasureeza`	`gasu#reeza`	*gas laser*
グラビトン	`gurabiton`	`gurabiton`	*graviton*
グリコカリックス	`gurikokarikkusu`	`gurikokarikkusu`	*glycocalyx*

4 Evaluation

We evaluate various aspects of the proposed method on sets of novel katakana strings not in the EDICT dictionary [20]. The first set consists of 150 katakana words extracted from the EDR Japanese corpus [21]. The second test comes from the NTCIR-2 test collection [22]. All 78 out-of-vocabulary katakana words from the topic section (49 short documents) were used. Several examples from this test set are shown in Table 2.

A collection of about 6,000 words in katakana together with the corresponding English translation extracted from the EDICT dictionary was used as training data. This set was expanded, so that for each katakana word containing a long vowel or a geminate consonant, we add one with these removed. The pronunciations for training the PM were obtained from the CMU pronouncing dictionary [23]. When no pronunciations were available the words were excluded from the training. The AT&T FSM library [24] was used for WFST manipulation.

For the EDR test set we used the complete CMU dictionary word set (around 120,000 words) compiled into a language model with word probabilities reflecting the corpus frequencies from the EDR English corpus [21]. For the NTCIR-2 test set we created a language model from about 110,000 words and their frequencies as counted in the English part of the NTCIR-2 collection. The transliterations were considered correct, if they matched the English translation, letter-for-letter, in a non-case-sensitive manner. Table 3 gives results for grapheme-model (GM), phoneme-model (PM), interpolated combination model (COMB) [7] and proposed method of direct combination (GPM).[9] Each model was evaluated with (+SEG) and without segmentation preprocessing module.

We can see that combination generally helps and that in all top-1 and most top-10 cases (regardless of the segmentation), GPM fares better than COMB method. Thus, direct combination seems to be an effective way of combining grapheme- and phoneme-based models. Only for the NTCIR-2 test set, the highest top-1 accuracy is still achieved by GM+SEG model. This can be attributed to a high number of scientific terms whose transliteration better reflects original

[9] For these experiments the combination weights are arbitrarily set at $\delta = \gamma = 0.5$ for the GPM whereas for the COMB model we use the trained values of δ and γ [7]. GM and PM models were trained with context $N = 2$.

Table 3. Transliteration results for the EDR test set (150 inputs) and for the NTCIR-2 test set (78 inputs)

	EDR test set		NTCIR-2 test set	
	Top-1 (%)	Top-10 (%)	Top-1 (%)	Top-10 (%)
SEG	33.33	34.00	25.64	25.64
GM	48.00	66.00	48.72	62.82
GM+SEG	58.00	72.00	**66.67**	78.21
PM	46.00	61.33	35.90	52.56
PM+SEG	57.33	68.00	57.69	74.36
COMB	49.33	72.00	44.87	67.95
COMB+SEG	59.33	75.33	62.82	**83.33**
GPM	54.00	70.00	48.72	70.51
GPM+SEG	**60.67**	**79.33**	62.82	78.21

spelling than pronunciation (e.g. グリコカリックス "`gurikokarikkusu`" *glycocalyx*) that are pushed lower in the combined result sets.

4.1 Evaluation on Chinese Test Data

In order to test whether our proposed method is also effective for other languages, we conducted an additional evaluation of back-transliteration on Chinese transliterations of foreign names. The data set used consists of 11,584 transliteration pairs listed by Xinhua News Agency [25]. We use 8,688 pairs to train the system and 2,896 pairs for the evaluation. The training procedure is identical to the one used for Japanese data with the exception of the romanization module which was modified to work with Chinese input and output pinyin transcriptions. We wanted to see whether tone marks and Chinese character (hanzi) segmentation affect the back-transliteration so we compared three different schemes: 1) **Toneless** where pinyin tone marks are removed from the training data, 2) **Tone** where pinyin tone marks are left as-is and **Hanzi** where unit-segmentation is enforced so that all alphabet characters corresponding to one Chinese character are in one unit. For example, for the transliteration 套戦 "`pei4li3`" of *Perry*, we would get the training pairs: (*perry*, `peili`), (*perry*, `pei4li3`) and (*perry*, `pei li`), respectively. Finally, a language model was constructed from all 11,584 words assigned equal weights.

The results of the experiment are given in Table 4. Besides the proposed method, we give the accuracy figures for 3-gram TM, a method proposed by [10] for the same test set (taken from their paper). We can see that for this data set the GM model generally performs better than PM,[10] but that combined models (COMB and GPM) achieve better accuracy than either of the individual models for all three experiment settings. The best performance is

[10] Somewhat worse performance of the PM model can be attributed to a smaller training set due to a large number of pronunciations missing from the CMU dictionary.

Table 4. Transliteration results for the Chinese test set (2,896 inputs)

	Toneless		Tone		Hanzi	
	Top-1 (%)	Top-10 (%)	Top-1 (%)	Top-10 (%)	Top-1 (%)	Top-10 (%)
3-gramTM	N/A	N/A	N/A	N/A	37.90	75.40
GM	68.99	95.99	70.79	94.30	52.14	67.09
GM+SEG	67.68	95.99	70.20	94.30	53.66	70.79
PM	62.81	91.71	65.68	91.69	41.95	62.40
PM+SEG	58.18	91.64	61.11	91.64	40.74	68.78
COMB	69.33	96.82	70.96	96.44	58.49	81.32
COMB+SEG	67.85	96.82	69.99	96.44	57.91	81.32
GPM	69.33	96.75	**72.76**	**96.96**	57.32	80.35
GPM+SEG	68.09	96.75	71.51	**96.96**	58.77	85.08

achieved by the GPM model trained on pinyin with tone marks, showing that not only is the modeling of pronunciation and spelling simultaneously beneficial but that the direct combination yields better results than interpolation. Finally, we can see that the segmentation negatively affects the top-1 performance in most test cases. This can be attributed to the fact that all the inputs in this data set are single words, thus any additional segmentation taxes performance.

4.2 Discussion

For Japanese evaluation sets, we observe similar trends as [7]. The performance of singleton models (GM,PM) can be significantly improved through combination and addition of segmentation module. However, the proposed combination model (GPM) performs better than the interpolated combination (COMB) for most of the test cases/settings. This leads us to believe that a back-transliteration system using direct combination is more robust than systems based on the singleton models or interpolated combination. Although we do not provide evaluation results, we have noticed that increase in the number of tokens in the source model negatively affects both the transliteration speed and accuracy. Thus, it would be beneficial to explore methods for reducing the size of the source model depending on the input and/or domain.

For Chinese evaluation set, we compare our back-transliteration model with 3-gram TM which outperformed other Chinese back-transliteration systems in evaluation given in [10]. 3-gram TM is a character-based model possibly outputting non-valid English strings. Thus, its accuracy could be increased by filtering the outputs against a list of valid tokens. However, such filtering is not an integral part of 3-gram TM. Furthermore, 3-gram TM forces Chinese string segmentation on Chinese character boundaries (akin to the **Hanzi** scheme above) and our experiments show that better results can be achieved by allowing finer

segmentation.[11] Although both models consider source and target string context simultaneously, our model considers both the preceding and following context while 3-gram TM model only considers preceding context. Furthermore, our model considers both pronunciation and spelling of the original string but 3-gram TM model only considers the spelling of the original. Given all this, it is not surprising that our model achieves significantly higher accuracy in the evaluation.

5 Conclusion

In this paper we propose a method for improving the back-transliteration accuracy by directly combining grapheme-based and phoneme-based information. Rather than producing back-transliterations based on grapheme and phoneme model independently and then interpolating the results as was previously proposed, we first combine the sets of allowed rewrites (i.e. edits) based on the two models and then calculate the back-transliterations using the combined set. We evaluate the proposed combination method on Japanese transliterations and show that the manner in which grapheme-based and phoneme-based information are combined can significantly affect the system performance.

Furthermore, we show the proposed method can easily be applied to back-transliteration of Chinese and that significant improvements can also be achieved by combining the grapheme and phoneme models.

Acknowledgments. We would like to thank Zhang Min for help with Chinese evaluation data and an anonymous reviewer for valuable comments.

References

1. Knight, K., Graehl, J.: Machine transliteration. Computational Linguistics **24** (1998) 599–612
2. Fujii, A., Ishikawa, T.: Japanese/English cross-language information retrieval: Exploration of query translation and transliteration. Computers and Humanities **35** (2001) 389–420
3. Lin, W.H., Chen, H.H.: Backward machine transliteration by learning phonetic similarity. In: Proc. of the Sixth Conference on Natural Language Learning. (2002) 139–145
4. Stalls, B.G., Knight, K.: Translating names and technical terms in Arabic text. In: Proc. of the COLING/ACL Workshop on Computational Approaches to Semitic Languages. (1998)
5. Jeong, K.S., Myaeng, S.H., Lee, J.S., Choi, K.S.: Automatic identification and back-transliteration of foreign words for information retrieval. Information Processing and Management **35** (1999) 523–540

[11] Admittedly, our model does not incorporate smoothing and is more susceptible to the data sparseness problem that arises when using this alignment scheme. Also [10] use the full set of 34,777 transliteration pairs for training as opposed to 8,688 pair subset we used.

6. Kang, B.J., Choi, K.S.: Effective foreign word extraction for Korean information retrieval. Information Processing and Management **38** (2002) 91–109
7. Bilac, S., Tanaka, H.: A hybrid back-transliteration system for Japanese. In: Proc. of the 20th International Conference on Computational Linguistics (COLING 2004). (2004) 597–603
8. Kang, B.J., Choi, K.S.: Automatic transliteration and back-transliteration by decision tree learning. In: Proc. of the Second International Conference on Language Resources and Evaluation. (2000)
9. Goto, I., Kato, N., Uratani, N., Ehara, T.: Transliteration considering context information based on the maximum entropy method. In: Proc. of the IXth MT Summit. (2003)
10. Li, H., Zhang, M., Su, J.: A joint source-channel model for machine transliteration. In: Proc. of the 42th Annual Meeting of the Association for Computational Linguistics. (2004) 159–166
11. Brill, E., Kacmarcik, G., Brockett, C.: Automatically harvesting katakana-English term pairs from search engine query logs. In: Proc. of the Sixth Natural Language Processing Pacific Rim Symposium. (2001) 393–399
12. Brill, E., Moore, R.C.: An improved error model for noisy channel spelling correction. In: Proc. of the 38th Annual Meeting of the Association for Computational Linguistics. (2000) 286–293
13. Damerau, F.: A technique for computer detection and correction of spelling errors. Communications of the ACM **7** (1964) 659–664
14. Levenshtein, V.: Binary codes capable of correcting deletions, insertions, and reversals. Soviet Physics–Doklady **10** (1966) 707–710
15. Oh, J.H., Choi, K.S.: An English-Korean transliteration model using pronunciation and contextual rules. In: Proc. of the 19th International Conference on Computational Linguistics. (2002) 758–764
16. Eppstein, D.: Finding the k shortest paths. In: Proc. of the 35th Symposium on the Foundations of Computer Science. (1994) 154–165
17. Bilac, S., Tanaka, H.: Improving back-transliteration by combining information sources. In: Proc. of the First International Joint Conference on Natural Language Processing (IJCNLP-04). (2004) 542–547
18. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete via the EM algorithm. Journal of the Royal Statistical Society **39** (1977) 1–38
19. Pereira, F.C.N., Riley, M.: Speech recognition by composition of weighted finite automata. In Roche, E., Shabes, Y., eds.: Finite-State Language Processing. MIT Press (1997) 431–453
20. Breen, J.: EDICT Japanese/English dictionary file (2003) Available: ftp://ftp.cc.monash.edu.au/pub/nihongo.
21. EDR: EDR Electronic Dictionary Technical Guide. Japan Electronic Dictionary Research Institute, Ltd. (1995) (In Japanese).
22. Kando, N., Kuriyama, K., Yoshioka, M.: Overview of Japanese and English Information Retrieval Tasks (JEIR) at the Second NTCIR Wordshop. In: Proc. of NTCIR Workshop 2. (2001)
23. Carnegie Mellon University: The CMU pronouncing dictionary (1998) Available: http://www.speech.cs.cmu.edu/cgi-bin/cmudict.
24. Mohri, M., Pereira, F.C.N., Riley, M.: AT&T FSM library (2003) Available: http://www.research.att.com/ mohri/fsm.
25. Xinhua News Agency: Chinese transliteration of foreign personal names. The Commercial Press (1992)

A Prosodic Diphone Database for Korean Text-to-Speech Synthesis System

Kyuchul Yoon

The Ohio State University, Columbus OH 43220, USA
kyoon@ling.osu.edu,
http://ling.osu.edu/∼kyoon

Abstract. This paper presents a prosodically conditioned diphone database to be used in a Korean text-to-speech (TTS) synthesis system. The diphones are prosodically conditioned in the sense that a single conventional diphone is stored as different versions taken directly from the different prosodic domains of the prosodically labeled, read sentences (following the K-ToBI prosodic labeling conventions [3]). Four levels of the Korean prosodic domains were observed in the diphone selection process, thereby selecting four different versions of each diphone. A 400-sentence subset of the Korean Newswire Text Corpora [5] were converted to its pronounced form as described in [8] and its read version was prosodically labeled. The greedy algorithm [7] identified 223 sentences containing 1,853 prosodic diphones (out of the 3,977 possible prosodic diphones) that can synthesize all four hundred utterances. Although our system cannot synthesize an unlimited number of sentences at this stage, the quality of the synthesized sentences strongly suggests that it is a viable option to use prosodically conditioned diphones in a text-to-speech synthesis system.

1 Introduction

Work on Korean shows that segmental properties are affected by the prosody of an utterance. In a study on the effect of prosodic domains on segmental properties of three Korean coronal stops /t, t^h, t*/, Cho and Keating [1] showed that initial consonants in higher prosodic domains are articulatorily stronger and longer than those in lower domains: the former has more linguopalatal contact and is longer in duration than the latter. Acoustic properties such as VOT, total voiceless interval, percent voicing during closure, and nasal energy minimum were also found to vary with prosodic position. Prosodic effects on segments have also been found in Korean fricatives. In her study on Korean coronal fricatives, Kim [4] observed prosodic effects on segmental properties. Linguopalatal contact was greater, acoustic duration was longer, centroid frequency was higher, and H1-H2 value for /s*/ was lower in higher domains than in lower domains. Yoon [9] looked at two Korean voiceless coronal fricatives and found that each fricative in different prosodic positions displayed characteristics that appear to signal its prosodic location by means of durational differences. Motivated by these findings,

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 425–428, 2005.

we have started working on a diphone-based concatenative TTS system within the Festival TTS framework [6]. Our aim was to build a prosodically conditioned diphone database. We believe that prosodic effects on segments can be properly modeled by recording different diphones in different prosodic contexts.

In this paper, we present our prosodically conditioned diphone database built from the read version of a 400-sentence subset of the Korean Newswire Text Corpora [5]. Although we have chosen to use a subset of the corpora, potentially making our system incapable of synthesizing an unlimited number of sentences, the quality of the synthesized utterances suggests that it is a viable option to use 'real' natural utterances in a diphone-based concatenative TTS system.

2 Methods

2.1 Prosodic Diphones

Our working hypothesis was that the effects of prosodic position are the primary type of allophonic variation that we need to model, and we can encode that variation more naturally by recording different diphones for each different relevant prosodic position. We will call these position-specific diphones "prosodic diphones" as opposed to conventional diphones that are not sensitive to their positions in an utterance. We also assumed that the prosodic positional effects extend half-way to the vowel of the domain initial or final syllables [9].

There are two tonally defined prosodic phrases above the level of the prosodic word (PW) in Korean, i.e. the intonational phrase (IP) and the accentual phrase (AP) [2]. An IP is marked by a boundary tone and phrase-final lengthening with an optional sense of pause. An AP, which is smaller than an IP but larger than a PW, is marked by a phrasal tone. An IP can have one or more APs and the final tone of the last AP within an IP is replaced with the boundary tone of the IP.

A conventional syllable-initial diphone such as *p-a* in Korean can occur in four different prosodic domains; initial to an IP, an AP, a PW and medial to a PW. In order to differentiate these prosodically different versions of the same conventional diphone, we employed three sets of boundary symbols, < and >, [and] and { and }, to represent the beginning and end of the three prosodic domains, an IP, an AP and a PW respectively. The diphones medial to a PW were not specified with any symbols.

A diphone with any one of the above boundary symbols will be called an "edge diphone" as opposed to a non-edge diphone medial to a PW. The four different prosodic versions of our example diphone thus will be *<p-a*, *[p-a*, *{p-a* and *p-a*. This notation represents our hypothesis that a syllable-initial phone *p* in the form of the diphone *p-a* will be realized as four different allophones in the prosodic hierarchy of Korean.

We used the phone set described in our earlier study [8]. When all possible combinations of prosodic diphones and phonotactic constrains of Korean are considered, 3,997 theoretically possible prosodic diphones were obtained; 423 IP edge diphones, 1,228 AP edge diphones, 1,228 PW edge diphones and 1,118 PW internal diphones.

2.2 Corpora

The text corpora that we used consisted of 400 sentences from dozens of newspaper articles provided by the Korean Newswire. The 400-sentence corpora contained 28,666 syllables or 9,246 space-delimited words, with an average of 72 syllables or 23 words per sentence. We followed Yoon [8] and obtained pronounced phonemic forms of the text corpus. In the next step, the text corpus was read by a native speaker of Korean because we intended to build a TTS system for the reading style of the speaker. The read speech corpus as well as the text corpus was then prosodically labeled by a trained labeler following the Korean ToBI (tones and break indices) labeling conventions [3].

The component prosodic diphones for each sentence were extracted based on the segmental compositions and the kind of prosodic domains present in each utterance. Given the following example sentence consisting of one IP and two APs, 21 prosodic diphones are extracted.

‡ <b o g o s eo n eu n] [s eo l my eo ng h e d dd a> ‡ *The report said*

‡-<b, <b-o, o-g, g-o, o-s, s-eo, eo-n, n-eu, eu-n], n]-[s, [s-eo, eo-l, l-my, my-eo, eo-ng, ng-h, h-e, e-d, d-dd, dd-a>, a>-‡

2.3 Diphone Selection

The greedy algorithm [7] implemented in a Praat script was applied to the component prosodic diphone sequences of each of the 400 sentences to search for the minimal number of sentences that will cover all the prosodic diphones present in the corpus. The search ended selecting 223 sentences. The number of prosodic diphones contained in the 223 sentences was 1,853 in total (47% of the 3,977 possible diphones), meaning that these diphones can synthesize all 400 sentences. The selected prosodic diphones from the greedy search were then segmented and labeled.

2.4 Diphone Database

Pitchmarks and LPC coefficients were extracted from the waveforms of the 223 sentences as explained in the Festival manual (version 2.0). A voice was created in Festival and sample sentences were synthesized to test the diphone database. These sample sentences were the remaining 177 sentences that were not selected in the greedy search. No fundamental frequency contour was given at this stage.

The sentence that had the most prosodic diphones as its components in the greedy selection process was also resynthesized with the labeled diphones. This sentence were composed of 268 prosodic diphones and 247 diphones were selected in the greedy search. In other words, the resynthesized version of this sentence had 92% of the component diphones from itself. This sentence was resynthesized to serve as the 'baseline' quality of the other synthesized sample sentences. An informal listening test with three native speakers of Korean showed encouraging results. A rigorous listening test is our next goal.

3 Discussion

This paper presented a prosodically conditioned diphone database created from a read speech corpus. Our hypothesis was that the effects of prosodic position on sound segments are the primary type of allophonic variation that we need to model. We encoded that variation by recording different diphones for a particular diphone from different relevant prosodic positions. We also hypothesized that the prosodic positional effects extend half-way to the vowel of the initial or final syllables of a particular prosodic domain. Prosodic diphones were extracted from a read version of a text corpus to reflect these hypotheses.

Although our work started as a limited domain synthesis in that our diphone database only covers a 47% of all possible prosodic diphones of Korean and that it can only synthesize the 400 hundred sentence corpus that we used, its outcome strongly suggests the plausibility of using K-ToBI labeled speech corpora in diphone database creation. With increasing availability of various genres of K-ToBI corpora, this is a promising start.

References

1. Taehong Cho and Patricia A. Keating: Articulatory and acoustic studies of domain-initial strengthening in Korean. Journal of Phonetics **29** (2001)
2. Sun-Ah Jun: The phonetics and phonology of Korean prosody: intonational pho nology and prosodic structure. The Ohio State University (1993)
3. Sun-Ah Jun: K-ToBI (Korean ToBI) labeling conventions. Version 3.1 http://www.linguistics.ucla.edu/people/jun/ktobi/K-tobi.html (2000)
4. Sahyang Kim: The interaction between prosodic domain and segmental properties: domain-initial strengthening of fricatives and post obstruent tensing rule in Korean. UCLA (2001)
5. Linguistic Data Consortium (LDC): Korean Newswire Text Corpus catalog number LDC2000T45, ISBN 1-58563-168-X (2000)
6. Paul Taylor and Alan W. Black and Richard Caley: The architecture of the festival speech synthesis system. Proceedings of the 3rd ESCA Workshop on Speech Synthesis (1998) 147–151
7. J.P.H. van Santen: Diagnostic perceptual experiments for text-to-speech system evaluation. Proceedings of the ICSLP (1992) 555–558
8. Kyuchul Yoon: Letter-to-sound rules for Korean. IEEE-SP2002 Workshop on Speech Synthesis (TTS 2002)
9. Kyuchul Yoon: The effects of prosody on segmental variation. Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2003)

On a Pitch Detection Method Using Noise Reduction

Jongkuk Kim[1], Ki Young Lee[2], and Myung Jin Bae [1]

[1] Department of Information & Telecommunication Engineering,
Soongsil University,
Sangdo 5-dong, Dongjak-gu, Seoul, 156-743, Korea
{Kokjk91, mjbae}@ssu.ac.kr
[2] Department of Information Communication Engineering,
Kwandong University,
7 San Imcheon-ri,Yangyang-eup, Yangyang-gun, Gangwon-do, Korea
kylee@mail.kwandong.ac.kr

Abstract. It can be used to easily change or to maintain the naturalness and intelligibility of quality in speech synthesis and to eliminate the personality for speaker-independence in speech recognition. In this paper, we proposed a new pitch detection algorithm. And Kalman filters are implemented for filtering speech contaminated by additive white noise or colored noise and an iterative signal and parameter estimator which can be used for both noise type is presented. The performance was compared with LPC and Cepstrum, ACF. we have obtained the pitch information improved the accuracy of pitch detection and gross error rate is reduced in voice speech region and in transition region of changing the phoneme. Also the results indicate that the additive white noise Kalman filters provide an audible improvement in output speech quality, and an improved pitch detection. This paper clearly shows the feasibility of using the Kalman filter for noise reduction.

1 Introduction

In speech signal processing, it is very important to detect the pitch exactly in speech recognition, synthesis, analysis. If we exactly pitch detect in speech signal, in the analysis, we can use the pitch to obtain properly the vocal tract parameter. It can be used to easily change or to maintain the naturalness and intelligibility of quality in speech synthesis and to eliminate the personality for speaker-independence in speech recognition. And a lot of methods for the pitch detection have been proposed until now.

However, these methods may be brought about the errors, when there are some phonemic transitions within the analysis frame and the speech signals are corrupted by background noises. And it is often necessary to perform speech enhancement through noise removal in speech processing systems operating in noisy environments. As the presence of noise degrades the performance of speech coders and voice recognition systems, it is therefore common to incorporate speech enhancement as a preprocessing step in these systems. In this paper, we proposed new pitch detection algorithm, and for the removal of additive white noise, we employed Kalman filtering.

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 429 – 432, 2005.

2 Pitch Detection Algorithm

Noise Reduction. The general method employed in this paper for coding noisy speech utilizes the cascade estimator and coder structure. The determination of the Kalman filter parameters and the actual filtering of the speech are performed iteratively [5]. The overall block diagram of the filter is shown in Figure 1.

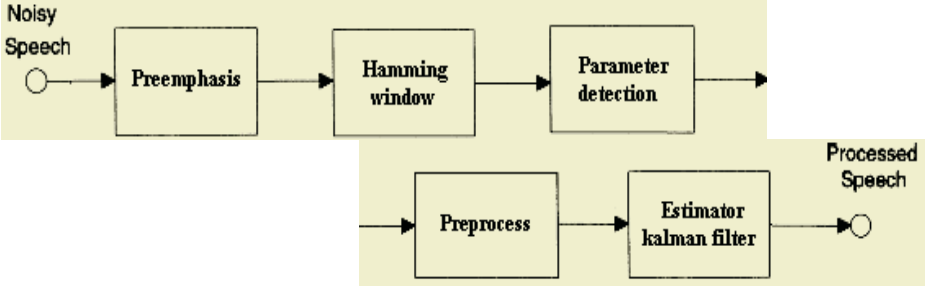


Fig. 1. Overall block diagram of noise reduction

Pitch Detection. Figure 2 shows a block diagram of the proposed method.

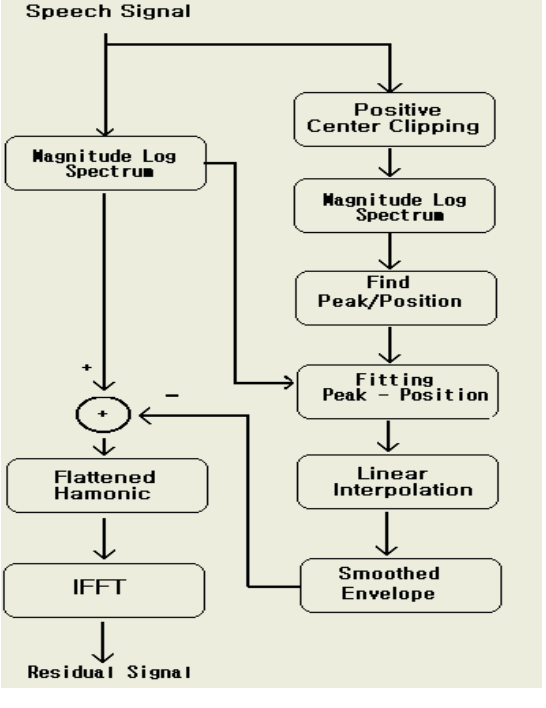


Fig. 2. Proposed Pitch Detection

3 Experimental Results

Computer simulation was performed to evaluate the proposed algorithm using an IBM Pentium(586MHz) interfaced with the 16-bit AD/DA converter. A frame size is 360 samples and subframe size is 120 samples. To measure the performance of the proposed algorithm, we used the following speech data. Speech data was sampled at 8kHz and was quantized with 16bits. And white Gaussian noise was added to each sentence with an average signal to noise ratio. A noise generator was used for each of the speech files. Consequently, a different white Gaussian noise was added. We now evaluate the performance of the proposed Kalman filtering algorithms for speech enhancement along and for coding of noisy speech when the additive noise is white.

Figure 3 shows the same speech with white noise added. Fig. 4 shows the results of enhancement using Kalman filters. As shown in table 1, it is gross error rate comparison in each method.

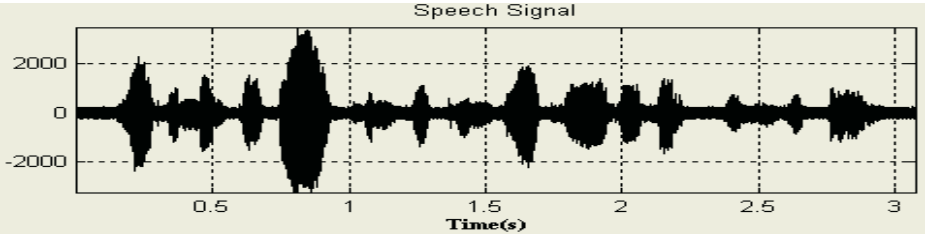


Fig. 3. White noise corrupted speech

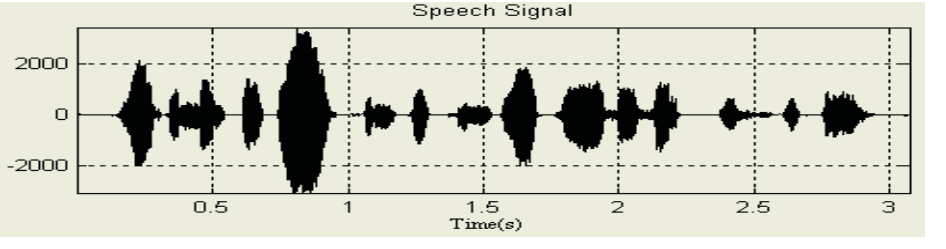


Fig. 4. Enhanced speech

Table 1. Comparison of gross error rate

Speech	gross error rate(%)			
	ACF	LPC	Cepstrum	Proposed
Utterance1	2.91	2.35	2.65	0.58
Utterance 2	0.80	1.85	2.34	0.55
Utterance 3	2.90	2.03	2.30	0.57
Utterance 4	2.39	2.21	2.58	0.66
Average	2.25	2.11	2.47	0.59

4 Conclusions

In speech signal processing, it is very important to detect the pitch exactly in speech. If we exactly pitch detect in speech signal, in the analysis, we can use the pitch to obtain properly the vocal tract parameter without the influences of vocal cord. It can be used to easily change or to maintain the naturalness and intelligibility of quality in speech synthesis and to eliminate the personality for speaker-independence in speech recognition. In proposed new method first, positive center clipping is process by using the incline of speech signals in order to emphasize pitch period with component of removed vocal tract characteristic in time domain as well we can detect more exactly pitch period through harmonics peak-fitting in frequency domain.

Owing to this algorithm, we obtained the pitch, improved the accuracy of pitch detection and extracted it in voice speech and transition region of changing the phoneme. Kalman filter algorithms have been developed for speech enhancement. This paper clearly shows the feasibility of using the Kalman filter for noise reduction. The drawback, however, was that the optimization of the parameters was a very difficult and tedious task when altering the noise and speech condition. There certainly remains considerable further work to be done towards a more significant improvement in mobile communication which remains a complex environment, mainly in non-stationary conditions.

Acknowledgement. This work was supported by the Korean Science and Engineering Foundation, grant no. R01-2002-000-00278-0.

References

1. Ing Yann Soon, Soo Ngee Koh, Chai Kiat Yeo, Noisy speech enhancement using discrete cosine transform, Speech communication 24, pp. 249–257, 1998.
2. A.M. Kondoz, Digital Speech, John Wiley & Sons, 1994.
3. W. B. Klejin *et al*., Speech Coding and Synthesis, Elsevier Science B.V., 1995.
4. Jerry D. Gibson, Speech coding in mobile radio communication, Processing of the IEEE, V. 86. N 7. July 1998.
5. M. Bae, I. Chung, and S. Ann, The Extraction of Nasal Sound Using G-peak in Continued Speech, KIEE, Korea, Vol. 24, No. 2 pp. 274–279, March 1987.
6. Y. Ephraim, D. Malah, A signal subspace approach for speech enhancement, IEEE Trans. Speech and Audio Process. 3, 1995 pp. 251–266.
7. Jean Rouat, A pitch determination and voiced / unvoiced decision algorithm for noisy speech, Speech Communication, 21, pp.191–207, 1997.
8. W. B. Klejin *et al*., Speech Coding and Synthesis, Elsevier Science B.V., 1995.
9. R.Le Bouquin, Enhancement of noisy speech signals: Application to mobile radio communication, Speech Communication, 18, pp.3–19, 1996.
10. Jerry D. Gibson, Filtering of colored noise for speech enhancement and coding, IEEE Transactions on Signal Processing. V. 39. N. 8, 1991.
11. J.K. Kim, M.J. Bae, A study of Pitch Extraction Method by using Harmonics Peak-Fitting in Speech Spectrum, Proc. ICSP2001, Vol. 2, pp. 617–621, 2001.

Toward Acoustic Models for Languages with Limited Linguistic Resources

Luis Villaseñor-Pineda[1], Viet Bac Le[2],
Manuel Montes-y-Gómez[1,3], and Manuel Pérez-Coutiño[1]

[1] Laboratorio de Tecnologías del Lenguaje,
Instituto Nacional de Astrofísica, Óptica y Electrónica
Sta. María Tonantzintla, Puebla, México
{villasen, mmontesg, mapco}@inaoep.mx
[2] Laboratoire CLIPS-IMAG,
385, Avenue de la Bibliothèque,
B.P. 53, 38041 Grenoble Cedex 9
Viet-Bac.Le@imag.fr
[3] Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia, España
mmontes@dsic.upv.es

Abstract. This paper discuses preliminary results on acoustic models creation through acoustic models already in existence for another language. In this work we show as case of study, the creation of acoustic models for Mexican Spanish, tagging automatically the training corpus with a recognition system for French. The resulting set of acoustic models for Mexican Spanish has gathered promising results at the phonetic level, reaching a recognition rate of 71.81%.

1 Introduction

A system for continuous speech recognition is formed by (i) a system, which using a set of acoustic models from the target language, builds a chain of symbols (usually phonemes) starting from acoustic boundaries extracted from the voice signal; and (ii) a system responsible for the reconstruction of words and sentences given a language model adapted to a language and, often adapted to the application domain of the recognition system [1]. Current statistical techniques used in the computation of acoustic models demands large volumes of data (oral and text corpus). Thus, specification, compilation and tagged of such data volumes are complex tasks and the human effort required is huge.

There are a diversity of initiatives in order to develop large acoustic data bases, like GlobalPhone data base [2], which has compiled data for Arab, Chinese, Croatian, German, Japanese, Korean, Portuguese, Russian, Spanish, Swedish and Turk languages. To date, the project has compiled 233 hours of speech from 1300 speakers approximately. Another effort is the SpeechDat project [3], currently with a total of 28 data bases for 11 European languages and some preponderant dialect variants and minority languages. These data bases have been compiled as basic elements for the development of telephonic applications like information services, transactions and other voice-based services. It is evident that such initiatives operate with a huge

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 433 – 436, 2005.

amount of human and material resources. Given this context, the treatment of minority languages within a lack of resources becomes extremely difficult. The aim of the present work is to take up this problem proposing a methodology in order to bring down the amount of required data to model languages with few resources, reusing both data and models existing for languages with abounding resources.

The remaining sections present a case of study which reuses the acoustic models initially developed for French, to construct the acoustic models for the Mexican Spanish. The relevance in the definition of this methodology is the possibility of the automatic treatment (from systems for language identification to specific recognizers) of the indigenous languages spoken by 54 ethnic groups distributed along the territory of Mexico.

2 Proposed Methodology

The aim of this work does not strive in the development of a system for automatic recognition of multilingual speech with recognition capabilities for several languages with a recognition quality equivalent between such languages. This work attempts to develop a monolingual system for a specific language (target language) reusing data and acoustic models from another language (source language).

The main idea lies on the hypothesis that there is some mapping between the phonemes of both languages. Thus, the challenge consists in the definition of the most pertinent correlation, i.e. what phonemes in the base language are the nearest to those in the target language? There are two approaches to answer this question: the knowledge-based methods and the data-based methods [4]. The first try to determine the correlation through phonetic similarity of the data, this require an expert whose define similarities and determine the correlation. One disadvantage of this approach is that the determination of acoustic units is performed independently from acoustic data. Then, the quality of the acoustic models depends on the quantity of oral data for the target language. Automatic methods derive the acoustic units using few acoustic data from source language, this require either, confusion matrixes analysis or distance metrics (usually based on relative entropy) to determine what model of the base language is the closer to the model of the target language. One disadvantage of such approach is the presence of particular sounds in the target language, which are not present in the base language.

In this work we explore the knowledge-based approach and propose an *a priori* mapping established with the help of linguists and our own expertise. As second step, after establish the mapping, a suite of acoustic models adapted for the target language is set up. With these models we start an automatic alignment process on a set of recordings. Thus, the first *real* version of the acoustic models is computed starting from these recordings and their approximated alignment. Then, the corpus is realigned with the first version and a second version of the acoustic models is computed. The process of alignment and computation for the acoustic models are repeated until the difference in the recognition between versions is minimal.

2.1 Spanish–French Mapping

The correlation between Spanish and French phonemes is relatively simple, given that the majority of Spanish phonemes are present in French. Only three phonemes require a special treatment. In these cases it was needed to approximate the model for Mexican Spanish starting from two French models. In the French system, a phoneme is modeled by a three state Hidden Markov Model: initial, intermediate and final. The approximation consists in take initial and intermediate state of a French model and combine them with the final state of a second model. This is the case for phonemes /tS/, /x/, /ll/. Table 1 shows the approximations applied in this experiment.

Table 1. Approximations for Mexican Spanish starting from French models[1]

Grapheme	Spanish	French	Approximations		
			initial	intermediate	final
ch	tS	t + S	**t**	**t**	**S**
j	x	k + h	**k**	**k**	**h**
ll	L	j + J	**j**	**j**	**NJ**

3 Preliminary Results

Starting from the 22 acoustic models adapted for Mexican Spanish, we made the first automatic alignment of recordings for a Mexican Spanish corpus [5]. Starting from this data set we compute the acoustic models for Mexican Spanish. For this step the corpus was divided. The training set is formed for 4694 sentences and test set for 1173 sentences. The size of the vocabulary is of 8754 different words. The corpus contains recordings from 100 speakers, for each of them, the corpus contains 60 recordings. Each recording was made with short sentences (3 seconds aprox.) for a total of 5 hours of recording.

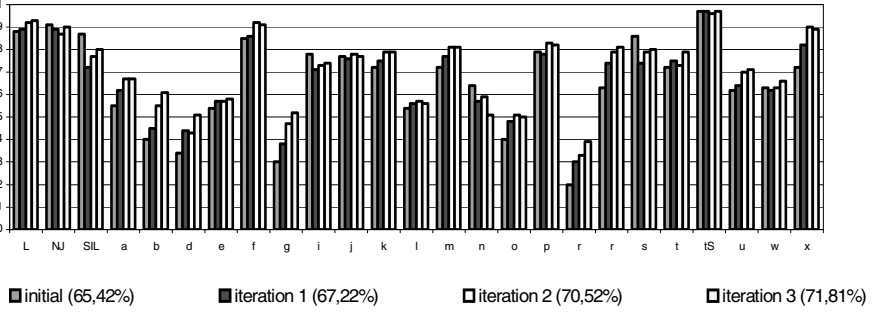


Fig. 1. Results gathered at the phonetic level

[1] The notation used for the phonologic transcription is defined in SAMPA, www.phon.ucl.ac.uk/home/sampa

The computation of the models required 43 features: 13 MFCC coefficients (besides their first and second derived), zero crossing, and the energy (plus first and second derived); besides we used the 3 state HMMs. For the computation we use the JANUS toolkit [6].

After the computation of the first version of the models, the process of realignment and computation of new models was repeated 3 more times, for each iteration the phonetic recognition was evaluated. In order to reach an accurate evaluation, we apply the JANUS recognition system, using a language model with equal probability for the phonemes. Thus, the evaluation is relative and only serves to establish a reference point[2]. Therefore the recognition rate at phonetic level is 71.81%. It is important to note that for the case of French, the recognition rate is 68%. Figure 1 shows the evolution of the results during the four iterations.

4 Conclusions

The treatment of languages in scenarios where resources are limited is extremely important, more over in the context of the Mexican reality. The proposed methodology shows promising results, at least, between Mexican Spanish and French. As further work, we envisage the combination of the knowledge and data based approaches, also the inclusion of other features; the later for both cases, the Mexican Spanish–French as well as Mexican indigenous languages. In the treatment of Mexican Spanish we will also perform some other experiments: (i) redefine the phonetic mapping of "r" and "rr" given that it was not satisfactory with the substitution of correlations r → R and rr → R for: r → l and rr → l; (ii) introduce phonologic variants in the dictionary, currently there is only one slot for each word; (iii) perform a full evaluation with the recognition system at the word level.

Acknowledgements. This work has been partly supported by the project "Man-Machine Spoken Interaction" LAFMI (Laboratorio Franco Mexicano de Informática).

References

1. Manning C. and Schütze, H. *Foundation of Statistical Natural Language Processing*. MIT Press, 2000
2. T. Schultz, T. and Waibel A. "*Language independent and language adaptive large vocabulary speech recognition*" Int. Conf. on Spoken Language Processing, Australia, 1998
3. www.speechdat.org
4. Beyerlein, P., Byrne, W., Huerta, J., Khudanpur, S., Marthi, B., Morgan, J., Peterek, N., Picone, J. and Wang, W. (1999) *Towards language independent acoustic modeling*, ASRU.
5. Pineda, L., Cuétara, J., Castellanos, H., López, I., Villaseñor, L. (2004). *DIMEx100: A New Phonetic and Speech Corpus for Mexican Spanish*. IBERAMIA, pp 948-957. Lecture Notes in Artificial Intelligence. Springer-Verlag. (in Press).
6. Rogina, I. and Waibel A. (1995). *The Janus Speech Recognizer*, Proceedings of the ARPA SLT workshop.

[2] A backward of this kind of evaluation using the language model mentioned will be, for instance, the preference for short chains of phonemes over large chains.

A Study on Pitch Detection in Time-Frequency Hybrid Domain

Wangrae Jo, Jongkuk Kim, and Myung Jin Bae

Information and Telecommunication Engr.,
Soongsil University, Seoul, Korea
wrjo@unitel.co.kr

Abstract. In this paper, we proposed a new method that can improve the accuracy of cepstrum pitch detection and can reduce the processing time. We control the phase information of cepstrum for making the pitch peak maximum. So we extract the exact pitch period easily. We shorten the processing time by omitting the bit-reversing process from the FFT and IFFT computation.

1 Introduction

The accurate pitch extraction is very important in speech signal processing. The accurate pitch extraction is very important in speech signal processing. If we can measure the pitch period accurately, the accuracy of speech recognition can be higher due to the decrement of speaker dependent effect and we can change the characteristic of synthetic voice easily. Because of this importance, various pitch detection methods have been proposed and it can be divided into time domain, frequency domain and time-frequency domain method.

There are ACM, AMDF, parallel processing method etc. in time domain method. It's processing algorithm is very simple but it is difficult to detect accurate pitch period in transition region of speech signal[1]. There are harmonics analysis method, Lifter method and Comb-filtering method etc. It is little affected at the phoneme transition but the large frame size for the high resolution makes the processing time longer and can't reflect the change of pitch period quickly. The time-frequency domain method has the advantages of time domain method and frequency domain method at the same time. But the computation complexity is the main drawback [2].

In this paper, we propose a new method that can improve the accuracy of cepstrum pitch detection method and can reduce the processing time. We adjust the phase information of cepstrum for making the pitch peak maximum so we extract the exact pitch period easily. And we shorten the processing time by omitting the bit-reversing process from the FFT and IFFT computation.

2 Enhanced Hybrid Domain Pitch Detection

If a signal is composed of three signals, the different phase of signals make the waveform complicate as shown in Fig. 1(b). This fluctuation of signal makes the pitch peak small in cepstrum domain. So it is difficult to estimate the accurate pitch period. If we

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 437 – 440, 2005.

adjust the phase of three signals to the same and synthesize them, the peak appears on every fundamental period and the peak is enlarged on the common period as Fig. 1(d). The speech signal also is constructed by the various waves so the synthetic wave has complicated form. For this reason it is difficult to estimate the accurate pitch period. But if we transform the signal into the spectrum and adjust the phase to the same and inverse transform into the cepstrum then enlarged pitch peak appears on the common period. As a general rule the pitch period is 2.5msec~25msec. So in case of 11kHz sampled signal, if we search the maximum peak in range of 27~270 sample, then the pitch period is obtained easily.

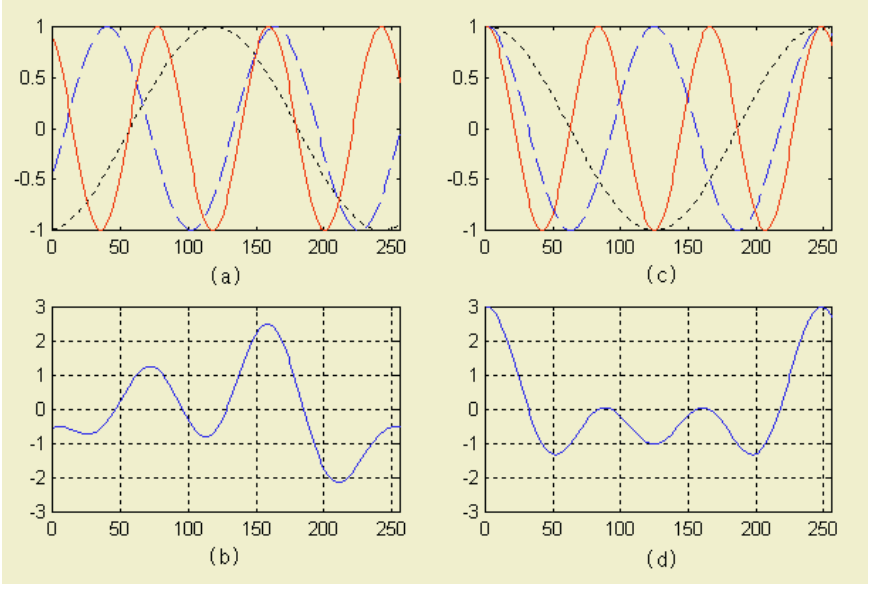


Fig. 1. The effect of phase control

The FFT cepstrum has two major drawbacks. One is that the radix 2 FFT only works on sequences with length which is a power of 2 and the other is that the FFT has a certain amount of overhead (e.g. bit-reversing) which is unavoidable. This overhead affects the processing time of time-frequency domain method e.g. cepstrum analysis.

In this paper we use the method that can reduce the processing time of FFT cepstrum by omitting the bit-reversing of FFT and IFFT. The conventional FFT cepstrum method uses the same algorithm for FFT and IFFT. Therefore the conventional method requires bit-reversing and has unavoidable overhead. But if we apply different algorithm to FFT and IFFT, the bit-reversing can be omitted from the FFT and IFFT of cepstrum analysis. In other word, if we use DIT algorithm with normal ordered inputs in FFT and use DIT algorithm with bit reversed inputs in IFFT then we can obtains cepstrum with normal ordered outputs. The block diagram of the conventional method and proposed method can be represented as in Figure 2.

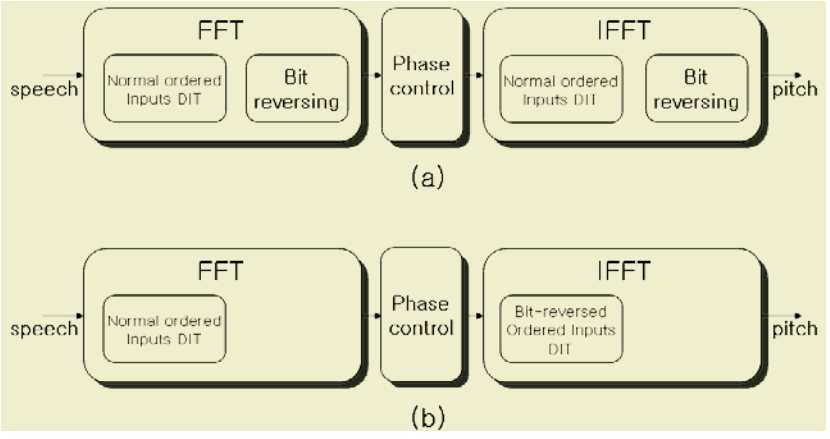


Fig. 2. Comparison of FFT cepstrum process
(a) by the conventional method (b) by the proposed method

3 Experimental Results

For the performance test we implemented the conventional algorithm and proposed algorithm of pitch detection method in IBM-PC/Pentium IV with C language. We estimate the pitch contour by conventional and proposed method, respectively, on the Korean sentence *Insune komaneun cheonjae sonyeunul joahanda*.

We showed the pitch detection of a frame by the conventional method and the proposed method in Fig. 3 (a) and Fig. 3 (b), respectively. The result represent that it is easy to extract the pitch period by proposed method for reason that the enlarged pitch peak as shown in figure.

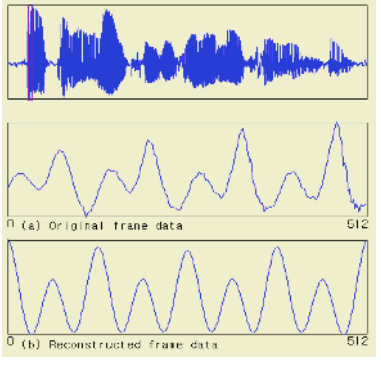


Fig. 3. The comparison of pitch detection (a) by the conventional method, (b) by the proposed method

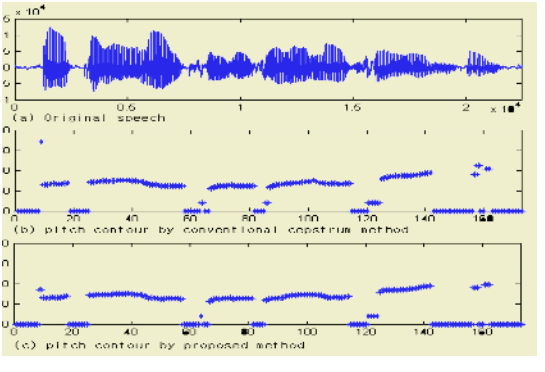


Fig. 4. The comparison of pitch contour (a) by the conventional method, (b) by the proposed method

Figure 4 represents the comparison of pitch contour about the whole sentence. We know that we can estimate the accurate pitch by the proposed method.

Also we measure the processing time of the conventional method and the proposed method of FFT cepstrum for 128 points, 256 points and 512 points inputs. In case of 512 points cestrum analysis, the processing time of proposed method and conventional method are 3,550 usec and 4,188 usec respectively as shown in table 1. As a result the proposed method can reduce the processing time to 84.4% compared to conventional FFT cepstrum method.

Table 1. Comparison of processing time

	Processing Time [us]		Rate (B/A)
	Conventional method (A)	Proposed method (B)	
128 points	835	712	85.3%
256 points	1,863	1,638	87.9%
512 points	4,188	3,550	84.8%

4 Conclusions

In this paper we proposed a new pitch detection method in time-frequency hybrid domain. We transform a speech signal to a speech spectrum by the FFT and control the phase for enlarging the pitch peak. The phase controlled speech spectrum is transformed into speech signal. The pitch of processed signal is detected easily. We also proposed a new method that can reduce the processing time. We shorten the processing time by omitting the bit-reversing process the FFT and IFFT computation.

Acknowledgement. This work was supported by the Soongsil University Research Fund.

References

1. L. R. Rabiner and R. W. Schafer, Digital Processing of Speech signals, Prentice-Hall, New Jersey, 1978.
2. M. Bae and S. Ann, “Fundamental Frequency Estimation of Noise Corrupted Speech Singals Using the Spectrum Comparison,” J., Acoust., Soc., Korea, Vol. 8, No. 3, June 1989.

VoiceUNL: A Semantic Representation of Emotions Within Universal Networking Language Formalism Based on a Dialogue Corpus Analysis

Mutsuko Tomokiyo[1] and Gérard Chollet[2]

[1] GETA-CLIPS-IMAG & ENST
BP 53 38041 Grenoble cedex 9 France
mutsuko.tomokiyo@imag.fr
[2] ENST
46 rue Barrault, 75634, Paris
{tomokiyo, frchollet}@tsi.enst.fr

Abstract. The paper aims to propose a semantic representation of emotions for oral dialogues, based on an analysis of real-life conversations, telephone messages and recorded TV programmes, for the purposes of a speech to speech machine translation. Lexicon and phatics are one of important emotion eliciting factors as well as gestures, prosody and voice tone in oral dialogues. So, the semantic representation is made in a way where these factors are taken into account at the same time. Also, it's done within **U**niversal **N**etworking **L**anguage (UNL) formalism, where UW (universal word) plays an important role.

1 Introduction

This work has been carried out in the framework of "VoiceUNL" [21], which is one of subprojects of "LingTour" and "Normalangue"[1] projects. The "VoiceUNL" is an extension of UNL, which is a text-oriented machine translation environment, to oral dialogues.

As for **s**peech to **s**peech **m**achine **t**ranslations (SSMT), the detection of emotions in source languages and its generation in target languages are an important issue from the viewpoint of the naturalness of dialogues [7], because *emotions entails distinctive ways of perceiving and assessing situations, processing information, and prioritising and modulating actions* [24]. It's the key reason for proposing a semantic representation of emotions.

In this paper, we introduce **U**niversal **N**etworking **L**anguage (UNL) briefly in section 2. In section 3, after having surveyed existing approaches to emotion detection, we define emotions and study emotion classes in section 3. In section 4, we detect emotion eliciting factors and extract emotional expressions in telephone messages, conversations between secretaries and callers, and an audio-visual corpus we have developed. In sections 5, we propose to annotate lexicon with a set of emotion labels,

[1] The Lingtour and Normalangue projects were launched in 2002 by the partnership which consists of TsingHua University (China), Paris 8 University (France), INT (France), ENST-Paris and Bretagne (France), and CLIPS (France). One of the objectives of the projects resides in R & D to enable multilingual-multimedia MT on user-friendly tools [1].

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 441 – 451, 2005.

followed by section 6, where we show a semantic representation of emotions within UNL formalism, by adding tags representing speech dialogue properties to UNL to suit it to SSMT.

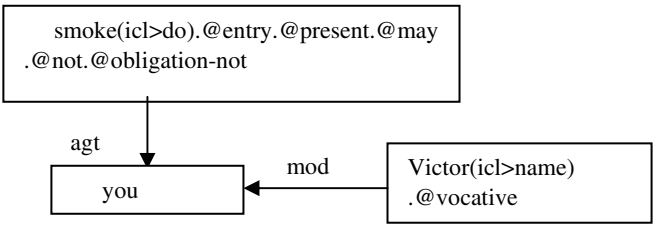
2 UNL

One of the main advantages of UNL is the Universal Word (UW) dictionary, which enables us to specify word meaning at the deep level and to perform lexical disambiguation in a semantic oriented formalism. The UNL consists of "UWs", "Relations", "Attributes" and the UNL knowledge base represented in the form of tags. The "UWs" form the vocabulary of UNL. "Relations" and "Attribute" mainly make up the syntax, and the knowledge base covers the semantics of UNL [4]. Here is an example of UNL specification and its graphs [5].

— *May I smoke?* [S:01]
— *No! You may not, Victor.* [S:02][2]

[S:01]
{org:e1}May I smoke?{/org}
{unl}agt(smoke(icl>do).@entry.@present.@may.@interrogative, I){/unl}
[/S]

[S:02]
{org:e2}No! you may not, Victor{/org}
{unl}agt(smoke(icl>do).@entry.@present.@may.@not.@obligation-not, you)
mod(smoke(icl>do).@entry.@present.@may.@not.@obligation-not, no)
mod(no, !(icl>symbol).@surprised)
mod(you, Victor(icl>name) .@vocative){/unl}
[/S]



An UW is made up of a character string followed by a list of constraints. The UW includes basic UWs (bare English words, called "Head words"), restricted UWs (English words with a constraint list), and extra UWs, which are a special type of restricted UWs [4].

In the example, "agt" and "mod" refer to Relation tags, which indicate dependency relation between a head word in a narrow linguistic sense and phrases dependent on

[2] We have not yet deal with discourse ellipsis like ' *you may not* ' for ' *you may not smoke* ' in this example. So, the specification is done being presupposed that the utterance is ' *you may not smoke* '.

the head word, based on a case grammar type specification. ".@entry", ".@present", ".@may", etc. are called Attribute tags, which indicate the grammatical conditions of a given utterance. The graphs in the example do not contain any tags a part from the UNL tags, so they will be merged with other tags added as well as embedded in another format for the purposes of SSMT. "icl" in the constraint list enables us to define subconcept of the "Head word". We also apply this constraint way to lexicon having emotional content for the purposes of an emotion representation. E.g.:

"state(icl>abstract thing)"	refers to the mental, emotional or physical condition.
"state(icl>country)"	refers to a country considered as a political organization.
"sad(icl>unhappy)"	refers to an unhappy sentiment, where something unpleasant has happened.

3 Previous Studies

3.1 Existing Approaches to Recognition and Generation of Emotions

Much work has been carried out in detection and identification of emotions in written texts or oral dialogues for various applications. Existing approaches are grouped into three types:

- observation of non-verbal elements such as prosody, facial and body movements in spoken languages,
- understanding the meaning of lexical items expressing emotions, or
- the analysis of the distribution of grammatical constituents in a sentence and applying it to templates to identify emotion class [6][7].

Our approach employs a method where spoken language properties such as lexicon, gestures, prosody, etc. are recognized, translated and generated, since one objective of our emotion representation is SSMT using UNL framework.

In fact, in order to determine the type of emotion, these elements are taken into account at the same time, because the same variable can express different classes of emotions. For example an increase of elocution speed or the rising tone can indicate joy as well as anger [22].

3.2 Emotion Definition and Its Class

We define emotions, according to Randall [8] as a feeling that has been caused by certain beliefs, directed toward a primarily conceptual and non-perceptual target that typically produces some physiological, behavioural, or cognitive effect.

How many and what kind of emotional expressions are used in general dialogues?

In OCC [9], class and intensity of emotions are analyzed from cognitive points of view: basic emotions, emotions as reactions to events, to objects, and to agents.
Plutchik [10] believes that emotions are like colours. Every colour of the spectrum can be produced by mixing the primary colours. His "emotion's wheel" consists of eight primary emotions: fear, surprise, sadness, disgust, anger, anticipation, joy, and acceptance.

Ekman [11] mentioned that there would be a linking of a second emotion with an initial emotion, and emotions rarely occur simply or in pure form.

Randall [8] states that most cultures have emotions and emotional vocabularies that have two components: a universal element, and a component or parameter that is peculiar to the beliefs and values of that culture. We have, however, not yet entered into such a complexity of emotions.

In this work, we adopt the 24 categories of human emotions defined by OCC [9], and simplify them into the following 10 categories by looking up our corpus :

happiness, sadness/disappointment, disgust, surprise, fear, anger, irritation, hesitation/uncertainty, anxiety, neutral

These class names are used later to annotate lexicon having emotional content.

4 Dialogue Corpus Analysis

4.1 Our Corpus

Emotion-eliciting factors are central to the concept of representing emotions. In our first approach to emotion eliciting factors, we have developed a corpus, which contains:

- 30 minutes of English instruction programmes on TV,
- a 40-minute French TV interview [5],
- 5½ hours of real-life vocal messages left on a telephone answering machine, sent from medical stuff to a group of computer engineers in a French public hospital [13],
- 1 hour of real-life telephone conversations between administration stuff of a French university [12] and
- 6 basic conversations on transport in English, French, Japanese and Chinese [25].

4.2 Emotion Eliciting Factors

In our corpus, the followings are the major emotion eliciting factors :

- lexicon (sad, happy, etc.)
- phatics (ah, hein, etc.)
- prosodies (fast, slow, strong, etc.)
- voice (noisy, soft, etc.)
- gestures (movement of hands, mouth, eyes, etc.)

As an example, “No!” in the example [S:02] expresses Victor’s father’s surprise, because Victor is a small boy.

Note that the surprise can be represented by the lexicon “No!”. However, at the same time, on TV, the father also made a grimace while saying “No!”. So, it can also be expressed by the movement of his eyebrows and voice tone.

4.3 Potential Emotion Expressions

In the following table, we illustrate the potential representation of different emotions in terms of those mentioned above.

Table 1. Potential representation of emotions

	happi-ness	sad-ness	dis-gust	sur-prise	fear	an-ger	irrita tion	hesi-tation	anxi-ety	neu tral
lexicon	*	*	*	*	*	*	*	*	*	*
phatics	*	*	*	*	*	*	*	*	*	
prosody	*	*	*	*	*	*	*	*	*	*
voice	*	*	*	*	*	*	*	*	*	*
hands	*						*			
mouth	*	*	*	*						
eyes	*	*	*	*	*	*				
eye-brows			*	*		*				
head		*		*				*		
shoul-ders			*	*						
fingers							*			

4.4 Lexicon Having Emotional Content

The subjects of telephone conversations recorded at a French university are room reservation, schedule arrangement, taking a message, order of office supplies, etc., and some chats also are contained.

In the telephone messages at a public hospital, callers complain about problems with their computers or the software they use, and ask for technical help from an engineer, or ask for a rapid validation of an electronic access card for newcomers. In this context, a typical lexicon, or set of phrases expressing irritation or uncertainty appear in the messages : *pénible, très pénible, drôlement embêté, une catastrophe, désespéré, relativement énervant, Ça me dérange beaucoup, ceci est assez désespérant, c'est embêtant*, etc.[3]

In Table 2, we illustrate lexicon of different emotions used in the telephone messages and the conversations.

On the other hand, prosodic manner common to this lexicon or set of phrases is, however, not necessarily found in the messages as shown in Table 3. For instance, "*c'est relativement énervant*" or "*c'est très pénible.*" is uttered at a neutral prosodic level.[4]

Therefore, emotion eliciting words or phatics for each emotion class are surely found, but prosodic manner for each emotion class is divergent, whereas there are

[3] We have also found utterances which emotional interpretation depends upon the contexts uttered : "*C'est vraiment très urgent, Pourriez-vous venir voir?, Si vous pourriez passer rapidement".*

[4] We also have verified prosodic characteristics for some lexicon and set of phrases in the messages on Praat [14], but further examinations should be made to study the variety of prosodic manners for each emotion class.

clear prosodic signs which are confined to only one word which is semantically less significant. For example, in an utterance "*Il faudrait impérativement résoudre ce problème ce matin*.", only '*matin*' is heavily accented, all of other words are uttered in a neutral tone, and we can interpret this accentuation as an implicit insistence on an urgent intervention.

Table 2. Lexicon or expressions of different emotions in our corpus

Emotions	lexicon in the telephone messages	lexicon in the administrative dialogs
Happiness		*Ah chouette!, Ouais!, impeccable!, Y a pas de souci!, C'est gentil. Merci!*
Disgust		
Fear	*J'ai peur que, je crains que,*	*J'ai peur que, je crains que*
irritation	*C'est embêtant, on est drôlement embêtés, ça me dérange, ça pose un reel problème, c'est relativement énervant, C'est très pénible*	*Ah non!*
hesitation uncertainty	*Je ne sais que faire, comment faire, nous aimerions savoir, je ne sais plus quoi faire.*	*Voyons voir, attends …je regarde, ben ben…, attends voir; heuh heuh, heueueueuh, Bof bof bof, hum hum, je ne sais pas, je vois pas bien, ça va faire un peu juste; on sait pas*
sadness / disappointment	*Ici infirmière en état désespéré*	*Oh la pauvre! ah mince, ah zut, c'est dommage*
surprise		*Oh, ça alors, ah bon?, tu crois?*
anger		*Y en a marre!, C'est pas vrai!,*
anxiety	*C'est une catastrophe*	*ça m'ennuie un peu, nous sommes ennuyés, c'est ennuyeux, y a une boulette, y a un souci, y a un truc qui me chiffonne,*

These phenomena are parameterized as emotion eliciting factors and are described in the structures of attributes and its values in the emotion representation.

Lexicon is a bench mark for detecting and identifying emotions as mentioned above. So, it's useful to mark lexicon with some labels in a dictionary used just like restricted UWs in UNL.

We propose, due to this fact, a set of emotion labels composing of 9 classes excluding "neutral" in our emotion classes and annotate lexicon in UNL manner. E.g.:

énervant(icl>sentiment>irritation)
catastrophe(icl>sentiment>anxiety)

Table 3. Lexical units and their prosody in Corpus Hotline CHRU

items	lexical units	particulars	examples	file ID
same lexical unit with different prosodies	*urgent*	emphasized	*on a un petit problème urgent*	mar5/11A 14,57
	extrême urgence	neutral	*il nous le faut d'extrême urgence*	mar5/11A 2,34
lexicon having emotional contents	*au secours, au secours*	neutral	*au secours, au secours, il faut absolument que je travaille sur cet ordi .....*	mar 24/12 24,14
	c'est infernal	neutral	*on passe des heures à connecter déconnecter l'ordinateur […] pour travailler, c'est infernal.*	vend 13/12 23,34
	énervant	neutral	*c'est relativement énervant*	jeu 19/1219,45
	désagréable	neutral	*ceci est assez désagréable, toutes les semaines*	mar17/12 8,52 19/1219,45
No lexicon having emotional contents in the entire utterance	*Il y a 9 étiquettes sur une et 16 étiquettes sur l'autre*	irritated	*L'imprimante nous imprime les étiquettes sur deux feuilles, qui sont toutes les deux incomplètes. Il y a 9 étiquettes sur une et 16 étiquettes sur l'autre*	vend 8/11-17,16
lexicon not having emotional contents	*connecter and lundi matin*	emphasized	*Je n'arrive absolument plus à me connecter […] Est-ce que vous pourriez intervenir lundi matin?*	mar 24/12 15,58
	toutes les semaines	insistent	*ceci est assez désagréable, toutes les semaines*	mar24/12 15,58

5 Semantic Representation of Emotions

The UNL semantic representation for written texts is actually designed by a set of 113 tags, which are divided into the Relation tags and Attribute one [4]. As for SSMT, some tags covering spoken language properties are merged with the UNL tag set : 9 emotion tags, 8 prosody tags from the W3C recommendation [15] for speech recognition and synthesis, 12 behaviour tags from MPEG-4 [16] for gesture control process-

ing and, in particular, 28 speech act tags from a speech act research team [17] [18] and 5 interaction manner tags from GDA [19] for dialogue discourse.

The UNL representation is a graph and consequently is not easy to encode in a linear data stream. However, it is feasible to project it onto a description format such as XML, which authorizes the definition of elements and attributes. The representation obtained offers the same expressive power as graphs, but in the form of tags, and is easy to transmit. It is therefore easily interpreted by a DTD (Document Type Definition) conforming to the XML norm [26]. Thus, we attempted to transform UNL graphs into XML format as it facilitates speech synthesis information the generation of the target language.

5.1 Linguistic and Paralinguistic Tags

The representation schema of emotions proposed is made by adding tags expressing emotions according to the UNL. There are three ways to add such tags, that's adding tags: "outside" of UNL makers as <VoiceUNL>, "inside" UNL text or a combination of both [21].

When emotions are formalized "inside" of the UNL makers, all tags representing prosody, behaviour and the speech act one are put in an UW. Therefore, in UNL graphs the arc concept representing a semantic relationship between two UWs might turn out to be unclear.

On the other hand, when emotions are formalized "outside" the UNL marker, in order to synchronize character's strings and speech and visual items occurring simultaneously in an utterance, the same character's string appears several times in a semantic representation. For example, the emotion of Victor's father could be interpreted as his surprise by looking up the prosody of his utterance and his eyebrow movements, when he cried "No!".

In such a dilemma, we create an additional UW type, which enables us to link speech, gesture, emotion and prosody tags : SP01, SP02, SP03...., and we use them in an "outside" and "combined" manner.

The following is a representation for the example [S:02] in the "combined" manner:

```
<?xml version="1.0" encoding="iso-8859-1 ?>
<D dn=" TV " on="mt" dt="2003">
<Paragraph number="1">
<Sentence  snumber="2">
<org lang="el"> No! you may not, Victor.</org>
<unlsem>
agt:SP01(smoke(icl>do).@entry.@present.@obligation-not,
       you.@emphasis)
mod:SP02(smoke(icl>do).@entry.@present.@obligation-
       not,no(icl>sentiment>surprise).@emphasis)
mod:SP03(no(icl>sentiment>surprise).@emphasis,
       !(icl>symbol>surprise).@surprised)
mod:SP04(you,Victor(icl>name).@vocative)
</unlsem>
<VoiceUNL><speech-act>type="inform" mod:SP01, type="No" agt:SP02
       </speech-act>
<interaction> ref="smoke" agt:SP01 </interaction>.
<emotion> class="surprise" mod:SP02 </emotion>
```

```
<gesture>eyebrows="left-and-right-raised" mod:SP02 </gesture>
</VoiceUNL>
</Sentence >
</Paragraph>
</D>
```

Note that "no" is annotated as "no(icl>sentiment>surprise)" by one of emotion class tags. It means that this "no" refers to a surprise as well as a negation[5].

On the other hand, prosody tags (.@emphasis) are attached on UWs between <unlsem> and </unlsem>, and the gesture, emotion and discourse tags are external to <unlsem>, because only prosody is identified at the level of UWs, and the rest is often associated with utterance fragments or an entire utterance.

"smoke", "you", "no", etc. are pivot languages called UW, and are converted into "fumer", "tu", "non" respectively in the French generation module [2] [20][23]. Therefore, the transcription of this utterance is: "Non! tu ne peux pas fumer, Victor".

5.2 Interaction Manner Tags

We have found overlapping of utterances, irregular turn taking, category omission, deictic expressions, etc. in our corpus [17]. Such interaction manners also are concerned with emotions of the speaker.

We actually use 5 tags from GDA[6] tag set [19] in the same way as paralinguistic tags to represent them as specificity of oral interaction manners. The GDA includes tags which enable previous utterance to be referenced, repeated utterance fragments to be annotated, or omitted category in an utterance and deictic expressions to be referred to. So, we synthesize some tags from the GDA in our semantic representation of emotions: anaphoric reference, deictic, overlap, repair and repeat tags.

For example, elliptic or anaphoric phenomena in context is tagged with a combination of a relational attribute 'id' and a referential index 'ref" in the GDA. We adopt this idea for a referentiality of the main verb "smoke", which is omitted in the example [S:02] and indicate the element to be referred to outside of UNL markers. E.g.:

in [S:01] <interaction> id="smoke" agt:SP01</interaction>
in [S:02] <interaction> ref="smoke" agt:SP01</interaction>

6 Conclusion

After having conducted a dialogue corpus analysis, we have suggested a delicate relationship between lexicon uttered and its prosodic manner. Thus we have proposed

---

[5] Many previous studies have indicated that F0 raising contour is evoked by the happiness, surprise and anger in contrast to F0 falling contour which is evoked by the sadness or the uncertainty [3] [22]. This "no" is uttered in strong raising F0 contour on Praat.

[6] The Global Document Annotation (GDA) Initiative research team has proposed (2001)a XML-based tag set to help computing machines automatically infer the underlying semantic/pragmatic structure of documents. The GDA tag set is designed so that the GDA-annotation reduces the ambiguity in mapping a document to a sort of entity-relation graph (or semantic network) representing the underlying semantic structure [19]. There is a mapping schema between UNL specification tags and GDA one.

a semantic representation of emotions in a way where all emotional expressions such as lexicon, prosody, gestures, etc. are described at the same time, by annotating lexicon with a set of labels, and adding speech property tags, speech act tags, interaction manner tags and behaviour tags to UNL in order to suit it to SSMT.

We also have discussed emotion representation in three ways within the UNL format and mentioned the advantage of the "combined" representation formalism.

The next step will be to develop a prototype with a speech and image interface as well as to enrich our corpus with speech and sound.

Acknowledgements. The authors thank Professor Ch. Boitet, Responsible at GETA in CLIPS, for many inspiring discussions and the opportunity to conduct this research in the framework of "UNL" and "Papillon" projects.

References

1. Lingtour, Chollet G., Project Brochure, Lingtour, Paris, 2003.
2. GETA-CLIPS-IMAG, French deconverter, 2000.
3. Scherer, K.R., Psychological models of emotion, in J.Borod. The neuropsychology of emotion, Oxford / NewYork University Press, 2000.
4. UNL Center and UNL Foundation, The Universal Networking Language(UNL) Specification (Version 3 edition 1), Japan, 2002.
5. TV programmes on the 5th of October, 'Victor', Arte, 2003.
6. Holzman Lars E.and Pottenger William M, Classification of Emotions in Internet Chat: An Application of Machine Learning Using Speech Phonems, 2003.
7. Fitrianie S., Wiggers P., and Leon J.M.,Rothkrantz L., A Multi-Modal Using Natural Language Processing and Emotion Recognition, Text, Speech and Dialogues, 6th International conference, TDS 2003, Czech Republic, Proceedings, LNAI 2807, V. Matousek and P.Mautner (Eds), Springer, 2003.
8. Randall, R.D., The nature and Structure of Emotions, U.S.Military Academy, USA, 1998, http://neologic.net/rd/Papers/EM-DEF19.html.
9. Ortony A., Clore G. L, Collins A, The Cognitive Structure of Emotions, Cambridge, 1990
10. Plutchik R., The nature of emotions, American Scientist 89 344, USA, 2001.
11. Ekman P., Emotions Revealed: Recognizing Faces and Feelings to Improve Communication and Emotional Life, Times Book, USA, 2003.
12. Corpus DialAdmin, Document CLIPS, Grenoble, 2003.
13. Corpus Hotline CHRU, Document CLIPS, Grenoble, 2003.
14. Boersma P. and Weenink D., Praat : doing phonetics by computer (version 4.1), Institute of Phonetics Science, University of Amsterdam, http://www.fon.hum.uva.nl/praat/, 2003.
15. W3C, Speech Synthesis Markup Language Version 1.0, W3C Working Draft 02, 2002, http://w3.org/TR/2002/WD-speech-synthesis-20021202, W3C, Semantic Interpretation for Speech Recognition, W3C Working Draft 02, 2003, http://w3.org/TR/2003/WD-semantic-interpretation-20030401.
16. Koenen P.R. (ed), MPEG-4 Overview, http://mpeg.telecomitalialab.com/standards/mpeg-4.html, 1999.
17. Tomokiyo M., Analyse discursive de dialogues oraux en français, japonais et anglais, Septentrion, Lille, 2001.
18. Seligman M., Tomokiyo M. and Fais L., A Bilingual Set of Communicative Acts Labels for Spontaneous Dialogues, Rap.ATR, TR-IT-161, Japan, 1996.
19. Hasida K.,The GDA Tag Set (version0.68), http://i-content.org/gda/tagman.html(version 0,71), 2003.

20. Tsai WJ., UNL news, http://www-clips.imag.fr/geta/User/wang-ju.tsai/showunl.html, 2003.
21. Tomokiyo M. and Chollet G.,VoiceUNL : a proposal to represent speech control mechanisms within the Universal Networking Digital Language, International Conference on the Convergence of Knowledge, Culture, Language and Information Technologies, Egypt, 2003.
22. Caelen-Haumont, G., BEL, B. Subjectivité et émotion dans la prosodie de parole et du chat: espace, coordonnées et paramètres. Colloque international «Emotions, Interactions & Développement» (2001 juin 28-29: Grenoble). In Coletta, Jean-Marc; Tcherkassof, Anna (eds.) Perspectives actuelles sur les émotions. Cognition, langage et développement. Mardaga: Hayen. 2002.
23. Sérasset G. & Boitet Ch., UNL-French deconversion as transfer & generation from an interlingua with possible quality enhancement through offline human interaction, MT-SUMMIT VII, Singapore, 1999.
24. HUMAINE, HUMAINE Technical Annex 2004, http://emotion-research.net/.
25. Corpus TRANSPORT: Basic conversations on transport in English, French, Japanese and Chinese, CLIPS, 2004.
26. Tsai WJ., UNL news, http://www-clips.imag.fr/geta/User/wang-ju.tsai/showunl.html, 2003.

Combining Multiple Statistical Classifiers to Improve the Accuracy of Task Classification

Wei-Lin Wu, Ru-Zhan Lu, Feng Gao, and Yan Yuan

Department of Computer Science and Engineering
Shanghai Jiao Tong University, Shanghai 20 00 30, China
{wu-wl, lu-rz, gaofeng}@cs.sjtu.edu.cn
yuanyan@sjtu.edu.cn

Abstract. Task classification is an important subproblem of Spoken Language Understanding (SLU) in automated systems providing natural language user interface, whose goal is to identify the topic of a query from the user. This paper presents a combination of multiple statistical classifiers to improve the accuracy of task classification in the context of city public transportation information inquiry domain. Three different typical types of statistical classifiers are trained on the same data to be the base classifiers of the combination system: naïve bayes classifier, n-gram model, and support vector machines. The combination method of two-stage classification is emplored to yield better overall performance. Our experiments showed that support vector machines outperform excessively the other base classifiers for task classification in our domain. The comparative experimental results between two-stage classification and voting strategy indicated, under the circumstance that the best base classifier has the overwhelming performance over the other base classifiers, the strategy of two-stage classification was more effective and could produce better results than the best component classifier.

1 Introduction

Task classification is a subproblem of Spoken Language Understanding (SLU) in automated systems providing natural language user interface, whose goal is to identify the topic of a query from the user (e.g. **ShowFare** is the topic for "*What is the minimum taxi fare?* ".) [1,2,3]. It is essentially one type of shallow semantic analysis of the input utterance. If the semantic representation is formalized as a frame with an internal structure consisting of slot/value pairs, then task classification can be regarded as identifying the frame type. Task classification is critical for SLU in many applications, for example, the well-known Airline Travel Information (ATIS) domain. Task classification may help the deep semantic analysis component such as rule-based robust parser by restricting the parser to only apply the grammar corresponding to the recognized task class [2]. It is a typical pattern recognition problem and suitable to be handled using statistical classification techniques. Similar works include call routing [4], classification of speech acts or dialog acts [5,6], etc.

In the literature of task classification, the reported error rates of task classification of text or speech input suggest it is necessary to endeavour to improve the accuracy [1,2,3,9]. There are at least two directions to obtain this goal: (1) Search for the meth-

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 452 – 462, 2005.

ods for extracting more powerful ways of knowledge respresentation (features) fed into the learner; (2) Rather than devising new features, combine different existing learning systems using normal features. We carried out experiments for task classification in the context of city public transportation information inquiry domain [9]. The experimental results on the clean test data showed, the performance of the classifier using deep-level features (Chinese words or non-terminals in the semantic grammar) was improved to a certian extent compared to that of the classifier using shallow features (Chinese characters). On the other hand, the results on the nosiy test data indicated, the performance of the classifier using deep-level features decreased drastically, however the performance of the classifier using shallow features degraded gracefully. Furthermore, one of disadvantages of the former way is that the cost of extracting deep-level features is fairly expensive. In addition, since task classification always runs on the front end and is applied earlier than other deep analysis, thus deep-level features can hardly be available. For these reasons, in this paper our philosophy follows the latter method, i.e. several different existing classifiers using shallow features are combined to reduce the error rate.

In recent years, it has been shown that combination of multiple complementary classifiers can improve the overall performance. These successes can be owed to the facts or observations as follows: (1) typical classifiers behave relatively well, but can hardly perform as well as expected for practical requirement in many applications; (2) different classifiers are based on different learning mechanisms, use various ways of knowledge representation (features) and employ diverse strategies to deal with sparse data problem. Those diversities combine to produce different errors, so that multiple classifiers can be integrated to improve the overall classification system [7]. In fact, decision fusion has made some achievements in several areas, such as part-of-speech tagging and text categorization [7,8]. A naïve combination method (simple voting) was exploited but no improvement had been yielded [2]. It implies that more sophisticated combination methods are required to achieve further improvement.

In this paper, we investigate a combination of a set of diverse statistical classifiers for Task classification in the context of a public transportation information inquiry domain. The classifiers include: naïve bayes classifier (NB), n-gram models (N-gram), support vector machines (SVM). Those learning methods are representative and can be built easily. Each of them employs different models, uses slightly different features, which will be in more details described in the later section. Therefore, those classifiers have the potential to be integrated to improve the overall performance. Another problem to be considered is the ensemble methods. Herein, we emplore the strategy of two-stage classification, in which the decision tree and maximum entropy model are used as second-level classifiers to select their output on the basis of the patterns of co-occurrence of the results from the various first-level classifiers [7]. Another combination strategy, (weighted) voting, is also implemented for comparison with two-stage classification.

The rest of this paper is organized as follows. Section 2 describes the features used by the single classifiers, the algorithms of base classifiers and their properties, and the combination methods. Section 3 reports our experimental setting and results. The last section draws the conclusion and presents discussion and future work.

2 The Component Classifiers and Combination Methodology

In this section, the features fed into the base classifiers are firstly introduced. Then, the component classifiers used in our combination system are briefly described: naïve bayes classifier (NB), N-gram Models (N-grams), Support Vector Machines (SVMs). Finally, the classifier combination methods employed in this paper are discussed.

2.1 The Features Set

In [9], we studied several kinds of features which can be used for Chinese input sentences: Chinese characters, Chinese words and non-terminals in the semantic grammar (concepts). A series of experiments showed that the classifier using Chinese characters as features performed relatively well and had the best robustness on the noisy data (recognized text), compared to other deep-level features available (Chinese words, non-terminals). Another advantage of including Chinese characters as features is its significantly cheaper preprocessing cost, compared to that of the other features. In addition, the classifier using Chinese characters as features is very portable and can generalize well. Therefore, we continue to introduce the GB2312-80 Chinese characters set as the initial feature set, which consists of 6,763 Chinese characters. To reduce computational complexity and improve the performance of the component classifiers, we filtered those hapax legomenon characters, which occur less than 50 times in the frequency table of Chinese character trained on TH-Rcorpus[1], and some characters with extremely high occurrence frequencey such as de ("的"). Finally, we obtain a feature set consists of 4,547 Chinese characters.

2.2 The Component Classifiers

2.2.1 Naïve Bayes Classifier

The naïve bayes classifier is a highly practical bayesian learning method and widely applied to many fields. The naïve part of such a model is the simplifying assumption of feature independence, which makes its computation extremely efficient because it does not use feature combination as predictors. However, it can still achieve the comparable performance to that of other approaches in many domains.

The naïve bayes classifier in our experiments uses a feature vector consisting of 4,547 Chinese characters. Following the practice in [2,9], we represent a query by a vector $\overrightarrow{ch} =< ch_1, \ldots, ch_{|\overrightarrow{ch}|} >$ of dimension $|\overrightarrow{ch}|$ (herein $|\overrightarrow{ch}|$ is equal to 4,547) with binary valued features: 1 if a given Chinese character is in this query or 0 otherwise. With the naïve bayes assumption and bayes theorem, we can get the following decision rule for task classification: choose $\hat{t}$ as the target task if

$$\begin{aligned} \hat{t} &= \arg\max_t P(t \mid \overrightarrow{ch}) = \arg\max_t P(t) P(\overrightarrow{ch} \mid t) \\ &= \arg\max_t P(t) \prod_i P(ch_i \mid t). \end{aligned} \tag{1}$$

[1] This frequency table of Chinese characters can be downloaded from www.lits.tsinghua.edu.cn/ainlp/source1.htm

In Equation (1), $P(ch_i = 1 \mid t)$ and $P(ch_i = 0 \mid t)$ are the probabilities that i^{th} Chinese character is and isn't present in a query of task t respectively. To avoid sparse data problem, $P(ch_i \mid t)$ is computed via maximum likelihood estimation with smoothing as follows:

$$P(ch_i = 1 \mid t) = \frac{N_t^i + \delta}{N_t + 2\delta}, \qquad P(ch_i = 0 \mid t) = 1 - P(ch_i = 1 \mid t) \tag{2}$$

In (2), N_t^i is the number of occurrence of the i^{th} Chinese character in those queries of task t in the training set, and N_t is the total number of queries for task t. δ was tuned to maximize the classification accuracy on the cross-validation data.

2.2.2 N-Gram Models

N-gram model can also be used to predict the task for a query. $P(t \mid \overrightarrow{ch})$, namely the probability that a query represented by a vector $\overrightarrow{ch}$ belongs to the task t, can be also decomposed as the probability $P(t)$ and the probability $P(ch_i \mid ch_{i-1}, \ldots, ch_1, t)$. Thus, the following scenario can be employed to assign each query a task $\hat{t}$:

$$\begin{aligned} \hat{t} &= \arg\max_t P(t) P(\overrightarrow{ch} \mid t) \\ &= \arg\max_t P(t) \prod_i P(ch_i \mid ch_{i-1}, ch_{i-2}, \ldots, ch_1, t) \end{aligned} \tag{3}$$

In accordance with different independence assumption, we can respectively build a task-specific unigram $P(ch_i \mid t)$, bigram $P(ch_i \mid ch_{i-1}, t)$ and so on.

The task n-gram classifiers use n-gram features respectively. The feature vector in the unigram case consists of the Chinese characters that are present in a given sentence, and the vector in bigram contains bigrams of characters is in a given sentence.

For n-gram models, the smoothing is a very important issue. In this paper, given the frequency of $ch_1, \ldots, ch_n$ in the training data is r, the n-gram probabilities for the class specific language models are calculated using the smoothing method of absolute discounting as follows:

$$P_{abs}(ch_1, \ldots ch_n) = \begin{cases} \dfrac{r - \delta}{N} & if \quad r > 0 \\ \dfrac{(B - N_0)\delta}{N N_0} & otherwise \end{cases} \tag{4}$$

Where N is the number of training instances, and N_r is the number of n-grams appeared r times in the training data, and B stands for the number of bins training instances are divided into [10]. In our experiments, δ is simply set an empirical value $N_1 / (N_1 + N_2)$.

2.2.3 Support Vector Machines

The support vector machines learning method is well-founded in terms of computational learning theory. Its basic idea can be seen as an attempt to find a hyper-surface among the space of possible inputs of feature vectors. This hyper-surface (decision surface) separates the positive training examples from the negative ones by the maximum margin with respect to the two classes. The property of SVM that it can be independent of the dimensionality of the feature space allows it can handle a large feature space. Thus, the same binary valued features vector used by naïve bayes classifier, that is, a bag of 4,547 Chinese characters, are fed into SVMs. We resorted to the LIBSVM toolkit to construct SVMs for our experiments, which allows for automatically scaling and parameter tuning on cross-validation data[2]. In our experiments, we trained SVMs using radial basic function (RBF) kernels.

2.3 The Combination Methods

For task classification, the idea of classifiers combination (ensemble) can be seen as applying k different classifier $TC_1,\ldots,TC_k$ to assign a query $\overline{ch}$ with a task t_i respectively, and then combining their results appropriately. Therefore, there are two main key issues to explore: (1) a choice of k individual classifiers, and (2) a choice of a combination function [11]. There are several methods both in the selection of the individual classifiers and in the way they are combined.

One way to create multiple classifiers resorts to the adaption of the training data, i.e. make use of the different sub-parts of the training data to train different classifiers. Bagging and Boosting belong to this kind. However, due to the diffculty of collecting the training data annotated in our domain, it may be insufficient to train the individual classifiers through adaptive resampling used in Bagging or Boosting. Therefore, in this paper we prefer to construct individual classifiers by training different learning methods on the same data. The different learning methods have been decribed in Section 2.2.

As for the classifiers combination methods, a straightforward way is voting: simple voting (majority voting), where each component classifier's vote is equal; or weighted voting, where each component classifier's vote is weighted by its perfromance. Obviously, simple voting will lead to majority effect. However, it also suffers from majority effect when the majority of component classifiers make wrong prediction. Weighted voting somewhat relieves this limitation through giving the accurate classifier bigger weight. This naïve strategy (both simple voting and weighted voting) fails under the circumstance that none of the results suggested by the component classifiers are correct. There is also another combination way: two-stage classification, in which a second-level classifier is trained to predict the correct output class when given as input the outputs of the base classifiers and other information [7]. This combination method can probably recall the correct result even under the situation that none of the component classifiers make a correct predication.

---

[2] LIBSVM is available from http://www.csie.ntu.edu.tw/~cjlin/libsvm

2.3.1 Voting

In (weighted) voting, the score is determined as follows for a given query:

$$Score(t,t_1^n)=\sum_{i=1}^{n}[\theta(t,t_i)\delta(t,t_i)+\overline{\theta}(t,t_i)\overline{\delta}(t,t_i)] \quad (5)$$

where t_i represents the task assigned by the i^{th} component classifier, parameters $\theta(t,t_i)$ and $\overline{\theta}(t,t_i)$ represent the voting weight of the i^{th} component classifier when its choice is t and isn't t respectively, and the $\delta(t,t_i)$ is the Kronecker function and $\overline{\delta}(t,t_i)=1-\delta(t,t_i)$.

Intuitively, $\theta(t,t_i)$ measures the confidence if the i^{th} component classifier assign a given query with t and $\overline{\theta}(t,t_i)$ forces the i^{th} classifier to give the vote a weight if it doesn't assign a given query with t. Then, the task t with the highest score is returned as the voting result. Especially, the voting strategy falls back to majority voting if $\theta(t,t_i)$ and $\overline{\theta}(t,t_i)$ are equal to 1 and 0 respectively. In our experiments, we investigated the majority voting and Precision-Recall weighted voting. In Precision-Recall weighted voting, $\theta(t,t_i)$ is weighted using the precision of the base classifier on the task t and $\overline{\theta}(t,t_i)$ using 1-recall [7]. When any ties occur, a random selection strategy is used to resolve them.

2.3.2 Two-Stage Classification

In this paper, we consider decision trees and maximum entropy model as the second-level classifiers.

2.3.2.1 Decision Tree Combination

Decision tree learning is a widely used and practical method for inductive inference, which is non-numeric instead of quantitative. A decision tree is a way to represent rules underlying training data, with hierarchical sequential structure that recursively partition the data [12]. In a decision tree, each node specifies a test of some attribute of the instance, and each branch descending from that node corresponds to one of the possible values for this attribute, and each leaf stands for a category [13]. During the tree construction procedure, greedy search strategy is employed, i.e. the attribute of the best classification performance on the training examples is selected at each step. An instance is classified on the decision tree by starting from the root, testing the corresponding attribute, then moving down the tree until a leaf node is reached or all the descending branches do not match the test instance, and finally returning the most frequent class at the last node [7]. There are some practical issues when decision tree is applied, for example, avoiding overfitting the training data. In our experiments, we used C4.5 as a second-level classifier[3]. C4.5 is one of the most popular standard packages for decision tree learning, which address most of these practical issues. More

[3] C4.5 is downloadable from http://www.cse.unsw.edu.cn/~quinlan

details about C4.5 can be seen in [14]. In decision trees, each instance is represented by attribute-value pairs. In our experiments, the outputs from the four base classifiers can conveniently be encoded into a feature vector.

2.3.2.2 Maximum Entropy Combination

Maximum entropy modeling is a framework for integrating information from many heterogeneous sources for classification [15]. The basic idea behind maximum entropy is to prefer the most uniform models that satisfy any given constraints. Each feature corresponds to a constraint on the model. This model represents examples (Cases) as sets of binary indicator features. The distribution is always of the exponential form:

$$p(t \mid Case) = \frac{1}{Z} \exp(\sum_i \lambda_i f_i(Case, t)) \tag{6}$$

where each $f_i(Case, t)$ is a feature, λ_i is a weight for the i^{th} feature and Z is simply a normalizing factor. The feature weights can be estimated using the algorithm of improved iterative scaling (IIS). For our experiments, we make use of the Maccent system implemented by Dehaspe as a second-level classifier[4]. The maximum entropy combiner also takes the same information as the decision tree as input. The Maccent system can automatically convert multi-valued features to binary features.

3 Experiments

3.1 Data Collection and Experimental Setting

With the context of city public transportation information inquiry domain, we define seven types of task: **ShowRoute**, **ShowFare**, **ShowTime**, **IsServeSite**, **IsServeTime**, **GroudService** and **ListServeSite**. When we define the task classes, we try to let the task classes also reflect the speech acts in the sentence since the speech acts will contribute much to understand the user's intention. The most frequent categories of speech acts are the four categories below: assertion, question, directive and response. Moreover, the question category is further subdivided into yes/no versus wh-questions [5]. Obviously, it is very important to distinguish these two sub-categories in our domain. Furthermore, according to the analysis on the data set, we observed that the users rarely used the yes/no questions except that when they were composing the queries about time and serve sites information. Therefore, only the two classes, i.e **IsServeTime** and **IsServeSite**, are added.

We collected 646 sentences when we carried out the experiments in [9]. Up to now, we have extended this data set to 1481 sentences. We split this data set into two parts: the training set consists of 1180 sentences, and the test set consists of 301 sentences. In addition, the test data has two versions as [9]: one is the transcription text and the other is the recognized text. The distribution of seven tasks in our domain is highly skewed to the ShowRoute class, which covers about 74.4% of total queries.

[4] Maccent is available from http://www.cs.kuleuven.ac.be/~ldh

To test its individual performance, each base classifier is trained on the total training set and evaluated on the whole test set. For the measure of the classifiers combination performance, we make use of the 10-fold training technique to avoid the overfitting problem [7]. The whole training data are firstly split into ten equal disjunctive subsets. Each subset can be made prediction by the component classifiers, which have been trained on the other nine parts. Then, all the prediction outputs are unified as the training set of the combiner. Finally, we use the test set to evaluate the various combiners trained on this set and compare their performance.

3.2 Base Classifier Performance and Agreement

Table 1 compares the n-best performances of four component classifiers on both the transcript testing set and recognition text testing set. From the 1-best results, i.e. the accuracy of the base classifiers, we observe that SVM outperforms excessively the other classifiers on both test sets and behaves very robustly on noise data (recognition text testing set). The unigram performs excellently on the transcript testing set, however, the performance of which decreases a little drastically on the noise data. The performances of the two other classifiers are close to each other on two test sets. It is somewhat surprising that bigram is outperformed by unigram on both test sets. We ascribe it to the fact that the sparse data problem in bigram is more serious than in unigram and the absolute discounting smoothing is less effective for bigram model.

In addition, we observe that most of the base classifiers always give the correct task a considerably high rank if they assign a query wrongly. Except SVM, the other base classifiers offer not only the best choice but also the n-best outputs. It enables the second-level classifiers to make use of the information of n-best candidates offered by the base classifiers to recall the misrecognized results. Since the performance of 2-best accuracy of the single classifier is good enough, we only consider at most 2-best results from the base classifiers. Addition of more low-ranked results of base classifiers will bring side-effects because these redundant information may interfere the decision of the second-level classifiers.

Table 1. Comparison of n-best performance of single classifiers on the transcript and recognition text set. Since SVM only gives one result, its 2-best accuracy is equal to 1-best accuracy

	Transcript (%Acc)		Recognition Text(%Acc)	
	1-best	2-best	1-best	2-best
NB	93.7	97.7	91.0	94.7
Bigram	95.7	99.7	92.0	97.0
Unigram	93.0	100.0	90.7	97.7
SVM	96.0	96.0	95.7	95.7
Standard deviation	1.5	1.9	2.3	1.3

Table 2. The distribution of agreement patterns among various classifiers for the two data sets. Both the percentage of the patterns occur in the two test sets and the corresponding cumulative percentage (%Cum) are listed

Pattern	Transcript		Recognition Text	
	%	%Cum	%	%Cum
All classifiers agree and are correct	88.4	88.4	83.1	83.1
A majority is correct	6.3	94.7	9.3	92.4
Correct task is present but is tied	2.0	96.7	4.0	96.4
A minority is correct	2.0	98.7	2.3	98.7
The classifiers vary but are all wrong	0.0	98.7	0.3	99.0
All classifiers agree but are wrong	1.3	100.0	1.0	100.0

For the classifier combination system, the performance improvement depends on how different outputs of the base classifiers are. Then, we also investigate the (dis)agreement of outputs from the base classifiers, which determine levels of combination quality, as shown in Table 2. Here, we use the same patterns of agreement as in [7] for the two test sets. From Table 2, we can conclude that the potential of majority voting is limited since the cumulative percentages of correct majority on the two test data are both lower than that of the best single classifier.

3.3 Combinational Results

Table 3 shows the results of our experiments with various combination methods. We list both the accuracy of the combiner (%Acc) and the error reduction in relation to the best base classifier (Δ_{err}). We can see that the two voting methods can hardly achieve satisfying improvements on the two test sets. Especially, the majority voting strategy is even outperformed by the best single classifier (SVM), in fact, which is determined by the cumulative percentage (%Cum) in the table 2. This is due to the fact that SVM has the overwhelming performance over the other component classifiers. Even so, the two-stage classification methods still demonstrate their combining power. The significant improvements were obtained through the addition of the 2-best outputs from the base classifiers for maximum entropy combiner. But it has little impact on decision tree combiner.

Table 3. Performance of the combination systems on two test sets

	Transcript		Recognition Text	
	%Acc	Δ_{err}	%Acc	$\%\Delta_{err}$
Best Single Classifier (SVM)	96.0	-	95.7	-
Majority Voting	96.0	0.0	94.4	-30.2
Weighted Voting (Precision-Recall)	96.0	0.0	96.3	14.0
Decision trees (1-best features)	96.7	17.5	96.3	14.0
Decision trees (2-best features)	96.7	17.5	96.3	14.0
Maximum entropy (1-best features)	96.3	7.5	96.3	14.0
Maximum entropy (2-best features)	97.7	42.5	96.7	25.0

4 Conclusion, Discussion, and Future Work

For the task classification in the context of city public transportation information inquiry domain, although the performance of the classifier is improved to a certain extent through including the deep-level features, it also has negative effect that degrading the robustness of the classifier on the noisy data. At the same time, it can hardly make use of the deep-level features since task classification is a shallow analysis process. In addition, the cost of extracting richer features is fairly expensive. Thus, in this paper we have tried to combine different classifiers using the shallow features to improve the overall performance.

Our experimental results have shown the combination of several different classifiers resulting from the same training data enables us to raise the overall performance of task classification with the context of city transportation information inquiry domain. The comparison of qualities of single classifiers has shown that SVM is an excellent candidate to be a base classifier for task classification in our domain. It may imply that the task classification problem in our domain is linearly separable. At the same time, for each query, the corresponding feature vector exploited in our methods always has high dimension, and contains only few entries which are not zero. Overfitting protection used in SVM allow it has the potential to handle the large feature spaces. Moreover, an inductive bias in SVM makes it well suited for problems with sparse instances. These factors all make attributes to the excellent performance of the SVMs in our domain.

We also have compared the performance of the classifier combination methods: two-stage classification and voting. Since SVM has the overwhelming performance over the other component classifiers, voting methods are incapable to provide any improvement over the accuracy of the best base classifier. In contrast, the two-stage classification methods (both decision tree combination and maximum entropy combination) can effectively raise the whole performance. Furthermore, the addition of n-best results from the base classifier also improves the performance of two-stage classification, especially for the maximum entropy combiner.

Future works include trying to explain why maximum entropy combiner yields higher performance than decision tree combiner when n-best results from the base classifier are added, and trying to find out a better second-level learner based on the analysis results.

Acknowledgements

The authors would like to thank Feng Jiang, Bo Du and the other members of our natural language processing group in Shanghai Jiao Tong University for help with the implementation of naïve bayes classifier and n-gram models. We would also like to thank to the creators of several well-known classifiers used here for making their systems available: LIBSVM, C4.5 and Maccent system.

References

1. M. Cettolo, A. Corazza, and R. De Mori.: A Mixed Approach to Speech Understanding. In *Proc. of ICSLP*, Philadelphia, USA, 1996
2. Ye-Yi Wang, Alex Acero, Ciprian Chelba, Brendan Frey and Leon Wong: Combination of statistical and rule-based approaches for spoken language understanding. In *Proc. of ICSLP*, pp.609–612, Denver Colorado, USA, Sep., 2002.
3. C. Chelba, M. Mahajan and A. Acero: Speech Utterance Classification, in *Proc. of ICASSP*, Hong Kong, Apr,, 2003.
4. Chin-Hui Lee, Bob Carpenter, Wu Chou, Jennifer Chu-Carroll, Wolfgang Reichl, Antoine Saad, Qiru Zhou: On natural language call routing, *Speech Communication*, v.31 n.4, pp.309-320, Aug., 2000
5. Marineau, J., Wiemer-Hastings, P., Harter, D., Olde, B., Chipman, P., Karnavat, A., Pomeroy, V., Graesser, A.C., & TRG: Classification of speech acts in tutorial dialog. In *Proceedings of the Workshop on Tutorial Dialogue at the Intelligent Tutoring Systems 2000 Conference*, pp57-63., Canada. 2000.
6. M. Mast, R. Kompe, S. Harbeck, A. Kiessling, H. Niemann, E. Nöth, E. G. Schukat-Talamazzini, and V. Warnke: Dialog act classification with the help of prosody. In *Proc. of ICSLP*, Philadelphia, 1996.
7. H. van Halteren, J. Zavrel, and W. Daelemans: Improving accuracy in word class tagging through the combination of machine learning systems. *Computational Linguistics*, 27(2): pp.199-230, 2001.
8. Larkey, Leah S. and Croft, W. Bruce: Combining Classifiers in Text Categorization. In *Proceedings of the 19th Annual International Conference on Research and Development in Information Retrieval (SIGIR 96)*, pp.289-297. Zurich, Switzerland.
9. Weilin Wu, Ruzhan Lu and Zheng Liu: Comparative experiments on task classification for spoken language understanding using naive Bayes classifier. In *Proceedings of International Conference on Natural Language Processing and Knowledge Engineering (IEEE) 2003*. pp. 492-497, Beijing, Oct.,2003
10. Ney, Hermann, Ute Essen, and Reinhard Kneser: On structuring probabilistic dependencies in stochastic language modeling. *Computer Speech and Language*, v.8, pp.1-38, 1994
11. Fabrizio Sebastiani: Machine learning in automated text categorization, *ACM Computing Surveys*. 34(1): pp.1-47, 2002.
12. L. Màrquez: *Machine Learning and Natural Language Processing*, Technical Report LSI-00-45-R, Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, Espana, 2000.
13. T.M. Mitchell: *Machine Learning*, McGraw-Hill, New York, US, pp.52-53, 1996.
14. Quinlan, J.R.: *C4.5: Programs for Machine Learning*, Morgan Kaufmann, San Mateo, CA. 1996.
15. Christopher D. Manning and Hinrich Schütze: *Foundation of Statistical Natural Language Processing*, The MIT Press, Cambridge, Massachusetts, USA, pp.589-597, 2000.

A Finite State Network for Phonetic Text Processing

Edward John Garrett

Eastern Michigan University
egarrett@emich.edu

Abstract. In the past, phonetic transcriptions were made using a wide variety of fonts and formats, which hampered the development of phonetic text processing tools. Today, however, the increasing number of language documentation projects making their data freely available over the Web, combined with the adoption of the Unicode Standard by linguists as "best practice" character encoding, present linguistic software developers with an unprecedented opportunity to develop powerful tools for the analysis of phonetic text. This paper describes the generation of a finite state transducer that converts text represented in the International Phonetic Alphabet into phonetic feature sets.

1 Introduction

In response to the crisis of language endangerment, the number and variety of language documentation projects has risen dramatically in recent years. Once content to produce printed grammars, dictionaries and texts, however, such projects are now taking advantage of developments in Internet technologies and multilingual computing to make diverse multimodal linguistic data freely available over the Web, for language researchers and community members alike.

This expansion of linguistic data on the Web presents both challenges and opportunities. The challenges include the problem of insuring that linguistic data is archived in accessible and flexible formats. Under the rubric of "best practices" in language documentation, much progress has already been made in this area [1]. Simultaneously, the adoption by linguists of best practices in language documentation is also creating opportunities for new computational methods and tools for linguistic processing. This paper explores one such opportunity: phonetic text processing.

1.1 Phonetic Text Processing and the Unicode Standard

Until recently, phonetic data represented by the International Phonetic Alphabet (IPA) had to be stored in a variety of fonts and encodings, determined by platform, software, or convenience. With the emergence of the Unicode Standard [2], however, this situation has changed: now, linguists are told to encode phonetic text as Unicode [3]. Since Unicode has the virtue of assigning characters from nearly all written scripts, including IPA, with uniform and invariant code points, IPA text encoded as Unicode can

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 463–473, 2005.

now be unambiguously identified as phonetic data, and subjected to phonetic text processing.[1]

Although it is already possible to do simple string and pattern matching on IPA symbols, this ability is not highly prized by linguists, because the value of the IPA for phonetic and phonological analysis lies not in the symbols themselves, but rather in the articulatory and acoustic properties which bind different symbols and combinations together. Indeed, one of the principles of the International Phonetic Association is that "the symbols of the IPA are shorthand ways of indicating certain intersections of categories. Thus [p] is a shorthand way of designating the intersection of the categories voiceless, bilabial, and plosive; [m] is the intersection of the categories voiced, bilabial, and nasal; and so on" [4].

To serve linguists' purposes, then, at minimum we need methods to (a) turn IPA strings into phonetic feature bundles; and (b) turn phonetic feature bundles into IPA strings. Other desirable features such as a phonetic query language could be built atop this basic functionality.

In this paper, I show that (a) can be implemented using a finite state transducer (FST) that recognizes legal IPA segments and converts them into phonetic feature sets. This process, as well as the complications involved in doing (b), are discussed in detail in Section 3. First, however, important background on the IPA and Unicode is presented in Section 2. Section 4 concludes.

2 The IPA and Unicode

The IPA consists of a set of symbols used for phonetic transcription, each of which is assigned a unique code point in Unicode. The relation between IPA symbols and phonetic segments or phonemes is many-to-one: generally speaking, a phonetic segment consists of a base symbol followed by zero or more diacritics. The architecture of Unicode, combined with the implicit logic of the IPA, impose certain constraints on the ways in which symbols can be combined to make a phonetic segment or phoneme. This section summarizes these constraints by focusing on the IPA, Unicode, and the mapping between the IPA and Unicode.

2.1 Architecture of the IPA

The IPA is divided into sub-charts of pulmonic consonants, non-pulmonic consonants, other symbols, diacritics, vowels, and suprasegmentals, as shown by Figure 1, which reproduces the IPA in its entirety. Although the general framework described here can be extended to the entire IPA, for reasons of space, this paper focuses exclusively on pulmonic consonants and diacritics that can modify them.

[1] Strictly speaking, this statement is false. Since some IPA symbols are used by Roman-script based writing systems such as English, many IPA words or phrases are ambiguous. For example, is /tot/ the English word "tot" or the phonetic transcription for "tote"? Moderately complex IPA strings exhibiting phonetic diacritics or IPA-specific symbols, however, can likely be identified as phonetic data using statistical analysis.

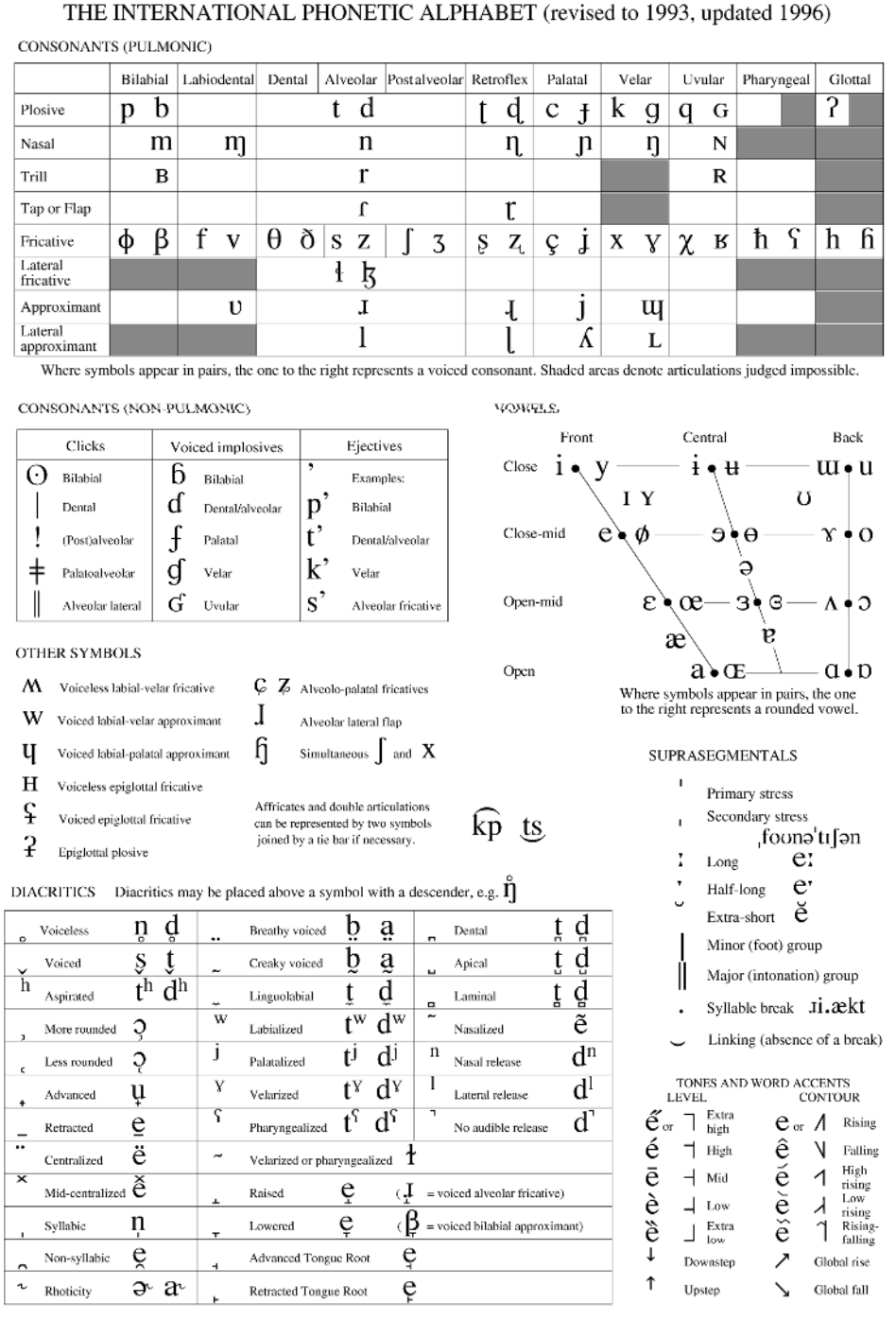


Fig. 1. Symbols of the International Phonetic Alphabet (IPA)

Consideration of the pulmonic consonant chart in Figure 1 illustrates three important properties of "base letters" or "base symbols" in IPA. First, as noted above, symbols such as [b] denote *the intersection of multiple phonetic categories*. Some articulatorily possible category intersections are left blank, because they are deemed to occur too infrequently in the world's languages to require their own symbols. As a consequence, category intersections such as voiceless nasals can only be represented with the help of diacritics.

Second, *the semantics of some base symbols is underspecified*. In particular, with the exception of [s] and [z], consonants directly under the alveolar heading are not specifically alveolar, but rather can be used to represent dental, alveolar, or postalveolar points of articulation. Roughly following existing practice, we can use "coronal" as a general cover term for these three places of articulation.

Third, *not all categorial decompositions are made explicit* in the chart. E.g., many linguists find it useful to speak of nasal plosives, to refer to classes of sounds such as rhotics and laterals, and to treat lateral fricatives and approximants as sub-types of fricatives and approximants. None of these assumptions are codified into the IPA.

Consideration of IPA diacritics, also shown in Figure 1, brings several additional properties of IPA to the mix. First, *base symbols are routinely suffixed by multiple diacritics*: witness [l̪ʲ], Irish's palatalized alveolo-palatal lateral approximant, or [t̪ʰ], Hindi's voiceless aspirated dental plosive.

Second, *certain sets of diacritics are mutually exclusive*. We do not find simultaneously lowered and raised consonants such as [β̞̝], creaky voiced and breathy voiced consonants such as [b̤̰], or even simultaneously palatalized and velarized consonants like [tʲˠ]. It is possible to imagine the IPA assigning principled interpretations to such sequences; however, until such recognized practices are observed, it is simpler to exclude such examples as uninterpretable.

Third, the *order of diacritics is often irrelevant*. There is no recognized practice by which [ɹ̪̊] is to be distinguished from [ɹ̪̊]. Both denote voiced postalveolar (rhotic) fricatives.

Fourth, *some diacritics impose selectional constraints* by only attaching to a proper subset of base symbol-diacritic sequences. E.g., the dental diacritic may attach to coronal consonants, turning the voiceless coronal plosive [t] into the voiceless dental plosive [t̪]; or to bilabials, turning the voiceless bilabial plosive [p] into the voiceless labiodental plosive [p̪] [5]. It does not, however, attach to velars: [k̪] is unattested.

Fifth, *diacritics may have different effects depending on what they attach to*. In addition to the just-cited case of the dental diacritic, another example of this is the retraction diacritic. While [k̟] denotes a retracted voiceless velar plosive, [t̠] conventionally denotes a voiceless postalveolar plosive (since the IPA chart lacks such an entry).

2.2 The Architecture of Unicode and How it Encodes the IPA

Unicode, now in its 4th major version, is an extraordinarily complex industry standard. Fortunately, the IPA barely touches its surface, so it is possible to concisely explain the concepts critical for this paper.

Each IPA symbol has a Unicode code point. Code points themselves fall into one of many code charts, or groupings of related characters. Although there is an IPA Extensions code chart, many IPA symbols are found in other code charts, including Basic Latin, Spacing Modifier Letters, and Combining Diacritical Marks, among others. This means that many IPA characters cannot be identified as uniquely IPA, since they may also be used by other writing systems.

One of the most important properties of a Unicode character is its general category. Most characters are sub-types of letters, marks, numbers, punctuation, symbols, or separators. In general, letters are characters which convey a basic unit of meaning in a writing system and which do not typographically depend on other Unicode characters. All of the base symbols in Figure 1 are letters. An important sub-class of letters are the modifier letters. This class includes several IPA diacritics, including aspiration, secondary articulations, release modifiers, and even the rhoticity diacritic.

The remaining IPA diacritics are classed as marks, and in particular, non-spacing marks. Non-spacing marks are diacritics that do not take up any horizontal space: instead, they typographically cohabit with the preceding character (for example, by overlaying it, or occurring above or below it). The syllabicity and voicing diacritics are examples of non-spacing marks.

Each Unicode character is assigned a numerical combining class. Letters are assigned to class 0, but non-spacing marks are assigned a number from 1-255. The basic rule is that marks that occur in the same general vacinity to a base letter tend to have the same combining class. Although there are over 20 commonly used combining classes, only a few of these are used by IPA diacritics:

1: The tilde overlay (U+0334).

220: Voicing, devoicing, breathy voice, creaky voice, advanced, retracted, raised, lowered, and all other diacritics that occur under a base character without touching it.

230: Tone markers, nasalization, centralization, mid-centralization, and all other diacritics that occur above a base character without touching it.

232: The no audible release diacritic (U+031A).

233: The linking bar, which is placed under two adjacent segments to mark the absence of a (word) break (U+203F).

234: The tie bar, which is placed above two adjacent segments to represent affricates and double articulations (U+0361).

Although it is possible to suffix a base letter with marks from any class in any order, Unicode employs a process of normalization which defines as equivalent certain otherwise distinct mark orders and character combinations [6]. Although normalization comes in several varieties, one constant is that after normalization is applied, non-spacing marks that follow a base letter cannot occur in decreasing numerical order. In other words, after normalization, marks of class 220 must occur closer to the base letter than marks of class 230. (Within a given combining class, order is free.)

Normalization also neutralizes the distinction between precomposed and decomposed representations, for example of [e̬] as either U+1E1B (precomposed creaky close-mid front vowel) or U+0065 U+0330 (decomposed close-mid front vowel plus creaky voice).

There is thus considerable advantage to normalizing IPA text before processing it. Unfortunately, however, some semantic equivalencies in IPA are not equivalent by

normalization and so must still be handled separately. For example, there are two ways of encoding the rhotacized vowels [ɚ] and [ɝ]: as ready-made rhotacized vowels (U+025A and U+025D, respectively), or as character pairs (U+0259 U+02DE and U+025C U+02DE, respectively). Since U+02DE MODIFIER LETTER RHOTIC HOOK is a spacing modifier with a combining class of 0 rather than a non-spacing mark, and decomposed sequences must consist of a class 0 character followed by one or more class>0 characters, U+025A and U+025D cannot be assigned decompositions.

3 A Finite State Network for Phonetic Text Processing

Finite state machines (FSMs) have become increasingly popular in computational linguistics, and are now used for diverse tasks such as spell-checking, morphological analysis, and phonological rule application, among others [7]. A finite state transducer (FST) is an FSM consisting of a set of states, including an initial and (possibly multiple) final states, along with input:output pairs <a:b> associated with pairs of states. The transition <a:b> between states q_1 and q_2 effectively defines a mapping at q_1 from an "input" of a to an "output" of b, resulting in a transition to state q_2.

Since FSTs are a demonstrably efficient way of translating from one logical language to another, and phonetic text analysis is a potentially high-volume task where efficiency is paramount, the application of FSTs to phonetic text analysis is attractive. Intuitively, the task is clear: we need a transducer whose input language is the set of valid IPA segments, and whose output language is the language of phonetic features. If we feed [t̪ʰ] into the transducer, we should get "aspirated voiceless dental plosive". If we feed in [m̥], we should get "voiceless bilabial nasal".

In this FST, unlike typical morphological analyzers, the order of elements in the output language is irrelevant: there is no difference between "voiceless bilabial nasal" and "nasal voiceless bilabial". Although this fact complicates the conversion from phonetic feature sets to IPA segments (see Section 3.4), the reverse conversion, from IPA segments to phonetic feature sets, remains straightforward.

3.1 Replacive Diacritics and Selectional Constraints

The central challenge in creating such an FST is that some IPA diacritics have a replacive rather than additive effect. Thus, while [m] by itself is voiced, [m̥] is voiceless, not both voiced and voiceless. Similarly, while [ɹ] is non-syllabic, [ɹ̩] is syllabic, not both non-syllabic and syllabic. The effect of this is that phonetic features that have the potential to be replaced by diacritics cannot be spelled out until it is known for sure that a replacive diacritic can no longer be attached.

This means that an FST that analyzes IPA segments from left to right cannot be sequential. That is, regardless of how it is constructed, there will always be states where two or more paths will have to be checked before knowing how a given input string should be processed. For example, to avoid [m̥] having both voiced and voiceless fea-

tures, suppose we have only one path from input [m], which does not output the voiced feature. This means that [m] itself would fail to acquire the voiced feature; to get it, an epsilon (null input) transition would have to be added. Adding an epsilon transition would present the machine with two choices when processing [m̥]: after [m], either to follow the episilon input to a final state (and fail, not having processed all the input), or to follow the voiceless input to a final state (and succeed). Alternatively, there could be two initial transitions: <m:voiced bilabial nasal> and <m:bilabial nasal>, only the first of which would lead to a final state. In both cases, the result is not only not sequential, but unsequentiable. That is, there is no way to modify the transducer to make it sequential [8].

Unsequentiability can be avoided, and efficiency increased, by forcing the transducer to apply to IPA segments from right to left. Encountering the voiceless diacritic first, as we transduce [m̥] from right to left, we know immediately that the overall segment must be voiceless. We can then deliberately fail to output [m]'s voicing feature. In other cases, the process may be more complicated, but the result is still the same. Since [t̠] denotes a voiceless *postalveolar* plosive, and [k̠] a *retracted* voiceless velar plosive, we cannot spell out the feature contributed by the diacritic [˕] until we know what precedes it. However, nothing stops us from outputting this feature on input of [t] or [k].

Both replacive and additive diacritics tend to impose selectional constraints (see above). Ideally, our FST should incorporate as many of these constraints as possible. By doing so, it can be used to diagnose illegal IPA, which could be useful in correcting transcription errors. Also, excluding illegal IPA saves the FST from having to make arbitrary decisions about what phonetic features to assign to such strings.

There is, of course, no place in the FST machinery itself to store selectional constraints. However, what we can do is to use these constraints to guide the construction of our FST. The mechanism I propose for implementing selectional constraints is simple. Each diacritic is associated with: (a) additive features: a list of features which the diacritic adds regardless of what it is attaching to; and (b) selectional conversions: an ordered list of pairs, specifying the phonetic features the diacritic selects for, and how those features are affected. Both properties are optional.

Let's consider some examples. The raised diacritic [˕] is purely additive: its additive feature list contains the feature "raised", and it has no selectional conversions. The voiceless diacritic [̥], on the other hand, has an additive feature list of "voiceless", and the single selectional conversion list <<voiced,0>>, which means that it selects for the voiced feature and then deletes that feature. The dental diacritic [̪] has no additive feature list, and the selectional conversion list <<coronal, dental>, <bilabial, labiodental>>, meaning that it either turns a coronal into a dental (as in [t̪]) or a bilabial into a labiodental (as in [p̪]), but nothing else (so, not [k̪]).

The selectional conversion list must be tracked as diacritics are attached, since it cannot be satisfied immediately. For example, when interpreting [t̪ⁿ], the FST first encounters [ⁿ], which outputs the feature “nasal release” and the selectional conversion

list <<plosive, plosive>>, meaning that it expects a plosive but retains the plosive feature. Next the FST encounters [̪], discussed above. The result is that there are two selectional conversion lists to satisfy: <<coronal, dental>, <bilabial, labiodental>> and <<plosive, plosive>>. In this case, selectional restrictions for both diacritics are met and so the segment is deemed legal. Had the segment been [l̪ⁿ] instead, the segment would be deemed illegal, since although the dental diacritic would have been happy, the nasal release diacritic would not, since [l] is not a plosive.

3.2 Ordering the Input

Having established that our FST should apply to IPA segments from right to left, and that selectional constraints can be implemented if the FST is designed in accordance with the above principles, our next task is to determine the order in which to process input characters.

If, as suggested above, an IPA segment consists of a Unicode letter followed by zero or more marks (of combining classes 1, 220, 230, 232, 233, or 234), followed by zero or more modifier letters, then the fact of normalization already imposes an initial constraint on input order: L < 1[=U+0334] < 220 < 230 < 232[=U+031A] < 233[=U+203F] < 234[=U+0361] < Lm. (Classes with Unicode values in brackets are classes with only one member.) Remember, even if a diacritic of class 230 (occurring above the base symbol) occurs before a diacritic of class 220 (occurring below the base symbol) in the original data, due to equivalence under normalization, such data cannot be treated differently from properly normalized data which has the diacritical ordering just given.

Additional ordering constraints can be imposed on the input by two of the points of Section 2.1: (1) that certain sets of diacritics are mutually exclusive; and (2) that the order of diacritics is often irrelevant. Combining these points, we can simplify our FST by dividing each diacritical class into several mutually exclusive subclasses.

In connection with IPA, the most populated combining class is class 220 (for diacritics occurring below the base symbol).Therefore, we use class 220 for illustration. By point (1), class 220 diacritics that attach to consonants can be divided into the following mutually exclusive sets: {advanced, retracted, linguolabial, dental, apical, laminal} < {voiceless, voiced, breathy, creaky} < {raised, lowered} < {syllabic, non-syllabic}.[2]

By point 2, an arbitrary order can be imposed on class 220 diacritics: intepret the greater than sign as ordering the diacritics from right to left. They are then input into

[2] To say that these sets contain mutually exclusive diacritics is not quite to say that they denote mutually exclusive phonetic features. Indeed, we may, for example, consider breathy consonants to be voiced. Rather, the point is that, to my knowledge, the *diacritical symbols themselves* are mutually exclusive. Should attested usages prove the contrary for some cases, then it would be easy enough to redefine the classes. The idea proposed in the text would only be invalidated if *most or all* diacritics were shown not to have *any such constraints*. Of the groupings in the text, probably the most controversial would be the one containing advanced, retracted, etc.

our FST, starting with a choice between `{syllabic, non-syllabic}` diacritics. In the same fashion, orderings of mutually exclusive sets of diacritics can be imposed on class 230 and the modifier letters.

Thus, for the input IPA segment to match the order expected by our FST, two processes must apply to it. First, it should be Unicode-normalized (using either NF-C, which favors precomposed forms, or NF-D, which favors decomposed forms). Second, it should be IPA-normalized. *IPA-normalization* is the reordering process which takes a Unicode-normalized IPA string and then reorders the diacritics within each combining class category according to imposed orders such as the one illustrated above for class 220.

3.3 Determining the Transitions

Having determined the order by which input characters may be fed into our FST, our final task is to compute the network's transitions. The initial transition itself is the freest, taking as input every character from all classes and subclasses, and outputting those phonetic features which cannot be replaced by subsequent characters.

The initial transition moves the FST into a state jointly determined by (a) the combining class and subclass of the input character, and (b) the selectional conversion list still to be satisfied. Two purely additive characters belonging to the same combining class and subclass, such as the 'raised' and 'lowered' diacritics, would therefore transition to exactly the same state. However, two characters with the same combining class and subclass, but with different selectional properties, like the voiced and voiceless diacritics, would transition to two different states.

Whenever transitioning on an input character from class/subclass x, the result state will be one that offers transitions with input from any class/subclass ordered before x. Thus, a transition on input [̩] 'raised' must be to a state which will only accept as input characters from class 220 aside from the syllabic, non-syllabic and raised and lowered diacritics, and characters from class 1 or the base letter class.

The number of states in the network thus rises largely due to the increasing complexity of the selectional conversion list as more diacritics are processed. This is because selectional conversion lists represent all those phonetic features of a segment that cannot be "spelled out" until we know what the base letter is.

At any point, input of a base letter causes our FST to transition to its (unique) final state. The output of this transition is a function of the selectional conversion list applied to the features of the base letter. If the phonetic features of the base letter are not compatible with the selectional conversion list (for example, if the base letter should be voiced but is actually voiceless), then no transition on that input is allowed. If no transitions on any base letters are allowed, then the FST fails, having reached a non-final state with no transitions.

Creating an FST implementing the above design principles thus offers a tool for (a) determining whether or not a string is a legal or illegal IPA segment; and (b) efficiently computing the phonetic feature set of a legal IPA segment.

3.4 Reversing Input and Output: Starting with Phonetic Features

This paper has provided a solution to one challenge – given an IPA segment as input, compute its phonetic feature set – but not another – given a phonetic feature set, compute the segment (or segments) which instantiate it.

This second challenge is more difficult than the first one, although the FST just described certainly helps. The first unpleasant reality is that whereas IPA segments unambiguously produce a single feature set, a given feature set may correspond to multiple IPA segments. For example, [d̥] and [t] are both voiceless coronal plosives, but they are two different strings. What we should get in response to a request for "voiceless coronal plosive" depends on the situation. We may want the canonical form, i.e. [t], or we may want all possible forms.

A second problem, already noted, is that while the characters in an IPA segment are strictly ordered, a phonetic feature set is unordered. There is no difference between "voiceless coronal plosive" and "plosive coronal voiceless". More importantly, because of this, our FST does not output phonetic features in any systematic way, but instead does so according to what is possible given the existence of selectional conversions.

A third problem relates to how phonetic feature sets are used in practice. Linguists do not often use phonetic feature sets to identify a single IPA segment. If referring to [t], for instance, linguists are more likely to simply use [t] than to say "voiceless coronal plosive" (or "voiceless alveolar plosive"). Phonetic feature sets are most useful when they identify *natural classes of sounds*. Linguists frequently do talk about classes such as the "voiceless plosives", since that identifies many segments, not just one. Thus, this conversion problem is confounded by the fact that the most common use case for phonetic feature sets is as a way to identify *sets* of IPA segments.

Work is underway to meet these challenges, but that is the topic of another paper.

4 Future Directions

For reasons of space, several types of IPA symbols were not discussed in this paper. A broader treatment would have also included discussion of vowels, vowel diacritics, suprasegmentals, non-pulmonic consonants, and the other symbols category.

One important symbol was deliberately excluded from the discussion because of the complications it introduces. The tie bar (U+0361) is used to join two segments together into a single phoneme, for example in affricates like [t͡s] and double articulations like [k͡p]. Understanding the full implications of combining two segments together and the resulting phonetic feature sets will require more research.

A related challenge concerns the imprecision of much existing data. For example, one often finds affricates written as [ts] without the tie bar. Without knowing the phonemic inventory of the language of the data, it is impossible to know whether an affricate has been intended, or whether this is a [t] followed by a [s]. It would appear that some questions cannot be resolved with certainty in the absence of metadata and language profiles.

Taking a broader perspective, the output of the FST described in this paper ought to be linked up to a general phonological ontology. Although it is possible to interpret IPA segments in terms of phonetic feature sets as I have done here, the fact remains that linguists have devised rich geometries for representing phonetic feature structures. Thus, a way to map back and forth between simple feature sets and more complex representations is much needed.

Still, in spite of these limitations, the proposal in this paper opens doors and invites extensions. The ultimate goal of this work is the construction of a phonetic search engine, which could scan the Web for data matching specific phonetic queries, a power tool for linguists. This goal remains distant, but other goals are perhaps closer. The application of machine learning techniques to discover associational patterns in phonetic text is one promising research direction made possible by this work.

For up to date information about this project, or to get involved in our open-source software development efforts, please visit our web site [9].

References

1. The LINGUIST List. E-MELD (Electronic Metastructure for Endangered Languages Data). http://emeld.org
2. The Unicode Consortium. The Unicode Standard, Version 4.0. Addison-Wesley, Boston (1991-2003)
3. Bird, Steven & Gary Simons. Seven Dimensions of Portability for Language Documentation. Language 79.3:557-582 (2003)
4. The International Phonetic Association. Handbook of the International Phonetic Association: A Guide to the Use of the International Phonetic Alphabet. Cambridge University Press, Cambridge (1999).
5. MacMahon, Michael K. C. Phonetic Notation. Pp. 821-846 in Peter T. Daniels & William Bright (eds.), The World's Writing Systems. Oxford University Press, Oxford (1996).
6. Davis, Mark & Martin Dürst. Unicode Standard Annex #15: Unicode Normalization Forms. http://www.unicode.org/reports/tr15/tr15-23.html
7. Beesly, Kenneth R. & Lauri Karttunen. Finite State Morphology. CSLI Publications, Stanford (2003).
8. Karttunen, Lauri. Finite-State Technology. Pp. 339-357 in Ruslan Mitkov (ed.), The Oxford Handbook of Computational Linguistics. Oxford University Press: Oxford.
9. Garrett, Edward. Tools for Field Linguistics. http://fieldling.sourceforge.net (2002-present).

Language Documentation: The Nahuatl Grammar

Mike Maxwell[1] and Jonathan D. Amith[2]

[1] Linguistic Data Consortium
maxwell@ldc.upenn.edu
[2] Gettysburg College
jonathan.amith@yale.edu

Abstract. We describe an on-going documentation project for Nahuatl, an indigenous language of Mexico. While we follow standard recommendations for documenting text corpora and for the dictionary, the usual recommendations are not explicit concerning the grammar. Since Nahuatl is an agglutinating language, the morphological component of the grammar is highly complex. Accordingly, we consider it essential to not only provide static information about the language, such as a lexicon and parsed text, but dynamic documentation in the form of a working morphological grammar. When compiled into a finite state transducer, this grammar provides parses for arbitrary inflected forms, including many not in the corpus, as well as the generation of the partial or full inflectional paradigms. In keeping with the archival goals of language documentation, we argue that this grammar should be simultaneously human readable and computer processable, so that it will be re-implementable in future computational tools. The notion of literate computing provides the appropriate paradigm for these dual goals.

1 Language Description and Documentation

> "It is to be lamented… that we have suffered so many of the Indian tribes already to extinguish, without our having previously collected and deposited in the records of literature, the general rudiments at least of the languages they spoke. Were vocabularies formed of all the languages spoken in North and South America, preserving their appellations of the most common objects in nature, of those which must be present to every nation barbarous or civilized, with the inflections of their nouns and verbs, their principles of regimen and concord, and these deposited in all the public libraries, it would furnish opportunities to those skilled in the languages of the old world to compare them with these, now or at a future time, and hence to construct the best evidence of the derivation of this part of the human race." –Thomas Jefferson (1781-1782) *Notes on the State of Virginia*

There are over 6000 languages in the world today [1]. The diversity of these languages has the potential to provide us with a window into the mind, an understanding of what it means to be human, and a way of reconstructing pre-history that we can attain in no other way.

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 474 – 485, 2005.

Sadly, many of these languages are on the verge of extinction. A common estimate is that during this century, at least a half of these languages will disappear; the worst-case predictions are considerably more grim.[1]

All languages have a story to tell about the human language capacity, but there are some questions which can only be answered by certain languages. For example, there are only a handful of languages which have the word order Object-Verb-Subject, and only one of these (Hixkaryana, [2]) is well documented. Had these few languages disappeared from the face of the Earth before being documented, we might not have known that this word order was even possible in human languages, much less what the properties associated with this word order might be.

While language change is inexorable, the loss of languages is not. Languages can be preserved by continuing to be spoken, but they can also be preserved by being written down and described. The long-term survival of knowledge of ancient languages such as Latin, Classical Greek, and Sanskrit is due in no small part to the efforts of a handful of speakers of these languages who wrote down their grammars. Some other ancient languages remain more or less accessible to this day because they have been preserved in written form, which we have been able to decode, usually with the help of bilingual documents such as the famed Rosetta Stone. In contrast, the complete loss of all other languages of that era is due to their lack of documentation.

Accordingly, linguists and speakers of many minority languages have focused increasing effort in recent decades on language preservation—preventing extinction—and documentation, the latter in recognition of the fact that at least some preservation efforts will fail.

Descriptions of languages are arguably the one field of human knowledge where scholarly writings of today will retain their value for the foreseeable future. Assuming that civilization survives and knowledge increases, all other fields of endeavor—with the possible exception of historical narrative—will some day be replaced by a better understanding. The works of today's astronomers, biologists, and engineers will be superseded by those of future generations of astronomers, biologists, and engineers. Not so descriptions of dying languages: once the primary data can no longer be produced because there are no more native speakers, no one can write more exhaustive descriptions than those that have already been made.

Naturally, not all language documentation is equal. At one end of the spectrum of methodologies, groups such as the Volkswagen Foundation, (www.volkswagenstiftung.de/foerderung/index_e.html) have advocated a breadth-first approach, calling for preservation of large text and audio or video corpora of languages, and little more. This has the advantage of spreading funding and human resources over as wide a variety of languages as possible. At the other end of the spectrum, one might advocate intensive investigation of individual languages, resulting in the in-depth knowledge that some research programs (such as generative linguistics) require.

Arguably both approaches are useful; if we cannot know a lot about all languages, then we should at least know a lot about a few languages, and a little about a lot of

[1] See e.g the Linguistic Society of America's FAQ at http://lsadc.org/faq/endangered.htm.

languages. But even at the intensive end of the spectrum, some focusing of effort is needed. Each language has its individual story to tell, hence there is some sense in concentrating particular effort on those aspects of each language that are unique. To take one example, Lushootseed, a nearly extinct Central Coast Salish language of Washington State (United States), employs a process called 'reduplication' in its morphology. Urbanczyk [3] examined reduplication in this language, with the goal of illuminating what a reduplicative process in a human language can be. A crucial point in the analysis turned on cases where a reduplicative affix occurred on stems with a particular stress pattern. The corpus which she used was reasonably large, and had been painstakingly collected over years by competent linguists. But it turns out that in this large corpus, there are only four instances of this affix on the relevant stems—and these four instances are evenly split over whether they support the analysis.

Clearly Lushootseed is a case where directed data gathering would have been helpful. But gathering data with the aim of answering a particular question is possible only if the questions are known. And the questions which a particular language is suited to answering can be known only if the language has been analyzed in enough detail to know where its individual genius lies.

Examples could be multiplied, but we believe the point is obvious: for at least some languages, data collection must be intensive, thorough, and focused. But how can we ensure that our data collection is accurate? While a linguist may be needed to ask the right questions and to propose the solutions, humans are less good at testing those solutions rigorously; we are much too apt to overlook problems, or not to apply the analysis consistently. Computers, on the other hand, are nothing if not thorough and rigorous; as programmers know all too well, computers will not overlook "minor" faults. Hence, if we can use the computer to test the grammatical analysis, thereby finding the holes in our analysis and in our data collection, we have an ideal marriage between human and machine.

2 The Nahuatl Project

One of us (Jonathan Amith) has been documenting Ameyaltepec and Oapan Nahuatl, two dialects of the Nahuatl language from the Balsas Valley of central Guerrero, Mexico. After having lived in these villages for five years in the early 1980s, in 2000 he returned to begin a long-term language documentation effort. This has involved writing the draft of a pedagogical grammar, compiling and making available on-line a 10,000 word dictionary of Ameyaltepec and Oapan Nahuatl, and beginning a long-term effort to compile an extensive textual corpus (in audio and transcribed form).

Nahuatl is an agglutinating language, that is, inflected words may carry several affixes. In particular, transitive verbs commonly bear at least two prefixes, indicating agreement in person and number with both subject and object. Additional prefixes may mark other features, such as direction of movement or non-specific arguments. Verbs also commonly take from two to several suffixes, marking tense and aspect, as well as number and other features. Nouns are generally marked as either possessed or unpossessed; they are often used as predicates, in which case subject marking is

obligatory. Moreover, there is relatively productive noun incorporation into verbs, putting Nahuatl into the class of at least weakly[2] polysynthetic languages.

In addition, allomorphy is common. Many prefixes have two forms, one used before a vowel, the other before a consonant. Some suffixal allomorphy occurs as well, but the most complex allomorphy involves stems. Verb stems regularly have three allomorphs; the differences among these allomorphs vary among verbs. While the origin of this variation can be traced back to historical stages of Nahuatl, the regularity has become obscured over time by sound changes, so that stem allomorphy is now best described in terms of verb classes. The Ameyaltepec/Oapan dictionary lists about a dozen such classes. By the judicious use of phonological rules, we can reduce that by half. Still, the complexity of the variation makes it difficult for the linguist to ensure by inspection that the written grammar accounts for all the forms.[3]

Two years ago, we began building a computationally implemented morphological grammar of Nahuatl. At present, the verbal morphology (the most complicated aspect of Nahuatl morphology) is substantially in place, with some irregular forms still to be accounted for. Using a finite state transducer engine allows us to both parse inflected words and generate arbitrary inflected forms from a suitable meaning representation.

The forms we work with are in a shallow orthography. The orthography abstracts away from some phonological processes, particularly across word boundaries.

In summary, we are engaged in intensive and thorough documentation of the Ameyaltepec/Oapan dialects of Nahuatl, and our work is focused in part on the morphology, particularly the inflectional morphology. Computational tools are thus essential for ensuring the accuracy and coverage of our analysis. For this purpose, we rely on a morphological transducer, capable of parsing words found in our text corpus into their constituent morphemes. Failure to parse a given word indicates a problem, either in the text (such as a misspelling) or in the grammar.[4]

3 Electronic Language Documentation Versus Archivability

> "…documentation projects are usually tied to software version, file formats, and system configurations having a lifespan of three to five years… In the very generation when the rate of language death is at its peak, we have chosen to use moribund technologies, and to create endangered data. When the technologies die, unique heritage is either lost or encrypted." –Bird and Simons, *Seven dimensions of portability for language documentation and description*.

[2] Incorporation is lexically governed; hence Nahuatl is not as productive as in a highly polysynthetic language such as Mohawk.

[3] A reviewer asked whether we have attempted automatic learning of phonological rules. We have not; the complexity of Nahuatl morphology exceeds the capabilities of any morphological and phonological learning tools that we know of, and in any case grammatical analysis was underway before computational implementation began. However, we expect that the corpus and grammar we are creating will prove useful in testing such learning tools.

[4] We are also using the computational grammar as part of a web-based language learning tool.

Bird and Simons [4] lay out standards for electronic language documentation and description.[5] High on their list is the requirement that documentation should be in plain text form, using plain text annotation (e.g. XML), as opposed to non-human-readable binary formats (such as those used in many databases or word processors). We have followed those guidelines with respect to our lexicon and texts, but the grammar is a different problem. As discussed above, we have built a computational grammar for our own purposes, and we want to include this in the final archivable language documentation. Our motivation for including the grammar is three-fold. First, a parser can use a computer-readable grammar to parse words in another corpus, or by elicitation from a native speaker. Second, a parser facilitates enrichment of the lexicon by processing texts and determining which forms are missing from the lexicon. Third, a generator can use such a grammar to generate forms which are not in a corpus but which (assuming the grammar has been written correctly) are nevertheless grammatically possible.[6]

This last point—the need to be able to reliably generate forms which do not happen to be attested in a corpus—has been highlighted by recent work in the phonology of Yawelmani (Yowlumne) Yokuts [5]. It turns out that over two thirds of the wordforms from this language which were used as crucial evidence in theoretical debates about phonology in the past thirty years have been constructed by non-speaker linguists on the basis of published descriptions. Many of these constructed forms appear to be erroneous [6]. If there were a morphological generator which had been tested[7] against corpora and/or native speakers, the discussion of Yawelmani—and theoretical phonology in general—might have taken a different turn.

A transducer is a morphological engine that combines the capabilities of a parser and a generator. In our project, we are using the Xerox finite state transducer [7]. This is available only in executable form, and runs under current versions of Microsoft Windows and Linux, Solaris, and Macintosh OS X. The choice of this tool is in part related to the morphological and phonological complexities of Nahuatl, as well as the need for bidirectionality (both parsing and generation). The Xerox transducer is simply the only general and currently available tool we know of which provides the ability to write rules in a more or less linguistically motivated formalism. In particular, Xerox-style grammar rules allow one to craft an item-and-arrangement grammar, [8] together with (morpho-)phonological rules to produce predictable allomorphs (and other mechanisms for non-predictable allomorphy, i.e. suppletion).

[5] Bird and Simons [4] make a distinction between language documentation (the primary data) and description (analyses). In our project, we are doing both. But for reasons of brevity, we will not repeat this dichotomy in the remainder of this paper, instead using the terms ambiguously.

[6] Of course the danger is that the grammar will produce forms which are not in the corpus because they are incorrect; that danger can only be avoided by careful checking. The danger of producing ungrammatical forms from a morphological grammar is less than the same danger with a syntactic grammar, since apart from productive compounding, the set of morphological forms is finite, unlike the set of sentences. Moreover, irregular forms are almost never rare, so that once the basic vocabulary has been covered, the remaining forms tend to be very predictable. But the point remains: careful checking is necessary.

[7] Methodically, but not necessarily exhaustively.

[8] The Xerox tools do provide for certain kinds of item-and-process affixation processes.

The formalism resembles that of the American structuralists, in the sense that there is no provision for the use of rules written in terms of phonological (distinctive) features (much less autosegmental-style analyses, or constraint-based theories). While this means that the analysis is couched in terms of a severely dated theory, for purposes of documenting a grammar, this is not a disadvantage: the changes which later theories of morphology made are largely irrelevant to Nahuatl (which has no large-scale non-concatenative morphological processes, the focus of much more recent work).[9] Likewise, the theories of phonology which came later were, for the most part, either in areas which are not crucial to Nahuatl (such as stress and tone), or were aimed at solving questions of descriptive or explanatory adequacy which are not essential to documenting the grammar of a language. In summary, we are interested in getting the facts of Nahuatl right, and at drawing attention to the generalizations that we know of, leaving it to future generations of linguists to draw conclusions of theoretical import. For our purpose, we have found the Xerox transducers to be perfectly adequate.

However, the use of a computationally interpreted grammar gives rise to a dilemma. Given that the best practice for documentation is to use plain text, while the inclusion of a working morphological grammar implies an executable program, there is a tension between the long-term goals of language documentation and our desire to provide a complete description of Nahuatl morphology. We want the grammar to be runnable for years to come. We do not want to rely on the continued existence of particular executable version of a morphological engine, an engine which may only be executable under a fixed set of operating systems on a fixed set of CPUs.[10]

In theory, one could overcome this problem by considering the archival form of the grammar to include a computer-readable form of the Xerox tools, together with an executable form of the Linux operating system. But this overlooks issues of hardware compatibility (does the archival grammar include an entire PC to run the software on, or perhaps an emulator?), as well as more mundane issues of copyright.

Alternatively, one might hope that the owners of proprietary software (Xerox, in this case) will some day release the source code for the software that we are dependent on. Whether this hope will be realized in any particular case is of course unknowable. Moreover, releasing the source code does not necessarily mean that anyone can produce an executable form of the program, since the programming languages the tools are written in may not exist in the long term.[11]

---

[9] More recent theories of morphology often assume a feature-based approach to grammatical meaning, which the Xerox tools also do not directly support. Again, this is not highly relevant to our grammar.

[10] This portability problem was brought forcibly to our attention after a recent server upgrade from the Linux OS to the very similar FreeBSD OS. Our Xerox tools ceased to work; the transducer could not compile the grammar. The problem was resolved by installing Linux compatibility libraries. But that solution may not be available ten years from now, and it will almost certainly not be available to investigators who might wish to use our grammar a century hence.

[11] This is not a minor quibble: one of the authors (Maxwell) released the source code to his own morphological parser (Hermit Crab, www.sil.org/computing/hermitcrab/) a decade ago. Unfortunately, it was coded in versions of Prolog and C that are no longer available, and porting to currently available versions of Prolog would be non-trivial. ‘Open source’ is not the complete answer.

For purposes of language documentation, therefore, the limitation of requiring a particular executable program, compiled for a particular combination of operating system and CPU, is unacceptable. This means that the grammar we have written for the Xerox transducer is, by itself, inadequate to the goal of language documentation.

Fortunately, the programming languages that the tools interpret (xfst and lexc) are based on the abstract language of regular expressions. The theory of finite state automatons and transducers, which generate and parse the languages defined by regular expressions, is well-understood, and forms the groundwork of much of modern computing theory. Thus, to the extent that the notation expected by the Xerox tools is generic, it meets our needs for longevity as well as could be expected from any such formalism. Unfortunately, the xfst/ lexc programming languages do have idiosyncrasies in their notation (as do all such notations).

One solution would be to simply document the xfst and lexc notation, perhaps by including a computer-readable version of the manual in the archival form of our grammar. But this is a high price to pay, not only in terms of the size of the manual, but more importantly in terms of the burden that it would impose on users. Anyone wishing to understand our grammar would need to first read the programming language manual.

A better approach, we argue in the next section, is to document the effect of the rules and other notation at the point in the grammar where they are used.

4 Two Forms of Documentation

> "…one deliberately writes a paper, not just comments, along with code."—Doug McIlroy. "Programming Pearls: A Literate Program", *CACM*, June 1986, pg. 478-479.

In language engineering, best practice for the documentation of computational grammars calls for "two types of documentation: one within the grammar itself and one as an overview of the grammar" [8:169]. Language engineering differs from language documentation with respect to the purpose of the grammar, but this recommendation at least is relevant to both communities

In our Nahuatl project, the task of writing the human-readable, or prose version of the grammar has fallen to one of us (Jonathan Amith, who speaks Nahuatl), while the task of writing the computer-processible grammar , with its own documentation, falls to the other (Mike Maxwell, who "speaks" the xfst programming language). In the process of writing the xfst version, Maxwell found himself frequently writing comments into the source code that were paraphrases of the prose grammar, often including references to section numbers in the human-readable grammar. The two documents were thus intertwined, in the sense that a thorough understanding of the source code and its comments required jumping back to the prose grammar. Of course, this meant that revisions had to be made in both places.

Moreover, the prose grammar was sometimes ambiguous, or omitted details that were necessary in order to write the machine-readable grammar. For example, there are two processes of consonant cluster simplification: degemination, which reduces of a sequence of two identical consonants to one, and affricate reduction, which changes an alveolar affricate to a fricative in the environment preceding another alveolar

consonant. The grammar gave examples of the application of these rules in which the second alveolar consonant was /n/, /l/, and /t/, but did not explain what happened to a sequence of two alveolar affricates /ts+ts/. In theory, both rules would be applicable. But since the two rules are mutually bleeding (the application of one precludes the application of the other), in fact only one of them could apply. Under xfst (as well as the particular theory of phonology on which xfst is roughly based), which rule would apply is determined by the rule order[12] For purposes of the xfst grammar, the author of the xfst grammar therefore had to know which result was appropriate, but the human-readable grammar was silent on this point. (As it happens, it is the affricate reduction rule which applies, and this is now documented.)

The two forms of the grammar, with their individual documentation, are thus complementary: the prose form is easier for the human reader to understand. The machine readable grammar, on the other hand, is more precise and less ambiguous.

In this sense, the distinction is like that between a traditional written grammar, and a generative grammar written in a precise formalism. But in a practical sense, the machine readable grammar is even more precise than the generative grammar, because it can be run on a computer and tested for correctness against data. While in theory it is possible to hand-test a generative grammar, in practice this becomes nearly impossible for grammars with any degree of complexity (and our Nahuatl grammar certainly exceeds this degree of complexity, at least for us).

Given that both a verbal description and an implementable form of the grammar are necessary for purposes of language documentation, the key issues are: how to minimize duplication between the two forms, keep them in synch, and make both grammars useable for decades, or better, centuries.

5 Literate Programming for Language Documentation

> "Literate programming…is the difference between performing and exposing a magic trick." –Ross Williams, *FunnelWeb Tutorial Manual*

For purposes of language documentation the ideal grammar would combine the human readable and computer readable versions into one. This is precisely what the discipline of literate programming was invented for.

Donald Knuth [9] created the term "literate computing" to describe a way of writing computer programs that inverted the usual way of documenting code. Instead of writing a document consisting of source code which the computer could understand, and then sprinkling in comments for humans who might need to decipher it (to fix bugs, or improve it in some way), the literate programming document becomes a way of explaining to a human being how the program works, and contains the source code at appropriate points. An executable program can be extracted from the document, a process which Knuth referred to as 'tangling'. The process of producing an elegantly formatted human-readable document he called 'weaving'.

The advantages of literate programming are even more important for language documentation than for computer programming. Computer programs have a longevity

[12] Or more precisely, the order in which the finite state transducers representing the two rules are composed.

measured in years, or at best, in decades. Language grammars, on the other hand, should remain accessible for centuries, even millennia.[13]

We sketch here what such a literate grammar looks like. Consider the fragment of the Nahuatl grammar shown in the figure below.

4.3 h-deletion

/h/ deletes everywhere except word-final (including before zero morphs). This applies only to underlying /h/, not to various /h/s that are derived from /w/ and /k/; that is, the rules deriving /h/ from /w/ and /k/ counterfeed *h-deletion.*

Any /h/ internal to a lexically listed stem is presumably non-derived, and should not be subject to this rule. This condition is encoded in the xfst version of this rule by the requirement that the right-hand environment must contain an overt morpheme boundary ('%-'; the '%' is an escape character required by the xfst notation before a special character like '-').

This rule bleeds the rule of *e-epenthesis* (below).

```
define hDeletion [h -> 0 || [? - c] _ %- [Cons | Vowel]];
```

The '[? - c]' in the left environment of *h-deletion* prevents this rule from applying to the 'h' of the grapheme 'ch' (it encodes the regular expression "any single character except 'c').

4.4 w → h

/w/ becomes /h/ before /k/. Again, this applies in derived environments, which we have encoded in xfst by requiring an explicit boundary marker.

This rule counterfeeds the above rule of *h-Deletion* (section 4.3). It should therefore also be bled by the rule of *k-Drop* (section 4.2), by transitivity of ordering. It is unclear whether *w → h* interacts crucially with any other rules.

```
define wk2h [w -> h || _ Nulls %- k];
```

The following is an example of the application of this rule:

chi:x-te:w-ka-0 'wait.for/V-upon.parting-PLUP-SG'

chi:xte:hka *w → h*

Fig. 1. A fragment of the Nahuatl grammar

The rules shown there are explained in prose, as is their interaction with other rules. As would be expected in a grammatical description intended for human consumption, we include examples of rule application in this grammar fragment. It is in fact common practice to include examples in traditional computational grammars [8]. Well-commented code is also preferred in programming languages, which

[13] One might question whether the longevity of the English or Spanish in which the grammatical description is written, not to mention the dictionary glosses, although we see no alternative here. Of course, the art of machine translation may render this a moot question.

includes computational grammars; but most code is in fact under-commented. The expository nature of literate programming, as applied to grammars of natural languages, makes it more natural to supply examples for the human reader's benefit.

The formatting—bold for section headings, italics for names of rule, the use of Courier font for programming code—is also intended to be an aid to understanding, and is not available in the comment fields of (most) computer languages. The formatting itself is of course not present in the XML file which is the archival form of the grammar, since that would be contrary to our aims of portability. Instead, the formatting is provided by style sheets, in standard XML fashion.

Walsh [10] has proposed a small extension to XML to support literate programming. The extension consists of primitives, `<src:fragment>` and `<src:fragref>`, which respectively allow the inclusion of programming code fragments, and references to those fragments. These extensions are well suited to being used with the DocBook [11]), an XML format commonly used for books and articles. Stylesheets are then used to produce the nicely formatted human-readable document (in printed format, PDF file, HTML format, etc.), as well as the actual program. We are currently engaged in producing a grammatical description of Nahuatl, complete with working programming code, in this literate programming format. This will then serve as the archival format for the Nahuatl grammar.

6 Conclusion

> "Always remember that you're not just writing for the next couple months or years, but possibly for the next couple of thousands of years."—Elliotte Rusty Harold and W. Scott Means, *XML in a Nutshell*, p. 96.

We have argued that language documentation calls for not only a verbal description of the grammar, but a computational analysis as well. We have also argued why this need is ill served by a traditional programming approach, and in its place have advocated a literate computing approach, intertwining the verbal description and the computational analysis. And finally, we have described how our own language documentation project, encompassing several dialects of the Nahuatl language, is using such a documentation method.

Epilogue: Object Oriented Grammar Editors

We argued above that the human-readable and machine-readable forms of the grammar are complementary. As linguists, we cannot help noting that the term 'complementarity' is related to the linguistic term 'complementary distribution', suggesting that the two forms of the grammar might, in fact, be best viewed as allo-grammars: dual manifestations of a single underlying structure. For example, consider the grammar fragment in the figure below, incorporating both prose and code:

The object-marking prefixes of the Oapan dialect are shown in the chart below:

	Singular	Plural
1	ne:ch-	te:ch-
2	mits-	me:ch-
3	k-	kim-

The xfst code for this is as follows:

```
define ObjPre [[[1SgO .x. {ne:ch}]%-"@D.PERS.1@"]
              |[[2SgO .x. {mits} ]%-"@D.PERS.2@"]
              |[[3SgO .x. k       ]%-               ]
              |[[1PlO .x. {te:ch}]%-"@D.PERS.1@"]
              |[[2PlO .x. {me:ch}]%-"@D.PERS.2@"]
              |[[3PlO .x. {kim}  ]%-              ]];
```

Fig. 2. Another fragment of the Nahuatl grammar

The xfst 'define' repeats much of the information in the table. This is obvious for the phonological form of the morphemes, but it is also true for the representation of their meaning: the glosses in the xfst code are directly related to the table labels: '1SgO' is equivalent to '1' (row label) 'Singular' (column label) 'Object' (from the caption); this relation between morphosyntactic features and glosses argued in more detail in [12]. The flag diacritics (the xfst constructs containing the '@' signs) repeat the person information. Even the hyphens at the right end of the morphemes in both representations (representing boundary markers) are in fact a "view" of the fact that this is a table of prefixes (as opposed to suffixes or roots).

The table and the xfst code are thus best seen as views of a single underlying structure: a slot in an inflectional schema, together with the affixes that can fill that slot. Suppose that we had a single computational object representing that structure, and that we could design views of that object that looked like the table or like the xfst code. Then we would simply insert a reference to that object in the prose of the grammar, choosing the table view to give human-readable form, and the xfst code view to produce the machine-readable form.

Similarly, the entire grammar—from individual affixes to phonological rules—can be conceived of as a collection of objects.

An object-oriented approach was taken in the LinguaLinks project [13-15], developed by SIL during the 1990s; and it is being taken in the current SIL FieldWorks project (http://fieldworks.sil.org/). Such an approach is capable of incorporating a literate computing view directly in the user interface.

An object-oriented grammar development system comes with a cost: the objects must somehow be integrated into a traditional word processor (or other document-producing tool), or else the capabilities of the traditional word processor must be replicated in the object-oriented programming environment. Additionally, the programmers of the object-based grammar tools must anticipate the potential needs of

all users. Not everyone will want to use such an environment. But even for grammars developed in such an environment, the object oriented database is a binary format; another format, such as an XML-based format, is needed for archival purposes.

Thus, regardless of the methodology used to create the grammar, we believe that the literate programming approach we have outlined in this paper constitutes the best practice for grammar documentation: a text (XML) based format which intertwines verbal description and the programming source code for the grammar.

References

1. Grimes, B.F.: Ethnologue: Languages of the World. 14th. ed. Summer Institute of Ling., Dallas, 2000
2. Derbyshire, D.: Hixkaryana. Lingua Descriptive Studies 1. North-Holland, Amsterdam, 1979
3. Urbanczyk, S.: Patterns of Reduplication in Lushootseed. Outstanding Dissertations in Linguistics. Garland Publishing, New York, 2001
4. Bird, S. and Simons, G.: Seven dimensions of portability for language documentation and description. Language 79, 2003, 557-582
5. Weigel, W.F.: The Yokuts Canon: A case study in the interaction of theory and description. Paper presented at Annual meeting of the Linguistics Society of America. San Francisco, 2002
6. Blevins, J.: A Reconsideration of Yokuts Vowels. International Journal of American Linguistics 70, 2004, 33-51
7. Beesley, K.R. and Karttunen, L.: Finite State Morphology. CSLI Studies in Computational Linguistics. University of Chicago Press, Chicago, 2003
8. Butt, M. and King, T.H.: Grammar Writing, Testing and Evaluation. In Farghaly, A. (ed) Handbook for Language Engineers. CSLI Publications, Stanford, 2003, 129-179
9. Knuth, D.E.: Literate programming. The Computer Journal 27, 1984, 97-111
10. Walsh, N.: Literate Programming in XML. In XML 2002. Baltimore, MD, 2002
11. Walsh, N. and Muellner, L.: DocBook: The Definitive Guide. O'Reilly & Associates, Inc., Sebastopol, California, 1999
12. Hayashi, L.S., Maxwell, M.B., and Simons, G.: A Morphological Glossing Assistant. In Proceedings of the International LREC Workshop on Resources and Tools in Field Linguistics. Las Palmas, Spain, 2002
13. Rettig, M., Simons, G., and Thomson, J.: Extended objects. Communications of the ACM 36, 1993, 19-24
14. Simons, G.F. and Thomson, J.V.: Multilingual Data Processing in the CELLAR Environment. In Nerbonne, J. (ed) Linguistic Databases. Center for the Study of Language and Information, Stanford, CA, 1997, 203-34
15. Simons, G.F.: The Nature of Linguistic Data and the Requirements of a Computing Environment for Linguistic Research. In Lawler, J. (ed) Using Computers in Linguistics: A Practical Guide. Routledge, New York, NY, 1998, 10-25

Creating Subjective and Objective Sentence Classifiers from Unannotated Texts

Janyce Wiebe[1] and Ellen Riloff[2]

[1] Department of Computer Science,
University of Pittsburgh, Pittsburgh, PA 15260
wiebe@cs.pitt.edu
[2] School of Computing, University of Utah,
Salt Lake City, UT 84112
riloff@cs.utah.edu

Abstract. This paper presents the results of developing subjectivity classifiers using only unannotated texts for training. The performance rivals that of previous supervised learning approaches. In addition, we advance the state of the art in objective sentence classification by learning extraction patterns associated with objectivity and creating objective classifiers that achieve substantially higher recall than previous work with comparable precision.

1 Introduction

There has been a recent swell of interest in the automatic identification and extraction of attitudes, opinions, and sentiments in text. Motivation for this task comes from the desire to provide tools for information analysts in government, commercial, and political domains, who want to automatically track attitudes and feelings in the news and on-line forums. How do people feel about recent events in the Middle East? Is the rhetoric from a particular opposition group intensifying? What is the range of opinions being expressed in the world press about the best course of action in Iraq? A system that could automatically identify opinions and emotions from text would be an enormous help to someone trying to answer these kinds of questions. Applications that could benefit from this technology include multi-perspective question answering, which aims to present multiple answers to the user based on opinions derived from different sources, and multi-document summarization, which aims to summarize differing opinions and perspectives.

There is also a need to explicitly recognize objective, factual information for applications such as information extraction and question answering. Linguistic processing alone cannot determine the truth or falsity of assertions, but we could direct the system's attention to statements that are objectively presented, to lessen distractions from opinionated, speculative, and evaluative language.

The goal of our research is to develop learning methods to create classifiers that can distinguish subjective from objective sentences. We strive to develop systems that excel at subjective classification as well as objective classification.

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 486–497, 2005.

In this paper, we present the results of developing subjectivity classifiers using only unannotated texts for training. The performance of the classifiers rivals that of previous supervised learning approaches to the same task. In addition, we advance the state of the art in objective sentence classification by learning new objective clues and creating objective classifiers that achieve substantially higher recall than previous work with comparable precision. Our approach begins with a seeding process that utilizes known subjective vocabulary to automatically create training data. This data is then used to train an extraction pattern learner and a probabilistic classifier. Finally, we add a self-training mechanism that improves the coverage of the classifiers, while still relying only on unannotated data.

2 The Data and Classification Task

The texts used in our experiments are English language versions of articles from the world press. The data is from a variety of countries and publications and covers many different topics (it was obtained from the Foreign Broadcast Information Service (FBIS), a U.S. goverment agency). 535 texts from this collection have been manually annotated with respect to subjectivity as part of a U.S. government funded program on automatic question answering.[1] These manually annotated texts comprise the *Multi-Perspective Question Answering (MPQA) corpus* and are freely available at nrrc.mitre.org/NRRC/publications.htm.

The test set used in our evaluations consists of 9,289 of the sentences in the MPQA corpus. None of this test data was used to produce any of the features included in our experiments. 5104 of the sentences in the test set (54.9% of the data) are subjective according to the definitions given below. Thus, the accuracy of a baseline classifier that chooses the most frequent class is 54.9%. Our unannotated text corpus consists of 298,809 sentences from the world press collection, and is distinct from the annotated MPQA corpus.

The annotation scheme and inter-coder reliability studies associated with the MPQA data are described in [1]. The scheme was inspired by work in linguistics and literary theory on *subjectivity*, which focuses on how opinions, emotions, etc., are expressed linguistically in context [2]. The goal is to identify and characterize expressions of *private states* in a sentence. *Private state* is a general covering term for opinions, evaluations, emotions, and speculations [3]. For example, in sentence (1), the writer is expressing a negative evaluation.

(1) "They are no more than frail excuses and pretexts to evade the peace process since this process does not agree with the ideology of expansion and the building of settlements."

Sentence (2) reflects the private state of Western countries. Mugabe's use of "overwhelmingly" also reflects a personal private state, his positive reaction to and characterization of his victory.

[1] The ARDA (Advanced Research and Development Activity in Information) AQUAINT (Advanced QUestion and Answering for INTelligence) program.

(2) "Western countries were left frustrated and impotent after Robert Mugabe formally declared that he had overwhelmingly won Zimbabwe's presidential election."

The annotators were asked to identify all expressions of private states in each sentence and to indicate various attributes, including strength (*low*,*medium*,*high*, or *extreme*). The gold-standard classes used in our evaluations are defined as follows: if a sentence has at least one private state of strength *medium* or higher, then the sentence is subjective; otherwise, it is objective. These are the same definitions that other researchers have used when performing experiments with the MPQA data [4, 5].

3 Learning Subjective and Objective Sentence Classifiers

3.1 Automatically Generating Training Data Using Rule-Based Classifiers

As a starting point for our research, we reimplemented the high precision, low recall subjective and objective classifiers that we previously developed [4]. We will refer to these as the *rule-based classifiers* because they do not involve learning but merely classify sentences by looking for well-established general subjectivity clues that have been previously published in the literature.[2] Some are drawn from manually developed resources, including entries from [6, 7], Framenet lemmas with frame element *experiencer* [8], and adjectives manually annotated for polarity [9]. Some were learned from corpora, including words distributionally similar to subjective seed words [10], n-grams [11, 12], and subjective nouns learned using extraction pattern (EP) bootstrapping [5]. The clues were divided into strong and weak subjective clues, where strong subjective clues have subjective meanings with high probability, and weak subjective clues have subjective meanings with lower probability.

The rule-based subjective classifier classifies a sentence as subjective if it contains two or more strong subjective clues (otherwise, it does not label the sentence). In contrast, the rule-based objective classifier looks for the absence of clues: it classifies a sentence as objective if there are no strong subjective clues in the current sentence, there is at most one strong subjective clue in the previous and next sentence combined, and at most 2 weak subjective clues in the current, previous, and next sentence combined (otherwise, it does not label the sentence).[3]

Our research uses these rule-based classifiers to generate training data for subsequent learning algorithms, which we will describe in the coming sections. Figure 1 shows the first stage of the training data creation process. The rule-

[2] We will be happy to make these clues available to other researchers.

[3] This is slightly more liberal than in [4], which did not allow a strong subjective clue in the previous or next sentence. This difference explains the higher recall figures reported here.

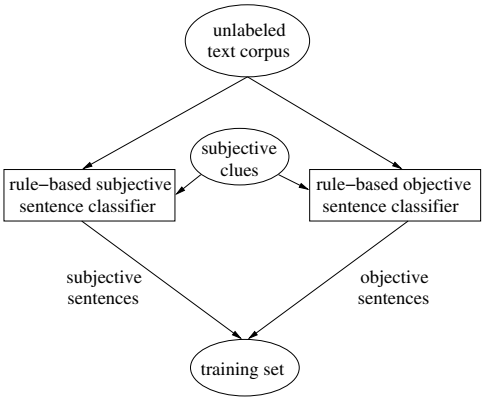


Fig. 1. Initial Training Data Creation

based subjective classifier is applied to the unlabeled corpus to identify sentences that it can label as subjective. Similarly, the rule-based objective classifier identifies sentences that it can label as objective. These subjective and objective sentences form our *initial training set*.

We use the following evaluation metrics in this paper. *Subjective precision (SubjPrec)* is the percentage of sentences automatically classified as subjective that are truly subjective. The *subjective recall (SubjRec)* is the percentage of true subjective sentences that are automatically classified as subjective. The *subjective F-measure (SubjF)* is the usual F-measure combining precision and recall. *ObjPrec*, *ObjRec*, and *ObjF* are defined similarly.

On the annotated test set, the rule-based subjective classifier achieved 34.2% subjective recall and 90.4% subjective precision. The rule-based objective classifier achieved 30.7% objective recall and 82.4% objective precision. Based on these results, we expect that the initial training set generated by these classifiers is of relatively high quality. Of the 298,809 sentences in the unannotated text corpus, the rule-based classifiers labeled 52,918 sentences as subjective and 47,528 as objective, creating a training set of over 100,000 sentences.

3.2 Extraction Pattern (EP) Learning

Previous research has shown that patterns designed for information extraction can effectively represent expressions associated with subjectivity [4]. Objectivity is a different beast because any objective statement can be made subjective by adding a subjective modifier to it. Consequently, it is not clear that individual expressions can be considered to be truly objective in an absolute sense. However, we hypothesized that in practice there are many expressions that are highly correlated with objective statements and therefore would be strong clues that a sentence is objective. In the Wall Street Journal, for example, sentences containing the words “profits” or “price” are very likely to be objective, even though there is no reason why a subjective sentence could not contain those words.

Consequently, we also decided to explore the idea of learning extraction patterns that are correlated with objectivity and then using them as features in a machine learning algorithm. To learn extraction patterns, we used the AutoSlog-TS [13] algorithm because it does not need annotated texts for training. Instead, AutoSlog-TS requires one set of "relevant" texts and one set of "irrelevant" texts. Extraction patterns are created by applying a set of syntactic templates to the corpus. The syntactic constructions recognized by AutoSlog-TS are described in [13] and reflect syntactic relationships identified by a shallow parser.

We trained the EP learner on the initial training set to generate patterns associated with objectivity as well as patterns associated with subjectivity. In our experiments, the subjective sentences were the relevant texts, and the objective sentences were the irrelevant texts. The patterns chosen as the subjective patterns are those that are strongly correlated with subjective sentences, while the patterns chosen as the objective patterns are those that are negatively correlated with subjective sentences (and hence positively correlated with objective sentences). AutoSlog-TS merely ranks patterns in order of their association with the relevant texts, so we automatically selected the best patterns for each class using two thresholds: θ_F is the frequency of the pattern in the corpus, and θ_P is the conditional probability (estimated from the training set) that a text is relevant if it contains the pattern: $\Pr(\text{relevant} \mid \textit{pattern}_i)$. For our experiments, *subjective patterns* were identified by setting $\theta_F \geq 5$ and $\theta_P \geq .95$ (i.e., at least 95% of its occurrences must have been in subjective sentences). *Objective patterns* were identified by setting $\theta_F \geq 5$ and $\theta_P \leq .15$ (i.e., at most 15% of its occurrences could have been in subjective sentences). Table 1 shows a few examples of subjective and objective patterns that were learned.

Table 1. Extraction Pattern Examples

Subjective Patterns	**Objective Patterns**
<subj> believes	<subj> increased production
<subj> was convinced	<subj> took effect
aggression against <np>	delegation from <np>
to express <dobj>	occurred on <np>
support for <np>	plans to produce <dobj>

Consider a simple classifier that classifies a sentence as subjective if it contains any of the learned subjective patterns. The subjective precision of this classifier on the manually annotated test set is 74.5% (i.e., 74.5% of the sentences with subjective patterns are subjective). The subjective recall is 59.8% (i.e., 59.8% of the subjective sentences contain at least one subjective pattern). The similar figures for a cooresponding objective classifier are 71.3% objective precision and 11.7% objective recall (i.e., 71.3% of the sentences with objective patterns are objective, and 11.7% of the objective sentences contain at least one objective pattern). The low objective recall reflects the fact that many fewer instances of objective patterns were found in the data (832 versus 6364 instances

Table 2. Rule-Based Classifier Results

	SubjRec	SubjPrec	SubjF	ObjRec	ObjPrec	ObjF	Acc
Subj RBC	34.2	90.4	46.6				61.9
Subj RBC w/Patterns	58.6	80.9	68.0				69.7
Obj RBC				30.7	82.4	44.7	65.8
Obj RBC w/Patterns				33.5	82.1	47.6	66.7

of subjective patterns). These results suggest that the EPs are good clues for distinguishing subjective sentences from objective sentences, but are not sufficient by themselves.

Next, we incorporated the learned EPs into the rule-based classifiers as follows. The subjective patterns were added to the set of strong subjective clues, which are used by both the subjective and objective rule-based classifiers. The strategy used by the rule-based subjective classifier remained the same. However, the strategy used by the rule-based objective classifier was augmented as follows: in addition to its previous rules, a sentence is also labeled as objective if it contains no strong subjective clues but at least one objective EP. Note that adding the subjective EPs to the set of strong subjective clues works to *decrease* the recall of the objective classifier because it looks for the absence of subjectivity clues. To balance that effect, the additional test for objective EPs can serve to *increase* the recall of the objective classifier.

The first row of Table 2 shows the results of the original rule-based subjective classifier (*Subj RBC*), and the second row shows the results after adding the subjective extraction pattern clues. Similarly, the third row shows the results for the original rule-based objective classifier (*Obj RBC*), and the fourth row shows the results after adding the objective EP clues. Comparing rows one and two, the subjective precision dropped from 90.4% to 80.9%, but subjective recall increased from 34.2% to 58.6%. Comparing rows three and four, the objective precision decreased only slightly (from 82.4% to 82.1%), and the objective recall increased from 30.7% to 33.5%. Adding EPs to the rule-based classifiers clearly expanded their coverage with relatively smaller drops in precision.

3.3 Naive Bayes Sentence Classification

The labeled sentences identified by the rule-based classifiers provide us with the opportunity to apply supervised learning algorithms to our sentence classification task. Previous work [14, 5, 15] found that naive Bayes performs well for subjectivity recognition, so we used naive Bayes as our learning algorithm. We

Table 3. Test Results of Naive Bayes Trained on Initial Training Data

	SubjRec	SubjPrec	SubjF	ObjRec	ObjPrec	ObjF	Acc
Naive Bayes	70.6	79.4	74.7	77.6	68.4	72.7	73.8

trained the naive Bayes classifier using the initial training set and several types of set-valued features. There are features for each of the following sets: the strong subjective clues used by the original rule-based classifiers; the weak subjective clues used by the objective rule-based classifier; the subjective patterns generated by the EP learner; and the objective patterns generated by the EP learner. We also added features for the following parts of speech, which were shown to be effective in previous work [5, 15, 11]: pronouns, modals (excluding 'will'), adjectives, cardinal numbers, and adverbs (excluding 'not'). A three-valued feature was defined for each set based on the presence of 0, 1, or ≥ 2 members of that set in the sentence. In addition, to incorporate contextual information in the classifier, another three-valued feature was defined for each set based on the presence of 0, 1, or ≥ 2 members of that set in the previous and next sentences combined.

Row one of Table 3 shows the performance of the naive Bayes classifier on the test set. The classifier achieves relatively balanced recall and precision for both subjective and objective sentences.

3.4 Self-Training the Sentence Classifier

The initial training data used by the naive Bayes classifier was generated by the rule-based classifiers, which simply look for the presence or absence of a set of general subjectivity clues. There are obvious concerns associated with this type of automatically created training data, such as potential biases introduced by the rules. A related concern is that the training sentences will be similar to one another and less heterogenous than the set of sentences that the classifier will ultimately be applied to.

We therefore saw an opportunity to try to improve the classifier by generating a new training set using the classifier itself. The naive Bayes classifier uses a greater variety of features than the rule-based classifiers and it exploits a probabilistic model to make classification decisions based on combinations of these features. We hypothesized that the naive Bayes classifier might be able to reliably label a different, and perhaps more diverse, set of sentences in the unlabeled corpus than the rule-based classifiers did.

The procedure we use is a variant of *self-training*, as the term is used by Nigam and Ghani [16]. They describe the procedure as follows: "Initially, self-training builds a single naive Bayes classifier using the labeled training data and all the features. Then it labels the unlabeled training data and converts the most confidently predicted document of each class into a labeled training example. This iterates until ..." (p. 90). Rather than adding one instance per class at a time to a cache of labeled data, we use our naive Bayes classifier to label all the sentences in the entire unannotated corpus from scratch, including those in the initial training set. Then, we select the top $N/2$ most confidently labeled sentences in each class to include in the new training data (where N = the size of the initial training set + 10,000 sentences). The chosen sentences form a brand new training set that we then use to retrain the EP learner and then the naive Bayes classifier. The overall process is depicted in Figure 2.

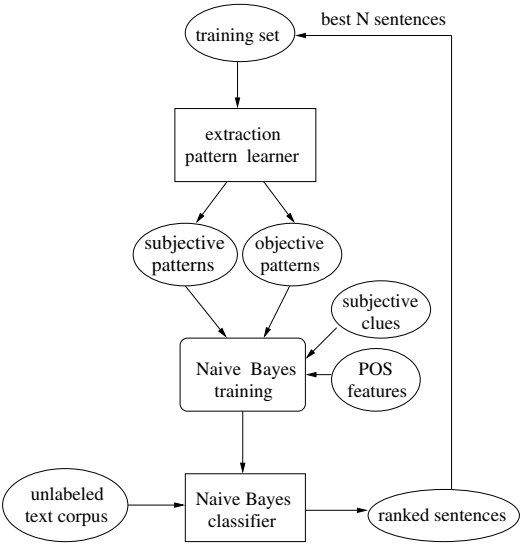


Fig. 2. Self-Training Process

The recall of the learned patterns improved substantially using the new training set, with just a minor drop in precision: subjective precision of the subjective patterns decreased from 74.5% to 73.1%, and objective precision of the objective patterns decreased from 71.3% to 68.9%, while subjective recall of the subjective patterns increased from 59.8% to 66.2% and objective recall of the objective patterns increased from 11.7% to 17.0%.

Table 4. Comparison of Results

	SubjRec	SubjPrec	SubjF	ObjRec	ObjPrec	ObjF	Acc
(a) Subj RBC w/Patterns 1	58.6	**80.9**	68.0				69.7
(b) Subj RBC w/Patterns 2	62.4	80.4	70.3				71.0
(c) Obj RBC w/Patterns 1				33.5	82.1	47.6	66.7
(d) Obj RBC w/Patterns 2				34.8	**82.6**	49.0	67.3
(e) Naive Bayes 1	70.6	79.4	74.7	**77.6**	68.4	**72.7**	**73.8**
(f) Naive Bayes 2	**86.3**	71.3	**78.1**	57.6	77.5	66.1	73.4
(g) RWW03 (supervised)	77	81	79	74	70	72	76

Table 4 shows the performance of the rule-based classifiers (RBC) with learned patterns and the naive Bayes classifiers on the test set after training on the initial training set (these are the rows labeled 1) and after retraining on the new training set (these are the rows labeled 2).

When the patterns learned on the new training set were incorporated into the rule-based classifiers, the classifiers showed increases in recall but with virtually

no drop in precision and even a slight increase for objective sentences (compare rows (a) and (b) for the subjective rule-based classifiers, and rows (c) and (d) for the objective rule-based classifiers).

Rows (e) and (f) show that the recall of the naive Bayes classifier swung dramatically toward subjective sentences (+15.7% recall for subjective sentences, -20% recall for objective sentences). At the same time, subjective precision decreased by 8 percentage points while objective precision increased by 9.

Finally, row (g) shows the performance of the best supervised subjectivity sentence classifier on the same type of data [5], which we will denote as RWW03. RWW03 was trained on a subset of the MPQA corpus containing 2197 sentences. 1296 (59%) of those sentences were subjective, so the accuracy of a baseline classifier that chooses the most frequent class was a bit higher for that dataset than for the one used in this paper (its baseline accuracy is 54.9%, as explained in Section 2).

The boldface numbers represent the best results achieved by our classifiers for each evaluation metric. For subjective sentences, the self-trained naive Bayes classifiers achieved the best recall, which was substantially higher than the recall obtained by RWW03, although our precision at the high recall level is lower. The best precision that we obtained is basically the same as RWW03, but with lower recall. For objective sentences, our initial naive Bayes classifier (e) achieved a slightly higher F-measure than RWW03. All in all, our classifiers achieved performance levels comparable to those obtained by a supervised learning system. Our highest precision objective classifier was the rule-based classifier with EPs after self-training (d).

4 Related Work

There has been a recent flurry of research in the related areas of opinion extraction, sentiment analysis, semantic orientation and polarity classification, and subjectivity analysis. Much of this work focuses on lexical aquisition, identifying subjective, positive, or negative words and phrases [9, 17, 9, 18, 10, 19, 20]. Riloff and Wiebe [4] used extraction pattern learning to find subjective expressions, but we know of no previous research on learning objective expressions.

Several projects have focused on document-level subjectivity classification. Some work identifies inflammatory texts (e.g., [21]) or classifies texts as positive or negative ([22, 17, 14]). Research in genre classification has included recognition of subjective genres such as *editorials* and objective genres such as *business* or *news* (e.g., [23, 24, 12, 15]).

In contrast, our work involves classifying individual sentences. Sentence-level subjectivity classification is useful because most documents contain a mix of subjective and objective sentences. For example, newspaper articles are typically thought to be relatively objective, but [12] reported that, in their corpus, 44% of sentences (in articles that are not editorials or reviews) were subjective.

Almost all previous evaluations of sentence-level subjectivity classifiers involved supervised learning systems (e.g., [15, 5, 11]). We compared our results

to [5] in Section 3.4. The precision achieved by [15] was lower, especially for objective sentences. The accuracies reported by [11] are higher (they do not report precision), but their baseline accuracy is very high. However, [15, 11] used different data sets with different annotation schemes, so our results cannot be directly compared.

As described in Section 3.1, [4] report high subjective and objective precisions, but achieve at most 40% subjective recall and 30% objective recall.

Automatic subjectivity/opinion/sentiment analysis is being applied to many interesting applications, including classification of reviews [19, 14, 11, 25, 26], analysis of product reputations [26, 25, 27], tracking sentiments toward events [28, 22, 29], and incorporating opinions into question answering and multi-document summarization systems [15].

5 Conclusions

We presented the results of developing subjectivity classifiers using only unannotated texts for training. The performance rivals that of previous supervised learning approaches. In addition, we advance the state of the art in objective sentence classification, by learning EPs associated with objectivity and creating objective classifiers that achieve substantially higher recall than previous work with comparable precision.

Acknowledgements

This research was supported in part by the National Science Foundation under awards IIS-0208985 and IIS-0208798, and this material is based upon work supported by the Advanced Research and Development Activity (ARDA) under Contract No. NBCHC040012. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the ARDA or the Department of Interior-National Business Center (DOI-NBC).

References

1. Wilson, T., Wiebe, J.: Annotating Opinions in the World Press. In: Proceedings of the 4th ACL SIGdial Workshop on Discourse and Dialogue (SIGdial-03). (2003) 13–22
2. Banfield, A.: Unspeakable Sentences. Routledge and Kegan Paul, Boston (1982)
3. Quirk, R., Greenbaum, S., Leech, G., Svartvik, J.: A Comprehensive Grammar of the English Language. Longman, New York (1985)
4. Riloff, E., Wiebe, J.: Learning Extraction Patterns for Subjective Expressions. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-2003). (2003) 105–112
5. Riloff, E., Wiebe, J., Wilson, T.: Learning Subjective Nouns Using Extraction Pattern Bootstrapping. In: Proceedings of the 7th Conference on Natural Language Learning (CoNLL-2003). (2003) 25–32

6. Levin, B.: English Verb Classes and Alternations: A Preliminary Investigation. University of Chicago Press, Chicago (1993)
7. Ballmer, T., Brennenstuhl, W.: Speech Act Classification: A Study in the Lexical Analysis of English Speech Activity Verbs. Springer-Verlag (1981)
8. Baker, C., Fillmore, C., Lowe, J.: The Berkeley FrameNet Project. In: Proceedings of the COLING-ACL. (1998)
9. Hatzivassiloglou, V., McKeown, K.: Predicting the Semantic Orientation of Adjectives. In: Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL-97). (1997) 174–181
10. Wiebe, J.: Learning Subjective Adjectives from Corpora. In: Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000). (2000) 735–740
11. Dave, K., Lawrence, S., Pennock, D.M.: Mining the Peanut Gallery: Opinion Extraction and Semantic Classification of Produce Reviews. In: Proceedings of the 12th International World Wide Web Conference (WWW2003). (2003) Web Proceedings.
12. Wiebe, J., Wilson, T., Bell, M.: Identifying Collocations for Recognizing Opinions. In: Proceedings of the ACL-01 Workshop on Collocation: Computational Extraction, Analysis, and Exploitation. (2001) 24–31
13. Riloff, E.: Automatically Generating Extraction Patterns from Untagged Text. In: Proceedings of the Thirteenth National Conference on Artificial Intelligence, The AAAI Press/MIT Press (1996) 1044–1049
14. Pang, B., Lee, L., Vaithyanathan, S.: Thumbs up? Sentiment Classification Using Machine Learning Techniques. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-2002). (2002) 79–86
15. Yu, H., Hatzivassiloglou, V.: Towards Answering Opinion Questions: Separating Facts from Opinions and Identifying the Polarity of Opinion Sentences. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-2003). (2003) 129–136
16. Nigam, K., Ghani, R.: Analyzing the Effectiveness and Applicability of Co-Training. In: Proceedings of the Ninth International Conference on Information and Knowledge Management. (2000) 86–93
17. Turney, P.: Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews. In: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL-2000). (2002) 417–424
18. Hatzivassiloglou, V., Wiebe, J.: Effects of Adjective Orientation and Gradability on Sentence Subjectivity. In: 18th International Conference on Computational Linguistics (COLING-2000). (2000)
19. Turney, P., Littman, M.: Measuring Praise and Criticism: Inference of Semantic Orientation from Association. ACM Transactions on Information Systems (TOIS) **21** (2003) 315–346
20. Gordon, A., Kazemzadeh, A., Nair, A., Petrova, M.: Recognizing Expressions of Commonsense Psychology in English Text. In: Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL-03). (2003) 208–215
21. Spertus, E.: Smokey: Automatic Recognition of Hostile Messages. In: Proceedings of the Eighth Annual Conference on Innovative Applications of Artificial Intelligence (IAAI-97). (1997) 1058–1065
22. Das, S.R., Chen, M.Y.: Yahoo! for Amazon: Opinion Extraction from Small Talk on the Web. In: Proceedings of the 8th Asia Pacific Finance Association Annual Conference. (2001)

23. Karlgren, J., Cutting, D.: Recognizing Text Genres with Simple Metrics Using Discriminant Analysis. In: Proceedings of the Fifteenth International Conference on Computational Linguistics (COLING-94). (1994) 1071–1075
24. Kessler, B., Nunberg, G., Schütze, H.: Automatic Detection of Text Genre. In: Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL-97). (1997) 32–38
25. Nasukawa, T., Yi, J.: Sentiment Analysis: Capturing Favorability Using Natural Language Processing. In: Proceedings of the 2nd International Conference on Knowledge Capture (K-CAP 2003). (2003)
26. Morinaga, S., Yamanishi, K., Tateishi, K., Fukushima, T.: Mining Product Reputations on the Web. In: Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2002). (2002)
27. Yi, J., Nasukawa, T., Bunescu, R., Niblack, W.: Sentiment Analyzer: Extracting Sentiments about a Given Topic using Natural Language Processing Techniques. In: Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM-2003). (2003)
28. Hearst, M.: Automatic Acquisition of Hyponyms from Large Text Corpora. In: Proc. of the 14th International Conference on Computational Linguistics (COLING-92). (1992)
29. Tong, R.: An Operational System for Detecting and Tracking Opinions in Online Discussions. In: Working Notes of the SIGIR Workshop on Operational Text Classification. (2001) 1–6

Instance Pruning by Filtering Uninformative Words: An Information Extraction Case Study

Alfio Massimiliano Gliozzo, Claudio Giuliano, and Raffaella Rinaldi

ITC-irst, Istituto per la Ricerca Scientifica e Tecnologica,
I-38050 Trento, Italy
{gliozzo, giuliano, rinaldiraf}@itc.it

Abstract. In this paper we present a novel instance pruning technique for Information Extraction (IE). In particular, our technique filters out uninformative words from texts on the basis of the assumption that very frequent words in the language do not provide any specific information about the text in which they appear, therefore their expectation of being (part of) relevant entities is very low. The experiments on two benchmark datasets show that the computation time can be significantly reduced without any significant decrease in the prediction accuracy. We also report an improvement in accuracy for one task.

1 Introduction

Information Extraction (IE) is the task of discovering a set of relevant domain-specific classes of entities and their relations in textual documents. Many of the state-of-the-art IE systems are based on supervised Machine Learning (ML) techniques such as, support vector machines (SVMs) [5], hidden Markov models [8], and boosting [7]. In general, they approach the task as a classification problem, assigning an appropriate classification label for each token in the input documents.

Some of the benchmark datasets used more often in IE are Job Posting, Seminar Announcements, Corporate Acquisition and the University Web Page collection [13]. Moreover, given the recent interest in the field of molecular biology and genetics, new datasets, such as the GENIA corpus [10], have become available. All these datasets have a highly unbalanced distribution of examples: the number of positive examples is sensibly lower than the number of negative ones.

The unbalanced distribution of examples can yield in many ML algorithms (e.g. boosting, SVMs) a drop off in classification accuracy [14]. On the other hand, very large datasets are a problem for supervised learning techniques. In addition, it becomes prohibitive to apply kernel methods designed explicitly for NLP (e.g. Word Sequence Kernels [1], Tree Kernels [3]) due to the high computational complexity of SVMs [11].

As a consequence, reducing the number of instances without degrading the prediction accuracy is a crucial issue for applying ML techniques in IE, especially in the case of highly unbalanced datasets. Furthermore, it would be useful to

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 498–509, 2005.

filter out only negative examples, while preserving the positive examples, since entities to recognize are very sparse in corpora.

In the literature, some approaches have been proposed to overcome either the computational complexity or the problems related to unbalanced distributions in IE tasks. [16] filtered out tokens according to their Part-Of-Speech (POS) tag. [15] defined a more general technique to remove text fragments. In general they suffer from the lack of portability across different domains.

In this paper, we propose a novel methodology for instance pruning in unbalanced datasets. The basic assumption behind this approach is as follows: tokens with a high frequency in the dataset do not provide any specific information about the text in which they appear, therefore their expectation of being (part of) relevant entities is very low. Our technique provides a way for removing uninformative tokens both in the training and the test sets. It can be exploited in any IE task by any supervised algorithm. We have tried our filtering technique uniformly on both the Seminar Announcements (SA) and the GENIA corpus (see subsection 4.1), using the filtered datasets for training and testing a supervised IE system based on SVMs. Methodologically, we adopted a simple common feature set in both tasks. The experiments show a drastic reduction in computation time in both tasks. We also report significant improvements in accuracy for the GENIA task.

The paper is structured as follows. In section 2 our instance pruning technique is proposed, and an evaluation metric for the quality of the filtering strategy is introduced. Section 3 describes SIE, the simple supervised IE system based on SVMs that we used for all the experiments. In section 4 the instance pruning technique is tested on both the SA and the GENIA tasks. Conclusions and future works are finally reported in section 5.

2 Term Filtering for Instance Pruning

Instance pruning techniques try to reduce the dataset size discarding the harmful and superfluous instances without degrading the prediction accuracy. In this section, we propose an instance pruning technique for IE. It is general, and can be exploited by any supervised IE system that casts IE as a classification problem. Our filtering strategy is based on the assumption that uninformative words are not likely to be entities to recognize, being their information content very low[1]. Following this assumption we filter out very frequent words in corpora because they are less likely to be relevant than rare words.

The Zipf's laws [18] also relate the word frequency to its polysemy: ambiguous words, in general, have higher probabilities than the monosemous ones. Those words need very strong contextual constraints to be disambiguated, while in isolation they denote many different concepts. In most of the cases domain-specific entities are not ambiguous. In addition, removing very frequent words

[1] The information content of a word can be measured by estimating its probability from a corpus by the formula $I(w) = -p(w) \log p(w)$.

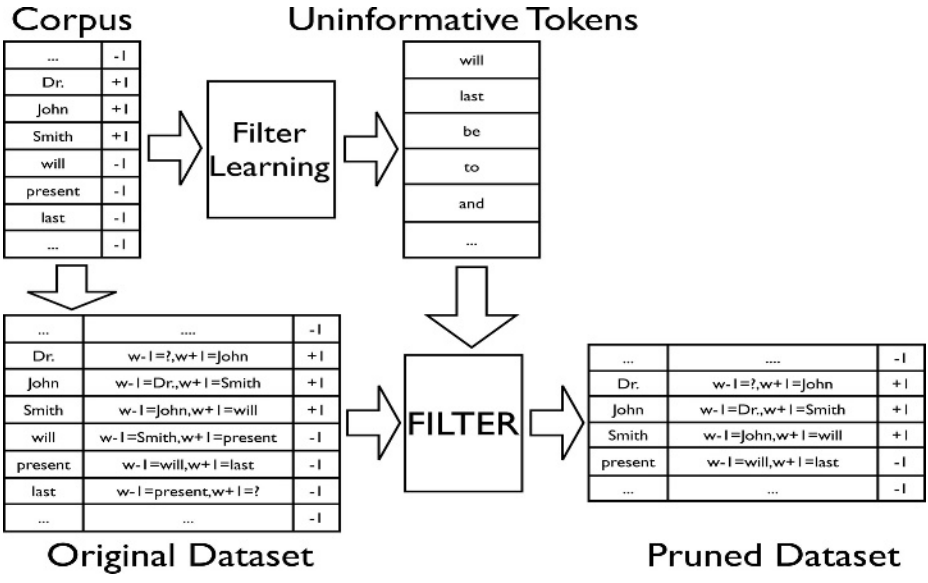


Fig. 1. Computational schema for instance pruning by filtering words

from texts reduce sensibly the dataset size. According to Zipf's Laws, most of the words contained in texts are the top ranked words in the vocabulary[2].

Filtering uninformative words is then a promising direction for instance pruning in IE: removing very frequent words from texts allows to drastically reduce the dataset size, losing just an irrelevant part of the information contained in texts. In the rest of this section, we will describe formally how to implement this intuition.

Figure 1 illustrates the computational schema we adopted for instance pruning by filtering words. The pruning algorithm consists of two stages. First, uninformative tokens are identified by computing statistics on the training corpus. This operation is performed by the FILTER LEARNING module whose aim is to identify a set U of words to remove. The second step is performed by the FILTER module: instances describing "uninformative" tokens (i.e. belonging to the set U) are removed from both the training and the test sets. Note that only *instances* are removed from the dataset, while words in U still appear in the feature description of the remaining instances. For example, the instance corresponding to the word *will* in figure 1 is removed from the dataset, while the word *will* still remains in the feature description for the contexts of the instances describing the contexts of both the word *Smith* and *present*.

We evaluate our instance pruning technique by comparing the total percentage of examples eliminated from the dataset with the percentage of positive examples (wrongly) removed, following the assumption that eliminating nega-

[2] The rank of a word is its position in the vocabulary sorted by increasing frequency.

tive instances without removing positive instances from both training and test sets simplifies the categorization problem. We use the term *filtering rate* (FR) to denote the total percentage of filtered tokens in a corpus, and the term *filtering rate for positives* (FR^+) to denote the percentage of positive tokens (wrongly) removed. Obviously, FR is expected to be maximized while minimizing FR^+ in both the training and test sets.

More formally, let $C = (t_1, t_2, \ldots, t_{|C|})$ be the list of all the tokens contained in the corpus; let $P(t_j)$ be a function returning 1 if t_j is a positive example and 0 otherwise; U be the set of uninformative words selected by the FILTER LEARNING module. The filtering rates are evaluated on the corpus C by equations 1 and 2.

$$FR(U, C) = \frac{|\{t_i | t_i \in U\}|}{|C|} \tag{1}$$

$$FR^+(U, C) = \frac{|\{t_i | t_i \in U \text{ and } P(t_i) = 1\}|}{\sum_{i=1}^{|C|} P(t_i)} \tag{2}$$

The harmonic mean between FR and $(1 - FR^+)$ is exploited to establish an unique criterion to evaluate the overall performances of the filtering strategy, as described in equation 3.

$$FR^*(U, C) = 2\frac{FR(U, C) \cdot (1 - FR^+(U, C))}{FR(U, C) + (1 - FR^+(U, C))} \tag{3}$$

In addition we would expect to preserve the ratio between the number of positive and negative examples in both the training and the test sets (respectively C_L and C_T). This requirement is expressed by equation 4.

$$\frac{FR^+(U, C_L)}{FR^+(U, C_T)} \simeq 1 \text{ and } \frac{FR(U, C_L)}{FR(U, C_T)} \simeq 1 \tag{4}$$

According to the already described framework, a simple strategy for filtering out uninformative words in a corpus is to identify all those words that have the highest probabilities in the training data. More formally, let V_C be the vocabulary of the corpus C and let $|C|$ be the number of tokens in C. Let $OCC(w, C)$ be the number of occurrences of word w in C. The smoothed probability of word w is estimated from C by equation 5.

$$p(w) = \frac{OCC(w, C) + 1}{|C| + |V_C|} \tag{5}$$

The set of uninformative words is then defined by equation 6.

$$U_\theta = \{w | p(w) > \theta \text{ and } w \in V_C\} \tag{6}$$

Even though this method satisfies our assumptions, it is vulnerable to the noise in the data. The entities to be extracted are often composed by more than

one word, and some of them could be very frequent in the corpus. It is well known in Computational Linguistics that ambiguity of words tends to disappear when they are composed in collocations [17]. It is the case of most of the entities to recognize in the IE tasks. For example names of proteins often contain parenthesis, whose frequency in the corpus is very high, names of seminar's speakers sometimes include titles (e.g. "Dr.", "Mr."), and so on. In order to deal with this problem we defined a more complex function performing a "shallow" supervision to identify frequent words that are often marked as positive examples. The basic idea is that words are uninformative when their probability to be negative examples is sensibly higher than their probability to be positive examples.

More formally let $OCC^+(w, C)$ and $OCC^-(w, C)$ be the number of occurrences of the word w respectively in the set of positive and negative examples. The smoothed probability for w to be a positive example is evaluated by formula 7. Similarly formula 8 estimates the probability to be a negative example.

$$p^+(w) = \frac{OCC^+(w, C) + 1}{|C| + |V_C|} \tag{7}$$

$$p^-(w) = \frac{OCC^-(w, C) + 1}{|C| + |V_C|} \tag{8}$$

Equation 9 describes a more robust formulation of the filtering function, designed according to the already introduced idea[3].

$$U_{\alpha,\theta} = \left\{ w \left| ln\frac{p^-(w)}{p^+(w)} - Z_{1-\alpha}\sqrt{\frac{1}{OCC^-(w, C)} + \frac{1}{OCC^-(w, C)}} \geq \Theta \right. \right\} \tag{9}$$

The first addend in formula 9 is a test for the statistical significance (we filter out the word w only if the *odds ratio* between $p^-(w)$ and $p^+(w)$ exceeds a predefined threshold)[4]. In addition we estimated the statistical significance of the test by introducing a one-tailed confidence interval, exploiting the approach introduced by [4]. Using this method, for each desired error probability $0 < \alpha < 1$, we may determine a value θ and state that with a probability of at least $1 - \alpha$ the true value $ln(p^-(w)/p^+(w))$ is greater than θ, where $Z_{1-\alpha}$ is the confidence coefficient, which may be found in statistical tables.

Varying α and θ a set of uninformative word sets $\Psi = \{U_{\alpha,\theta} | \alpha \in [0, 1[\text{ and } \theta \in [0, 10]\}$ is generated. Fixing an upper bound $\epsilon \geq 0$ for FR^+ it is always possible to univocally select the set $U_{\alpha,\theta}$ such that $FR^*(U_{\alpha,\theta}, C)$ is maximized and $FR^+(U_{\alpha,\theta}, C) \leq \epsilon$. This parameter optimization step can be performed by cross validation on the training corpus. Different filtering rates are then obtained varying ϵ, while fixing in advance a reasonable low filtering rate for the positive examples.

[3] If either $OCC^-(w, C) = 0$ or $OCC^+(w, C) = 0$ we used the simplified test $U_{\alpha,\theta} = \left\{ w \left| ln\frac{p^-(w)}{p^+(w)} \geq \Theta \right. \right\}$.

[4] In the literature odds ratio has been used to prune noisy rules in decision lists [4, 9] for Word Sense Disambiguation. Our filtering technique is inspired by those works.

In the rest of the paper the FILTER LEARNING module will adopt equation 9 to select the set of uninformative words, while equation 3 will be used as a maximization criterion for the filtering rate after fixing the upper bound ϵ for the filtering rate for positives.

3 A Simple IE System Based on SVM

We implemented a Simple IE (SIE) system based on SVM for evaluating the impact of our filtering strategy in IE tasks. The SIE system takes in input a dataset in an EOB notation. This notation does not allow nested and overlapping entities. Tokens outside entities are tagged with *O*, while the first token of an entity is tagged with **B-[entity_type]**, and the last token is tagged **E-[entity_type]**, where **entity_type** is the type of the marked entity (e.g. protein, speaker). We approach IE as a classification problem, assigning an appropriate classification label for each token in the dataset. In particular, we identify the *boundaries* that indicate the beginning and the end of each entity as two distinct classification tasks, following the approach introduced in [7, 2]. All tokens that begin(end) an entity are considered positive instances for the begin(end) classifier, while all the remaining tokens are negative instances. All the positive predictions produced by both the begin and the end classifier are then paired. If nested or overlapping entities occur, the entity having the most likely length is selected (the probability that an entity has a given length is estimated from the training set). We solved the multi-class problem training $2n$ binary classifiers (where n is the number of entity_types for the task) and combining them in a one-versus-all classification schema. The predicted class is the one of the most confident classifier.

As input to the begin and end classifiers, we use a bit-vector representation. Each instance is represented encoding all the following basic features for the actual token and for all the tokens in a context window of fixed size.

Token. The actual token.

POS. The Part of Speech of the token. Each token is tagged with its corresponding POS using the TNT tagger[5].

Token Shapes. This feature maps each token onto equivalence classes that encode attributes such as *capitalization*, *numerals*, *single character*, and so on.

4 Evaluation

To test the proposed instance pruning technique we exploited the filtering strategy described in section 2 as a pre-processing mudule for the SIE system (see section 3). As SIE learns two distinct classifiers for the beginning and the end boundaries of each entity, we have pruned the dataset differently for the begin and end classifiers. In subsection 4.2 we will report the filtering rates obtained

[5] The TNT tagger can be downloaded from `http://www.coli.uni-sb.de/~thorsten/tnt/`.

by varying ϵ. It will be demonstrated that the proposed pruning technique decreases drastically the computation time (see subsection 4.3) while preserving or improving the overall accuracy of the system (see subsection 4.4). We tested our technique on two different tasks (see subsection 4.1) using exactly the same system, in order to demonstrate its domain independence.

4.1 The Tasks

To show the domain independence of our filtering strategy we use two benchmark datasets: GENIA and Seminar Announcements.

The *JNLPBA shared task* (GENIA) [12] is an open challenge task proposed at the *"International Joint Workshop on Natural Language Processing in Biomedicine and its Application"*. The dataset consists of 2,000 Medline abstracts of the GENIA corpus version 3.02 [10]. The abstracts were annotated with five entity_types: `protein`, `DNA`, `RNA`, `cell-line`, and `cell-type`. The JNLPBA task splits the GENIA corpus into two partitions: the training partition consists of 492,551 tokens, the test partition consists of 101,039 tokens. The fraction of positive examples with respect to the total number of tokens in the training set varies from about 0.2% to about 6%.

The *Seminar Announcements* (SA) collection [6] consists of 485 electronic bulletin board postings. The purpose of each document in the collection is to announce or relate details of an upcoming talk or seminar. The document were annotated for four entities: `speaker`, `location`, `stime`, and `etime`. The corpus is composed by 156,540 tokens. The fraction of positive examples varies from about 1% to about 2%. We performed a 5-fold cross-validation.

4.2 Filtering Rates

We exploited the filtering strategy described in section 2 to obtain the list of uninformative words from the training set, then we pruned both the training and test sets according to formula 9. To evaluate the filtering rates we selected a predefined set of values for the parameter ϵ (i.e. the upper bound for the filtering rate of positive examples) then we maximized the overall filtering rate by performing 3-fold cross-validation on the training data.

The obtained filtering rates in the training set for each classifier are reported in figure 2. For both tasks our filtering strategy achieved excellent results, allowing to filter out at least the 80% of the instances in the dataset, while loosing less than the 5% of positive examples.

Figure 3 reports the divergences between the filtering rates in the training and test sets achieved using the same filter. Even though the filtering rate for the training set is higher then the one reported for the test set in both tasks, the divergences between the two curves are minimal, satisfying the requirement expressed by equation 4. Our filtering strategy is then robust to overfitting. In the rest of the paper we will report the filtering rates evaluated on the training datasets, assuming that the same proportion is maintained on the test datasets.

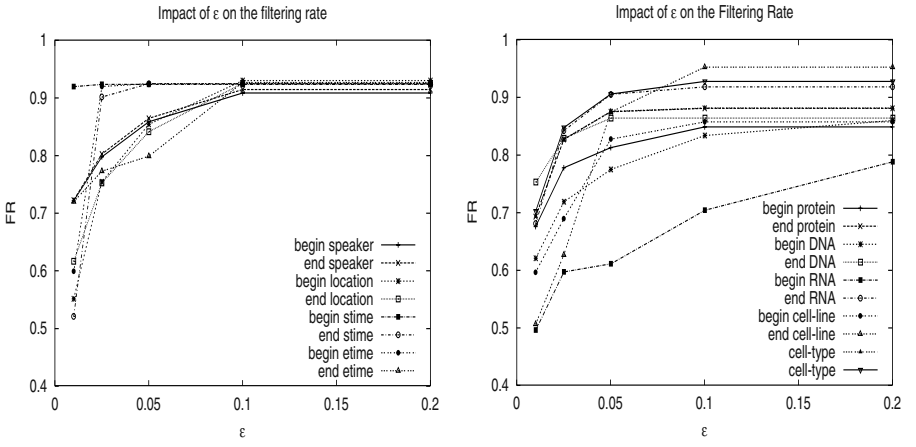


Fig. 2. Filtering rates varying ϵ for SA (left) and GENIA (right) on the training dataset. Filtering rates for begin and end tags for each entity are reported

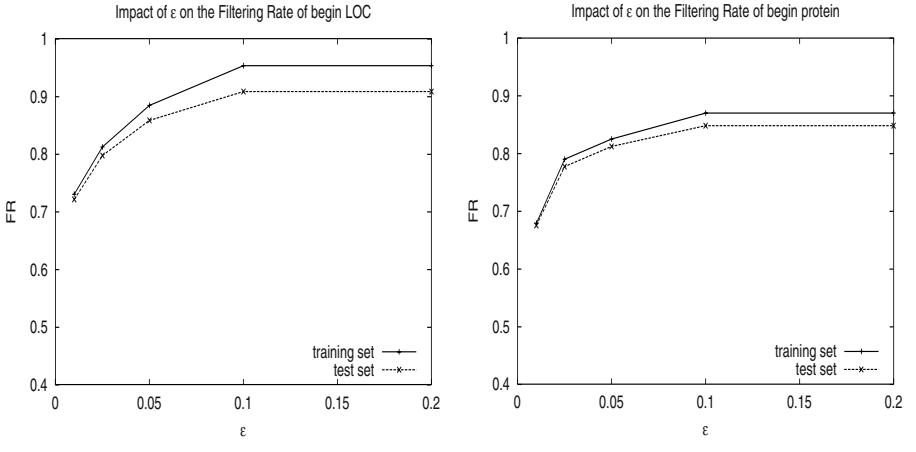


Fig. 3. Comparison among filtering rates on training and test sets varying ϵ for the begin classifier for the entity *speaker* in SA (left) and the begin classifier for the entity *protein* in GENIA (right)

4.3 Computation Time Reduction

Figure 4 reports the impact of instance pruning on the computation time. The graphics show clearly the drastic reduction of the time required to perform the overall process of training and testing the begin and end classifiers for each

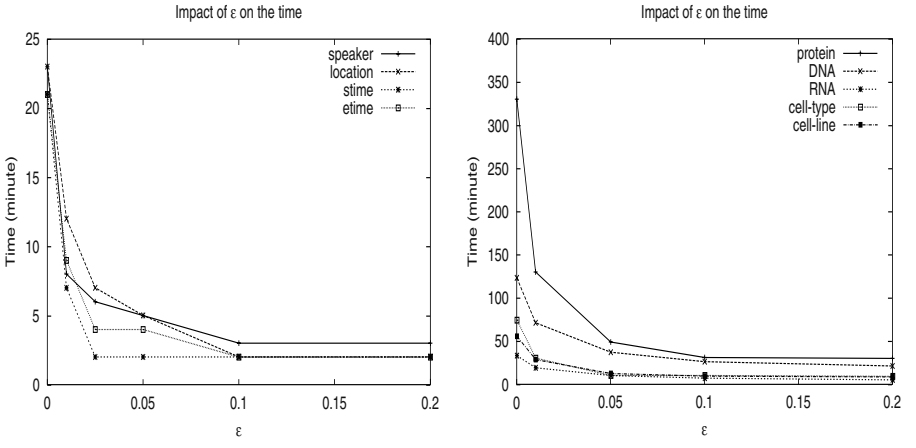


Fig. 4. Computation time required by the SIE system to perform the overall IE process ϵ for SA (left) and GENIA (right)

entity[6]. In particular, for the GENIA task the time requirements are a limitation for supervised approaches. Our strategy is then useful to work and experiment with those data. For example, the time required for the entity *protein* decreased from more than 300 to less then 50 minutes. The same behavior has been found for all the entities, both in SA and in GENIA.

4.4 Accuracy

A fundamental property of our pruning technique is that it does not decrease the overall performance of the system, while it substantially reduces the computation time. Figure 5 reports the variation of F1 for each entity, evaluated according to the methodology introduced above. For all the entities in SA, the values of F1 are stable when the filtering rate increases, while for most of the entities in GENIA F1 measures significantly increase.

Table 1 summarizes the micro averaged performance of the SIE system varing the ϵ parameter[7]. For the GENIA task the F1-micro significantly increases by about 1,5 points. As expected precision increases, while recall is stable. Both for GENIA and SA the computation time required to complete the overall learning and test process was reduced by more than 80%.

In our experiments, SIE has been trained on a basic feature set. Nevertheless, it compares well to the state-of-the-art systems for both the GENIA and SA

[6] All the experiments have been performed using a Macintosh G5 dual processor. For SA we report the time required to perform the overall cycle of 5-fold cross-validation, while for GENIA we trained and tested the system on the official splitting provided by the task organizers.

[7] $\epsilon = 0$ means that the original dataset has not been pruned.

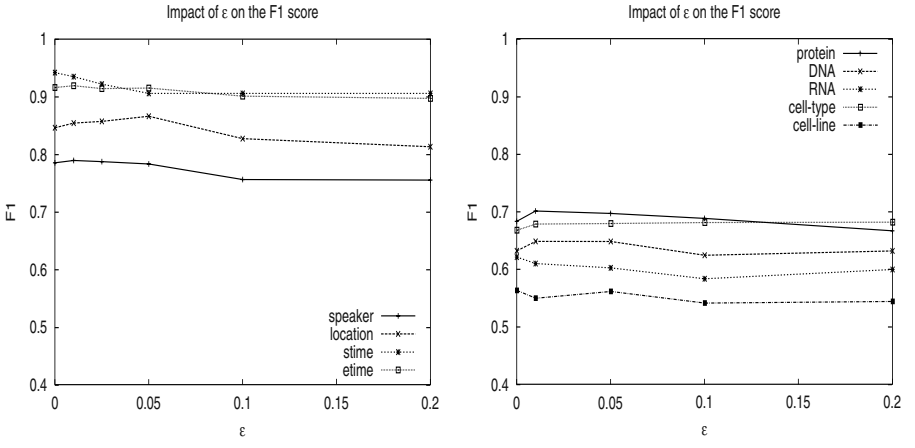


Fig. 5. F1 for each entity-type varying ϵ for the SA and the GENIA

Table 1. Micro averaged Recall, Precision and F1 compared to time reduction varying the filtering rate for GENIA and SA

Task	ϵ	Recall	Precision	F1	Time(m)
GENIA	0	0.664	0.670	0.667	615
GENIA	0.01	0.679	0.684	0.681	278
GENIA	0.05	0.668	0.690	0.679	108
SA	0	0.840	0.912	0.870	88
SA	0.01	0.837	0.915	0.875	36
SA	0.05	0.832	0.910	0.870	16

tasks. For example, according to the results of the JNLPBA shared task[8], SIE is the fourth system. Note that almost all the systems designed to partecipate to the GENIA task adopted external knowledge sources (e.g. gazetteers), ad hoc preprocessing and/or postprocessing steps, while SIE is designed to be domain independent and does not require any external knowledge source.

5 Conclusion and Future Work

In this paper we presented a novel instance pruning technique for IE, based on filtering out uninformative words. The technique is based on linguistic assumptions about the information content of words in the language, and can be applied uniformly to any supervised system. In order to demonstrate the gener-

[8] More details can be found at `http://www-tsujii.is.s.u-tokyo.ac.jp/GENIA/ERtask/report.html`.

ality and the domain independence of our filtering technique, we evaluated it in two different IE tasks.

In most of the cases the filtering technique was able to automatically discard more than the 80% of the instances from the datasets, while maintaining more than the 95% of the positive examples. In addition the technique is robust to overfitting, preserving the filtering rate from the training set to the test set.

To demonstrate the usefulness of the instance pruning technique we implemented a simple supervised IE system based on SVMs with a linear kernel. All the experiments have shown a drastic reduction of the computation time in both tasks. The overall system accuracy augmented significantly for GENIA, while for SA it has been preserved.

The filtering strategy has been developed on the context of a wider research about kernel methods and their applications to NLP (detailed information about the project can be found at `http://tcc.itc.it/research/textec/projects/lsk/`). For the future we plan to exploit more complex kernel functions for pattern recognition (e.g. word sequence kernels and tree kernels). Those techniques have not yet been applied to the IE task due to their high computational cost. In addition, we plan to investigate on the relations between Information Theory and text understanding, in order to develop a wider common framework for information extraction, text summarization, and text indexing.

Acknowledgments

Claudio Giuliano is supported by the IST-Dot.Kom project sponsored by the European Commission (Framework V grant IST-2001-34038). Raffaella Rinaldi is supported by the WebFAQ project, funded by the Provincia Autonoma di Trento. Thanks to Ido Dagan for its help in defining the mathematical details of the filtering formula. Thanks to Carlo Strapparava, Alberto Lavelli and Cesare Furlanello for useful suggestions and comments.

References

1. N. Cancedda, E. Gaussier, C. Goutte, and J. M. Renders. Word sequence kernels. *Journal of Machine Learning Research*, 3:1059–1082, 2003.
2. F. Ciravegna. Learning to tag for information extraction. In F. Ciravegna, R. Basili, and R. Gaizauskas, editors, *Proceedings of the ECAI workshop on Machine Learning for Information Extraction*, Berlin, 2000.
3. A. Culotta and J. Sorensen. Dependency tree kernels for relation extraction. In *42nd Annual Meeting of the Association for Computational Linguistics*, Barcelona, Spain, 2004.
4. I. Dagan and A. Itai. Word sense disambiguation using a second language monolingual corpus. *Computational Linguistics*, 20(4):536–596, 1994.
5. A. Finn and N. Kushmerick. Multi-level boundary classification for information. In *AAAI-04 Workshop on Adaptive Text Extraction and Mining (ATEM-2004)*, San Jose, California, 2004.

6. D. Freitag. *Machine Learning for Information Extraction in Informal Domains.* PhD thesis, Carnegie Mellon University, 1998.
7. D. Freitag and N. Kushmerick. Boosted wrapper induction. In *AAAI/IAAI*, pages 577–583, 2000.
8. D. Freitag and A. McCallum. Information extraction with HMM structures learned by stochastic optimization. In *AAAI/IAAI*, pages 584–589, 2000.
9. A. Gliozzo, C. Strapparava, and I. Dagan. Unsupervised and supervised exploitation of semantic domains in lexical disambiguation. *Computer Speech and Language,18(3)*, pages 275–299, 2004.
10. T. O. J. Kim, Y. Tateishi, and J. Tsujii. Genia corpus - a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19(Suppl.1):180–182, 2003.
11. T. Joachims. Making large-scale support vector machine learning practical. In A. S. B. Schölkopf, C. Burges, editor, *Advances in Kernel Methods: Support Vector Machines.* MIT Press, Cambridge, MA, 1998.
12. J. Kim, T. Ohta, Y. Tsuruoka, Y. Tateisi, and N. Collier. Introduction to the bio-entity recognition task at JNLPBA. In N. Collier, P. Ruch, and A. Nazarenko, editors, *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications (JNLPBA), Geneva, Switzerland*, pages 70–75, August 28–29 2004. held in conjunction with COLING'2004.
13. A. Lavelli, M. Califf, F. Ciravegna, D. Freitag, C. Giuliano, N. Kushmerick, and L. Romano. IE evaluation: Criticisms and recommendations. In *AAAI-04 Workshop on Adaptive Text Extraction and Mining (ATEM-2004)*, San Jose, California, 2004.
14. J. Leskovec and J. Shawe-Taylor. Linear programming boosting for uneven datasets. In T. Fawcett and N. Mishra, editors, *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), August 21-24, 2003, Washington, DC, USA*, pages 456–463. AAI Press, 2003.
15. D. Roth and W. tau Yih. Relational learning via propositional algorithms: An information extraction case study. In *Seventeenth International Joint Conf. on Artificial Intelligence, 2001*, 2001.
16. Y. Song, E. Yi, E. Kim, and G. G. Lee. Posbiotm-ner: A machine learning approach for bio-named entity recognition. In *The 20th International Conference on Computational Linguistics*, 2004.
17. D. Yarowsky. One sense per collocation. In *ARPA Workshop on Human Language Technology*, 1993.
18. G. K. Zipf. *Human Behavior and the Principle of Least Effort.* Addison-Wesley, 1949.

Incremental Information Extraction Using Tree-Based Context Representations

Christian Siefkes

Berlin-Brandenburg Graduate School in Distributed Information Systems*
Database and Information Systems Group, Freie Universität Berlin
Takustr. 9, 14195 Berlin, Germany
christian@siefkes.net

Abstract. The purpose of *information extraction* (IE) is to find desired pieces of information in natural language texts and store them in a form that is suitable for automatic processing. Providing annotated training data to adapt a trainable IE system to a new domain requires a considerable amount of work. To address this, we explore *incremental learning*. Here training documents are annotated sequentially by a user and immediately incorporated into the extraction model. Thus the system can support the user by proposing extractions based on the current extraction model, reducing the workload of the user over time.

We introduce an approach to modeling IE as a token classification task that allows incremental training. To provide sufficient information to the token classifiers, we use rich, tree-based context representations of each token as feature vectors. These representations make use of the heuristically deduced document structure in addition to linguistic and semantic information. We consider the resulting feature vectors as ordered and combine proximate features into more expressive joint features, called "Orthogonal Sparse Bigrams" (OSB). Our results indicate that this setup makes it possible to employ IE in an incremental fashion without a serious performance penalty.

1 Introduction

The purpose of *information extraction* (IE) is to find desired pieces of information in natural language texts and store them in a form that is suitable for automatic querying and processing. IE requires a predefined output representation (*target structure*) and only searches for facts that fit this representation. Simple target structures define just a number of *slots*. Each slot is filled with a string extracted from a text, e.g. a name or a date (*slot filler*).

To adapt an IE system to a new domain, it is necessary to either manually rewrite the rules used in the system (in case of *static* rule-based systems) or to provide annotated training data (in case of *trainable* systems). Manual rewriting of rules is a time-consuming and intricate task that must be done by experts

* This research is supported by the German Research Society (DFG grant no. GRK 316).

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 510–521, 2005.

which are usually hard to get. Providing annotated training data is less costly but still requires a considerable amount of work.

To address this, some approaches use *active learning* [5,14] where the system actively selects texts to be annotated by a user from a pool of unannotated training data. Thus adaptation to a new domain still requires a large amount of raw (unannotated) training data (which are usually cheap), but only a reduced amount of annotated (and thus expensive) training data which are chosen to be especially valuable for building the extraction model, e.g. those texts or fragments whose annotation is least certain.

An alternative setup is *incremental learning*. Here training documents are annotated sequentially by a user and immediately incorporated into the extraction model. Except for the very first document(s), the system can support the user by proposing slot fillers. Thus the work to be done by the user is reduced over time, from largely manual annotation of slot fillers to mere supervision and correction of the system's suggestions.

While incremental learning generally requires more annotated documents than active learning to reach the same level of accuracy (since the system cannot select the most informative samples), the work required by the user for annotating each document is reduced. Also the user keeps control about which documents are processed. Moreover an incremental setup fits better in situations where information is to be extracted from a stream of incoming documents ("text stream management"), for example email messages or newspaper articles.

In an incremental setup, the workflow for processing a document comprises the following steps:

1. Extract text fragments to fill the slots defined by the given target structure.
2. Show the predicted information to a user; ask the user to review the information and to correct any errors and omissions.
3. Adapt the extraction model based on the user's feedback.

In this paper we introduce an approach to IE that allows incremental training. In the following section we explain our approach to modeling IE as a classification task and the classification algorithm we use. We then describe the preprocessing steps our of system and the rich tree-based context representations we generate as input for the classifier. After reporting experimental results, we conclude by discussing related approaches and future work.

2 Classification-Based Extraction

2.1 Fragment Extraction

The extraction of slot fillers can be handled as a token classification task, where each token (typically a word) in a text is classified as the begin of a slot filler

of a certain type (B-*type*), as a continuation of the previously started slot filler, if any (I-*type*), or as not belonging to any slot filler (O).[1]

Thus there are $2n+1$ classes for n slot types. Not all of them are allowed for all tokens—the B-*type* classes and the O class are always allowed, but there is at most one I-*type* class allowed, and only if the preceding token has been classified to be part of a slot filler of the same *type* (B-*type* or I-*type*).

2.2 The Winnow Classification Algorithm

Most refined classification algorithms are unsuitable for incremental training. One exception is the *Winnow* algorithm [11].

We use a variant of Winnow introduced in [16] which is suitable for both binary (two-class) and multi-class (three or more classes) classification. It keeps an n-dimensional weight vector $w^c = (w_1^c, w_2^c, \ldots, w_n^c)$ for each class c, where w_i^c is the weight of the ith feature. The algorithm returns 1 for a class iff the summed weights of all active features (called the score Ω^c) surpass a predefined threshold θ:

$$\Omega^c = \sum_{j=1}^{n_a} w_j^c > \theta.$$

Otherwise ($\Omega^c \leq \theta$) the algorithm returns 0. $n_a \leq n$ is the number of active (present) features in the instance to classify.

The goal of the algorithm is to learn a linear separator over the feature space that returns 1 for the true class of each instance and 0 for all other classes on this instance. The initial weight of each feature is 1.0. The weights of a class are updated whenever the value returned for this class is wrong. If 0 is returned instead of 1, the weights of all active features are increased by multiplying them with a *promotion factor* α, $\alpha > 1$: $w_j^c \leftarrow \alpha \times w_j^c$. If 1 is returned instead of 0, the active weights are multiplied with a *demotion factor* β, $0 < \beta < 1$: $w_j^c \leftarrow \beta \times w_j^c$.

The used threshold is not fixed, but set to the number n_a of features that are active in a given instance: $\theta = n_a$. Thus initial scores are equal to θ since the initial weight of each feature is 1.0.

We use a *thick threshold* for training Winnow (cf. [3,16]). Instances are trained even if the classification was correct if the determined score was near the threshold. Two additional thresholds θ^+ and θ^- with $\theta^- < \theta < \theta^+$ are defined and each instance whose score falls in the range $[\theta^-, \theta^+]$ is considered a mistake. In this way, a large margin classifier will be trained that is more robust when classifying borderline instances.

We use the parameter values recommended in [16], setting the promotion factor $\alpha = 1.23$, the demotion factor $\beta = 0.83$, and the threshold thickness to 5%.[2]

[1] This is the so-called **IOB2** combination strategy. There are other combination strategies for combining single-token classification decisions into slot fillers that can comprise several tokens (cf. Sec. 5); but in preliminary experiments we found this one to perform best.

[2] In either direction, i.e. $\theta^- = 0.95\,\theta$, $\theta^+ = 1.05\,\theta$.

The scores Ω^c generated by Winnow are converted into confidence estimates using the sigmoid function $\sigma^c(\theta, \Omega^c) = 1/(1 + \frac{\theta}{\Omega^c})$. The resulting σ^c values are normalized by dividing them by $\sum \sigma^c$ so they sum up to 1.

While our Winnow variant supports multi-class classification, initial experiments indicated that is advantageous to use multiple binary classifiers in a "one-against-the-rest" setup. We train a separate classifier for each B-*type* and I-*type* class, considering the context representations of all tokens of a class as positive instances for the corresponding classifier and all other token contexts as negative instances. If several classifiers predict their positive class, the most confident classifiers wins.

2.3 Orthogonal Sparse Bigrams (OSB)

Winnow is a linear separator in the Perceptron sense, but by providing a feature space that itself allows conjunction and disjunction, complex non-linear features may be recognized by the composite feature-extractor + Winnow system.

For this purpose, the OSB (orthogonal sparse bigrams) technique introduced in [16] has proved valuable. OSB slides a window of length N over the original feature list. For each window position, joint output features are generated by combining the right-most (newest) input feature with each of the other input features in the window, memorizing the order of the two input features and the distance between them.

Each of these joint features can be mapped to one of the numbers from 1 to $2^N - 1$ with two bits "1" in their binary representations ($2^n + 1$, for $n = 1$ to $N - 1$) where original features at "1" positions are visible while original features at "0" positions are hidden and marked as skipped. Thus $N - 1$ combinations with exactly two input features are produced for each window position. We use OSB with a window length of $N = 5$ as recommended in [16].

With a sequence of five input features, $i1, \ldots, i5$, OSB produces four output features:

$$\begin{array}{rrrr} & & & i4 \quad i5 \\ & & i3 & <skip> \; i5 \\ & i2 & <skip> & <skip> \; i5 \\ i1 & <skip> & <skip> & <skip> \; i5 \end{array}$$

3 Preprocessing and Context Representation

3.1 Preprocessing

Regarded naively, an input text feed to an IE system might appear to be flat data without visible structure; just a sequence of characters. This is a wrong impression—there is structure in any text. At a low level, text can be considered as a sequence of tokens (words, numbers, punctuation). In natural language texts, tokens are arranged in sentences. Several sentences are grouped in paragraphs, which are grouped in sections (which in turn might be grouped in higher-order sections).

In structured text formats the higher-level structure (usually down to the paragraph level) is explicitly coded, but the lower-level structure (sentences; sentence constituents such as verb groups or noun phrases; tokens) must usually be induced.

The native format of our IE system is XML-based; any well-formed XML document is accepted as input. Documents in other formats must be converted to an XML dialect before they can be processed. Currently, converters from SGML-based (legacy) HTML to XHTML (*JTidy* [9]) and from plain text to XHTML (*txt2html* [20]) are integrated into the system—the latter uses heuristics to deduce the text structure from ASCII markup, recognizing section headers, lists and tables, emphasized text etc. Other document formats can be processed by integrating a suitable converter into the system or by converting them to XML or HTML prior to processing.

In a second step, the text is augmented with explicit linguistic information. We use the well-known *TreeTagger* [19] to:

- Divide a text into sentences;[3]
- Split sentences into "chunks" such as verb groups, noun phrases and prepositional phrases;[4]
- Tokenize the input into a sequence of *parts-of-speech* (words, numbers and punctuation) and determine their syntactic categories and normalized base forms.[5]

The output of the tagger is converted to the XML markup mentioned in the footnotes and merged with the explicit markup of the source document. The merging algorithm is described in [15]. After preprocessing, a text is represented as a DOM (Document Object Model) tree. The structure of the DOM tree for a simple HTML document (containing a section heading and several paragraphs) is shown in Fig. 1.

3.2 Tree-Based Context Representation

Typically, the *context window* considered by IE algorithms comprises either the nearest tokens/words (e.g. [2]) or some predefined syntactic elements of the current sentence (e.g. [17]). The hierarchical tree structure obtained by preprocessing yields a more flexible context model: the context of a node contains the nearest nodes around it. The context we consider for each token includes features about:

- The token itself and the POS (part-of-speech) element it is in.
- Up to four preceding and four following siblings[6] of the POS element (neighboring parts-of-speech, but only those within the same sentence chunk).

[3] `sent` element

[4] `const` element with a `type` attribute that identifies the chunk/constituent type

[5] `pos` element with `type` and `normal` attributes

[6] We use the terms *preceding sibling*, *following sibling*, *parent*, and *ancestor* as defined by XPath [21].

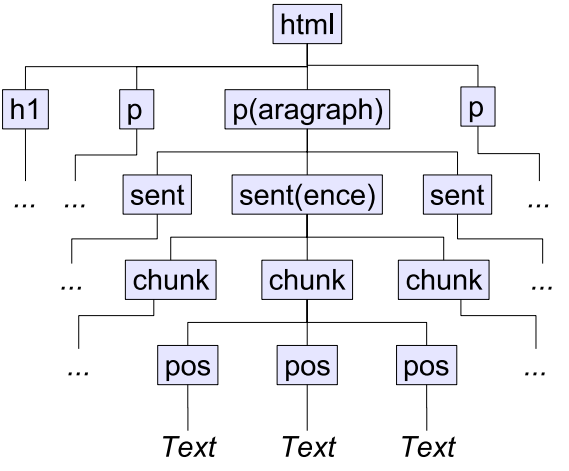


Fig. 1. Partial DOM Tree of a Simple HTML Document with Linguistic Annotations

- Up to four ancestors of the element (typically the embedding chunk, sentence, paragraph or related unit, etc.)
- Preceding and following siblings of each ancestor—the number of included siblings is decremented for each higher level of ancestors (three for the direct parent, i.e. three preceding and three following chunks; two for the "grandparent", i.e. sentence; etc.)

In addition to this DOM tree–based context, we add information on the last four slot fillers found in the current document, similar to the *lastTarget* variable used in [12].

In the DOM tree creating during preprocessing, all leaf nodes are POS elements. Each POS element contains a text fragment for which we include several features:

- The text fragment, both in original capitalization and converted to lowercase;
- Prefixes and suffixes from length 1 to 4, converted to lower-case;[7]
- The length of the fragment;[8]
- The type of the fragment (one of *lowercase, capitalized, all-caps, digits, punctuation, mixed* etc.)

Additionally, the semantic class(es) the fragment belongs to are listed, if any. For this purpose a configurable list of dictionaries and gazetteers are checked. Currently we use the following semantic sources:

[7] Pre-/suffixes that would contain the whole fragment are omitted.
[8] Both the exact value and the rounded square root as a less sparse representation.

- An English dictionary;[9]
- Name lists from US census;[10]
- Address suffix identifiers from US Postal Service;[11]
- A list of titles from Wikipedia.[12]

All other elements are inner nodes which contain child elements, they do not directly contain text. For *chunk* elements, we include the normalized form of the last POS that is not part of a sub-chunk as head word. For elements containing chunks (such as *sentence*), the head words of the left-most and the right-most chunk are included. For other elements, no features are generated, except the name of the element and any attributes stored in the DOM tree[13] which are included for all elements. For the represented POS element and its ancestors, we also store the position of the element within its parent.

Th result is a fairly high number of features representing the context of each token. The features are arranged in an ordered list to allow recombination via OSB (Sec. 2.3); the resulting feature vector is provided as input to the classifier (cf. Sec. 2.2).

4 Experimental Results

We have tested our approach on the *CMU Seminar Announcements*[14] corpus, a standard corpus that is used very often to evaluate IE systems. The corpus contains 485 seminar announcements (plain text files) collected from university newsgroups; the task is to extract up to four fragment slots from each document (if present): *speaker*, *location*, *start time (stime)* and *end time (etime)* of the talk.

We randomly shuffled the order of documents in the corpus and used the first 50% of documents for training and the other 50% for evaluation, averaging results over five random shuffles.[15] As usual for this corpus, we used the "one answer per slot" approach for evaluation (cf. [10]): at most one instance of each slot is to be extracted from each document; if there are several annotated fragments in a document, it is sufficient to find one. If our system finds multiple extraction candidates, it selects the most probably one. Only exact matches are accepted—partial matches are counted as errors.

Extraction results are evaluated in the usual way by counting *true positives tp* (correct slot fillers), *false positives fp* (spurious slot fillers), *false negatives fn*

[9] `http://packages.debian.org/testing/text/wamerican`
[10] `http://www.census.gov/genealogy/names/`
[11] `http://www.usps.com/ncsc/lookups/abbreviations.html`
[12] `http://en.wikipedia.org/wiki/Title`
[13] Type of parts-of-speech and chunks, normalized form of parts-of-speech, etc.
[14] `http://www-2.cs.cmu.edu/~dayne/SeminarAnnouncements/__Source__.html`
[15] The most typical evaluation setup for this corpus; some other systems average over ten shuffles.

Table 1. F1 Results Compared to Other Approaches

Approach	**TIE**		**BWI**	**ELIE**		**HMM (LP)^2**	**MaxEnt**	**MBL**	**SNoW-IE**
	Inc.	**Iter.**		**L1**	**L2**				
Reference			[7]	[6]		[8]	[2]	[1]	[22]	[13]
etime	96.7	97.5	93.9	87.0	96.4	59.5	95.5	94.2	96	96.3
location	79.3	80.6	76.7	84.8	86.5	83.9	75.0	82.6	87	75.2
speaker	80.9	85.2	67.7	84.9	88.5	71.1	77.6	72.6	71	73.8
stime	99.2	99.3	99.6	96.6	98.5	99.1	99.0	99.6	95	99.6
Average	88.3	89.9	83.9	88.8	92.1	81.7	86.0	86.9	86.6	85.3

Table 2. Results With Incremental Feedback

F1	**Evaluation Set**	**All Files**
etime	97.8	94.2
location	80.2	73.2
speaker	83.9	77.0
stime	99.2	98.0
Average	89.5	84.8

(missing slot fillers) and calculating *precision* $P = \frac{tp}{tp+fp}$ and *recall* $R = \frac{tp}{tp+fn}$. *F1 measure* is the harmonic mean of precision and recall:

$$F1 = \frac{2 \times P \times R}{P + R}.$$

To combine the results from all slot types into a single measure, we report the *weighted average* as used in [1] where each slot type is weighted by the total number of expected slot fillers in the corpus (485 *start times*, 464 *locations*, 409 *speakers*, 228 *end times*). All reported performance figures are *F1* percentages.

Table 1 compares our system (called TIE, "Trainable Information Extractor") with other approaches evaluated in the same way.[16] When trained incrementally (first column), our system is better than all other approaches, except one (the ELIE system described in [6]).[17] ELIE uses Support Vector Machines in a two-level approach, while our system so far is limited to a single level. When resigning incrementality and iteratively training our system until accuracy of the token classifiers on the training set stops increasing (second column), our system

[16] One other approach, BIEN [12], is not directly comparable, since it uses an 80/20 split instead of 50/50. When run with an 80/20 split, the overall result of our system (in incremental mode) is 89.5%; BIEN reaches 88.9%.

The reported results are all from *trainable* systems (mainly statistical ones, while some—BWI, (LP)^2—use rule-learning). In the past, domain-specific rule-based systems haven often been able to outperform trainable approaches. However, for this corpus we are not aware of comparable or superior results reached by static, handcrafted systems.

[17] Testing the statistical significance of performance differences is not possible since it would require detailed test results of the other systems which are not available.

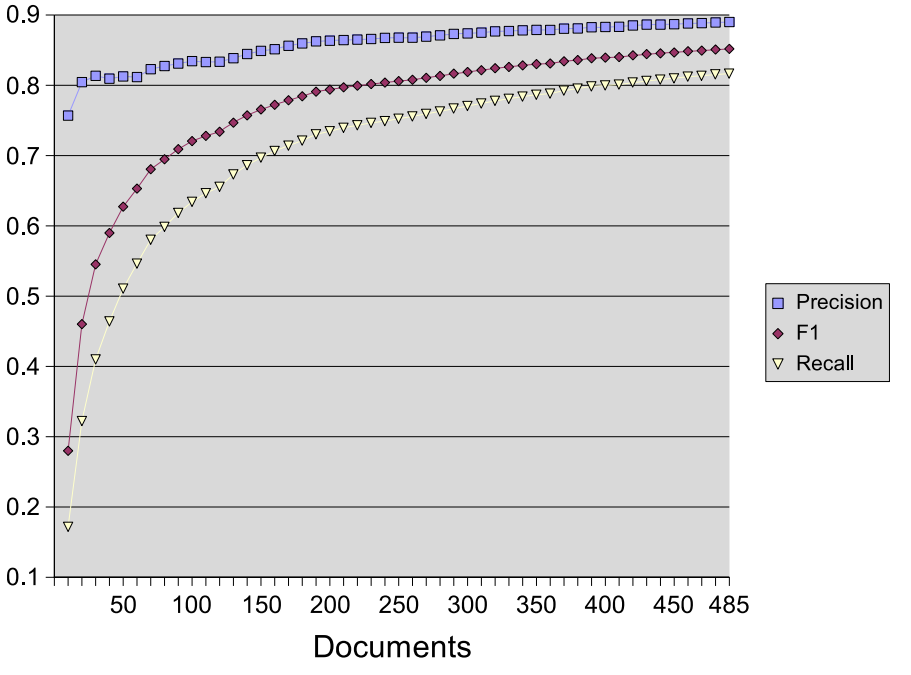


Fig. 2. Incremental Feedback: Learning Curve (precision, recall, and F1 on all documents processed so far)

outperforms their first level by more than 1%. Further improvement should be possible by adding a second level similar to theirs.

In the usual setup, 50% of all documents are used for training and the rest for evaluation (50/50 split). In an incremental approach, it is possible to adapt the extraction model even during the evaluation phase, by allowing the classifier to train the correct slot fillers from each document after evaluating its own proposals for this document.[18] With this feedback added, the F1 measure on the evaluation set increases to 89.5% (Table 2, left column).

With this feedback mechanism it is not strictly necessary to start with a training-only phase; the system can be used to propose slot fillers to be evaluated from the very start, using the whole corpus as evaluation set (0/100 split). Tested in this way, our system still reaches almost 85% F1 over *all documents* (right column). This means the system can be beneficial to use very soon, without requiring a tedious manual annotation phase to provide initial training data.

[18] This corresponds to the workflow from the Introduction where the system proposes slot fillers which are reviewed and corrected by a human supervisor. After the supervisor has corrected a document, the system updates its extraction model prior to processing the next document. In this way the quality of the extraction proposals will continually improve.

Table 3. Ablation Study

F1	Default	No Semantic	No HTML	No Linguistic	No OSB
etime	96.7	97.2	97.0	89.2	95.5
location	79.3	78.0	76.8	68.0	69.3
speaker	80.9	77.0	72.8	53.6	64.9
stime	99.2	99.4	99.4	99.1	98.7
Average	88.3	87.0	85.6	76.8	80.9

Fig. 2 shows the learning curve in this setup. As can be seen, precision is high from the very start—more than 75% after the first 10 documents, more than 80% after 20. Initial recall is far lower, but it exceeds 50% after processing 50 documents and 70% after 160 documents.

Table 3 shows the relative importance of different sources of information. Semantic information is less important than for other systems—without it, F1 drops by only 1.3%, while $(LP)^2$ reports a performance drop by 23% (from 86% to 63.1%); for BIEN it is 11% (from 88.9% to 77.8%). This indicates that our approach makes efficient use of syntactic and linguistic features to compensate for missing explicit semantic data.

More relevant is the heuristic preprocessing step to recognize document structure in the plain text input (*txt2html*, cf. Sec. 3.1). Linguistic annotation (*TreeTagger*) contributes most to the results, not surprisingly. We also find that the OSB feature combination technique (cf. Sec. 2.3) is indeed useful—without it, F1 degrades by 7.4%.

5 Related Work

There are several other approaches modeling IE as a classification task: [1] uses Maximum Entropy modeling with four classes for each slot type (*X*-begin, *X*-continue, *X*-end, *X*-unique). [6] uses two SVM classifiers for each slot type, one for detecting the begin and the other for detecting the end of slot fillers. [22] uses Memory-based Learning (MBL) with the *IOB1* strategy[19].

While there are multiple approaches to statistical IE, most of them use methods that are unsuitable for incremental training. One other approach, SNoW-IE [13], employs the Winnow algorithm, but since it uses several parameters that are determined from the training data it cannot be trained incrementally.[20] We are not aware of any approach that supports incremental training.

[19] Which differs from *IOB2* (Sec. 2.1) in using B-*type* only when necessary to avoid ambiguity; otherwise I-*type* is used even at the beginning of slot fillers.

[20] SNoW-IE realizes a two-step approach. Among a small number of possible candidate fragments identified in a *filtering* stage, the (presumably) correct text fragment is determined and extracted in a *classifying* stage. The complete training data is inspected to determine minimum scores necessary to pass the filtering stage as well as specific conditions fulfilled by all or most positive instances (such as the maximum length of slot fillers).

The employment of a (DOM) tree-based representation of the input documents and the use of heuristics to recognize document structure (*txt2html* conversion) appear to be other novel traits of our approach.

6 Conclusion and Future Work

We have introduced an approach to modeling information extraction as a token classification task that allows incremental updating of the extraction model. To provide sufficient information to the token classifiers, we use rich, tree-based context representations of each token as feature vectors. These representations make use of the heuristically deduced document structure in addition to linguistic and semantic information. We consider the resulting feature vectors as ordered and combine proximate features into more expressive joint features, using the OSB combination technique. Our results indicate that this setup makes it possible to employ IE in an incremental fashion without a serious performance penalty.

There are several directions for future work. We plan to try our system for other tasks and to explore variations of the used context representations in more detail. To augment our current single-level setup, we will add a correction mode that reconsiders misclassified tokens near extraction candidates. We are also experimenting with a sentence filtering step to reduce the number of tokens to be presented to the token classifiers, similar to the approach proposed in [4].

Currently our system is limited to very simple target structures—it handles only *slot filling* (extraction of text fragments). We plan to add support for more complex target structures by extending the system to handle *relationship recognition* (e.g. a **works-for** relation between a *person* and a *company*) and *template unification* (deciding which slot fillers give details about the same complex object, e.g. a seminar). In the used CMU task this isn't necessary because each document contains only one seminar announcement, but in real-life applications there will often be multiple relevant objects per document.

The IE system presented in this paper is available as free software [18].

References

1. H. L. Chieu and H. T. Ng. A maximum entropy approach to information extraction from semi-structured and free text. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI 2002)*, pages 786–791, 2002.
2. F. Ciravegna. $(LP)^2$, an adaptive algorithm for information extraction from Web-related texts. In *Proceedings of the IJCAI-2001 Workshop on Adaptive Text Extraction and Mining*, Seattle, USA, 2001.
3. I. Dagan, Y. Karov, and D. Roth. Mistake-driven learning in text categorization. In C. Cardie and R. Weischedel, editors, *Proceedings of EMNLP-97, 2nd Conference on Empirical Methods in Natural Language Processing*, pages 55–63, Providence, US, 1997. Association for Computational Linguistics.

4. A. De Sitter and W. Daelemans. Information extraction via double classification. In *Proceedings of the International Workshop on Adaptive Text Extraction and Mining (ATEM-2003)*, 2003.
5. A. Finn and N. Kushmerick. Active learning selection strategies for information extraction. In *Proceedings of the International Workshop on Adaptive Text Extraction and Mining*, 2003.
6. A. Finn and N. Kushmerick. Information extraction by convergent boundary classification. In *AAAI-2004 Workshop on Adaptive Text Extraction and Mining*, San Jose, USA, 2004.
7. D. Freitag and N. Kushmerick. Boosted wrapper induction. In *AAAI/IAAI*, pages 577–583, 2000.
8. D. Freitag and A. K. McCallum. Information extraction with HMMs and shrinkage. In *Proceedings of the AAAI-99 Workshop on Machine Learning for Information Extraction*, 1999.
9. JTidy. `http://jtidy.sourceforge.net/`.
10. A. Lavelli, M. Califf, F. Ciravegna, D. Freitag, C. Giuliano, N. Kushmerick, and L. Romano. A critical survey of the methodology for IE evaluation. In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC 2004)*, 2004.
11. N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2:285–318, 1988.
12. L. Peshkin and A. Pfeffer. Bayesian information extraction network. In *IJCAI*, 2003.
13. D. Roth and W.-t. Yih. Relational learning via propositional algorithms: An information extraction case study. In *IJCAI*, 2001.
14. T. Scheffer, S. Wrobel, B. Popov, D. Ognianov, C. Decomain, and S. Hoche. Learning hidden Markov models for information extraction actively from partially labeled text. *Künstliche Intelligenz*, (2), 2002.
15. C. Siefkes. A shallow algorithm for correcting nesting errors and other well-formedness violations in XML-like input. In *Extreme Markup Languages (EML) 2004*, 2004.
16. C. Siefkes, F. Assis, S. Chhabra, and W. S. Yerazunis. Combining Winnow and orthogonal sparse bigrams for incremental spam filtering. In J.-F. Boulicaut, F. Esposito, F. Giannotti, and D. Pedreschi, editors, *Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD 2004)*, volume 3202 of *Lecture Notes in Artificial Intelligence*, pages 410–421. Springer, 2004.
17. S. Soderland, D. Fisher, J. Aseltine, and W. Lehnert. CRYSTAL: Inducing a conceptual dictionary. In *IJCAI*, 1995.
18. Trainable Information Extractor. `http://www.inf.fu-berlin.de/inst/ag-db/software/tie/`.
19. TreeTagger. `http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/`.
20. txt2html. `http://txt2html.sourceforge.net/`.
21. *XML Path Language (XPath) 2.0*, 2004. W3C Working Draft, 29 October 2004.
22. J. Zavrel and W. Daelemans. Feature-rich memory-based classification for shallow NLP and information extraction. In J. Franke, G. Nakhaeizadeh, and I. Renz, editors, *Text Mining, Theoretical Aspects and Applications*, pages 33–54. Springer Physica, 2003.

Learning Information Extraction Rules for Protein Annotation from Unannotated Corpora

Jee-Hyub Kim and Melanie Hilario

Artificial Intelligence Lab, University of Geneva,
CH-1211 Geneva 4, Switzerland
{Jee.Kim, Melanie.Hilario}@cui.unige.ch

Abstract. As the number of published papers on proteins increases rapidly, manual protein annotation for biological sequence databases faces the problem of catching up with the speed of publication. Automated information extraction for protein annotation offers a solution to this problem. Generally, information extraction tasks have relied on the availability of pre-defined templates as well as annotated corpora. However, in many real world applications, it is difficult to fulfill this requirement; only relevant sentences for target domains can be easily collected. At the same time, other resources can be harnessed to compensate for this difficulty: natural language processing provides reliable tools for syntactic text analysis, and in bio-medical domains, there is a large amount of background knowledge available, e.g., in the form of ontologies. In this paper, we present a method for learning information extraction rules without pre-defined templates or labor-intensive pre-annotation by exploiting various types of background knowledge in an inductive logic programming framework.

1 Introduction and Background

With the progress of genome sequencing projects, a large number of gene and protein sequences have been newly found and their functions are being investigated by biological researchers. Such research results are published mostly in the form of scientific papers, and there have been strong needs for storing information on each gene and protein in efficient ways for easier access, amounting to building biological sequence annotation databases. For the moment, many of those databases are constructed by manual annotation, and this approach has the problem of catching up with the speed of daily-published research papers on new genes and proteins. Information Extraction (IE) can provide a solution to this situation in two respects: automating annotation tasks, and structuring annotation output for further data-mining analysis.

In the last ten years, there has been a significant amount of work on IE due in particular to the impetus provided by the Message Understanding Conference (MUC) series. In the MUC problem setting, a team of domain experts and knowledge engineers pre-define precise information needs (i.e., what to extract)

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 522–534, 2005.

in the form of templates and pre-annotate corpora corresponding to these templates. With these pairs of templates and annotated corpora, machine learning systems can learn IE rules to fill up the pre-defined template slots. However, this type of setting is not suitable for some cases where pre-defining templates or gathering annotated corpora is very difficult. For instance, in the broad and complex field of protein research, it is not realistic to expect a few annotators to pre-define all possible templates for extracting protein-related information. As a consequence, it is also difficult to build annotated corpora. On the other hand, annotators can easily collect relevant and non-relevant sentences for some target concepts without pinpointing what to extract. As opposed to the MUC setting, real-world situations require *IE without templates* and *without annotated corpora*.

In *IE without templates*, there is no specification of the types of information (e.g., which entities, which relationships, etc.) to be extracted for a target domain; these have to be learned from a corpus provided by domain experts, with some help of prior knowledge if available. Domain ontologies, or conceptualizations of a given domain, are ideal sources of domain knowledge as they indicate what types of entities and relationships exist in a domain. In bio-medical fields, there exist dozens of ontologies with different and often complementary views on the domain. *IE without annotated corpora* raises a different challenge. Generally, corpus annotations provide guidance on the boundaries of fragments to be extracted from text. In the absence of annotated corpora, boundaries have to be identified with the support of general linguistic knowledge provided by natural language processing tools.

In this paper we present a method for IE in the absence of pre-defined templates and annotated corpora. This approach learns IE rules by using Inductive Logic Programming (ILP) to combine domain knowledge in the form of ontologies and general knowledge in the form of linguistic heuristics.

2 Task Statement and Proposed Solution

In this paper, the main task is to learn IE rules for target relations involving proteins, *given only sentences that have been pre-labeled as relevant or non-relevant.* In the absence of pre-defined templates, the relation is unspecified; thus the task can be formulated as a question *"how is X related to Y?"*, where X is a protein or set of proteins and Y is any protein-related concept (e.g., function, structure, disease, post-translation modification). Throughout this paper, we shall use the term "target concept" to refer to concept Y, with the understanding that learning a target concept Y means more precisely learning a relation between protein(s) X and concept Y.

In the absence of annotated sentences, we *do not know which fragments of sentences should be extracted for which specific information concerning the target concept*. Nevertheless, it is certain that there is at least one fragment containing an information on the target concept in a positive sentence. On the other hand, we are sure that there is no fragment relevant to the target concept in negative

sentences. With this minimal information, *we have to decide what types of information to extract for the target concept and the boundaries of fragments from which to extract this information.*

The basic approach used here is to generate possible candidates of IE rules from a positive sentence, and calculate the occurrences of those candidates in the remaining positive sentences and in all negative sentences respectively. If one candidate IE rule occurs more frequently in the other positive sentences than in all negative sentences, we could consider this candidate IE rule as relevant to the target concept.

In order to learn extraction rules, we should first know what they look like. Based on a survey of IE rule patterns [13], we identified essential and optional components in IE rules. Triggers and boundary information on extracts (elements to be extracted) are essential components while syntactic, semantic or lexical constraints are optional. For instance, in a sentence like 'X is phosphorylated', the trigger alone, *is phosphorylated*, is enough to extract relevant information for the target concept, *phosphorylation*. Overly strong constraints would decrease the coverage of an IE rule, whereas overly weak constraints would reduce the accuracy of the rule. As for semantic-type constraints, different ontologies provide different sets of semantic types, thus also affecting performance. In this paper, we show how these different types of optional constraints can be learned trading off coverage and accuracy, instead of fixing constraint types from the outset.

To generate candidate IE rules, we utilized a shallow-parser, domain ontologies, and linguistic heuristics in the framework of ILP. These design options can be justified as follows. First, a shallow-parser gives us syntactic boundary information on fragments (or chunks) in a sentence, such as noun phrases, verb phrases, etc.; using this information, we can, for instance, select noun phrases as extracts. Second, domain ontologies provide semantic types of noun phrases, and these semantic types can be used as constraints in IE rules. In addition, semantic types provide hints for slot names when generating templates from IE rules. Besides semantic types, we have also utilized head words as constraints because sometimes ontologies are not expressive for certain target concepts and in the bio-medical domain where the average length of terms is quite long (e.g., RNA-binding *domain*), head words can serve as semantic types. Relationships between sentences, IE rules, and templates are shown in Table 1. Third, by using linguistic heuristics for IE rule formation, we can generate candidate rules to reduce the size of the hypothesis space. Finally, ILP provides a framework for bringing background knowledge to bear in the process of learning rules from pre-classified sentences. Algorithm 1 shows all the steps involved in learning IE rules.

3 Task Representation

3.1 Background Knowledge

Background knowledge is essential to our task, since it compensates the lack of pre-defined templates and annotated corpora. We used background knowledge

Table 1. Relations between shallow-parsed sentences, IE rules, and templates

shallow-parsed sentences	IE rules	templates	providers
NP	extract	filler	shallow-parser
verb in VP	trigger	event name	shallow-parser
semantic type of NP	semantic constraint	slot name	ontology
head word of NP	lexical constraint	slot name	shallow-parser

Algorithm 1 The algorithm for learning IE rules

- Given:
  - a collection of sentences relevant and non-relevant to target concept T
  - a domain ontology
  - a set of linguistic heuristics
  - a shallow-parser
- Do:
  1. Parse all the sentences with the shallow-parser.
  2. If possible, tag each noun phrase in the sentences with semantic types in the given ontology.
  3. Keep only the head word in each noun phrase as lexical constraints and lemmatize it.
  4. Formalize the task and learn IE rules in an ILP setting.
  5. Select some IE rules, and build template manually.

in the form of domain-independent linguistic heuristics and domain-specific ontologies.

Linguistic Heuristics for IE Rule Formation: There are some rules on how to construct the essential parts of candidate IE rules from shallow-parsed sentences. For instance, we can consider a rule like 'given a contiguous sequence of NP and VP, use a head verb in VP as a trigger and extract NP', which could be represented in Prolog as '`ep('np_vp',S,A,T) :- has(S,A,'np',_), has(S,B,'vp',T), next(A,B).`'. This type of IE rule can be considered a verb subcategorization frame, and in this paper, we only consider this type of IE rules for specific purposes (e.g., to make it easier to build templates from these IE rules). Some examples of linguistic heuristics are shown in Figure 1(a). Generally speaking, this step can be viewed as first-order feature construction in ILP.

Ontologies: An ontology is a conceptualization of a given domain, and in this task it has two major roles: one is to provide semantic types as constraints in IE rules and the other is to generalize these rules using *is-a relationships* between these semantic types. Some examples of is-a relationships are shown in Figure 1(a). For template building, the user can consider learned semantic types as candidate template slot names. As different ontologies could provide different views on the same corpus, we explored the effects of using different ontologies, GENIA and UMLS. Table 2 shows comparison between GENIA [10] and UMLS[1] [3].

[1] UMLS consists of different vocabulary sources. Among them, we used Gene Ontology, MEDLINE, MeSH, UMLS Metathesaurus, NCBI Taxonomy, NCI Thesaurus, and OMIM.

(a) Background knowledge file

```
% Mode declarations & Type specifications
:- modeh(1,disease(+s)).
:- modeb(*,ep(#type,+s,-c,#trigger)).
:- modeb(1,head(+c,#word)).
:- modeb(1,protein(+c)).
:- modeb(1,cell(+c)).
% Determinations
:- determination(disease/1,ep/4).
:- determination(disease/1,head/2).
:- determination(disease/1,protein/1).
:- determination(disease/1,cell/1).
% Linguistic heuristics
ep('vp_np',S,B,T) :-
has(S,A,'vp',T),has(S,B,'np',_),next(A,B).
ep('np_vp',S,A,T) :-
has(S,A,'np',_),has(S,B,'vp',T),next(A,B).
ep('vp_pp_np',S,C,T) :-
...
% is-a relationships in an ontology
organism(X) :- multi_cell(X).
organism(X) :- virus(X).
nucleic_acid(X) :- 'DNA'(X).
% A shallow-parsed sentence
s(s1).
c(c1_1).
has(s1,c1_1,np,'it').
head(c1_1,'it').
c(c1_2).
has(s1,c1_2,vp,'be_developed').
head(c1_2,'be_developed').
next(c1_1,c1_2).
```

(b) Positive examples file

```
disease(s1).
disease(s2).
```

(c) Negative examples file

```
disease(s3).
...
```

Fig. 1. Excerpts of background file, positive examples file, and negative examples file used to learn target concept 'disease'

Table 2. Comparison between GENIA and UMLS

	# semantic types	# is-a relationships	domain
GENIA	48	around 10	biological reactions
UMLS	135	133	various fields in the bio-medical domain

3.2 Formalization in ILP

Logic is a very appropriate formalism for dealing with natural language. In our case, example sentences, all necessary background knowledge and a final hypothesis, that is a set of IE rules, are all easily represented in logic language. ILP comes naturally to mind as an approach to learning language in logic [7]. Muggleton [12] defines the general ILP learning setting as follows:

$$B \wedge H \models E$$

where B is background knowledge, H is a hypothesis, and E is a set of examples. Here, the goal is to find H given B and E. In our IE task, domain knowledge like ontologies maps to B, labeled sentences to E, and a set of IE rules to H.

To solve our problem, we used Aleph (A Learning Engine for Proposing Hypotheses), an implementation of multiple ILP approaches [16]. Its basic algorithm is shown in Algorithm 2. Aleph is based on Mode Directed Inverse Entailment (MDIE); this means the user has to define candidate predicates for the head and the body of a rule to restrict the hypothesis space. An attractive feature of Aleph is the availability of different search strategies, evaluation functions, etc., for theory construction, allowing users to adapt learning processes and models to their specific problems.

In Aleph, all information should be stored in three files containing background knowledge, positive examples, and negative examples respectively.

Algorithm 2 Aleph's basic algorithm

1. Select example: Select an example to be generalized. If none exist, stop, otherwise proceed to the next step.
2. Build most-specific-clause: Construct the most specific clause that entails the example selected, and is within language restrictions provided.
3. Search: Find a clause more general than the bottom clause.
4. Remove redundant: The clause with the best score is added to the current theory, and all examples made redundant are removed.

- *Background knowledge file* contains background knowledge in the form of Prolog clauses that encode information relevant to the problem. It can also contain any directives understood by the Prolog compiler being used. This file also contains language and search restrictions for Aleph. The most basic among these refer to modes, types and determinations; *mode* declarations declare the mode of call for predicates that can appear in any hypothesized clause, *types* have to be specified for every argument of all predicates to be used in constructing a hypothesis, and *determination* statements declare the predicates that can be used to construct a hypothesis. Other background knowledge (e.g., linguistic heuristics and ontologies mentioned in Section 3.1) can also be included.
- *Positive examples file* contains positive examples of a concept to be learned, and some examples are shown in Figure 1(b).
- *Negative examples file* contains negative examples of a concept to be learned, and some examples are shown in Figure 1(c).

Figure 1(a) shows an excerpt of the background knowledge file used to learn the target concept 'disease'. Here, modeh(1,disease(+s)) means that the predicate disease, which is the target concept, can be used as the head of a rule and the type of its argument should be s (sentence). As another example, modeb(*,ep(#type,+s,-c,#word)) declares that predicate ep can be used in the body of a rule and takes four arguments: type (extraction pattern type, e.g., vp_np), s (sentence), c (chunk), and trigger (trigger). In this mode declaration, a '+' sign prefixes an input variable and a '-' sign an output variable, and a '#' a constant. Based on these declarations, Aleph tries to learn the definition of disease with other predicates like ep, head, protein, and semantic-type predicates.

In ILP systems, background knowledge is typically classified into two categories: background knowledge describing rules in a given domain, and background knowledge related to descriptions of examples. By convention, such facts are grouped with background knowledge. In our task, is-a relationships in ontologies and linguistic heuristics (mentioned in 3.1) belong to the first category whereas shallow-parsed sentences and semantic types of NPs belong to the second. A shallow-parsed sentence is a positive or negative example which has been broken down into a list of chunks with syntactic tags; each chunk can have several properties such as syntactic roles, head words, and semantic types. Table 3 shows the predicates and their descriptions used to represent shallow-parsed sentences and semantic types.

Table 3. Predicates used to represent shallow-parsed sentences and semantic types

predicates	argument types	descriptions
s/1	Sentence (S)	type declaration for a sentence
c/1	Chunk (C)	type declaration for a chunk
has/4	S, C, SyntacticRole, HeadWord	states the relation between a sentence and a chunk
head/2	C, HeadWord	states that a chunk has a headword.
next/2	C, C	states that a chunk follows another chunk.
protein/1	C	states that the semantic type of a chunk is a protein
other semantic-type predicates such as 'protein/1'		

4 IE Rule Induction

4.1 How Does the Generalization of a Sentence Work?

The generalization of a given sentence is the core of IE rule learning; it is performed in two stages:

- Data representation stage: After shallow-parsing, a given sentence appears as a list of syntactic-tagged chunks which can be generalized in several ways: first, by reducing NPs and VPs to their head words; second, by lemmatizing these head words; and third, by replacing the (lemmatized) head words with their semantic types.
- Learning stage: The shallow-parsed and semantic-tagged sentences are described in Prolog using the predicate definitions in Table 3. The next step is to build the most specific-clause from those predicates. In this step, we use linguistic heuristics to build a list of candidate IE rules and is-a relationships to generalize the semantic types of the chunks. From this most specific-clause, we perform general-to-specific search to find the best combination of predicates (ep, head, protein, and other semantic-type predicates) that satisfy certain selection criteria (e.g., coverage, accuracy, etc.). Notice that predicates like head, protein, and other semantic-type predicates cannot be used alone without predicate ep introducing an output variable for a chunk (c-), since in mode declarations, we declared those predicates to require a chunk as input (c+). Here, the best combination of predicates is the generalized IE rule of the given sentence. Finally, we remove examples covered by the learned IE rule.

Figure 2 illustrates the different steps in the generalization of a given sentence.

4.2 Examples of IE Rules and Templates

Table 4 shows some examples of learned IE rules for the target concepts by our learning algorithm. Based on these learned IE rules, we manually built templates based on Table 1. Examples of those templates are shown in Table 5. Using IE rules for building templates is much easier than starting from the original corpora, because IE rules show some statistics on the relevancy of the rules as well as the semantic types the rules can extract.

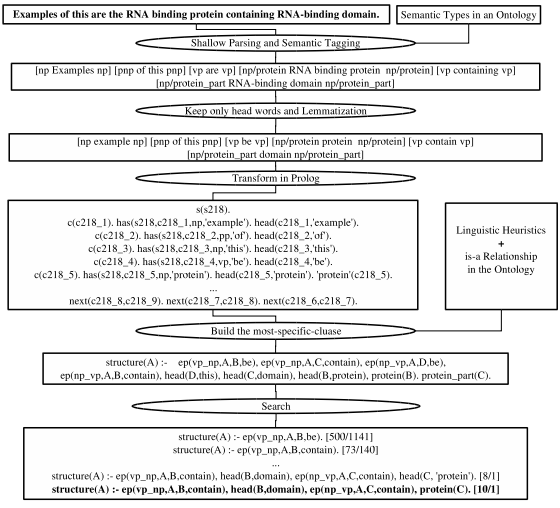


Fig. 2. Illustration of the generalization of a sentence. [x/y] = number of positive examples covered by the rule/number of negative examples covered by the rule

Table 4. Examples of the learned IE rules for protein annotation

disease(A) :- ep(vp_np,A,B,encode), ep(np_vp,A,C,encode), chemical(C). [3/0]
disease(A) :- ep(np_vp,A,B,contain), ep(vp_np,A,C,contain), head(C,tumor). [3/0]
disease(A) :- ep(vp_np,A,B,have), ep(np_vp,A,C,have), pathologic_function(C). [5/1]
disease(A) :- ep(np_vp,A,B,alter), ep(vp_np,A,C,alter), biologic_function(C). [2/0]
function(A) :- ep(np_vp,A,B,regulate), organic_chemical(B). [5/0]
function(A) :- ep(vp_np,A,B,catalyze). [6/0]
function(A) :- ep(np_vp,A,B,stimulate), organic_chemical(B). [6/0]
function(A) :- ep(np_vp,A,B,induce), biologic_function(B). [8/1]
structure(A) :- ep(np_vp,A,B,contain), head(B,peptide). [2/0]
structure(A) :- ep(np_vp,A,B,contain), head(B,sequence). [4/0]
structure(A) :- ep(vp_np,A,B,contain), head(B,motif). [5/0]
structure(A) :- ep(vp_np,A,B,contain), head(B,domain), ep(np_vp,A,C,contain). [8/2]

Table 5. Examples of templates built manually

disease	*event name*: alter *slot names*: protein, pathologic_function
source IE rule:	disease(A) :- ep(vp_np,A,B,have), ep(np_vp,A,C,have), pathologic_function(C).
function	*event name*: regulate *slot names*: protein, organic_chemical
source IE rule:	function(A) :- ep(np_vp,A,B,regulate), organic_chemical(B).
structure	*event name*: contain *slot names*: protein, domain
source IE rule:	structure(A) :- ep(vp_np,A,B,contain), head(B,domain), ep(np_vp,A,C,contain).

5 Experiments

5.1 How to Evaluate Learned IE Rules?

Since we have no pre-defined types of information for the target concepts, we cannot directly evaluate the learned IE rules. Alternatively, borrowing an idea

Table 6. Summary of IE development corpora

target concept	positives	negatives	all	% positives
disease	1087	1927	3014	36%
function	1018	4049	5067	20%
structure	613	1433	2046	30%

Table 7. Various parameters used for the experiments. *search strategies*: bf (breadth first), df (depth first), ibs (iterated beam search) heuristic (best first). *evaluation functions*: evaluates individual rules. *noise*: sets an upper bound on the number of negative examples allowed. *clauselength*: sets an upper bound on the number of literals in a clause

search strategies (ss)	evaluation functions (ef)	noise	clauselength
bf, df	accuracy, coverage, entropy, mestimate	2, 4	3
ibs, ils, heuristic	auto_m, wracc, compression, laplace, gini	6, 8, 10	4, 5

from the Information Retrieval (IR) community where indexing methods are evaluated indirectly within an IR system [11], we tested a set of learned IE rules from a text classification viewpoint. The underlying idea is that if we find a set of relevant IE rules for a target concept, the learned IE rules should be applicable to sentences relevant to the concept but not to non-relevant sentences. In a sense, we are using a set of IE rules as a sentence classifier. In this case, we can use well-established evaluation methods in text classification such as precision, recall, and F-measure. We have chosen F-measure to select a best set of IE rules. If we considered only high precision, the resulting rules would be too specific rules; if we considered only high recall, the rules would be too general. As parsimony is a desirable feature of any learned hypothesis, another useful measure involves the number of rules generated for a target concept. We used a measure called *compress*, defined as $1-(numberOfLearnedRules/numberOfPositiveExamples)$.

5.2 Experimental Conditions

Experiments were conducted using sentence-level-labeled corpora from MEDLINE abstracts provided by the PRINTS database annotators [1]. These are summarized in Table 6. In these corpora, sentences are judged relevant if they relate a (set of) protein(s) to the concept of disease, function, and structure respectively. Sentences were pre-processed using the Memory-Based Shallow Parser

Table 8. Different types of constraints used in IE rules

types	descriptions
def	using only syntactic information (e.g., NP, VP, ...)
head	head words of NPs
genia	semantic types in GENIA
head_genia	head words + semantic types in GENIA
umls	semantic types in UMLS
head_umls	head words + semantic types in UMLS
umls_isa	semantic types in UMLS + is-a relationships
head_umls_isa	head words + semantic types in UMLS + is-a relationships

(MBSP) [6, 4], which is adapted on GENIA corpus and shows 97.6% POS-tagging accuracy and 66.1% concept-tagging accuracy. To learn IE rules and select those which yield the highest F-measure, we explored a variety of parameter settings available in Aleph (summarized in Table 7). One of the issues we examined is how different types of constraints affect the performance of sets of IE rules. Table 8 shows different types of constraints used in the experiments. A training set composed of 80% of each available corpus was used to learn IE rules with the different parameter values and and types of constraints mentioned above. The learned rules were then evaluated using the remaining 20% of each corpus.

6 Results and Discussion

Table 9 shows the best settings for the target concepts. Performance results shown in the table were based on the 20% holdout set. In this section we take a closer look at how the choice of evaluation function and types of constraints used in IE rules impact performance.

Effects of Evaluation Functions for Individual Rules: The evaluation function used has a significant impact on the selection of candidate IE rules.

Table 9. Best parameter settings with highest F-measures

	F-measure	precision	recall	compress	data	ss	ef	noise	clauselength
disease	0.56	0.46	0.72	0.59	head_genia	df	wracc	10	4
function	0.49	0.42	0.59	0.53	head_umls_isa	ibs	laplace	6	5
structure	0.50	0.39	0.69	0.58	head	ils	wracc	10	4

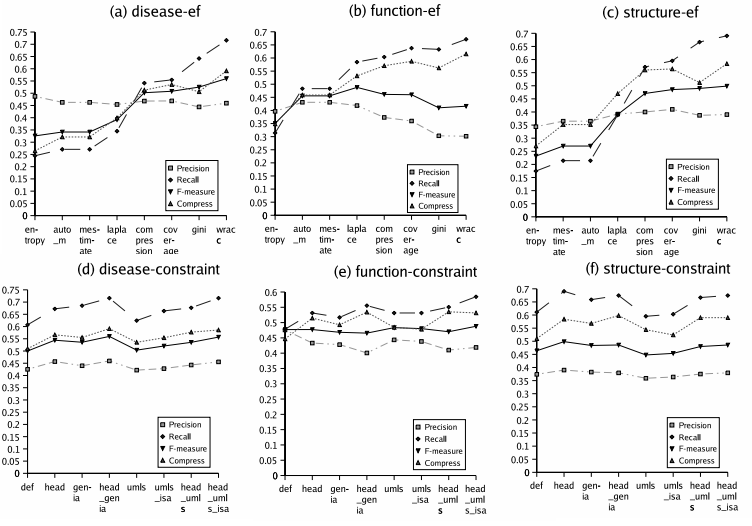


Fig. 3. Results of using different evaluation functions and different types of constraints

Generally, evaluation functions can be divided into three classes: coverage-based measures (coverage), accuracy-based measures (accuracy, laplace, mestimate, etc.), and measures trading off between accuracy and coverage (wracc and compression). For more details on each evaluation function, cf. [8]. Figure 3 shows the results of using different evaluation functions. In particular, wracc (weighted relative accuracy) performs outstandingly on recall without loss of precision. Given a rule $H \leftarrow B$, with H being the head and B the body¸ $wracc(H \leftarrow B) = coverage(H \leftarrow B) * (accuracy(H \leftarrow B) - accuracy(H \leftarrow true))$¸ where the default rule $(H \leftarrow true)$ predicts all instances to satisfy H. The wracc function tends to prefer a slightly inaccurate but very general rule; it also exhibits high *compress*, suggesting it is suitable for our task, building templates based on IE rules.

Effects of different types of constraints: As mentioned before, the idea is to generate IE rules with different types of constraints and to find out which types are suitable for a given target concept. In Figure 3, it is clear that using only syntactic constraints shows the worst performance; on the other hand using semantic types from ontologies and heads shows different results depending on the target concept. Overall, using only head words performs better than using only semantic types; however, the combined use of head words and semantic types performs best except for two of the three target concepts. The exception is the structure corpus, where we found important head words such as domain, motif, loop, etc., which seem to play a more effective role than semantic types. Overall, the use of ontologies reduces the number of IE rules without degrading precision.

7 Related Work

There has been very little previous work on *IE without annotated corpora*. In Riloff's pioneering work [14], pre-labeled corpora and linguistic heuristics were used to generate IE rules and relative frequency was applied to filter out spurious ones. Domain experts then linked selected IE rules to pre-defined templates. With regard to *IE without templates,* Basili noticed drawbacks of the MUC setting for real world domains and applied a terminology extraction method (i.e., using the frequency of a pattern) to a large domain corpus in order to find templates [2]. He did not use positive and negative examples for a target concept, thereby generating some templates which were too general.

Considering IE on the bio-medical literature, there have been work on various topics: subcelluar localization, binding, interaction, inhibition, and other topics. Especially, [15] has addressed the problem of classifying different between diseases and treatments. However all these tasks required *predefined templates* and *preannotated corpora,* whereas our method can be applied to new topics using relevant and non-relevant sentences. There has been some work on the application of ILP to IE [9][5], but this was also based on annotated corpora.

8 Conclusion

IE without pre-defined templates and without annotated corpora is a difficult task which commonly arises in real-world text mining applications. In this paper, we showed how this task can be addressed using only sentences that have been classified as relevant or non-relevant to a target concept. Our method extracts relational knowledge from texts and builds first-order IE rules by combining a shallow parser with background knowledge in the form of ontologies and linguistic heuristics. We also described a method for evaluating learned IE rules and applied this method to the current prototype. Experimental results provide a proof of concept while revealing a large margin for improvement.

Our agenda for future work includes the use of domain knowledge to adapt existing ILP refinement operators and evaluation functions to the needs of protein-related IE. In addition, templates are derived manually from the learned IE rules in the current version of our system. During this handcrafting process, we are exploring ways of improving the generalization process from the initial corpus to the final template. For instance, we noticed that IE rule triggers (verbs) could be generalized using general-purpose background knowledge like WordNet or FrameNet. Such incremental improvements would give rise to more general rules and bring us closer to the goal of automated template construction.

Acknowledgements. The work reported in this paper was partially funded by the European Commission and the Swiss Federal Office for Education and Science in the framework of the BioMinT project (QLRI-2002-02770, 2003-2005).

References

1. T. K. Attwood, P. Bradley, D. R. Flower, A. Gaulton, N. Maudling, A. L. Mitchell, G. Moulton, A. Nordle, K. Paine, P. Taylor, A. Uddin, and C. Zygouri. Prints and its automatic supplement, preprints. *Nucleic Acids Research*, 31(1):400–402, 2003.
2. Roberto Basili, Maria Teresa Pazienza, and Fabio Massimo Zanzotto. Learning IE patterns: a terminology extraction perspective. In *Workshop of Event Modelling for Multilingual Document Linking at LREC 2002*, 2002.
3. Olivier Bodenreider. The Unified Medical Language System (UMLS): integrating biomedical terminology. *Nucleic Acids Research*, 32:D267–D270, 2004.
4. S. Buchholz, J. Veenstra, and W. Daelemans. Cascaded grammatical relation assignment. In *In EMNLP-VLC'99, the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, 1999.
5. Mary Elaine Califf and Raymond J. Mooney. Bottom-up relational learning of pattern matching rules for information extraction. *Journal of Machine Learning Research*, 4:177–210, 2003.
6. Walter Daelemans, Sabine Buchholz, and Jorn Veenstra. Memory-based shallow parsing. In *Proceedings of CoNLL-99*, 1999.
7. Saso Dzeroski, James Cussens, and Suresh Manandhar. An introduction to Inductive Logic Programming and learning language in logic. *Lecture Notes in Artificial Intelligence*, 1925:3–35, 2000.

8. Peter Flach and Nada Lavrac. *Intelligent Data Analysis*, chapter 7 Rule Induction, pages 229–267. Springer, 2002.
9. Markus Junker, Michael Sintek, and Matthias Rinck. Learning for text categorization and information extraction with ILP. *Lecture Notes in Artificial Intelligence*, 1925:247–258, 2000.
10. J.-D. Kim, T. Ohta, Y. Tateisi, and J. Tsujii. GENIA corpus-a semantically annotated corpus for bio-textmining. *Bioinformatics*, 2003.
11. Jee-Hyub Kim, Byung-Kwan Kwak, Seungwoo Lee, Geunbae Lee, and Jong-Hyeok Lee. A corpus-based learning method for compound noun indexing rules for Korean. *Information Retreival*, 4:115–132, 2001.
12. Stephen Muggleton and Luc De Raedt. Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 19(20):629–679, 1994.
13. Ion Muslea. Extraction patterns for information extraction tasks: A survey. In *AAAI-99 Workshop on Machine Learning for Information Extraction*, 1999.
14. Ellen Riloff. Automatically generating extraction patterns from untagged text. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, pages 1044–1049, 1996.
15. Barbara Rosario and Marti A. Hearst. Classifying semantic relations in bioscience texts. In *Proc. ACL*, 2004.
16. Ashwin Srinivasan. The Aleph manual. Technical report, Computing Laboratory, Oxford University, 2000.

Transformation-Based Information Extraction Using Learned Meta-rules

Un Yong Nahm

Ask Jeeves, Inc., 1551 South Washington Avenue,
Suite 400, Piscataway, NJ 08854
pebronia@acm.org

Abstract. *Information extraction* (IE) is a form of shallow text understanding that locates specific pieces of data in natural language documents. Although automated IE systems began to be developed using machine learning techniques recently, the performances of those IE systems still need to be improved. This paper describes an information extraction system based on transformation-based learning, which uses learned meta-rules on patterns for slots. We plan to empirically show these techniques improve the performance of the underlying information extraction system by running experiments on a corpus of IT resumé documents collected from Internet newsgroups.

1 Introduction

The goal of an information extraction system is to fill out the pre-determined template by finding relevant data in natural-language text [1]. In order to reduce human efforts to build an information extraction systems, automatic construction of complex IE systems began to be considered lately. One of the typical problems often found in existing information extraction systems is that the *recall* (percentage of correct slot fillers extracted) of an IE system is significantly lower than its *precision* (percentage of extracted slot fillers which are correct) [2]. In this paper, we present a method to boost the performance of given IE systems, especially recalls, by learning meta-rules by finding relationships between slots and applying these meta-rules on weakly-labeled data repeatedly. For example, we found that a pattern "<degree> in <major>" appears frequently in resumé postings on USENET newsgroups, e.g. "B.S. in Mathematics". This rule can be applied to partially-labeled data, such as "M.S. in <major>" or "<degree> in Mechanical Engineering", to extract additional fillers, e.g. M.S. for degree or Mechanical Engineering for major.

Since meta-rules only require target texts to be tagged and do not assume anything about the tagger, it becomes clear that meta-rules are not restricted to the initial weakly-labeled data tagged by the underlying IE system, but recursively to the data already labeled by meta-rules themselves. This notion of recursiveness resembles that of transformation-based learning which were found to be successful in some natural language processing tasks such as part-of-speech (POS) tagging [3]. The main advantage of our approach is that it is completely independent of the underlying information extraction systems and is therefore very flexible.

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 535–538, 2005.

2 Background and Related Work

In transformation-based error-driven learning, rules acquired as linguistic knowledge are applied to corpus iteratively in order to get accumulatively better results on the target corpus [3]. The general flow for transformation-based learning is described as follows. First, unannotated text is passed to an initial annotator. Second, the initially annotated text is compared with manually annotated text, which is considered to be the gold standard in the given task. Then an ordered set of transformation rules are learned by trying to make the initially annotated text better resemble the manually annotated text. It has been shown that transformation-based learning is one of promising tools in several natural-language processing tasks including POS tagging, prepositional phrase attachment disambiguation, and syntactic parsing [4]. However, while there has been a large amount of research exploring automation of acquiring IE patterns, we are unaware of any existing works on transformation-based IE learners.

3 Transformation-Based Information Extraction

Figure 1 illustrates our approach to learning and applying meta-rules for information extraction. We propose two methods for learning meta-rules specifying relationships between slot values. The first one is to generate decision lists of meta-rules independently from the underlying IE systems (as shown in Figure 1) while the second one is to induce rules by utilizing the existing learners for IE patterns in a recursive manner (IE rule learning and meta-rule learning modules are combined) [5]. In this paper, we will focus on the former approach.

We present an algorithm to discover meta-rules by pattern matching. The input document is a series of N tokens, from $token_1$ to $token_N$. A set of slots to be extracted, S, is given and L is a set of labeled examples. Labeling a document is done for each slot, by producing a list of start positions and end positions, e.g. $(s_1, s_2, ..., s_m)$ and $(e_1, e_2, ..., e_m)$ when there are m labels for that slot. The goal of this algorithm is to generate an ordered list of extraction patterns, P. An extraction pattern consists of two

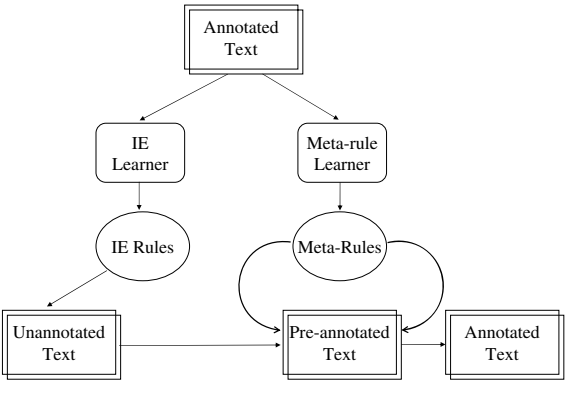


Fig. 1. Transformation-Based Information Extraction

Input Set of slots S and labeled documents L.
Output List of extraction patterns P.
$P := \emptyset$.
For each slot pair $(s_i, s_j) \in S$
  For each document D in L
    For each position where s_i and s_j appears $\in D$
      Let $infix$:= tokens between s_i and s_j.
      If length of $infix$ is less than a predetermined infix length **Then** break.
      Let $prefix$:= tokens preceding s_i of a prefix length.
      Let $postfix$:= tokens succeeding s_j of a postfix length.
      P: = $P \cup (prefix, slot_i, infix, slot_j, postfix)$.
For each pattern $p \in P$ do
  If coverage of p is too low **Then** P: = P - p.
Sort P by coverage.
Return P.

Fig. 2. Algorithm specification: Generating rules by pattern-matching

Table 1. Sample rules discovered by pattern matching algorithm

Rules	Examples
<app>, <app> and <app>.	Flash, Director 5 and AuthorWare.
<degree> in <major>	BS in Mathematics
in writing <lang>, <lang>, and	in writing Java, Javascript, and
<city>: <lang> Programmer /	Austin: VBScript Programmer/Administrator
in either <lang>, <lang> [end-of-line]	in either C, C++ [end-of-line]

slots and prefix, infix, and postfix, when each is a list of tokens. The lengths of prefix, infix, and postfix are predetermined. The extraction pattern can be specified as a 5-tuple of ($prefix$, $slot_1$, $infix$, $slot_2$, $postfix$). Pseudocode for generating rules is shown in Figure 2.

Sample rules discovered from 300 resumé postings from the USENET newsgroup `misc.resume` are shown in Figure 1. In this examples, "app", "degree", "major", and "lang" indicates applications, degree, major, and programming languages slot, respectively. Meta-rules can be used in aiding information extraction by first applying normal information extraction rules and then applying meta-rules iteratively. The iteration of applying meta-rules is repeated until no more updates in extraction labels are observed. Many extraction systems provide relatively high precision, but recall is typically much lower [2]. We expect that our approach possibly improves low recalls of typical information extraction system by pulling out more fillers with extra cues provided by extraction labels.

4 Evaluation

To test the overall system, 300 user-labeled IT resumé postings were collected from the USENET. 10-fold cross validation was used to generate training and test sets. BWI

[6], a learner for simple wrapper-like extraction patterns, is adopted for our base IE learner. BWI is a typical precision-biased information extraction algorithm which favors precisions over recalls. Parameters for BWI are set to the default values. To evaluate our system, we compared the performance of BWI alone and BWI aided with meta-rules learned through the transformation-based algorithm. As hypothesized, Meta-rules provides higher recall of up to 10% of the recall of the original BWI, and although it decreases precision somewhat, overall F-measure is moderately increased. These results demonstrate the advantage of learning and utilizing meta-rules for improving extraction accuracy.

5 Conclusion and Future Work

Information extraction is an emerging field in natural language processing with many useful applications. Typical information extraction has relatively low recalls in comparison with high precisions. This paper presents a framework and initial experiments on improving recalls by using transformation-based learning with learned meta-rules. Learning meta-rules based on relationships among pre-specified slots, extra information about the given text can be obtained and thus to be employed in improving the performances of information extraction task. In this paper, we presented only a generic approach towards information extraction using transformation-based techniques. A number of issues are to be addressed. Currently we have a fixed size of patterns for pre-fillers, in-fillers and post-fillers as well as pre-determined length limit for fillers when we generate and apply meta-rules. Probabilistic information about the length of such patterns could be collected from the corpus and utilized to learn more expressive meta-rules.

Acknowledgements

The author wishes to acknowledge helpful comments from Dr. Raymond J. Mooney at the University of Texas at Austin and anonymous reviewers.

References

1. Muslea, I., ed.: Papers from the AAAI-2004 Workshop on Adaptive Text Extraction and Mining (ATEM-2004) Workshop, San Jose, CA, AAAI Press (2004)
2. DARPA, ed.: Proceedings of the Seventh Message Understanding Evaluation and Conference (MUC-98), Fairfax, VA, Morgan Kaufmann (1998)
3. Brill, E.: Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. Computational Linguistics **21** (1995) 543–565
4. Ramshaw, L.A., Marcus, M.P.: Text chunking using transformation-based learning. In: Proceedings of the Third Workshop on Very Large Corpora. (1995)
5. Nahm, U.Y.: Transformation-based information extraction using recursive rules. Submitted for publication (2004)
6. Freitag, D., Kushmerick, N.: Boosted wrapper induction. In: Proceedings of AAAI-2000, Austin, TX, AAAI Press / The MIT Press (2000) 577–583

A Machine Learning Approach to Information Extraction

Alberto Téllez-Valero[1], Manuel Montes-y-Gómez[1,2], and Luis Villaseñor-Pineda[1]

[1] Language Technologies Group, Computer Science Department, National Institute of Astrophysics, Optics and Electronics (INAOE), Mexico
{albertotellezv, mmontesg, villasen}@inaoep.mx
[2] Department of Information Systems and Computation, Polytechnic University of Valencia, Spain
mmontes@dsic.upv.es

Abstract. Information extraction is concerned with applying natural language processing to automatically extract the essential details from text documents. A great disadvantage of current approaches is their intrinsic dependence to the application domain and the target language. Several machine learning techniques have been applied in order to facilitate the portability of the information extraction systems. This paper describes a general method for building an information extraction system using regular expressions along with supervised learning algorithms. In this method, the extraction decisions are lead by a set of classifiers instead of sophisticated linguistic analyses. The paper also shows a system called *TOPO* that allows to extract the information related with natural disasters from newspaper articles in Spanish language. Experimental results of this system indicate that the proposed method can be a practical solution for building information extraction systems reaching an F-measure as high as 72%.

1 Introduction

The technological advances have brought us the possibility to access large amounts of textual information, either in the Internet or in specialized collections. However, people cannot read and digest this information any faster than before. In order to make it useful, it is often required to put this information in some sort of structured format, for example, in a relational database.

The information extraction (IE) technology is concerned with structuring the relevant information from a text of a given domain. In other words, the goal of an IE system is to find and link the relevant information while ignoring the extraneous and irrelevant one [2]. The research and development in IE have been mainly motivated by the Message Understanding Conferences (MUC[1]). These conferences provide a decade of experience in the definition, design, and evaluation of this task.

According to the MUC community, the generic IE system is a pipeline of components, ranging from preprocessing modules and filters, to linguistic components for

[1] www.itl.nist.gov/iaui/894.02/related_projects/

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 539 – 547, 2005.

syntactic and semantic analysis, and to post-processing modules that construct a final answer [4]. These systems deal with every sentence in the text and try to come up with a full-scale syntactic, semantic, and pragmatic representation. Evidently, they have serious portability limitations since their construction demands a lot of hand-crafted engineering to build the required grammars and knowledge bases.

On the other hand, empiric or corpus based methods are encouraging for the development of IE systems, and in general for many computational linguistics tasks (see [7] for a study). These methods automate the acquisition of knowledge by means of training on an appropriate collection of previously labeled documents. Unlike the traditional approach, they are based on pattern recognition instead of language understanding, and use shallow knowledge instead of deep knowledge. Their main advantages are portability and robustness.

Most current IE systems apply linguistic techniques for text pre-processing and use empiric methods to automatically discover morpho-syntactic extraction rules. This combined scheme produces satisfactory results even when the common errors at the pre-processing stage impose a barrier at the output accuracy. It facilitates the domain portability, but complicates the extensive usage of the IE technologies in other languages than English that lack of robust natural language processing resources.

In this paper we propose a general empiric method for building IE systems. This method avoids using any kind of sophisticated linguistic analysis of texts. It models the IE task as a text classification problem [13]. Basically, it is supported on the hypothesis that the lexical items around the interesting information are enough to learn most extraction patterns. Therefore, the main characteristic of this proposal is its small dependence to the target language.

In order to evaluate this method, we present a system called *TOPO*. This system allows to extract information about natural disasters from news reports in Spanish language. Our results demonstrate that our approximation can be fruitfully used to extract information from free-text documents.

The rest of the paper is organized as follows. Section 2 describes previous work on information extraction using machine learning techniques. Section 3 presents our approach to information extraction based on text classification methods. Section 4 shows a general IE system architecture based on this approach. Section 5 describes a real-world application and shows the results. Finally, section 5 concludes the discussion.

2 Related Work

The use of machine learning (ML) methods in IE applications is mainly focused on the automatic acquisition of the extraction patterns. These patterns are used to extract the information relevant to a particular task from each single document of a given collection (see [9,10,17] for a survey). Current IE approaches, supported on supervised ML techniques, are divided in the following three categories:

Rule Learning. This approach is based on a symbolic inductive learning process. The extraction patterns represent the training examples in terms of attributes and relations between textual elements. Some IE systems use propositional learning (i.e.

zero order logic), for instance, AutoSlog-TS [11] and CRYSTAL [15], while others perform a relational learning (i.e. first order logic), for instance WHISK [16] and SRV [3]. This approach has been used to learn from structured, semi-structured and free-text documents.

Our method is related to the SRV system in that it models the IE task as a classification problem. However, it applies Inductive Logic Programming and uses information about negative examples.

Linear Separators. In this approach the classifiers are learned as sparse networks of linear functions (i.e. linear separators of positive and negative examples). It has been commonly used to extract information from semi-structured documents (see for instance SnoW-IE [12]). It has been applied in problems such as: affiliation identification and citation parsing [1], extraction of data from job ads [18], and detection of an e-mail address change [5].

In general, the IE systems based on this approach present an architecture supported on the hypothesis that looking at the words combinations around the interesting information is enough to learn the required extraction patterns. Their main advantage is that a deep linguistic analysis is not necessary; instead classification techniques are used to find the desired information.

Our method is similar to all these systems. It is based on the same hypothesis. However, it is suited for extracting more general and diverse kinds of information. In some degree our research attempts to empirically determine the limits of this approach when dealing with a complex domain and free texts instead of semi-strcutured documents.

Statistical Learning. This approach is focused on learning Hidden Markov Models (HMMs) as useful knowledge to extract relevant fragments from documents. For instance, [14] presents a method for learning model structure from data in order to extract a set of fields from semi-structured texts. This method is similar to ours in that it considers just the lexical information of texts .

3 Information Extraction as a Classification Problem

Our IE method, like the linear separator approach, is supported on the idea that looking at the words combinations around the interesting information (i.e. the context) is enough to learn the required extraction patterns. Therefore, this method considers two main tasks:

1. Detect all the text segments having some possibility to be part of the output template.
2. Select, from the set of candidate text segments, those that are useful to fill the extraction template.

Figure 1 illustrates this process with a simple example about a hurricane news report. The following subsections describe the purpose and techniques used on each task.

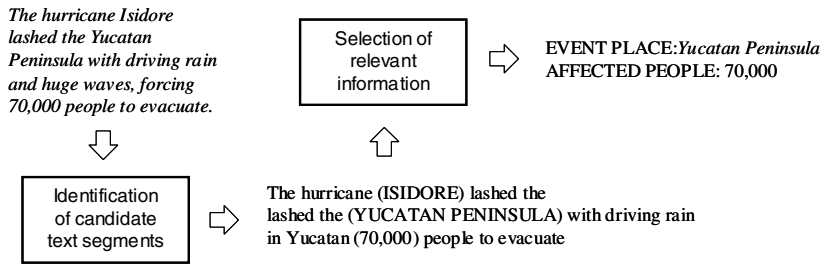


Fig. 1. Information extraction as text classification

3.1 Detecting Candidate Text Segments

The goal of this task is to detect the majority, if not all, of the text segments having some possibility to take place in the extraction template. Since most IE applications consider just the extraction of simple factual data, our method is focused on detecting the text segments expressing names, quantities and temporal data.

In order to identify the candidate text segments we use a *regular expression analysis*. This kind of analysis is general and robust, produces high levels of recall, and is consistent with our purpose of using the less as possible of linguistic resources.

The first part of the figure 1 shows this task. The uppercase words correspond to the candidate text segments of the input text. For each candidate text segment its context (the k neighbor words from left and right) is also extracted.

3.2 Selecting the Relevant Information

The goal of this task is to capture the text segments that must be part of the output template, in other words, it is responsible to classify the text segments into relevant and irrelevant (i.e. to extract or not).

The classification is based on *supervised learning techniques*. In this framework, each candidate text segment is classified according to its lexical context.

In contrast to the previous task, the selection of the relevant information must achieve a high precision rather than a high recall. This situation motivates us to use a pool of learning methods in order to specialize a different classifier for each type of output data. For instance, build a classifier for names, other for dates and another for quantities.

The second part of the figure 1 illustrates this task. There, the classifier uses the contextual information to discard the text segment (ISIDORE) as not relevant to the output template, and also to define (YUCATAN PENINSULA) and (70,000) as the dissaster place and the number of affected people respectivaly.

4 A General IE System Architecture

This section describes a general IE system achitecture based on our approach of "information extraction by text classification" (refer to the section 3). This architecture is shown in the figure 2. It consists of two basic stages: text filtering and information extraction.

It is important to notice that both stages are fully suppported on supervised machine learning algorithms. Moreover, both stages are trainned with the same corpus, and both considers just the lexical information for training.

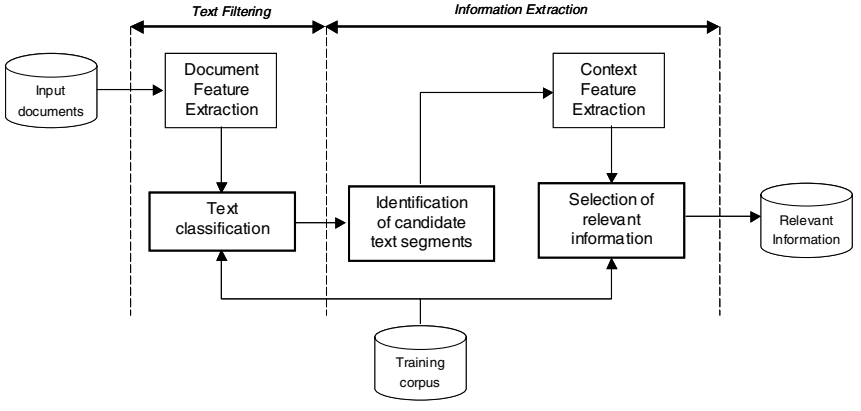


Fig. 2. General IE system architecture

The main characteristic of this architecture is its portability. It is language independent since the training features and the candidate text segments are selected and identified basen on simple lexical patterns and criteria. Also, it can be easily adapted to different domain applications by constructing a small training corpus. Our experiments, refer to the following section, indicates that some hundreds of training examples are enough to reach an acceptable level of output accurancy.

5 A Case Study: Natural Disasters Reports

In this section we introduce the system *TOPO*. A system that extracts information related with natural disasters from newspaper articles in Spanish language. This system was inspired by the work carried out by the network of Social Studios in Disasters Prevention in Latin America [6].

TOPO allows to extract the following information: (i) information related with the disaster itself, i.e. it date, place and magnitude; (ii) information related with the people, for instance, number of dead, wounded, missing, damaged and affected persons; (iii) information related with the buildings, e.g. number of destroyed and affected houses; and (iv) information related with the infrastructure, that is, number of affected hectares, economic lost, among others.

Currently, *TOPO* works with news reports about: hurricanes, forest fires, inundations, droughts, and earthquakes. The following subsections present its technical characteristics and some experimental results.

5.1 Technical Characteristics

Document feature extraction. The documents are represented as boolean vectors indicating the presence or absence of certain words in the texts. The Information Gain

technique was used to avoid a high dimensionality of the feature space. The result was a vector formed by 648 terms obtained from a vocabulary of 26,501 terms from a collection of 534 news reports.

Text classification. We experimented with four different machine learning algorithms [8]: Support Vector Machines (SVM), Naive Bayes (NB), C4.5 and k-Nearest Neighbors (kNN). This selection was based on recent studies that define these classifiers as the best ones for text processing tasks [13].

Candidate text selection. In order to identify the candidate text segments (i.e., names, dates and quantities) from Spanish texts we use the following grammar:

```
Entity_name      →  name |
                    Name connect_name entity_name
Entity_date      →  month |
                    month connect_date number |
                    number connect_date entity_date
Entity_quantity  →  number(. number)? |
                    number(. number)? entity_quantity
```

In this grammar, the terminals symbols generate groups of chains given by the following regular definitions:

```
name          →  [A-Z][A-Za-z]*
connect_name  →  de | la | ... | ε
month         →  enero | ... | diciembre
connect_date  →  de | - | ... | ε
number        →  [0-9]+
```

In addition, we are using a dictionary of names and numbers to treat some grammar exceptions (e.g.: to identify textual quantity expressions and to eliminate words starting with a capital letter but expressing a not valid named entity).

Context feature extraction. This process represent the context of the candidate text segments as a vector of nominal attributes, i.e. the words surrounding the text segments.

In the experiments, we consider context sizes from 1 to 14 terms. In addition, we evaluate several ways of defining this context: (i) using the original surrounding words; (ii) not using stop words as attributes; (iii) using the root of the words; and (iv) using entity tags, i.e., substituting candidate text segments in the context for a tag of name, date or quantity.

Selection of relevant information. It considered the same classifiers used on the text classification task. However, as said elsewhere above, we attempt to specialize each classifier in a different type of output data (i.e., one for the names, other for the dates and another one for the quantities).

5.2 Experimental Results

Text filteirng stage. It was evaluated on a test set of 134 news reports. The evaluation considered the metrics of precision, recall and F-measure[2] adapted to the text classification task [13].

Table 1 resumes the best results we obtained using the SVM algorithm. It is important to mention that these results are equivalent to those reported for similar domains. For instance [13] reports an F-measure from 72% to 88% on the classification of news reports from the Reuters collection.

Table 1. Results for the text filtering task

Disaster	Precision	Recall	F-measure
Forest fire	100	96	98
Hurricane	93	87	90
Inundation	82	93	88
Drought	86	60	71
Earthquake	92	100	96

Information extraction stage. This stage was evaluated on a training set of 1353 text segments –that represent the context of names, dates, and quantities– taken randomly from 365 news reports about natural disasters. Just the 55% of the training examples represent relevant information to be extracted.

In order to evaluate the performance of the information extraction task, we used the precision, recall, and F-measure metrics as defined by the MUC community.

$$precision = \frac{Number\_correct}{Number\_correct + Number\_incorrect + Number\_spurious} \quad (1)$$

$$recall = \frac{Number\_correct}{Number\_correct + Number\_incorrect + Number\_mis\sin g} \quad (2)$$

$$F - measure = \frac{2 \times precision \times recall}{precision + recall} \quad (3)$$

The table 2 resumes the experimental results. This outcome correspond to a context of size eight (i.e., four words to the left and four words to the right) for names and dates text segments, and a context of size six (i.e., three words to the left and three words to the right) for the quantities text segments. The best classifiers were SVM for names and quantities, and kNN for dates.

In general, we obtained a 72% average F-measure for the information extraction task. The precision measures were greater than the recall ones. This indicates that our system is more accurate than complete. We think this situation can be compensated with the redundancies existing in the news reports.

[2] Precision is the proportion of documents placed in the category that are really in the category, and recall is the proportion of documents in the category that are actually placed in the category. The F-measure is a lineal combination of both proportions.

Table 2. Results for the information extraction task

Information	Precision	Recall	F-measure
Disaster date	95	95	95
Disaster place	42	81	55
Disaster magnitude	75	89	82
People dead	65	91	76
People wounded	89	86	88
People missing	79	73	76
People damaged	72	64	68
People affected	50	51	50
Houses destroyed	59	82	69
Houses affected	63	37	47
Hectares affected	66	96	78
Economic lost	80	76	78

These results are equivalent to those reported for similar IE applications. For instance, at MUC-6, where were analyzed news about managerial successions, the participants obtains F-measures lower than 94% for the entity recognition task and measures lower than 80% for the template filling (information extration task).

Finally, it is important to mention that *TOPO* is currently being used for automatically populating a database of natural disasters from Mexican news reports. The system was implemented in Java using the Weka open source software.

6 Conclusions

This paper presents a general approach for building an IE system. This approach is supported on the idea that looking at the word combinations around the relevant text segments is sufficient enough to learn to discriminate between relevant and irrelevant information.

In the proposed approach the information extraction is done by a combination of regular expressions and text classifiers. The use of these methods allows to easily adapt an IE application to a new domain. In addition, it avoids the employment of any kind of sophisticated linguistic recourse, which defines this approach as language independent.

Our experiments demonstrated the potential of this approach. Using a very small training set we reached an average F-meausre of 72% for the extraction task.

The main disadvantages of the proposed approach are: on the one hand, that it is not possible to extract the information expressed in an implicit way. On the other hand, that it is complicated to extract and link the information from documents reporting more than one interesting event. We believe that these problems can be partially solved using some level of linguistic analysis as a preprocessing stage, just before applying the regular expression analysis.

Acknowledgements. We would like to thank CONACyT for partially supporting these work under grants 171610, 43990A-1 and U39957-Y, and to the Secretaría de Estado de Educación y Universidades de España.

References

1. Bouckaert, R.: Low level information extraction. In Proceedings of the workshop on Text Learning (TextML-2002), Sydney, Auxtralia (2002)
2. Cowie, J., Lehnert, W.: Information Extraction. Communications of the ACM, Vol. 39, No. 1 (1996) 80-91
3. Freitag, D.: Machine Learning for Information Extraction in Informal Domains. Ph.d. thesis, Computer Science Department, Carnegie Mellon University, (1998)
4. Hobbs, J. R.: The Generic Information Extraction System. In proceedings of the Fifth Message Understanding Conference (1993)
5. Kushmerick, N., Johnston, E., McGuinness, S.: Information Extraction by Text Classification. Seventeenth International Join Conference on Artificial Intelligence (IJCAI-2001), N. Kushmerick Ed. Adaptive Text Extraction and Mining (Working Notes), Seattle, Washington (2001) 44-50
6. LA RED: Guía Metodológica de DesInventar. OSSO/ITDG, Lima (2003)
7. 7. Manning, C., Schütze, H.: Foundations of Statistical Natural Language Processing. MIT Press (1999)
8. Michel, T.: Machine Learning. McGraw Hill, (1997)
9. Muslea, I.: Extraction Patterns for Information Extractions Tasks: A Survey. In Proceedings of the AAAI Workshop on Machine Learning for Information Extraction (1999)
10. Peng, F.: Models Development in IE Tasks - A survey. CS685 (Intelligent Computer Interface) course project, Computer Science Department, University of Waterloo (1999)
11. Riloff. E.: Automatically Generating Extraction Patterns from untagged text. In proceedings of the 13th National Conference on Artificial Intelligence (AAAI), (1996) 1044-1049
12. Roth, D., Yih, W.: Relational Learning Via Propositional Algorithms: An Information Extraction Case Study. In Proceedings of the 15th International Conference on Artificial Intelligence (IJCAI), (2001)
13. Sebastiani, F.: Machine Learning in Automated Text Categorization: a Survey. Technical Report IEI-B4-31-1999, Istituto di Elaborazione dell'Informazione (1999)
14. Seymore, K., McCallum, A., Rosenfeld, R.: Learning Hidden Markov Model structure for Information Extraction. In Proceedings of the 20th National Conference on Artificial Intelligence (AAAI), (1999).
15. Sonderland, S., Fisher, D., Aseltine, J., Lehnert, W.: CRYSTAL: Inducing a Conceptual Dictionary. In Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI), (1995) 1314-1321
16. Sonderland, S.: Learning Information Extraction Rules for Semi-Structured and Free Text. Machine Learning, No. 34 (1999) 233-272
17. Turno, J.: Information Extraction, Multilinguality and Portability. Revista Iberoamericana de Inteligencia Artificial. No. 22 (2003) 57-78
18. Zavrel, J., Berck, P., Lavrijssen, W.: Information Extraction by Text Classification: Corpus Mining for Features. In Proceedings of the workshop Information Extraction meets Corpus Linguistics, Athens, Greece (2000).

Automatic Time Expression Labeling for English and Chinese Text

Kadri Hacioglu, Ying Chen, and Benjamin Douglas

Center for Spoken Language Research,
University of Colorado at Boulder,
Boulder, Colorado 80309

Abstract. In this paper, we describe systems for automatic labeling of time expressions occurring in English and Chinese text as specified in the ACE Temporal Expression Recognition and Normalization (TERN) task. We cast the chunking of text into time expressions as a tagging problem using a bracketed representation at token level, which takes into account embedded constructs. We adopted a left-to-right, token-by-token, discriminative, deterministic classification scheme to determine the tags for each token. A number of features are created from a predefined context centered at each token and augmented with decisions from a rule-based time expression tagger and/or a statistical time expression tagger trained on different type of text data, assuming they provide complementary information. We trained one-versus-all multi-class classifiers using support vector machines. We participated in the TERN 2004 recognition task and achieved competitive results.

1 Introduction

Extraction of temporal expressions from an input text is considered a very important step in several natural language processing tasks; namely, information extraction, question answering (QA), summarization etc. ([Mani 2004]). For example, in the summarization task, temporal expressions can be used to establish a time line for all events mentioned in multiple documents for a coherent summarization. Recently, there has been growing interest in addressing temporal questions in QA systems ([Schilder and Habel 2003]; [Saquete et. al. 2004]). In those systems, a highly accurate temporal expression recognizer or tagger (statistical or rule-based) is required for effective treatment of temporal questions yielding high-quality end-to-end system performance.

An official evaluation, sponsored by the DARPA automatic content extraction (ACE) program, has been organized by MITRE and NIST for time expression recognition and normalization (TERN) in 2004. The TERN task requires the recognition of a broad range of temporal expressions in the text and normalization of those expressions according to ([Ferro et. al. 2004]). The annotation is intended to mark information in the source text that mentions *when* something happened, *how long* something lasted, or *how often* something occurs. Temporal

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 548–559, 2005.

Table 1. Word and document statistics of TERN corpus. Numbers in parentheses indicate the total number of documents

	Development	Test
English	265K (767)	55K (192)
Chinese	147K (466)	67K (256)

expressions in text vary from explicit references, e.g. *June 1, 1995*, to implicit references, e.g. *last summer*, to durations, e.g. *four years*, to sets, e.g. *every month*, and to event-anchored expressions, e.g. *a year after the earthquake*.

We participated in the recognition task for both English and Chinese text. Here, we adopt an end-to-end statistical approach by identifying three sub-tasks; (i) data representation, (ii) feature engineering and (iii) model learning. The original XML representation of time expressions is changed into a bracketed representation by assigning tags to each token. The bracketed representation is chosen to account for embedded structures in the time expressions. Tags indicate whether a token is inside a time expression, it begins a time expression, it ends a time expression, or it is outside a time expression. Several lexical, syntactic and semantic features are chosen based on intuition, experience and data analysis. One-versus-all classifiers ([Allwein et. al 2000]) are trained using support vector machines (SVMs) ([Vapnik 1995]; [Burges 1998]) and all system settings (e.g. polynomial degree, regularization parameter and context window) are optimized using a held-out data set. Our labeling (or tagging) scheme being language independent yields almost identical systems for both English and Chinese languages. We test the final systems on the official evaluation data set and report competitive results.

The paper is organized as follows. In the next section, we describe the TERN corpus. The English system is described in Section 3. Section 4 introduces the Chinese system. Experimental set-up and results are presented in Section 5. Section 6 concludes the paper with some remarks and possible future work.

2 Description of Data

The TERN corpus is composed of text selected from broadcast news programs, newspapers and newswire reports. It is available in two different sets for both Chinese and English languages; one set is for system training/development and the other set is for evaluation. Table 1 summarizes word and document statistics of the TERN corpus.

3 English System

In this section we describe the TERN system developed for English text. We cast the time expression extraction as a supervised tagging problem. We illustrate this for the sentence *That's 30 percent more than the same period a year ago.*,

which contains an embedded time expression. This text appears in the training set and it is tagged with the xml-style TIMEX2 tags in the following way:

> That's 30 percent more than <TIMEX2> the same period <TIMEX2> a year ago </TIMEX2> </TIMEX2>.

The sentence is converted to a vertical token-level representation by using bracketed representation to incorporate the embedded structure as illustrated below:

That	O
's	O
30	O
percent	O
more	O
than	O
the	(*
same	*
period	*
a	(*
year	*
ago	*))
.	O

The time expressions in the example sentence are enclosed between the brackets. Each word is assigned a tag depending on its position with respect to the time expressions in the sentence. In the example, "O" indicates an outside word (or token), "(*" indicates the beginning of a time expression, "*" indicates a word inside a time expression and "*))" indicates a word that ends the embedded time expression. This representation requires a sentence segmenter and a tokenizer for a given raw document that contains several sentences. Therefore, the TERN data is first segmented into sentences and next tokenized using MXTERMINATOR[1] ([Reynar and Ratnaparkhi 1997]) and a slightly modified version of the Penn Tree Bank tokenizer [2], respectively. Finally, the original TIMEX2 tags are converted into the bracketed representations illustrated earlier. In doing so, the TERN training data is organized as one-token per line with sentences separated by blank lines. This data is passed to other modules for the creation of token specific features.

We define a number of features for each token. Our features can be grouped into four broad classes; lexical, syntactic, semantic and external features. The lexical features are the token itself, its lower-case version, its part of speech tag, a set of features that indicates a specific token pattern (e.g. is hyphenated or not, is a number etc.) and its frequency (e.g. Rare/Frequent/Unknown) with respect to a lexicon (with counts) created from the training data. The syntactic

[1] `http://www.cis.upenn.edu/~adwait/statnlp.html`
[2] `www.cis.upenn.edu/~treebank/tokenization.html`

features that we extract are base phrase chunks ([Ramhsaw and Marcus 1995]; [Kudo and Matsumato 2000]) represented using IOB2 tags as considered in the paper ([Sang and Veenstra 1999]). The head words and dependency relations between the tokens and their respective heads are considered as semantic features. We use a part-of-speech (POS) tagger, trained in-house, to determine the POS tag for each word. This tagger is based on the Yamcha SVM toolkit [3] and trained on a relatively large portion of the Penn TreeBank. Similarly, the base phrase chunks are obtained using an in-house trained SVM-based chunker. The dependency features are assembled from the output of Minipar[4] ([Lin 1998]), a rule-based dependency parser. In addition to those features, we use external features as the decisions from a rule-based time expression tagger (distributed by TERN organizers [5] which covers many of the types of time expressions contained in the TIMEX2 2001 guidelines) and BBN IdentiFinder ([Bikel et. al 1999]). Summarizing, the following features are used within a predefined, finite-size sliding window:

- tokens
- lower-cased tokens
- POS tags
- token pattern flags
- token frequency
- base phrase chunks
- head words
- dependency relations
- rule-based time expression tags
- BBN-Identifinder time expression tags
- previous time expression decisions

A total of 10 one-versus-all SVM classifiers were trained using a polynomial kernel of degree 2. The regularization parameter of SVMs was set to C=1.0. The class labels are illustrated below:

((*, ((*), (*, (*), (*)), *, *), *)), *))), O

The general architecture of the English system is shown in Figure 1. After the SVM classification we employ a simple post-processing algorithm to maintain the consistency of bracketing that might have been violated due to tagging errors. In the following, we summarize the steps taken in the system for the extraction of time expressions:

Step 1. Sentence segmentation
Step 2. Tokenization
Step 3. Pattern and frequency checking

[3] http://cl.aist-nara.ac.jp/taku-ku/software/TinySVM
http://cl.aist-nara.ac.jp/taku-ku/software/yamcha
[4] http://www.cs.ualberta.ca/~lindek/downloads.htm
[5] http://timex2.mitre.org/taggers/timex2_taggers.html

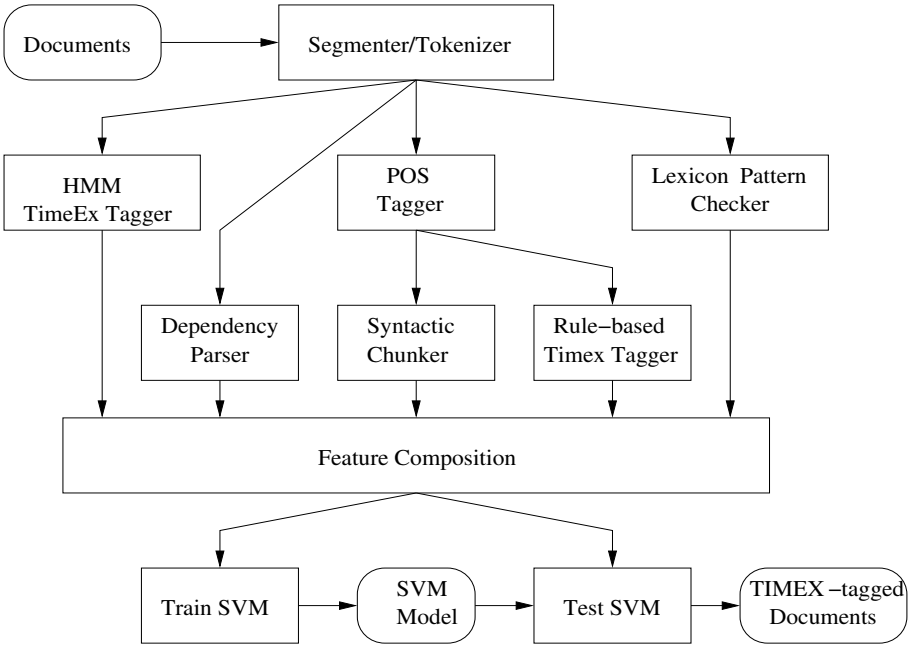


Fig. 1. General system architecture for English automatic time expression labeler (see text for details)

Step 4. Dependency Parsing
Step 5. Third-party time tagging (statistical, HMM)
Step 6. POS tagging
Step 7. Base phrase chunking
Step 8. Third-party time tagging (rule-based)
Step 9. Feature composition
Step 10. Multi-class SVM classification
Step 11. Post-processing

4 Chinese System

In this section we describe the TERN system developed for Chinese text. The same approach outlined in the preceding section is followed to obtain a word-level representation using the bracketing scheme. As mentioned earlier, this representation requires a sentence segmenter and a tokenizer for a given raw document that contains several sentences. Similar to the English system, the Chinese TERN data is also segmented into sentences and tokenized into words. We train a sentence segmenter and tokenizer for Chinese using the Chinese Tree Bank (CTB). The performance of the sentence segmenter is in the high 90's while the performance of the tokenizer is in the mid 90's.

We define a number of features for each token; the token itself, its part of speech, n-gram suffixes and prefixes, pattern flags and the time tag from a rule-based (hand generated) Chinese time expression tagger that we have developed. We used a part-of-speech (POS) tagger, trained in-house, to determine the POS tag for each word. This tagger is based on the Yamcha SVM toolkit and trained on the CTB. The token pattern flags are determined by looking up a number of hand-generated character lists that indicate months, dates, numbers, ordinals and foreign names. In addition to those features we also use a number of previous tag decisions as features. Summarizing, the following features are used within a predefined, finite-size sliding window:

- tokens
- POS tags
- n-gram suffixes and prefixes
- token pattern flags
- rule-based time expression tags
- previous time expression decisions

We note that the feature set for the Chinese system is not as rich as the English system due to lack of NLP resources for Chinese. However, with the availability of the CTB this is changing. We have been developing a syntactic chunker and a syntactic parser for Chinese. However, at the time of evaluation, we were not able to successfully incorporate some features extracted using those systems.

A total of 8 one-versus-all SVM classifiers were trained using a polynomial kernel of degree 2. The regularization parameter of SVMs was set to C=1.0. The class labels are illustrated below:

((*, ((*), (*, (*), (*)), *, *), O

The system architecture is shown in Figure 2. It is very similar to the English system with a few components missing. As in the English system, after the classification we employ a simple post-processing algorithm to maintain the consistency of bracketing. However, we observe that the bracketing after detection was perfectly consistent. This is in contrast with our English system, which has relatively more complex time expressions than Chinese. In the following we summarize the steps taken in the Chinese system for the extraction of time expressions:

Step 1. Sentence segmentation
Step 2. Word segmentation (tokenization)
Step 3. POS tagging
Step 4. Rule-based time tagging
Step 5. Feature composition
Step 6. Multi-class SVM classification
Step 7. Post-processing

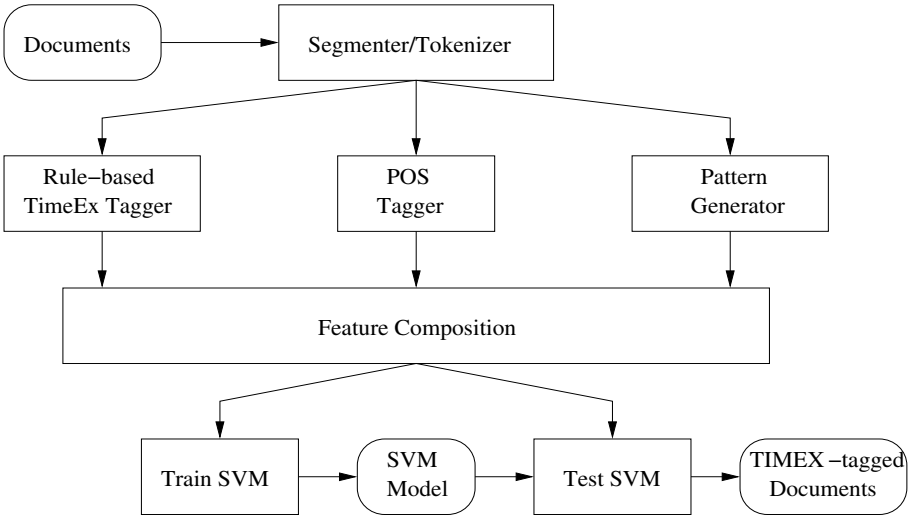


Fig. 2. General system architecture for Chinese automatic time expression labeler

5 Experimental Results

5.1 Labeling Performance

In this section we report the performance of the systems as scored against the evaluation data. We note that the system development and its optimization have been made over a development set withheld from available training data. This data was 20% and 10% of the development data for the English and Chinese systems, respectively. For evaluation, the development data was put back into the training pool and the final systems were trained using the whole training set described in Section 2.

We used the official TERN scoring script developed by MITRE for evaluating the systems. The results are reported using precision and recall numbers along with the $F_{\beta=1}$ metric. Table 2 presents the detection performance and Table 3 presents the bracketing performance as defined in the TERN 2004 Evaluation Plan [6]. In the former, the system output is counted correct if it overlaps (even if only one character) with the reference time expression. However, in the latter an exact match between the system output and the reference expressions is required. All SVM classifiers, for POS tagging, sentence segmentation, Chinese word segmentation etc. were implemented using TinySVM with a polynomial kernel of degree 2 and the general purpose SVM based chunker YamCha . In these experiments the accuracy of all those SVM-based systems were comparable to that of the state-of-the-art systems.

[6] `http://timex2.mitre.org/tern_evalplan-2004.29apr04.pdf`

Table 2. Detection Performance

	Precision	Recall	$F_{\beta=1}$
English System	97.8%	89.4%	93.5
Chinese System	96.5%	85.2%	90.5

Table 3. Bracketing Performance

	Precision	Recall	$F_{\beta=1}$
English	91.9%	84.0%	87.8
Chinese	83.8%	74.0%	78.6

Table 4. Comparison of detection and bracketing performances of word and character based Chinese systems

System	Detection	Bracketing
Word-based	90.5	78.6
Character-based	90.1	80.3

It is interesting to note that the bracketing performance (BP) of the Chinese system is considerably worse than that of the English system. This is in contrast with the detection performance (DP) which is moderately lower than the DP performance of the English system. Preliminary error analysis has shown that this is partly due to the so-called "boundary noise" at character level introduced by the Chinese word segmenter. This has motivated us to shift from using the words as tokens to using the characters as tokens for time expression labeling. We provide a result in Table 4 to show that such processing paradigm shift does indeed improve the bracketing performance with a statistically insignificant drop in detection performance. The latter is expected since the character based system has a narrower context when compared to the word-based system.

5.2 Computational Performance

In this section we provide some figures that will give a sense of how long it took for the SVM training and the final system evaluation.

For the English system, after segmentation and tokenization we had created approximately 316K distinct labeled examples for SVM training. The training took almost 10 hours. The number of binary features was 58907. For the Chinese system, we had approximately 137K distinct labeled examples. The training time was 2.5 hours. The number of binary features was 60174. Table 5 summarizes these training statistics.

Approximate processing times during evaluations for some of the system components are shown in Table 6. Both the English and Chinese systems were run on PCs with a single CPU, Pentium 4 at 2.4GHz. The machines had 1GB of RAM.

Table 5. Training statistics

	Examples	# Features	Training Time (hr)
English System	316K	58907	10
Chinese System	137K	60174	2.5

Table 6. Run times (RTs) for several components in English and Chinese systems

	English System's RT (min.)	Chinese System's RT (min.)
Segmentation/Tokenization	3	26
Part-of-Speech Tagging	34	40
Base Phrase Chunking Tagging	2	na
Dependency parsing	3	na
Rule-based Time Tagging	1	1
HMM-based Time Tagging	< 1	na
Feature Composition	< 1	< 1
SVM Time Tagging	5	3

5.3 Feature Sensitivity

In this section we explore the impact of adding or removing some of the features on performance. Computationally, it is not feasible to try all configurations; for example, we have $2^9 - 1 = 511$ possible configurations for the English system. Instead, we picked several configurations that are believed to shed light on the behavior of the system with respect to the features. The results are summarized in Table 7.

The difference in performance between the baseline system (that uses only the tokens) and the final system is 6.3% absolute in BP and 3.1% absolute in DP. This clearly indicates the significance of feature engineering. Although each feature only marginally contributes to performance, it becomes significant when all the contributions are summed up. The results show that the information provided by external time expression classifiers based on different paradigms contributed the most. It can also be seen that the syntactic features in terms of base phrase chunks did not contribute significantly. This is attributed to the fact that the temporal expressions have an unrestricted distribution with respect to syntax.

Another useful view for feature sensitivity can be obtained by grouping features into broad classes, such as

- baseline features: tokens
- lexical features: POS, lower-case, patterns, hyphen
- syntactic features: base phrase chunks
- "semantic" features: heads and grammatical relations
- external features: rule-based time tags, hmm-based time tags

and perform experiments by adding one group of features at a time. The results are presented in Table 8.

Table 7. English system performance at different feature combinations; tok: tokens, low: lower-cased tokens, pos: part-of-speech tags, bp: base phrase chunks, hyp: hyphenation flag, pat: patterns, rbtt: rule-based timex tags, iftt: identifinder timex tags, DP: detection performance and BP: bracketing performance

tok	low	pos	bp	hyp	pat	dep	rbtt	iftt	DP	BP
+	-	-	-	-	-	-	-	-	90.4	81.5
+	-	+	-	-	-	-	-	-	90.1	82.7
+	-	+	+	-	-	-	-	-	90.1	82.8
+	+	+	-	-	-	-	-	-	90.0	82.9
+	-	+	-	+	-	-	-	-	90.9	83.2
+	+	+	-	+	-	-	-	-	90.7	82.8
+	-	+	-	-	+	-	-	-	91.5	83.5
+	-	+	+	-	+	-	-	-	91.3	83.9
+	-	+	-	-	+	+	-	-	91.3	84.8
+	-	+	+	-	+	+	-	-	91.3	84.8
+	-	+	-	-	+	-	+	-	92.2	84.9
+	-	+	-	-	+	-	-	+	92.6	85.6
+	-	+	-	-	-	-	+	+	92.6	86.2
+	-	+	-	-	+	-	+	+	92.9	86.3
+	-	+	-	-	-	+	+	+	92.8	87.3
+	-	+	-	-	+	+	+	+	93.0	87.4
+	-	+	+	-	+	+	+	+	93.1	87.6
+	-	+	+	-	-	+	+	+	92.8	87.7
+	+	+	+	+	+	+	+	+	93.5	87.8

Table 8. System performance with respect to broad classes of features; lex: lexical features, syn: syntactic features, sem: "semantic" features, ext: external features, DP: detection performance and BP: bracketing performance

	DP	BP
baseline	90.4	81.5
baseline+lex	91.9	83.5
baseline+lex+syn	91.7	83.9
baseline+lex+syn+sem	91.7	85.4
baseline+lex+syn+sem+ext	93.5	87.8

Similar experiments have also been performed for the Chinese system. The results are shown in Table 9. Feature engineering contributed about 15% absolute to both detection and bracketing performances. It can be easily seen that each feature consistently contributed to the system performance. Results with different combinations clearly show the overlapping nature of the features. For example, the impact of POS tags when the other features are absent is quite different form the impact when all other features are present. A similar argument can also be made for the rule-based timex tags. The table also shows that they are the most useful features.

Table 9. Chinese system performance at different feature combinations; tok: tokens, pos: part-of-speech tags, 2-gram: prefixes and suffixes of length 2, pat: patterns, rbtt: rule-based timex tagger, DP: detection performance and BP: bracketing performance

tok	pos	2-gram	pat	rbtt	DP	BP
+	-	-	-	-	74.9	63.8
+	+	-	-	-	87.0	74.2
+	-	+	-	-	80.3	69.9
+	-	-	+	-	86.9	74.2
+	-	-	-	+	90.2	77.0
+	-	+	+	-	87.9	76.8
+	+	+	+	-	88.8	77.8
+	-	+	+	+	90.5	78.1
+	+	+	+	+	90.5	78.6

5.4 Error Analysis

We performed some preliminary error analysis without any quantification by looking at some phenomena that can be categorized at two levels; namely, token and phrase levels. At token levels, anaphoric temporal mentions were consistently missed, e.g. *this,it*, and frequent "time mentions" appearing as or part of proper nouns, or appearing as cardinal/ordinal numbers are spuriously detected, e.g. *the tonight show, midnight cowboy, 2000, first*. At phrase levels, time expressions with embedded structures are frequently detected as flat structures covering either the maximum extent or, one or multiple of its shorter mentions. This is probably due to quite small numbers of embedded time expressions in the training data. Besides, time mentions covering longer noun phrases were prematurely terminated, and, in some cases, however, shorter noun phrase time mentions were elongated by including VPs. Currently, a more detailed error analysis towards finding consistent contextual cues to avoid those phenomena in labeling is in progress.

6 Conclusions

We have introduced an end-to-end language independent statistical approach for labeling time expressions occurring in English and Chinese text. The architecture of the final systems for English and Chinese have been found very similar. The systems have had competitive performances at the evaluation workshop, although the Chinese system has not performed as well as the English system. We believe that this is partly due to the difficulty of language, partly due to the relatively smaller size of training data and partly due to the relatively small number of features employed when compared to the English system. We plan to perform detailed error analysis, use additional features (e.g. from WordNet, syntactic trees) and employ feature selection in a principled manner for further performance improvement.

Acknowledgement

We extend our special thanks to Wayne Ward, James H. Martin and Dan Jurafsky for their support, encouragement and useful feedback during this work. We also thank to Ashley Thornton for her careful and neat error analysis. This work is supported by the ARDA Aquaint II Program via contract NBCHC040040.

References

[Allwein et. al 2000] Allwein, E. L., Schapire R. E., and Singer, Y.: Reducing multiclass to binary: A unifying approach for margin classifiers. Journal of Machine Learning Research, 1:113-141, (2000).

[Bikel et. al 1999] Bikel, D. M., Schwartz, R. L., and Weischedel, R. M. , An algorithm that learns what's in a name. Machine Learning. Vol. 34, no. 1-3, pp. 211-231, (1999).

[Burges 1998] Burges, C. J. C.: Tutorial on Support Vector Machines for Pattern Recognition. Data Mining and Knowledge Discovery, 2(2), pages 1-47, (1997).

[Ferro et. al. 2004] Ferro, L., Gerber, L., Mani I., Sundheim B. and Wilson, G.: 2003 standard for the annotation of temporal expressions. Technical Report, MITRE, (2004).

[Kudo and Matsumato 2000] Kudo, T. and Matsumato, Y.: Use of support vector learning for chunk identification. Proc. of the 4th Conference on Very Large corpora, pages 142-144, (2000).

[Lin 1998] Lin D.: Dependency-based Evaluation of MINIPAR. Workshop on the Evaluation of Parsing Systems, Granada, Spain, May, (1998).

[Mani 2004] Mani, I.: Recent Developments in Temporal Information Extraction. to appear in Proceedings of RANLP'03, (2004).

[Ramhsaw and Marcus 1995] Ramhsaw, L. E. and Marcus, M. P.: Text Chunking Using Transformation Based Learning. Proceedings of the 3rd ACL Workshop on Very Large Corpora, pages 82-94, (1995).

[Reynar and Ratnaparkhi 1997] Reynar, J. C., and Ratnaparkhi, A.: A Maximum Entropy Approach to Identifying Sentence Boundaries. Proceedings of the Fifth Conference on Applied Natural Language Processing. March 31-April 3, (1997).

[Sang and Veenstra 1999] Sang, E. F. T. J. and Veenstra, J.: Representing text chunks Proceedings of EACL'99, pages 173-179, (1999).

[Saquete et. al. 2004] Saquete E., Martinez-Barco P., Munoz, R. and Vicedo J.L.: Splitting Complex Temporal Questions for Question Answering systems. Association for Computational Linguistics (ACL). Barcelona, SPAIN, (2004).

[Schilder and Habel 2003] Schilder, F. and Habel, C.: Temporal information extraction for temporal question answering. Proceedings of the 2003 AAAI Spring Symposium in New Directions in Question Answering. Stanford University, Palo Alto, USA, (2003).

[Vapnik 1995] Vapnik, V.: The Nature of Statistical Learning Theory. Springer Verlag, New York, USA, (1995).

Integrating Natural Language Techniques in OO-Method

Isabel Díaz[1,2], Lidia Moreno[1], Inmaculada Fuentes[1], and Oscar Pastor[1]

[1] Universidad Politécnica de Valencia – Dpto. de Sistemas Informáticos y Computación, Camino de Vera s/n, 46022 Valencia, España,
[2] Universidad Central de Venezuela, Ciudad Universitaria, Edif. FACES, Piso 2, Caracas 1051, Venezuela
{idiaz, lmoreno, opastor}@dsic.upv.es, infuegil@inf.upv.es

Abstract. An approach that involves natural language analysis techniques for the treatment of software system functional requirements is described in this paper. This approach is used as the basis for a process developed to generate sequence diagrams automatically from the textual specification of use cases. This facility has been integrated in the Requirements Engineering Phase of OO-Method, an automatic production environment of software. For this purpose, a translator that is based on natural language parser is used. The translator provides grammatical information to each use case sentence and it identifies the corresponding interaction. The automatic transformation is conceived and specified following an orientation that is based on models and patterns. The results of the validation of the transformation patterns are presented.

1 Introduction

The OO-Method is an automatic production environment of object-oriented software that has been created at the Universidad Politécnica de Valencia [1]. It is supported by a tool whose industrial version was given the name OlivaNova Model Execution® (ONME). In the OO-Method, the construction of the *Conceptual Model* plays a leading role from which it is possible to generate the *Execution Model* automatically (Fig. 1). The *Conceptual Model* graphically describes the problem space from a structural, dynamic, and functional perspective, and from the point of view of the presentation. Each piece of the Conceptual Model's graphic information can be automatically transformed into an OASIS concept, an object-oriented formal specification language based on dynamic logic [2]. The OASIS specification is used to generate the Execution Model.

The construction of the Conceptual Model is supported by the models obtained during the OO-Method Requirements Engineering Phase [3]. This phase begins by defining the *Mission Statement* which describes its purpose and the main functionalities of the system. Taking into account the system's possible interactions with its environment, the *Functions Refinement Tree* (FRT) is obtained. The remaining nodes form a hierarchy of the system's functionalities at different abstraction levels. An FRT leaf node is an elementary function that can be activated

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 560–571, 2005.

directly by an actor or as a result of a temporal event. Each one of the ARF elementary functions is a use case in the *Use Case Model*. By applying an iterative strategy, this model is refined by identifying the actors that interact in these use cases and by describing them in natural language. A use case models the communication between an actor and the system for the exchange of information, as well as the actions that must be carried out internally by the system to respond to these requests for information [4].

The Use Case Model is the main input for the development of the *Sequence Diagram Model*. A sequence diagram is built by each use case scenario. The Sequence Diagram Model is used as a link between the Use Case Model (which specifies the interaction between the system and its environment) and the OO-Method Conceptual Model (whose purpose is to describe the system's internal components, relationships and restrictions).

Originally, the construction of the Sequence Diagram Model was formulated as a manual task, to be undertaken exclusively by the stakeholders. Nevertheless, traceability mechanisms have recently been established and make it possible to deduce the sequence diagrams automatically, based on the use case text. In order to do so, a linguistic approach has been used with the intention of establishing this fourth point of automatic translation in the OO-Method. This approach is supported by a framework that is based on patterns that are compliant with MDA (Model Driven Architecture) and with UML (Unified Modeling Language) [5,6].

The defined linguistic framework and its integration in the OO-Method is described in this paper. This article has seven sections. Section 1 is the introduction. Section 2 shows the phases of the translation process, its objectives, activities, inputs and outputs. Section 3 describes the transformation model based on patterns that support the translation. Section 4 explains the strategy of transformation pattern application. Section 5 describes the validation process of these patterns and the translator tool that has been developed. Sections 6 and 7 present our conclusions and references.

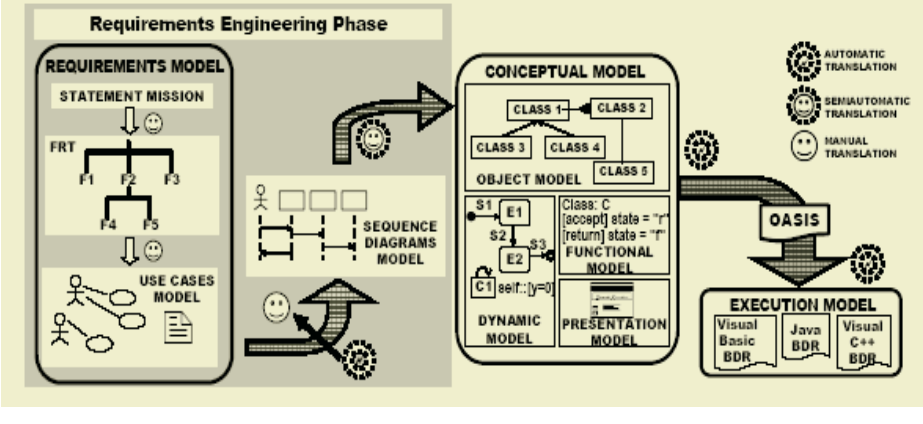


Fig. 1. The OO-Method Models

2 OO-Method Linguistic Approach

The automatic generation of the OO-Method Sequence Diagram Model is supported by linguistic information processing and control techniques [7]. This information is obtained through use case specification. It assumes that this specification is expressed as a document written in natural language that describes an elementary function of a software system [4,8]. The use case language is described by a previously defined grammar [9,10]. The translation of use cases into sequence diagrams is an iterative process developed through four sequential phases (Fig. 2). The result obtained in each phase is illustrated by an example in Figure 3 (based on the specification of a use case of a Sales Terminal System for Stores).

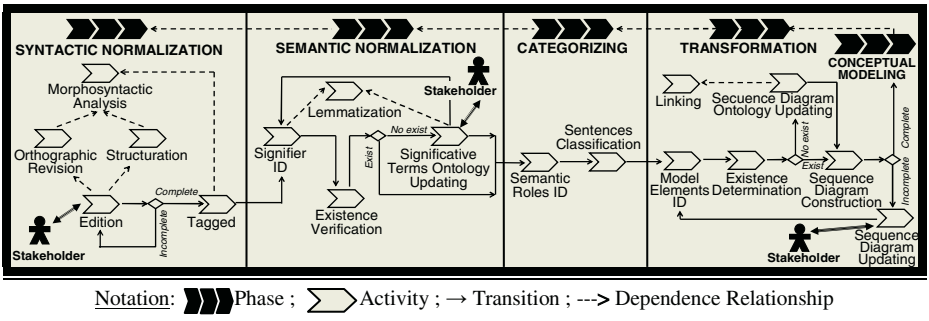


Fig. 2. Activities Diagram for the OO-Method Linguistic Approach

2.1 Syntactic Normalization

The goals of this phase are: (a) to generate a standard specification of the use case for the purpose of developing system documentation, and (b) prepare the use case text to be used to obtain information that enables the Sequence Diagram Model to be constructed automatically. In addition to improving the quality of requirements documentation, this phase acknowledges its lexical constituents in each sentence of the use case and provides them with useful morphological information so that it will be possible to identify the elements of a sequential diagram such as instances and parameters later on. As a result of this phase, a use case, which is structured according to the predefined grammar and is grammatically correct and enriched with morphosyntactic information, is obtained.

The Syntactic Normalization starts with the *editing* or transcription activity of the use case body and its descriptive information. The *spelling check* of the use case text includes the identification of words that do not exist in the language dictionaries. The *morphosyntactic analysis* enables possible morphological interpretations of each word in the use case body to be obtained and allows their respective grammatical features to be determined. It covers the disambiguation of those words that permit more than one morphological interpretation. Through *structuring*, the morphological constituents are grouped into syntactic categories of a superior level, i.e., in noun phrases. This activity makes it possible to determine whether or not the editing of the

use case has respected the structure imposed by pre-established grammar and style rules. In addition to this, its function is to guarantee the grammatical agreement of the text sentences of a use case. Once the use case has been edited, each word is *tagged* with information related to its morphological features and the syntactic category to which it belongs (Fig. 3).

2.2 Semantic Normalization

The main objective of this phase is to guarantee the terminological consistency of the use case text. Upon completion, each word of the use case body will be given a sole meaning and useful information about its grammatical relationships (e.g. equivalence, antithesis, generalization and composition).

In order to study the vocabulary of a use case, the words that describe domain significant information are distinguished from those that lack it. The words that have a semantic content are called significant terms [11]. The main components, properties and restrictions of the significant terms of the use cases form an ontology of the domain. The purpose of this ontology is to define the *common vocabulary* that enables *information* about the *requirements* of the software system under development to be shared [12,13]. The *updating* of this ontology is the central activity of the Semantic Normalization phase. The *signifier identification* or symbol that represents each significant term recognized in the use case text is undertaken for this purpose. The canonical form of the signifier is obtained through *lemmatization.* In order to avoid information redundancy, it is necessary *to check* that the significant term has not been previously defined in the ontology.

2.3 Categorization

In this phase, the significant terms and the use case sentences are classified according to their role in the Use Case Model. The *identification of the semantic roles* of each significant term consists of determining the function of the elements involved in the communication modeled by the use case. Thus, "issuer" and "action" are semantic role examples. Syntactic patterns are used to identify them. Hence, semantic roles are intermediaries between syntactic patterns and abstractions of the use case, making them independent of the way they are expressed in natural language. This enables each semantic role to be related to equivalent syntactic patterns in different languages (Fig. 3).

Lastly, the Categorization *classifies each sentence* of the use case as follows: *actor-system interface* (the actor is the issuer of the communication), *system-actor interface* (the issuer of the communication is the system and the recipient is the actor), and process (the sentence describes a certain behaviour that is able to change the state of the system). The predefined grammar and semantic roles identified by the sentence are used for this classification.

2.4 Transformation

The main purpose of this phase is to build the sequence diagram that corresponds to a scenario. The first activity is the *identification of the sequence diagram elements*. It is

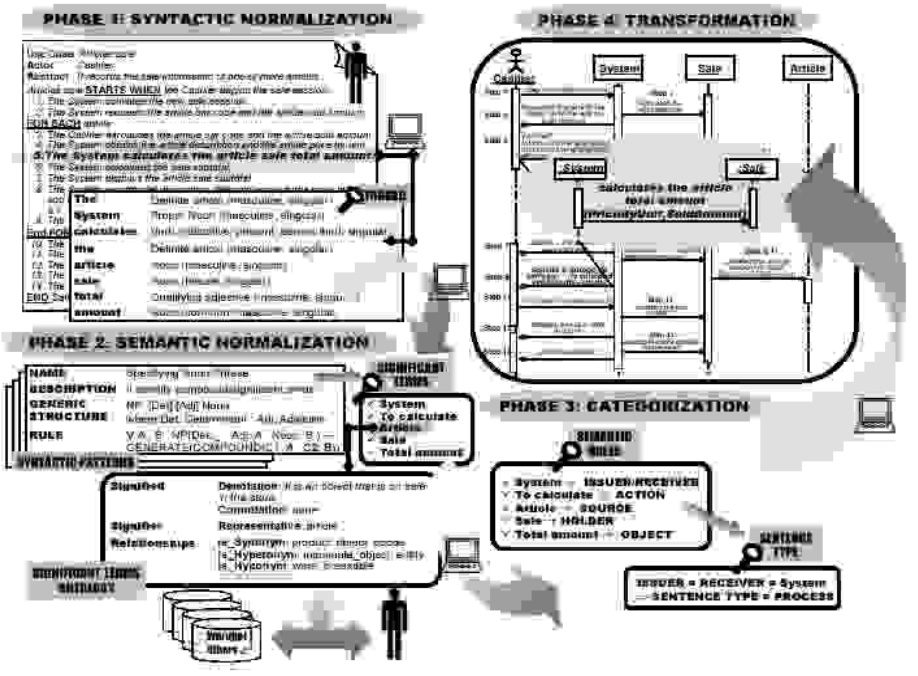


Fig. 3. An example

all a matter of recognizing instances, messages and parameters in each use case sentence. To do this, the information related to the semantic roles of the significant terms identified with the syntactic patterns is used. In order to guarantee that an element is described only once, its *pre-existence* is *determined*. Therefore, *updating* the analysis ontology involves defining a new Sequential Diagram Model element and *linking* it to the ontology of the domain significant terms. The description of the elements of the Sequence Diagram Model and their restrictions makes up an analysis ontology.

The *construction of the sequence diagram* is the graphic representation of these elements. This consists of acknowledging the interaction pattern associated with the syntactic pattern of each sentence type. An interaction pattern describes the interchange of messages between two or more objects. After the preliminary version of the sequence diagram that corresponds to a scenario has been automatically generated, the stakeholder can modify whatever he considers appropriate. The *diagram update* makes it possible to register the information concerned with these changes and verify their consistency.

3 Transformation Model Based on Patterns

The automatic transformation from the Use Case Linguistic Model to the Sequence Diagram Model has been conceived and specified following an orientation based on models. Figure 4 shows the application of the transformation model using the MDA

framework (Model Driven Architecture) [6]. The target and source models that take part in the transformation are platform-independent models (PIMs). These models don't display implementation details. The transformation model assumes that the Use Case Linguistic Model has been syntactically and semantically normalized.

The use of *patterns* is decisive in the transformation [14, 15]. These patterns allow us to identify generic conceptual structures and to describe how they can be reused whenever it is necessary to provide a solution to the same type of transformation. Each pattern is specified using a basic schema of five elements: *name* (identification that distinguishes a pattern from others), *source structure or context* (informal, formal or graphical representation that describes the situation in which the transformation can be applied), *target structure or context* (informal, formal or graphical representation of the transformation) and *transformation rules* (formal specification of a target structure from a source structure or context). Furthermore, a pattern can describe specific cases, contain application examples and attach observations. The patterns have been specified using the following metalanguages: (a) *EBNF* (Extended Backus Normal Form) for the specification of lexical component sequences [16] and (b) the combination of *OCL* (Object Constraint Language) and *UML* (Unified Modeling Language) to describe the participant models in the transformation [5,17].

The types of patterns used in OO-Method are described in the following sections. The patterns were designed for the Spanish language with the intention of also considering them in other languages. They were designed following the linguistic approach to transform the use case text into sequence diagrams.

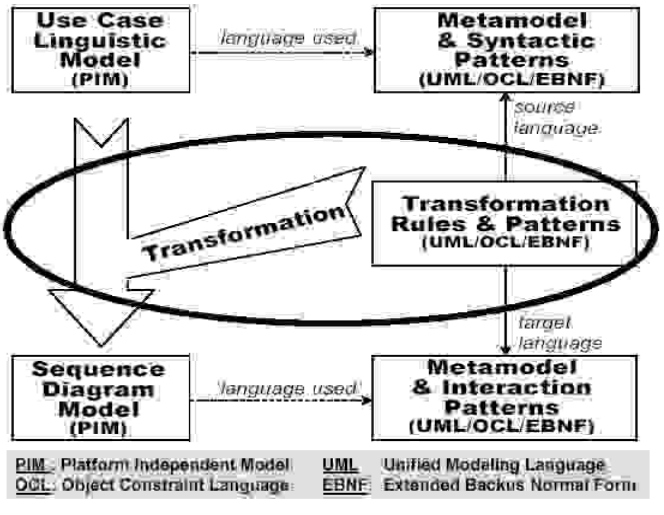


Fig. 4. Linguistic Transformation Model

3.1 Syntactic Patterns

They allow the recognition of types of lexical component sequences from use case text [18]. They can be atomic or molecular. Each atomic syntactic pattern allows the

deduction of a modeled element from generic lexical component sequences. The composition of two or more atomic syntactic patterns gives rise to a molecular syntactic pattern. Syntactic patterns of this type facilitate the acquisition of modeled elements and help to determine how these elements collaborate with each other. Figure 5 shows a molecular syntactic structure that is described by the "Properties Chain Pattern". This structure corresponds to a grammatical context of a particular type of use case process sentence.

3.2 Interaction Patterns

The interaction patterns specify generic types of sequence diagram fragments [5]. Figure 5 shows an interaction structure that is specified by the "Domino Effect Pattern". The structure is conformed by a border object and two or more domain objects[1]. The interaction initiates with a message that is sent by the border object to a domain object. This message induces the receiving instance to send another message to another domain instance and so on, until each instance sends a message with its respective answer.

3.3 Transformation Patterns

The transformation patterns describe how the grammatical contexts (recognized by the syntactic patterns from use case text) are turned into sequence diagram fragments (in accordance with the interaction patterns). The next section explains how the OO-Method transformation patterns are applied.

4 Applying Transformation Patterns

The transformation patterns act within the scope of each use case step. As a result of the syntactic and semantic normalization and the categorization, each use case step contains the following information: (a) an identification that indicates its position in the use case with respect to the other steps, establishing a partial order among them; (b) a part-of-speech tag according to the predefined grammar for each word in a step; and (c) the type of the step, depending on the sentence type that it contains: interface or process sentences (see Section 2.3).

This information allows the transformation pattern to recognize a certain grammatical context and to deduce the interaction structure that corresponds to it. The recognition of the grammatical context that underlies a step also implies the recognition of the transformation pattern that must be applied. Thus, according to the guidelines established in the transformation pattern, the information relative to the participants of the interaction is extracted from the grammatical context.

The transformation matches an interaction with each use case step. The interaction can be compounded by one or more messages and by one or more instances that fulfil the roles of senders and receiver of these messages. In order to deduce complementary information on the interaction, it is necessary to make a later analysis

[1] Hereafter, we will use the terms "instance" and "object" interchangeably. The definitions of "border class object" and "entity class object" correspond to the ones given in [8].

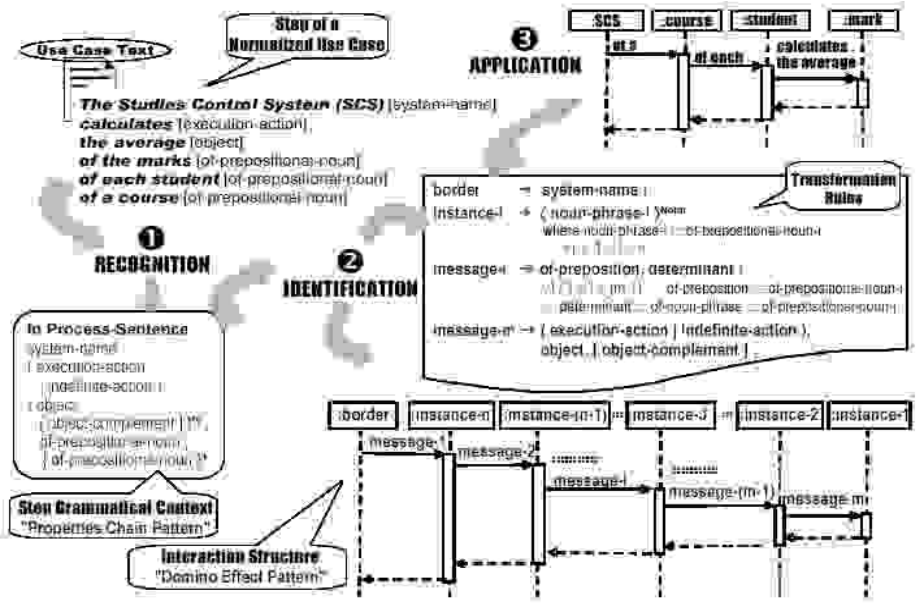


Fig. 5. Simple Communication Transformation Pattern from Different Sources

considering groups of two or more steps. This analysis allows us to recognize the parameters of a message or to determine if it is synchronous or asynchronous. This analysis is also done to incorporate information relative to conditionals and iterations (deduced from special sentences).

A sequence diagram is obtained by combining the interactions deduced from the use case steps. One or more sequence diagrams are matched to each use case. One of these corresponds to the basic path of the use case. There is also a sequence diagram for each alternative path. The process of application of a transformation pattern is described in Figure 5. Some details have been omitted for reasons of brevity.

4.1 Phase 1: Grammatical Context Recognition

The part-of-speech of a use case step allows us to recognize the grammatical context. A grammatical context is described by a syntactic pattern. A transformation pattern is specified from this syntactic pattern. Thus, the recognition of the grammatical structure helps determine which transformation pattern must be applied. In the example, the "Different Origin Simple Communication Transformation Pattern" was applied because its grammatical context corresponded to the part-of-speech of the step (Figure 5).

4.2 Phase 2: Participants and Interaction Type Identification

A transformation pattern always ties a grammatical context (described in a syntactic pattern) to a generic type of interaction (specified in an interaction pattern). The

transformation pattern also describes how to deduce information from the part-of-speech to obtain the elements that participate in the interaction. In the example, the transformation pattern establishes that the border instance name is the system name that is being modeled (Figure 5). The names of the other instances are obtained from the noun phrases contained in each one of the "OF prepositional phrase" in the step.

4.3 Phase 3: Transformation Pattern Application

The transformation pattern uses the grammatical information that is recognized in the step, the type of interaction, and the participant elements identified. By rewriting, the pattern infers the interaction contained in this step. This way, a specific fragment of the sequence diagram of the use case is obtained. The combination of these fragments (one fragment per step) allows us to complete this diagram.

In the example, the obtained interaction contains three domain instances: "course", "student" and "mark" (Figure 5). These objects were recognized from noun phrase content in each "OF prepositional phrase". The canonical form of these noun phrases was taken into account. Therefore, the label "marks" was substituted by label "mark" by means of the word normalization function: ⟨noun-phrase-i⟩Norm. Furthermore, the name given to the border instance was the same as the name given in the system been developed (SCS: Studies Control System). The three synchronous messages that were sent and received by these instances were identified. The order of the messages was inverse to the order of "OF prepositional phrases" in the step. The second and first messages allowed the referencing of their respective receiving instances ("course" and "student"). The third message was responsible for activating the execution of the "calculates the average" operation in the "mark" instance.

5 The Experience

In principle, the transformation patterns defined were designed through the direct observation of a sample of sequence diagrams obtained from the use cases of some academic and commercial information systems. A strategy was devised to validate these patterns. The strategy permitted us to establish the limitations of the transformation patterns designed initially and then improve and enrich them. The automatic validation strategy was supported by a translator developed by way of a prototype. The following sections make a description of the validation process.

5.1 The RETO-UPV Translator

The characteristics of RETO-UPV (Requirements Engineering TOol of the Universidad Politécnica de Valencia) were taken into account in the translator implementation and design [3][19]. This tool supports all the activities of the Requirements Engineering Phase of the OO-Method (see Section 1). Figure 6 shows the translator architecture and its interaction with the RETO-UPV components.

The stakeholder must use RETO-UPV to elicit and specify the use cases. This implies defining the Statement Mission, constructing the FRT and developing the Use Case Model (see Figure 1). Every use case text is normalized and then every step is considered as the input of the translator. The first action of the RETO-UPV Translator

is to obtain the tag of each word indicating its part-of-speech. To do this, we used the MS-Analyze tool which was developed at the Research Centre on Language and Speech Technologies and its Applications (TALP) at Universitat Politècnica de Catalunya (Spain) [20]. This tool is responsible for splitting use case text into tokens. Each token is tagged with its part-of-speech.

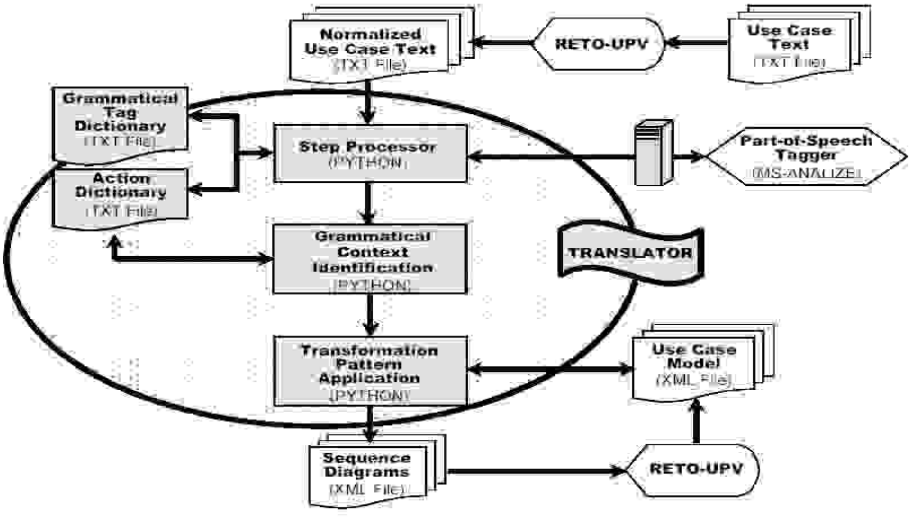


Fig. 6. The RETO-UPV Translator Architecture

With the help of dictionaries, a set of grammar rules and the tagged step, the translator recognizes the structures that correspond to the grammar symbols. This permits the translator to identify the grammatical context of the step, including the type (interface or process sentence). The grammatical context determines the transformation pattern that the translator must apply. The translator must provide itself with additional information about the Use Case Model so that the transformation process that indicates the pattern can be made. This process generates the interaction fragment specification that corresponds to analyzed step as output. Finally, the RETO-UPV Translator combines the fragments of every use case step until the sequence diagram specification is obtained. This information is held in XML File. Then, the RETO-UPV Sequence Diagram Editor displays the graph representation of this specification.

5.2 Validation

A manual validation of the transformation patterns proposed at the beginning of this study was made. To do this, the Use Case Model of the Car Rental System (CRS) following the OO-Method guidelines was developed. After normalization of the use cases, the Sequence Diagram Model was constructed. Both models were exhaustively

revised by stakeholders in order to reach a consensus on the results obtained manually. The sequence diagrams were then compared with the sequence diagrams generated using the RETO-UPV Translator in order to determine differences and similarities. Forty-one use cases were analyzed. This included a total of 574 steps of which only 14% were special (conditionals, iterations, etc.).

The interactions manually obtained were compared with the interactions generated automatically for each step of the CRS use cases. The comparison had to establish whether automatically generated interactions were the one expected by stakeholders. This implied determining if both interactions were equal, equivalent or different. We considered them equal when they were compounded by the same instances and the messages that these instances exchanged. We considered two interactions equivalents if both represented the same interaction goal even though the instances and messages weren't the same.[1] If the interactions were neither equal nor equivalent, we considered them to be different. Using these criteria, 66% of the transformation patterns, 23% were equivalent and only 11% were categorized as different.[2] This experience allowed us to establish which of the transformation patterns had to be improved or rejected. It was also possible to identify new transformation patterns of the grammatical contexts that were not considered by the designed ones initially.

6 Conclusions

In this paper, a linguistic approach for the automatic deduction of sequence diagrams from the use case textual specification has been presented. The deduction process has been defined following a software development approach that is based on the Use Case Model transformation in a Sequence Diagram Model. The transformation assumes the semantic and syntactic normalization of the use cases. This linguistic approach has been integrated into the OO-Method, a software automatic production environment. To do this, a translator was developed that was incorporated into the Requirements Engineering tool of OO-Method. The translator uses a natural language tool to provide each use case sentence with the necessary information to recognize its grammatical context. This context determines the type of transformation pattern that the translator must apply to obtain the interaction that corresponds to each sentence. The sequence diagram is obtained by the ordered combination of all the interactions of the use cases. An experiment was designed and executed that allowed us to validate the transformation patterns used. Actually, we are working on the definition of new transformation patterns and the design of an evolution strategy of sequence diagrams to guarantee the bidirectional traceability between sequence diagrams and their corresponding use cases to improve and to enrich the transformation process defined.

Acknowledgments. This work has been supported by the research projects CICYT TIC2003-07158-C04-03 and ICT EU-India (ALA/95/23/2003/077-054); it has been

[1] This decision was taken by stakeholders and who designed the transformation patterns.

[2] The interactions that did not come from grammar contexts recognized by a transformation pattern were also considered like different interactions. In these situations, the translator supposed that the interactions were formed by a single self-message on a border object.

financed also by *Consejo de Desarrollo Científico y Humanístico* of the *Universidad Central de Venezuela*.

References

1. Pastor O., Gómez J., Insfrán E., Pelechano V.: The OO-Method Approach for Information Systems Modeling: from Object-Oriented Conceptual Modeling to Automated Programming. Information Systems 26 (2001): 507-534.
2. Pastor O., Ramos I.: Oasis 2.1.1. A Class-Definition Language to Model Information Systems Using and Object-Oriented Approach. Departamento de Sistemas Informáticos y Computación. Universidad Politécnica de Valencia. España, 1995.
3. Insfrán E., Pastor O., Wieringa R.: Requirements Engineering-Based Conceptual Modeling. Requirements Engineering, 7(2), 61-72. Springer-Verlag. March 2002
4. Díaz I., Losavio F., Matteo A., Pastor O.: A Specification Pattern for Use Cases. Information & Management, Vol. 41/8 (2004). Pp. 961-975. Elsevier Science B.V.
5. Object Management Group: Unified Modeling Language Specification: Superstructure. Version 2.0. August 2003. http://www.omg.org/uml.
6. Object Management Group: MDA Guide. Version 1.01. Jun 03. http://www.omg.org/uml
7. Métais E.: Enhancing IS Management with Natural Language Processing Techniques. Data & Knowledge Engineering. 41(2002), 247-272. Elsevier Science B.V.
8. Jacobson I., Christerson M., Jonsson P., Övergaard G.: Object-Oriented Software Engineering. A Use Case Driven Approach. Addison-Wesley, 1992.
9. Díaz I., Pastor O., Moreno L., Matteo A.: Una Aproximación Lingüística de Ingeniería de Requisitos para OO-Method. Memorias del VII Workshop Iberoamericano de Ingeniería de Requisitos y Desarrollo de Ambientes de Software (IDEAS'04). Perú. Mayo, 2004.
10. Díaz I., Moreno L., Pastor O.: Traducción de Casos de Uso en Patrones de Interacción de Instancias: una Aproximación Lingüística. Memorias de las 3eras. Jornadas Iberoamericanas de Ingeniería de Software e Ingeniería del Conocimiento. Chile, 2003.
11. Rolland C., Ben-Achour C.: Guiding the Construction of Textual Use Case Specifications. Data & Knowledge Engineering 25(1998), 125-160. Elsevier Science.
12. Aussenac-Guilles N., Biébow B., Szulman S.: Revisiting Ontology Design: a Method based on Corpus Analysis. Proceedings of the 12th European Workshop on Knowledge Acquisition, Modeling and Management (EKAW 2000). Pp. 172-188. Springer-Verlag.
13. Velardi P., Fabriani P., Missikoff M.: Using Text Processing Techniques to Automatically Enrich a Domain Ontology. Proceedings of the International Conference on Formal Ontology in Information Systems (FOIS'2001). Pp. 270-284. ACM C. P.
14. Fowler M. Analysis Patterns: Reusable Object Models. Addison-Wesley, 1997.
15. Gamma E., Helm R., Johnson R., Vlissides J.: Design Patterns. Elements of Reusable Object-Oriented Software. In Professional Computing Series, Addison-Wesley, 1992.
16. International Standard ISO/IEC 14977. Extended Backus-Naur Form. 1996.
17. Object Management Group. OCL 2.0. October 2003. http://www.omg.org/uml.
18. Juristo N., Moreno A., López M. How to Use Linguistic Instruments for Object-Oriented Analysis. IEEE Software Vol. 17 Issue 3. May/June 2000. Pp. 80-89.
19. Requirements Engineering Tool (RETO-UPV). Universidad Politécnica de Valencia. DSIC. http://retoweb.europe.webmatrixhosting.net/home.aspx
20. Carreras X. Padró L. A Flexible Distributed Architecture for Natural Language Analyzers. TALP Research Center Departament de Llenguatges i Sistemes Informàtics Universitat Politècnica de Catalunya, Barcelona, España.

Document Re-ordering Based on Key Terms in Top Retrieved Documents

Yang Lingpeng, Ji Donghong, Nie Yu, and Zhou Guodong

Institute for Infocomm Research,
21, Heng Mui Keng Terrace,
Singapore 119613
{lpyang, dhji, ynie, zhougd}@i2r.a-star.edu.sg

Abstract. In this paper, we propose a method to improve the precision of top retrieved documents by re-ordering the retrieved documents in the initial retrieval. To re-order the documents, we first automatically extract key terms from top N (N<=30) retrieved documents, then we collect key terms that occur in query and their document frequencies in top N retrieved documents, finally we use these collected terms to re-order the initially retrieved documents. Each collected term is assigned a weight by its length and its document frequency in top N retrieved documents. Each document is re-ranked by the sum of weights of collected terms it contains. In our experiments on 42 query topics in NTCIR3 Cross Lingual Information Retrieval (CLIR) dataset, an average 17.8%-27.5% improvement can be made for top 10 documents and an average 6.6%-12% improvement can be made for top 100 documents at relax/rigid relevance judgment and different parameter setting.

1 Introduction

For Chinese Information Retrieval where query is a short description by natural language, many retrieval models, indexing strategies, query expansion strategies and document re-ordering methods have been proposed. Chinese Character, bi-gram, n-gram (n>2) and word are the most widely used indexing units. The effectiveness of single Chinese Characters as indexing units has been reported in [7]. The comparison between the three kinds of indexing units (single Characters, bi-grams and short-words) is given in [5]. It shows that single character indexing is good but not sufficiently competitive, while bi-gram indexing works surprisingly well and it's as good as short-word indexing in precision. [9] suggests that word indexing and bi-gram indexing can achieve comparable performance but if we consider the time and space factors, it is preferable to use words (and characters) as indexes. It also suggests that a combination of the longest-matching algorithm with single characters is a good method for Chinese IR and if there is a module for unknown word detection, the performance can be further improved. Some other researches give similar conclusions. Bi-gram and word are considered as the top two indexing units in Chinese IR and they are also used in many reported Chinese IR systems.

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 572 – 583, 2005.

Regarding retrieval models, two models are most widely used in Chinese Information Retrieval, i.e., Vector Space Model [12] and Probabilistic Retrieval Model [2].

For query expansion, most strategies make use of the top *N* retrieved documents in initial retrieval [11]. Generally, it selects *M* indexing units from the top *N* documents according to some criteria and adds these *M* indexing units to original query to form a new query. In such a process of query expansion, it's supposed that the top *N* documents are related with original query. However in practice, such an assumption is not always true. Although many literatures report that query expansion can improve the recall in many situation, they also suggest that the actual relevance quality of top retrieved documents affects the effectiveness of query expansion.

While query expansion tries to improve the recall of top retrieved documents, document re-ordering is used to improve the precision of top retrieved documents.

Lee, K. et.al. propose a document re-ranking method which uses document clusters [6]. Firstly, they build a hierarchical cluster structure for the whole document set; secondly, they divide top retrieved documents into some clusters, that is, they find sub-trees in hierarchical cluster structure which contain some retrieved documents by some criteria; finally, they calculate similarity between each cluster and each query topic, and use the similarity to adjust the similarity between query and each document in this document cluster. It's reported their method achieves significant improvements on their experiments on Korean corpus. One difficulty of this method is it needs to build hierarchical cluster structure for document set.

Kamps, J. [4] propose a method to re-order retrieved documents by making use of manually assigned controlled vocabularies in documents. By building a controlled vocabulary - controlled vocabulary matrix on co-occurrences, each document can be represented as a vector by controlled vocabularies which occur in and each query can be represented as a vector by the vectors of top N retrieved documents. Finally, each document is re-ordered by the distances between the document vector and query vector. It's reported this re-ranking strategy significantly improves retrieved effectiveness on their experiments on German GIRT and French Amaryllis collections. This method depends on the controlled vocabularies assigned to document, but in most case, no controlled vocabulary is assigned to documents.

Qu, Y. L. [10] uses manually built thesaurus to re-rank retrieved documents. Each term in query topic is expanded with a group of terms in thesaurus. It's a hard job to manually build a large thesaurus for unexpected query topics.

Bear J. el al. [1] use manually constructed or automatically learned small grammars for topics to re-order documents by matching grammar rules in some segment in articles. But grammar construction itself is a difficult problem in Chinese language.

Yang, L.P., et. al [14,15] use extracted long terms in query and document to re-order retrieved documents in Chinese IR. Firstly, they cluster the whole document set into some clusters; secondly, they automatically extract global key terms from these clusters; thirdly, they make use of these global terms and their frequencies to find local terms in a query or a document; finally, they use long local terms to re-calculate the similarity between query and document, and use the new similarity value to re-order retrieved documents. Their experiments show that long terms play an important

role in document re-ordering, since they tend to be more significant for the retrieval precision than short terms. It's reported their experiments based on NTCIR3 CLIR dataset can achieve an average 10%-11% improvement for top 10 documents and an average 2%-5% improvement for top 100 documents. One difficulty of this method is how to identify local key terms in query and document because there are a few parameters needed to set.

In this paper, we propose an approach to re-order retrieved documents. Firstly, we automatically extract key terms from each document in document set; secondly, we use key terms in top N retrieved documents and their document frequencies to re-order top retrieved K (N<K) documents.

The rest of this paper is organized as following. In section 2, we describe how to automatically extract key terms from document. In section 3, we describe how to re-order retrieved documents. In section 4, we evaluate the performance of our proposed method on NTCIR3 CLIR dataset and give out some result analysis. In section 5, we present the conclusion and some future work.

2 Term Extraction

We use a seeding-and-expansion mechanism to extract terms from documents. The procedure of term extraction consists of two phases, seed positioning and term determination. Intuitively, a seed for a candidate term is an individual word (or Chinese character) within the term, seed positioning is to locate the rough position of a term in the text, while term determination is to figure out which string covering the seed in the position forms a term.

To determine a seed needs to weigh the individual words to reflect their significance in the text in some way. To do so, we make use of a very large corpus *r* as a *reference*. Suppose *s* is the text of the collected summaries, *w* is an individual word in the text, let $P_r(w)$ and $P_s(w)$ be the probability of *w* occurring in *r* and *s* respectively, we adopt 1), *relative probability* or *salience* of *w* in *s* with respect to *r* [13], as the criteria for evaluation of seed words.

1) $P_s(w) / P_r(w)$

We call *w* a *seed* if $P_s(w) / P_r(w) \geq \delta (\delta > 0)$.

We have the following assumptions about a term.

i) a term contains at least a seed.
ii) a term occurs at least *L (L>1)* times in the text.
iii) a *maximal word string* meeting i) and ii) is a term.
iv) for a term, a *real maximal substring* meeting i) and ii) without considering their occurrence in all those terms containing it is also a term.

Here a *maximal word string* meeting i) and ii) refers to a word string meeting i) and ii) while no other longer word strings containing it meet i) and ii). A *real maximal substring* meeting i) and ii) refer to a real substring meeting i) and ii) while no other longer real substrings containing it meet i) and ii).

Figure 1 describes the procedure to extract key terms from a document *d*.

```
let F_d(t) represents the frequency of t in d;
let O is a given threshold (O>1);
T = { };
collect Seeds in d into S;
for all c∈S
        let Q = {t: t contains c and F_d(t)≥O};
        while Q ≠ NIL
                max-t ← the longest string in Q;
                T ← T + { max-t };
                Remove max-t from Q;
                for all other t in Q
                        if t is a substring of max-t
                                F_d(t)← F_d(t)- F_d(max-t);
                                if F_d(t)<O
                                        removing t from Q;
return T as key terms in document d;
```

Fig. 1. Key Term Extraction from Document *d*

3 Document Re-ordering

We make use of the information of key terms and their document frequencies in top N (N<=30) retrieved documents to re-order top K (N<K) retrieved documents. Firstly, we automatically extract key terms from each document; secondly, we collect key terms which are sub-string of query topic and their document frequencies in top N retrieved documents; thirdly, we assign collected key terms weight by their length and document frequency, that is, more weight is given to more long key term and more weight is given to more document frequent key term; finally, we re-rank retrieved documents by the sum of weight of key terms they contain.

Given query *q*, following is the procedure to re-order the *K* initial retrieved documents by making use of top *N* (*N*<*K*) retrieved documents:

Step 0: Let $\{d_1, d_2, \ldots, d_i, \ldots, d_K\}$ denote the top *K* initial retrieved documents;

Let $R=\{R_1, R_2, \ldots, R_i, \ldots, R_K\}$ denote the similarity values between *q* and d_i in initial retrieval;

Let $S=\{S_1, S_2, \ldots, S_i, \ldots, S_K\}$ denote the similarity values between *q* and d_i after document re-ordering;

Step 1: Extract key terms from top *N* retrieved documents;

Step 2: Collect key terms which occur at query *d*;

Let these collected key terms form term set $T=\{T_1, T_2, \ldots, T_n\}$; their document frequencies in top *N* retrieved documents form set $D=\{DF_1, DF_2, \ldots, DF_n\}$;

Step 3: Assign each term T_i in T a weight W_i by following formula:

$$W(T_i) = \text{sqrt}(|T_i|) \times \text{sqrt}(DF_i).$$

where $|T_i|$ is the length of term T_i, i.e., the number of Chinese characters in term T_i and sqrt(x) is the square root of x.

Step 4: Calculate new similarity value S_i between query q and each document d_i in K initial retrieved documents;

Step 4.1: Calculate re-ranking weight w of d_i by key terms in d_i;

```
w = 0;
T' = T;
do while (T' is not empty)
        Find the longest key term t in T';
        If t occurs in document d_i, Then
                w = w + W(t)
                Remove all occurrence of t in d_i
                Discard t from T';
```

Step 4.2: Calculate new similarity value S_i between q and d_i

```
if w > 0 then
        S_i = w × R_i
else
        S_i = R_i
```

Step 5: Re-order top K retrieved documents by their new similarities values $S=\{S_1, S_2, \ldots, S_i, \ldots, S_K\}$.

4 Experiments and Evaluation

We use NTCIR3 CLIR dataset as our test dataset. The dataset contains Chinese document set CIRB011 (132,173 documents from China Times, China Times Express, Commercial Times, China Daily News and Central, Daily News) and CIRB20 (249,508 documents from United Daily News). We also use the Chinese-Chinese D-run query topics in NTCIR3 CLIR as query topics. There are 50 query topics released in NTCIR3, but only 42 topics are finally used to evaluate. Each query is a simple description of a topic by Chinese language. (Appendix lists the 42 query topics. You may also find more information about NTCIR3 CLIR task from http://research.nii.ac.jp/ntcir-ws3/work-en.html).

For initial retrieval, we use bi-gram as index unit and we use vector space model to represent documents and queries. Each document or query is represented as a vector in vector space where each dimension of vector is a bi-gram. The weight of bi-gram t in document d is given by the following TF/IDF weight scheme:

$$w(t, d)=\log(\mathrm{T}(t, d)+1) \times \log(N/\mathrm{D}(t)+1)$$

where $w(t, d)$ is the weigh given to t in d, $\mathrm{T}(t, d)$ is the frequency of t in d, N is the number of documents in document set, $\mathrm{D}(t)$ is the number of documents in document set which contain t.

The weight of bigram t in query q, $w(t, q)$, is given by the following weight scheme:

$$w(t, q) = \mathrm{T}(t, q)$$

where $\mathrm{T}(t, q)$ is the frequency of t in q.

The similarity (distance) between a document d and a query q is calculated by the cosine of the document vector and the query vector.

The initial retrieval result is used as 1st baseline to evaluate our proposed method; we also use Yang L.P et.al. [14]'s result on NTCIR3 CLIR dataset as 2nd baseline.

Our experiments re-rank the top 1000 initial retrieved documents and evaluate the effectiveness by precisions at different document levels. We use NTCIR3's relax relevance judgment and rigid relevance judgment to measure the precision of retrieved documents. Relax Relevance Judgment considers highly relevant documents, relevant documents and partially relevant documents, while Rigid Relevance Judgment only considers highly relevant documents and relevant documents. We use PreAt10 and PreAt100 to separately represent the precision of top 10 retrieved documents and top 100 retrieved documents.

Our experiments focus on two parts: Which kind of key terms in documents will be used to re-order retrieved documents? How many top retrieved documents should we use to extract key terms from? For the first part, we extract different key terms by using different parameters in our term extraction method. There are two parameters in our term extraction method. One parameter is δ - the minimum saliency of seed in term, the other parameter is L - the minimum occurrence of term in document. For the second part, we only test parameter N - the number of top retrieved documents that are used to extract terms from. Following is the parameter setting in our experiments:

δ=1, 10: We consider terms which contain at least a seed whose salience is 1 or 10;

L=2, 3, 4: We consider terms which occur at least 2 times, 3 times or 4 times in document;

N=20, 25, 30: We consider top 20, 25 or 30 retrieved documents as related documents and extract key terms from them to re-order retrieved documents.

Table 1-6 gives the comparison of precisions at different parameters setting. In table 1-6, column [PreAt10(relax)] represents the average precision of 42 topics on PreAt10 relax relevance judgment; Column [PreAt10(rigid)] represents the average precision of 42 topics on PreAt10 rigid relevance judgment; Column [PreAt100(relax)] represents the average precision of 42 topics on PreAt100 relax relevance judgment; Column [PreAt100(rigid)] represents the average precision of 42 topics on PreAt100 rigid relevance judgment. Row [BaseLine1] represents the initial retrieved result; Row [BaseLine2] represents experiment result reported on Yang et. al [14]; Row [N=20] represents the re-ordered result which make use of key terms in top 20 retrieved documents; Row [N=25] represents the re-ordered result which make use of key terms in top 25 retrieved documents; Row [N=30] represents the re-ordered result which make use of key terms in top 30 retrieved documents. Each item in table represents the precision and its improvement over [BaseLine1] at the conditions expressed by Column and Row.

Table 1. Statistics on (δ=1, L=2)

	PreAt10(relax)	PreAt10(rigid)	PreAt100(relax)	PreAt100(rigid)
BaseLine1	0.3619	0.2595	0.1886	0.1279
BaseLine2	0.4052 (12%)	0.2871 (10.6%)	0.1926 (2.1%)	0.133 (4%)
N=20	0.4143 (14.5%)	0.3024 (16.5%)	**0.2055 (9%)**	**0.1376 (7.6%)**
N=25	**0.4262 (17.8%)**	**0.3143 (21.1%)**	0.2052 (8.8%)	0.1371 (7.2%)
N=30	0.4167 (15.1%)	0.3119 (20.2%)	0.2048 (8.6%)	0.1369 (7%

Table 2. Statistics on (δ=1, L=3)

	PreAt10(relax)	PreAt10(rigid)	PreAt100(relax)	PreAt100(rigid)
BaseLine1	0.3619	0.2595	0.1886	0.1279
BaseLine2	0.4052 (12%)	0.2871 (10.6%)	0.1926 (2.1%)	0.133 (4%)
N=20	0.4119 (13.8%)	0.3001 (15.6%)	0.205 (8.7%)	0.1376 (7.6%)
N=25	**0.4333 (19.7%)**	**0.3167 (22%)**	0.2079 (10.2%)	0.1381 (8%)
N=30	**0.4333 (19.7%)**	**0.3167 (22%)**	**0.2083 (10.4%)**	**0.1388 (8.5%)**

Table 3. Statistics on (δ=1, L=4)

	PreAt10(relax)	PreAt10(rigid)	PreAt100(relax)	PreAt100(rigid)
BaseLine1	0.3619	0.2595	0.1886	0.1279
BaseLine2	0.4052 (12%)	0.2871 (10.6%)	0.1926 (2.1%)	0.133 (4%)
N=20	0.4262 (17.8%)	0.3143 (21.1%)	**0.2117 (12.2%)**	**0.14 (9.5%)**
N=25	**0.4357 (20.4%)**	0.319 (22.9%)	0.2098 (11.2%)	0.1393 (8.9%)
N=30	0.4333 (19.7%)	**0.3214 (23.9%)**	0.2105 (11.6%)	0.1395 (9.1%)

Table 4. Statistics on (δ=10, L=2)

	PreAt10(relax)	PreAt10(rigid)	PreAt100(relax)	PreAt100(rigid)
BaseLine1	0.3619	0.2595	0.1886	0.1279
BaseLine2	0.4052 (12%)	0.2871 (10.6%)	0.1926 (2.1%)	0.133 (4%)
N=20	0.4262 (17.8%)	0.3119 (20.2%)	**0.2043 (8.3%)**	**0.1369 (7%)**
N=25	**0.4381(21.1%)**	**0.3214 (23.9%)**	0.2038 (8.1%)	0.1364 (6.6%)
N=30	0.4357(20.4%)	**0.3214 (23.9%)**	0.2038 (8.1%)	0.1362 (6.5%)

Table 5. Statistics on (δ=10, L=3)

	PreAt10(relax)	PreAt10(rigid)	PreAt100(relax)	PreAt100(rigid)
BaseLine1	0.3619	0.2595	0.1886	0.1279
BaseLine2	0.4052 (12%)	0.2871 (10.6%)	0.1926 (2.1%)	0.133 (4%)
N=20	0.4286 (18.4%)	0.3119 (20.2%)	0.2076 (10.1%)	0.1379 (7.8%)
N=25	**0.4476 (23.7%)**	**0.331(27.5%)**	0.2064 (9.4%)	0.1383 (8.1%)
N=30	0.4405 (21.7%)	0.319 (22.9%)	**0.2086 (10.6%)**	**0.14 (9.5%)**

Table 6. Statistics on (δ=10, L=4)

	PreAt10(relax)	PreAt10(rigid)	PreAt100(relax)	PreAt100(rigid)
BaseLine1	0.3619	0.2595	0.1886	0.1279
BaseLine2	0.4052 (12%)	0.2871 (10.6%)	0.1926 (2.1%)	0.133 (4%)
N=20	**0.4405 (21.7%)**	**0.3262 (25.7%)**	**0.2129 (12.9%)**	**0.141 (10.2%)**
N=25	**0.4405 (21.7%)**	0.3238 (24.8%)	0.2112 (12%)	0.1402 (9.6%)
N=30	0.4381(21.1%)	0.3238 (24.8%)	0.21 (11.3%)	0.139 (8.7%)

Table 7. Statistics on (δ=1, N=25)

	PreAt10(relax)	PreAt10(rigid)	PreAt100(relax)	PreAt100(rigid)
BaseLine1	0.3619	0.2595	0.1886	0.1279
BaseLine2	0.4052 (12%)	0.2871 (10.6%)	0.1926 (2.1%)	0.133 (4%)
L=2	0.4262 (17.8%)	0.3143 (21.1%)	0.2052 (8.8%)	0.1371 (7.2%)
L=3	0.4333 (19.7%)	0.3167 (22%)	0.2079 (10.2%)	0.1381 (8%)
L=4	**0.4357 (20.4%)**	**0.319 (22.9%)**	**0.2098 (11.2%)**	**0.1393 (8.9%)**

Table 8. Statistics on (δ=10, N=25)

	PreAt10(relax)	PreAt10(rigid)	PreAt100(relax)	PreAt100(rigid)
BaseLine1	0.3619	0.2595	0.1886	0.1279
BaseLine2	0.4052 (12%)	0.2871 (10.6%)	0.1926 (2.1%)	0.1330 (4%)
L=2	0.4381(21.1%)	0.3214 (23.9%)	0.2038 (8.1%)	0.1364 (6.6%)
L=3	**0.4476 (23.7%)**	**0.331(27.5%)**	0.2064 (9.4%)	0.1383 (8.1%)
L=4	0.4405 (21.7%)	0.3238 (24.8%)	**0.2112 (12%)**	**0.1402 (9.6%)**

From table 1-6, our proposed method gets better result than [BaseLine1] and [BaseLine2] in every parameter setting. If only considering PreAt100, it seems we may get better result by using terms in top 20 retrieved documents; but if only considering PreAt10, it seems we may get better result by using terms in top 25 or top 30 retrieved documents. If considering PreAt10 and PreAt100 together, we regard that we may get better and stable result by using terms in top 25 retrieved documents.

Tables 7 and 8 gives the comparison of precisions on different term extraction parameter settings using terms in top 25 retrieved documents.

From Table 7 and 8, our proposed method can improve PreAt10 by 17.8%-23.7% from 0.3619 to 0.4262-0.4476 in relax relevance judgment and improve PreAt10 by 21.1%-27.5% from 0.2595 to 0.3143-0.331 in rigid relevance judgment. In PreAt100 level, our method can improve 8.1%-12% and 6.6%-9.6% in relax relevance judgment and rigid relevance judgment. Even in worst case, our proposed method get better result than [BaseLine2] with 18.8%, 21.1%, 8.1% and 6.6% improvement at PreAt10(relax), PreAt10(rigid), PreAt100(relax) and PreAt100(rigid) level compared with 12%, 10.6%, 2.1% and 4% improvement in [BaseLine2].

From table 7 and table 8, we may conclude that using key terms that occur at least 3 times or 4 times in documents may get better results.

The above experiments on NTCIR3 dataset show our method can achieve significant improvements on PreAt10 and PreAt100 results.

Fig. 2-5 gives the comparison of the precisions of 42 query topics before and after document re-ordering at parameter setting (δ=1, N=25, L=4).

From Fig. 2–5, for 42 topics in NTCIR3, there are only 2 query topics (topic 9 and 43) whose precisions are slightly decreased after document re-ordering, the other 40 topics are all improved after document re-ordering.

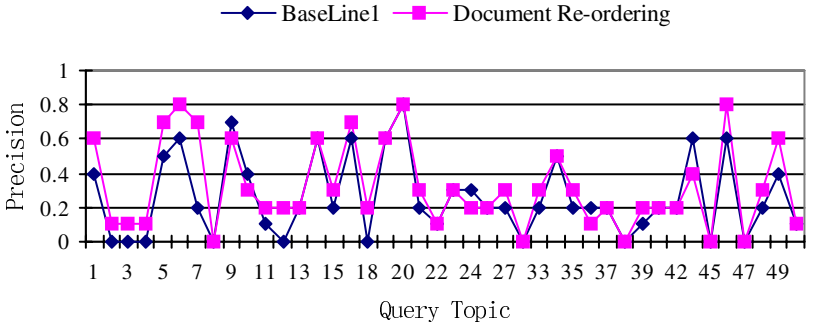


Fig. 2. PreAt10 at rigid relevance judgment (δ=1, N=25, L=4)

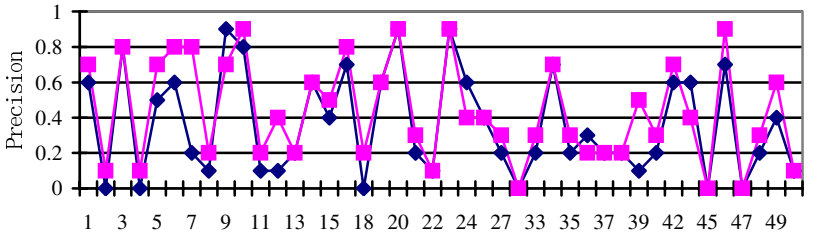


Fig. 3. PreAt10 at relax relevance judgment (δ=1, N=25, L=4)

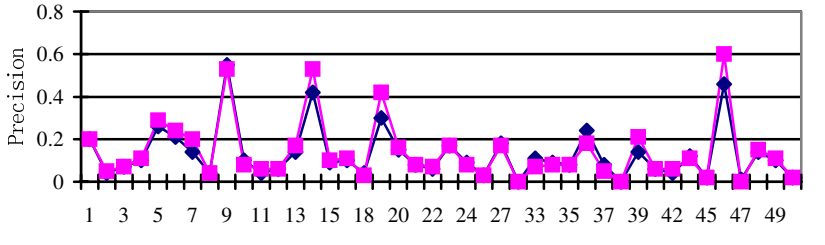


Fig. 4. PreAt100 at rigid relevance judgment (δ=1, N=25, L=4)

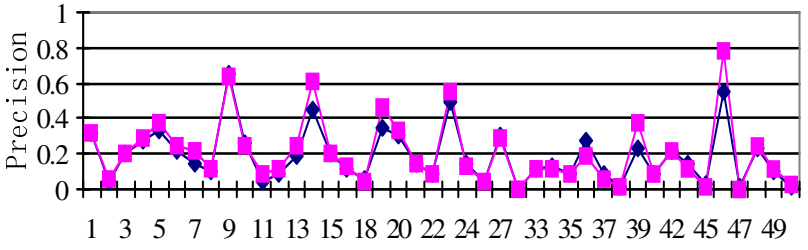


Fig. 5. PreAt100 at relax relevance judgment (δ=1, N=25, L=4)

5 Conclusion and Future Work

Document re-ordering is very important for improving the precision. In this paper, we introduce our approach to re-order retrieved documents in Chinese IR. For each query topic, firstly, we automatically extract key terms from top N (N<=30) retrieved documents; secondly, we collect these terms that occur at query topic and their document frequencies in top N retrieved documents; thirdly, we re-order top K (N<K) retrieved documents by collected key terms each document contains. Each collected key term is given a weight by its length and its document frequency in top N retrieved documents. Weight is given to reflect an observation: long key term may contain more precise information and it'll be given more weight; key term occurred in more top retrieved documents tends to play more important role in distinguishing query topic and it'll be given more weight.

With bi-gram as indexing units and vector space model as retrieval model, our experiments in the Chinese tasks of CLIR in NTCIR3 show that our method using key terms can improve the average performance of Chinese IR on 42 query topics by 17.8%-27.5% at top 10 documents and 6.6%-12% at top 100 documents at all kinds of parameter settings and relax relevance judgment or rigid relevance judgment.

In future, we'll apply our method on Chinese IR which uses Probabilistic Retrieval Model as retrieval model or uses word as indexing unit. We also want to do more experiments on Chinese IR which uses long description as query topic.

References

1. Bear J., Israel, D., Petit J., Martin D.: Using Information Extraction to Improve Document Retrieval. Proceedings of the Sixth Text Retrieval Conference, 1997.
2. Fuhr. N.: Probabilistic Models in Information Retrieval. The Computer Journal. 35(3):243-254, 1992.
3. Ji D.H., Yang L.P., Nie Y.: Chinese Language IR based on Term Extraction. In The Third NTCIR Workshop, 2002.
4. Kamps, J.: Improving Retrieval Effectiveness by Reranking Documents Based on Controlled Vocabulary. The 21th European Conference on Information Retrieval, 2004.
5. Kwok K.L.: Comparing Representation in Chinese Information Retrieval. In Proceedings of the ACM SIGIR-97, pp. 34-41.1997
6. Lee K., Park Y., Choi, K.S.: Document Re-ranking model using clusters. Information Processing & Management V, 2001.

7. Li. P. Research on Improvement of Single Chinese Character Indexing Method, Journal of the China Society for Scientific and Technical Information, Vol. 18 No. 5. 1999.
8. M. Mitra., A. Singhal. and C. Buckley. Improving Automatic Query Expansion. In Proc. ACM SIGIR'98, Aug. 1998.
9. Nie J.Y., Gao J., Zhang J., Zhou M.: On the Use of Words and N-grams for Chinese Information Retrieval. In Proceedings of the Fifth International Workshop on Information Retrieval with Asian Languages, IRAL-2000, pp. 141-148, 2000
10. Qu, Y.L., Xu, G.W.,Wang J.: Rerank Method Based on Individual Thesaurus. Proceedings of NTCIR2 Workshop. 2000.
11. Robertson S.E., Walker, S.: Microsoft Cambridge at TREC-9: Filtering track: In NIST Special Pub. 500-264: The Eight Text Retrieval Conference (TREC-8), pages 151-161, Gaithersburg, MD, 2001.
12. SALTON, G., MCGILL, M.: Introduction to Modern Information Retrieval, McGraw-Hill.1983
13. Schutze, H.: The hypertext concordance: a better back-of-the-book index. Proceedings of First Workshop on Computational Terminology. 101-104, 1998.
14. Yang L.P., Ji D.H., Tang L.: Document Re-ranking Based on Automatically Acquired Key Terms in Chinese Information Retrieval. Proceedings of 20th International Conference on Computational Linguistics (COLING'2004). 2004.
15. Yang L.P., Ji D.H., Tang L.: Chinese Information Retrieval based on Terms and Ontology. Proceedings of NTCIR4 Workshop. 2004.

Appendix: 22 Query Topics in NTCIR3 (First Part of 42 Topics)

001: 查询故宫博物院所举办之千禧汉代文物大展相关内容
(Find information of the exhibition "Art and Culture of the Han Dynasty" in the National Palace Museum)
002: 查询台湾加入WTO後各产业可能面对的问题
(Find possible problems that industries will meet after Taiwan's joining WTO.)
003: 查询大学学术追求卓越计划的相关内容
(Find the content of Program for Promoting Academic Excellence of Universities.)
004: 查询何谓电子商务及电子商务之内容
(Find what E-Commerce is and its contents)
005: 查询朱熔基担任中国总理後所提出的经济改革计划。
(Find Zhu Rong ji's economic reform after his serving as the premier)
006: 查询有关一九九八年诺贝尔物理学奖的相关报导
(Retrieve reports relating to 1998 Nobel Prizes in Physics)
007: 查询有关华航於桃园中正机场失事的相关报导
(Retrieve reports about China Airlines' crash while trying to land at Taoyan international airport.)
008: 查询一九九八年电影「铁达尼号」获得奥斯卡奖之相关报导
(Retrieve reports of Oscar winners, Titanic, in 1998)
009: 查询有中新一号卫星相关报导及评论
(Find reports and comments related to satellite ST1)
010: 查询何谓反圣婴现象及其与圣婴现象的比较与影响
(Find what the anti-El Nino is and the comparison with El Nino)

011: 查询阿里山蒸气火车的历史及其与观光业、森林业的关系
(Find the history of steam locomotive in Mount Ali and its relationship with forestry and sightseeing)
012: 查询曼谷亚洲运动会的相关报导
(Find news of Asian Games in Bangkok)
013: 查询精省的法规内容有哪些，以及台湾省废省後前省长宋楚瑜的态度
(Find the content of Province-refining enactment and Mr. James Soong's attitudes after the Province-refining)
014: 有关於感染电脑病毒引起的问题之文章
(Articles about problems caused by computer virus infection.)
015: 与使用被称为体细胞核移植的技术创造复制牛相关的文章
(Articles relating to the birth of cloned calves using the technique called somatic cell nuclear transfer.)
017: 有关於北野武导演的电影之文章
(Articles relating to Director Takeshi Kitano's films.)
018: 有关最後审判日或世界末日的宗教思想的关连事件。
(Incidents relating to religious thought about doomsday, or the end of the world.)
019: 有关於欧洲货币组织的经济影响之文章
(Articles relating to economic influence of European monetary union.)
020: 有关於日产与雷诺汽车公司资本结合的文章
(Articles relating to a capital tie-up of Nissan Motor Company of Japan and Renault of France.)
021: 有关於1999年西土耳其大地震造成的破坏与救援行动及灾害情形与难民的文章
(Articles relating to the damage, the rescue operations, and the damage situation and victims of a big earthquake in Western Turkey in 1999.)
022: 描述有关柬埔寨的前首相Pol Pat 战争罪行的文章
(Articles describing the war crimes of former Prime Minister Pol Pot of Cambodia.)
023: 有关金大中总统对亚洲的政策之文章
(Articles relating to President Kim Dae-Jung's policy toward Asia)

Merging Case Relations into VSM to Improve Information Retrieval Precision

Wang Hongtao[1], Sun Maosong[1], and Liu Shaoming[2]

[1] The State Key Laboratory of Intelligent Technology and Systems,
Department of Computer Science and Technology,
Tsinghua University, Beijing 100084, China
wanght02@mails.tsinghua.edu.cn
[2] Future Technology Institute, Fuji Xerox Co. Ltd, Japan
Liu.shaoming@fujixerox.co.jp

Abstract. This paper presents an approach that merges case relations into the well-known Vector Space Model (VSM), leading to a new model named C-VSM (Case relation-based VSM). A Chinese case system with 23 case relations is established, and a Chinese Olympic news corpus of 7,662 sentences, denoted COCS, is constructed by manual annotation with these 23 case relations. We use 50 queries on COCS as a test set. Experimental results on the test set show that C-VSM outperforms W-VSM (Word-based VSM) by 3.4% on the average 11-point precision. It is worth pointing out that almost all the previous studies on semantic IR obtained no better, even worse, results than W-VSM, our work thus validates the usefulness of case relations in IR through the validation is still preliminary. The proposed model is believed to be language-independent.

1 Introduction

A majority of traditional models of information retrieval (IR) mainly make use of surface linguistic information such as words/terms. It is reasonable to expect better retrieval results if we can exploit deep linguistic information further. Previous studies of this sort have been carried out at both syntactic and semantic levels. Most of them focused on the former, because the recognition of syntactic structures is easier than that of semantic structures. Syntactic information possibly exploited in IR can be a simple syntactic relation between a pair of words, and can also be a complex structure tree. The use of simple syntactic relations in IR has found a small improvement in retrieval effectiveness (Croft, Turtle and Lewis, 1991; Hyoudo, Niimi and Ikeda, 1998). But the results of using complex structure trees are worse than keyword matching (Smeaton, O'Donnell and Kelledy, 1995).

It is natural to assume that semantic information is more useful in IR since it can capture the meaning of a sentence more precisely than syntax. Semantic information, both intra-sentential and inter-sentential, is usually represented by the so-called semantic relations between various entities involved.

Case relation is an intra-sentential semantic relation that exists between the core verb and other constituents of a sentence (Fillmore, 1968; Somers, 1987). Lewis

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 584–592, 2005.

(1984) addressed the possibility of IR based on case relation matching. Lewis' major hypothesis is that if index terms of a query and indexed terms of a document are more likely to co-occur with similar case relations, there will be a more significant similarity between the query and the document. In other words, if a document is judged to be associated with a query, the document would not only share many identical index terms with the query, but would share similar case relations between those index terms as well. Lewis just put forward this idea, without giving any experiment result. Lu (1990) proposed a simple structure tree-matching method in IR, according to the case-frame system in (Young, 1973), to formulate semantic meaning of sentences. The Experiment on a small test set demonstrated that the performance of this proposed method is worse than the vector-based keyword matching. Lu's study also suggested that the strict matching between case relations may hurt the performance of IR due to the resulting data sparseness problem. Liu (1997) incorporated the word concept, abstracted as the semantic category of a word, and the semantic role of the concept in a sentence into the Vector Space Model (VSM), taking a 2-tuple (word concept, semantic role of the concept), instead of the literal word, as the basic element of vectors. This method is named 'partial relation matching' because it is not based on full semantic structure tree (we shall continue to use this term in Section 3.1). The vector dimension can be controlled using upper-lower relations between semantic categories. The method yields an increasing in recall and a drop in precision, and almost same F-measure compared to conventional word-based VSM (W-VSM).

Inter-sentential semantic relations often exist between words beyond separate sentences in text. Khoo (2001) made an intensive study on exploring just one relation – the cause-effect relation. An algorithm is developed for recognizing cause-effect relations in text automatically. But the experiment on the Wall Street Journal corpus did not give better results than proximity-based word matching.

As can be observed, previous efforts of taking both syntactic and semantic information into consideration in IR have not reached satisfactory performance so far, and, obviously, the research concerned is very preliminary. This implies that there may be a large room of improvement for relation-based IR (in particular, for semantic relation-based IR).

This paper tries to introduce case relation into VSM, leading to a C-VSM (Case relation-based Vector Space Model). In C-VSM, the classic TF*IDF formula is adjusted by multiplying a weighting factor to each word according to its case relation in the sentence. Experiments on a test set show that the average 11-point precision of this model reaches 87.2% and outperforms the baseline, W-VSM, by 3.4%.

The rest of the paper is organized as follows: Section 2 describes a semantically annotated Chinese corpus used and the case relations defined in it, Section 3 discusses experiment-based design of the algorithm, in the context of comparing with W-VSM and the strategy of partial relation matching, and Section 4 is conclusion.

2 Semantically Annotated Corpus and Case Relations Defined

Case relations are semantic relations that hold between the core verb and other constituents in a sentence (Fillmore, 1968; Somers, 1987). For example, in the sentence *Harry loves Sally*, the case relation *experiencer* holds between *Harry* and

love, and the case relation *patient* holds between *love* and *Sally*. The verb *love* is said to assign the case relation of *experiencer* to *Harry* and the case relation of *patient* to *Sally*. Case relations can be sub-categorized into two groups, i.e., essential case relations and peripheral case relations. Essential case relations are those necessary for the verb while peripheral case relations are those optional to the verb.

Inspired by Fillmore's theory, Lin (1999) designed a Chinese case system with 22 cases. We simply adopt Lin's system with a minor expansion by adding one case particularly for the Olympic domain. As a consequence, a Chinese case system with 23 cases is established. A Chinese Olympic news corpus of 7,662 sentences, denoted COCS, is then constructed by manual annotation with these 23 case relations. Case relations defined and their distribution in COCS are listed in Table 1.

Table 1. A Chinese case system and the distribution of case relations in a semantically annotated Chinese corpus

Case Relation	Symbol	Coverage for case relations in COCS (%)
Agent (施事)	S	21.4
Experiencer (当事)	D	17
Genitive (领事)	L	0.12
Patient （受事）	O	12.6
Accusative (客事)	K	4.7
Comitative (共事)	Y	3.7
Link (系事)	X	4.3
Type (类别)	B	0.1
Object (对象)	T	2.4
Result (结果)	R	6.8
Manner (方式)	Q	3.3
Quantity (数量)	N	1.3
Scope (范围)	E	8.7
Time (时间)	H	8.8
Part (分事)	F	0.15
Benchmark (基准)	J	0.6
Instrument (工具)	I	0.03
Material (材料)	M	0
Location (位置)	P	2.8
Direction (方向)	A	0.07
Warranty (依据)	W	0.45
Cause (原因)	C	0.6
Purpose (目的)	G	0.4

To ensure the quality of the annotated corpus, a two-round annotation is performed. An sample sentence from COCS is as follows:

[S 中国/ns 选手/n 龚智超/nr]S1S2 [D 周五/t]H [D 在/p 奥运会/j 羽毛球/n
Chinese player Gong Zhichao Friday in Olympic Games badminton
女单/j 决赛/vn 中/f]E ，/w [D 以/p 2/m ：/w 0/m]Q
Women's singles final within with 2 : 0
[P 战胜/v]V1 [O 前/f 世界/n 排名/v 第一/m 的/u 丹麦/ns 名将/n
defeat former world ranking NO.1 of Denmark well-known player
马尔廷/nr]T1 ，/w [D 为/p 中国/ns 代表团/n]Y2 [P 夺得/v]V2 [O 本届/r
Camilla Martin for Chinese delegacy win this
奥运会()/j 上/f 的/u 第 1 4 ()/m 块/q 金牌/n]R2 。/w
Olympic Games in of the fifth piece Gold medal

(Gong Zhichao from China, defeated Camilla Martin, a well-known and former world ranking No.1 player from Denmark, on Friday, yielding the Women's Singles Badminton title. It is the 14th gold medal China has won in this Olympic Games.)

where: 'w/x' stands for part-of-speech x for word w, '[……]' gives a chunk in a given sentence, '[X' indicates the grammatical function of the associated chunk in the sentence (for example, '[S' means Subject), ']X#' indicates the case relation of the associated chunk to the core verb of the sentence ']V#'(for example, ']S' means Agent of V), and '#' is a sequence number for multiple sentences. The word underlined is the head of the associated chunk.

3 Experiment-Based Algorithm Design

COCS concerns news for Olympic sports. Each article in COCS is usually quite short, – on average, there are only 2-3 sentences in each article. So both query and retrieval in experiments here are based on a single sentence rather than a full article. We selected 100 sentences from COCS as queries, and hand-crafted the retrieval outputs, 635 sentences in total, for these 100 queries accordingly. Then we randomly split this data set into two equal parts: 50 queries for parameter estimation of the proposed model, and the remaining 50 queries for testing. W-VSM is regarded as the baseline throughout the experiments.

3.1 Solution 1: Partial Relation Matching

In attempting to incorporate case relations into IR, we try a solution similar to 'partial relation matching' at first. Each word and its associated case relation in a sentence constitute a 2-tuple (Note: the case relation of a word is conveyed from the case relation of the chunk containing that word), and this 2-tuple is used as an index item in vectors. For example, there exist three index items, (龚智超, S), (马尔廷, T) and (战胜, V), for the sentence "龚智超(*Gong Zhichao*) 战胜(*defeat*) 马尔廷(*Camilla Martin*)" (*Gong Zhichao defeated Camilla Martin*). Each 2-tuple is then weighted with TF*IDF where TF is the frequency of the 2-tuple in a sentence and IDF is the inverse sentence frequency of the word involved in the 2-tuple. We say that an index item (or a 2-tuple) is matched if and only if its word and case relation are matched simultaneously.

We explore two strategies in the experiment. Strategy 1 takes all of the 2-tuples in sentences as index items, whereas strategy 2 only takes 2-tuples relating to heads of chunks as index items (the others are still simply indexed as words). We compare the two strategies with the baseline, W-VSM, as illustrated in Fig. 1.

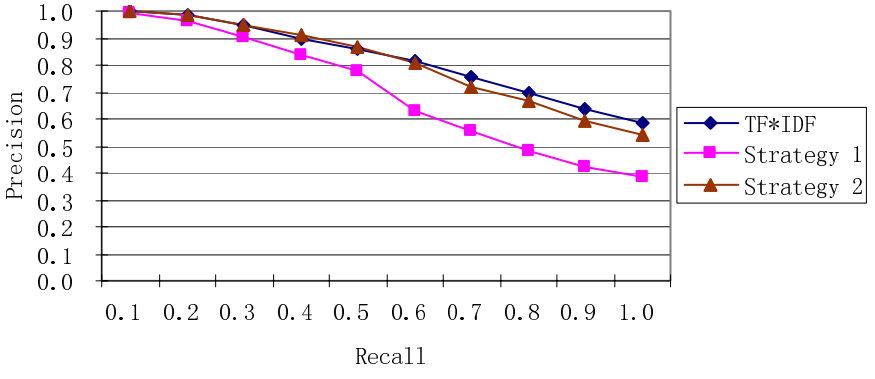


Fig. 1. Comparisons between partial relation matching and W-VSM

As can be seen, strategy 1 is poorer than strategy 2, and both strategies do not give better results than the baseline, suggesting that solution 1 may not be feasible for integration.

3.2 Solution 2: Merging Case Relations into Word Weighting

In experience gained in Section 3.1, we find that the condition for matching two 2-tuples is too strict. We thus present another possible solution: instead of using a case relation explicitly in a 2-tuple, we use it in a way of being viewed as just a weighting factor added to the traditional W-VSM model, that is, we still use a word as an index item, but re-estimate its TF*IDF weighting by multiplying a factor according to the type of the associated case relation. As stated earlier, the case relation of any word in a chunk is derived from the case relation of the chunk containing that word.

Obviously, the contributions of case relations to the meaning of a sentence are not identical. We categorize case relations into several groups in order to decrease the number of parameters to be estimated in the model. Case relations falling into the same group will be given an identical weighting factor, meanwhile those falling into different groups will be assigned distinct factors.

Agent, Experiencer and Genitive all belong to the source of an action, so we classify these 3 case relations into a group, denoted Group 1; In parallel, Patient, Result, Link, Part, Objective, Type and Accusative, the direct target of an action, are classified into another group, denoted Group 2; Verb is not a case relation, but deserves special attention in weighting, so it is treated as a separate group, denoted Group 3; Comitative and Benchmark belong to the indirect target of an action, so we classify them into a group, denoted Group 4; Scope describes an important situational aspect of an action (especially for sports news), we let it stand alone as a group,

denoted Group 5; and, all the rest case relations are classified into a group, denoted Group 6. The classification for case relations is summarized in Table 2.

Table 2. Classification for case relations

Group	Case Relation	Group	Case Relation
1	S D L	4	Y J
2	O R X F T B K	5	E
3	V	6	I, M, P, A, W, C, G, Q, N, H

Thus, we have six weighting factors that need to be estimated.

Suppose $wt(w)$ is the TF*IDF value of a word w in W-VSM, $q(w)$ is its weighting factor according to case relation of w in a sentence, then we have an adjusted weight for w:

$$wt(w)*q(w) \tag{1}$$

Furthermore, the head of a chunk is expected to be more significant than the other words in the chunk for IR. So we particularly design a set of weighting factors for heads, resulting in another six weighting factors, in accordance with the six groups in Table 2 respectively.

Consequently, the weighting for a head w is adjusted as:

$$wt(w)*r(w) \tag{2}$$

where $r(w)$ is head-related weighting factor according to case relation of w in a sentence.

We need to determine 12 weighting factors in total. We fixed the factor for Group 6 to be 1. Genetic algorithm (GA) is used to train the rest 10 factors based on the 50 queries in the training set. The setting of GA is: binary encoding, 40 populations in a generation, and the average 11-point precision as fitness function. The weighting factors obtained from GA are listed in Table 3.

Table 3. Weighting factors obtained from GA

Type	Group 1	Group 2	Group 3	Group 4	Group 5	Group 6
q	2.147	1.575	1.315	1.100	1.545	1
r	2.761	2.144	2.383	2.900	0.133	1

Now, we yield a new IR model – Case Relation-based Vector Space Model, denoted C-VSM. Fig. 2 compares the performance of C-VSM with that of W-VSM on the test set.

As shown in Fig. 2, C-VSM outperforms W-VSM: there is a 3.4% improvement on the average 11-point precision.

We demonstrate the effectiveness of C-VSM with the following example.

Query：2000 年 9 月 25 日，北京时间周一下午刚刚结束的女子 400 米决赛上，澳大利亚名将弗里曼夺得金牌。

(Fuliman, an Australian well-known player, won the gold medal of women 400m in the afternoon, Monday 25 Sep 2000, Beijing time.)

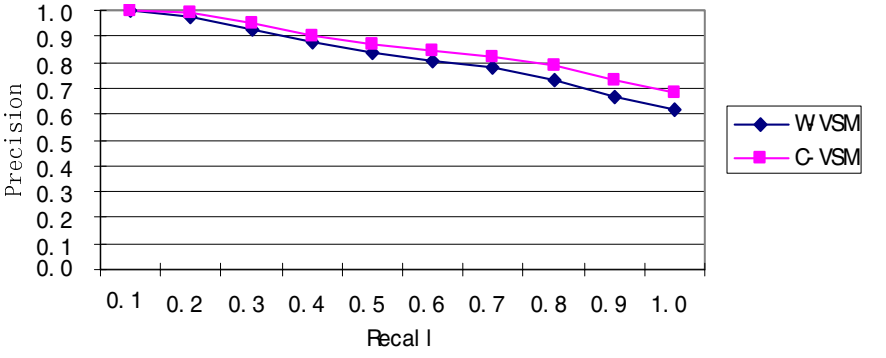


Fig. 2. Comparison between C-VSM and W-VSM

The top 3 retrieved sentences from C-VSM and W-VSM for this query are given in Table 4. Sentences with '*' are correctly retrieved results according to human judgments.

Table 4. An example for comparison between C-VSM and W-VSM

Response from C-VSM	Response from W-VSM
* 2000 年 9 月 25 日，北京时间周一下午刚刚结束的女子 400 米决赛上，澳大利亚名将弗里曼夺得金牌。*(Fuliman, an Australian well-known player, won the gold medal of women 400m in the afternoon, Monday 25 Sep 2000, Beijing time.)*	* 2000 年 9 月 25 日，北京时间周一下午刚刚结束的女子 400 米决赛上，澳大利亚名将弗里曼夺得金牌。*(Fuliman, an Australian well-known player, won the gold medal of women 400m in the afternoon, Monday 25 Sep 2000, Beijing time.)*
* 澳大利亚名将弗里曼夺得女子 400 米金牌。*(Fuliman, an Australian well-known player, won the gold medal of women 400m.)*	2000 年 9 月 25 日，北京时间周一下午刚刚结束的男子 400 米决赛中，美国名将约翰逊夺得金牌。*(Michael Johnson, an American well-known player, won the gold medal of men 400m in the afternoon, Monday 25 Sep 2000, Beijing time.)*
* 9 月 25 日，澳大利亚选手弗里曼在奥运会女子 400 米决赛中获得金牌。*(Fuliman, an Australian player, won the gold medal of women 400m on 25 Sep.)*	2000 年 9 月 25 日，北京时间周一下午刚刚结束的女子撑杆跳决赛上，美国德拉吉拉夺得金牌。*(Dragan, an American player, won the gold medal of women's pole vault in the afternoon, Monday 25 Sep 2000, Beijing time.)*

3.3 Smoothing with Similarity Between Words

An observation on C-VSM indicates that many unmatched words share the same or similar meanings, as '获得'(*gain*) and '夺得'(*seize*), and '金牌'*(gold medal*) and '冠军'(*champion*). A possible improvement for C-VSM is thus to resort to a sort of thesaurus as a means of smoothing as matching between two words is being done.

A thesaurus of words is constructed by automatic clustering on COCS. We consider the context of a word w to be a window with k words on the left and right of w respectively (k = 1 here). Two words are said to be semantically associated with each other if their contexts are similar in a document collection. In the process of automatic clustering, TF*IDF is used for word weighting, and cosine is used for similarity measuring.

In the process of retrieval, for any unmatched word w_1 in a query, we compute similarities between w_1 and any word in the target sentence. The word with the biggest similarity in the target sentence is fixed, denoted w_2. If the similarity between w_1 and w_2, $Sim_{w1,w2}$, is greater than a threshold, then we assert that w_1 is approximately matched with w_2, and the weighting of w_1 is estimated by:

$$Sim_{w1,w2}*TF_{w2}*IDF_{w1} \tag{3}$$

We try three strategies:

Strategy A: C-VSM + Approximate matching on all unmatched words in the query;
Strategy B: C-VSM + Approximate matching on all unmatched head words in the query;
Strategy C: C-VSM + Approximate matching only on the core verb in the query.

Experimental results are listed in Table 5:

Table 5. Experimental results for introducing word similarity into IR

Recall	Precision of C-VSM	Precision of Strategy A	Precision of Strategy B	Precision of Strategy C
0.1	1	1	1	1
0.2	0.993	0.987	0.990	0.993
0.3	0.953	0.927	0.944	0.952
0.4	0.911	0.896	0.911	0.914
0.5	0.880	0.869	0.871	0.880
0.6	0.853	0.820	0.843	0.852
0.7	0.822	0.806	0.825	0.824
0.8	0.791	0.778	0.788	0.792
0.9	0.725	0.706	0.715	0.729
1	0.669	0.654	0.661	0.678
The average 11-point precision	0.872	0.858	0.868	0.874

We can see from Table 5 that the noise brought by word similarity may hurt the performance of IR: only strategy C gains a bit improvement on the average 11-point precision (0.2%) compared to C-VSM.

4 Conclusion

Experimental results in the paper indicate that case relations can benefit the effectiveness of IR if they are properly combined with W-VSM, though the improvement is not as significant as expected. Approximate matching between words may also be beneficial to case relation-based IR.

We believe the largest contribution of this work is that we obtain a better performance with C-VSM (Note that the model is in fact language-independent) than W-VSM, whereas previous studies on semantic IR often obtained no better, even worse results. We preliminarily validate by experiments an assumption that semantic information is useful for IR, though the road ahead in this direction is still very long.

Acknowledgements. This research is sponsored by Fuji Xerox Co. Ltd.

References

1. Khoo, S.G.: Using Cause-effect Relations in Text to Improve Information Retrieval Precision. Information Processing and Management, 37, (2001) 119-145
2. Liu, G.Z: Semantic Vector Space Model: Implementation and Evaluation. Journal of the American Society for Information Science, 48(5), (1997) 395-417
3. Lin, X.G.: Lexical Semantics and Computational Linguistics. YuWen Press, Beijing, (1999)
4. Lu, X.: An Application of Case Relations to Document Retrieval, Doctoral dissertation, University of Western Ontario, (1990)
5. Fillmore, C.J.: The Case for Case. In: Universals in Linguistic Theory, New York: Holt, Rinehart and Winston, Inc, (1968)
6. Somers, H.L.: Valency and Case in Computational Linguistics. Edinburgh University Press, (1987)
7. Lewis, D.A.: Case Grammar and Functional Relations. Doctoral dissertation, University of Western Ontario, (1984)
8. Young, C.: Development of Language Analysis Procedures with Application to Automatic Indexing. Doctoral dissertation, The Ohio State University, (1973)
9. Croft, W.B., Turtle, H.R., Lewis D.D.: The Use of Phrases and Structured Queries in Information Retrieval. In: Proc. of the Fourteenth Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval, (1991)
10. Hyoudo, Y., Niimi, K., Ikeda, T.: Comparison between Proximity Operation and Dependency Operation in Japanese Full-text Retrieval. In: Proc. of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, (1998)
11. Smeaton, A.F., O'Donnell, R., Kelledy, F.: Indexing Structures Derived from Syntax in TREC-3: System Description. In: Overview of the Third Text REtrieval Conference (TREC-3), National Institute of Standards and Technology Special Publication 500-225, (1995) 55-67

Evaluating Document-to-Document Relevance Based on Document Language Model: Modeling, Implementation and Performance Evaluation*

Ge Yu, Xiaoguang Li, Yubin Bao, and Daling Wang

School of Information Science and Engineering, Northeastern University
Shenyang 110004, P.R.China
xgli7312@mail.163.com, yuge@mail.neu.edu.cn

Abstract. To evaluate document-to-document relevance is very important to many advanced applications such as IR, text mining and natural language processing. Since it is very hard to define document relevance in a mathematic way on account of users' uncertainty, the concept of topical relevance is widely accepted by most of research fields. It suggests that a document relevance model should explain whether the document representation describes its topical contents and the matching method reveals the topical differences among the documents. However, the current document-to-document relevance models, such as vector space model, string distance, don't put explicitly emphasis on the perspective of topical relevance. This paper exploits a document language model to represent the document topical content and explains why it can reveal the document topics and then establishes two distributional similarity measure based on the document language model to evaluate document-to-document relevance. The experiment on the TREC testing collection is made to compare it with the vector space model, and the results show that the Kullback-Leibler divergence measure with Jelinek-Mercer smoothing outperforms the vector space model significantly.

1 Introduction

The evaluation of document relevance is a very important task for many advanced applications such as information retrieval (IR), text mining, natural language processing, and has been extensively studied until now. In general, document relevance can be divided into two categories: query-to-document relevance and document-to-document relevance [1]. The former focuses on document relevance to a user's query, while the latter aims at an entire document in contrast to a small number of words in a specific user query. On account of the inherent redundancies and ambiguities in textual descriptions and high dimension problem, document-to-document relevance is more complex.

* Supported by the National Natural Science Foundation of China under Grant No.60173051 and the Teaching and Research Award Program for Outstanding Young Teachers in Higher Education Institution of the Ministry of Education, China.

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 593–603, 2005.

Because it is hard to define document relevance in a mathematic way on account of users' intention uncertainty, the concept of topical relevance is widely accepted by most of research fields [3]. It suggests that a document relevance model should explain whether the representation of document describes its topical content, and whether the matching method reveals its topical difference from others. While these issues are very important to evaluate the relevance, the current models, such as vector space model, string distance model, don't explain explicitly whether they are based on the topical relevance. The purpose of our paper is to devise a document representation from the viewpoint of topical relevance. Thus, we exploit a document language model to capture the document topical content, and two distributional similarity measures are established to evaluate document-to-document relevance. The experiment on the TREC testing collection shows that our method outperforms vector space model.

The rest of the paper is organized as follows. Section 2 briefly reviews document-to-document relevance in previous work and the applications of language model in information retrieval. Section 3 discusses document-to-document relevance. The document language model and model estimation are introduced in section 4 and 5 respectively. Section 6 presents the measures of evaluating document relevance based on document language model. The experiments to evaluate the performance of measures are presented in section 7. Section 8 gives our conclusion.

2 Related Work

Language model, also called statistical language model (*SLM*), is a distribution to capture the generation rule in natural language. After the first model was proposed in 1980, the *SLM* was used in many applications such as speech recognition, optic character recognition, machine translation. Only very recently, since 1998, the *SLM* is applied to information retrieval as query-to-document relevance model [6]. Due to statistic foundation and promising performance of language model, in the past it shows a remarkably large number of publications. Its basic idea is to compute the conditional probability $P(Q|D)$, i.e. the probability of generating a query Q given the observation of a document D [6, 7, 8, 4, 9, 10]. These methods emphasized mainly on the estimation accuracy of language models. They are not suitable for the application of document-to-document relevance, because a document, compared a query, consists of many topic-irrelevant words so that generating measure is usually invalid.

As for document-to-document relevance, there are three kinds of models in the past: string distance model, statistics of term model and structure model. String distance model [11, 12] considers the relevance as the distance, which is the amount of the difference between strings e.g., the difference between two strings is the number of insertions, deletions, or substitutions of letters required to transform one string into another. Document components or structure model [13] considers the structure or components of documents to judge the similarity between two documents. This approach is well suited for situations in which documents are of a specific type, or have special components or structure. In the case of research papers, for instance, they have

similar structure and components, such as title, abstract, keywords, references. Statistics of term model [14] considers the frequency of terms in the documents to judge the similarity. The representative model is Vector Space Model (*VSM*), which is the most popular similarity model in the field of text mining and information retrieval. The most typical model in *VSM* is $tf \bullet log(N/df)$ weight family in SMART and cosine-distance, where *tf* is the term frequency in a given document, and *df* is the number of documents that include that term.

Although these three models have achieved a promising performance in many applications, there are actually little or no explicit explanations that whether they are based on topical relevance. The term weight is always determined by the experiments and users do not intuitively understand the document representation and the judgment of relevance obtained by these models.

3 Document-to-Document Relevance

In general, document relevance is divided into two cases [2, 3]: system-oriented relevance and user-oriented relevance. System-oriented relevance, also called topical relevance, is intuitively based on the assumption that the document representation can describe its topic, and the relevance judgment is measured by the matching degree between the representations of requirement and document. In this case document relevance is considered as the system's property and irrelevant to the users, i.e. if the requirement representations are the same, regardless of the users' intention, the relevance judgment to a document is uniform. User-oriented relevance emphasizes that the requirement representation and the matching function can reveal users' intention. But because of the users' uncertainty, it is very hard to define user-oriented relevance in a mathematic way so that the topical relevance has been accepted by most of research fields.

Since there's no a mathematic definition of document relevance up to now, we give only a general definition as follow.

Definition 1. Given the topical representations d and d', *Document-to-document topical relevance* is defined as a matching function $FT(d, d')$, $FT \in \boldsymbol{R}^+$. If the document d' is more relevant topically to d than d'', then $FT(d, d') > FT(d, d'')$.

As described above, there are two issues when applying the topical relevance into practice: (1) how to devise the document representation, which implies the document topical content, and (2) how to devise the matching measure between the representations as the criteria of relevance judgment.

4 Document Language Model

In this section we introduce document language model as document representation, and explain why it can capture the topical content of document in some sense. Firstly we give some definitions.

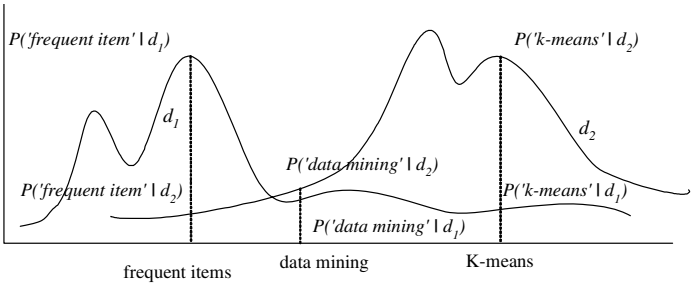


Fig. 1. Illustration for document language model

Definition 2. A *term* is the feature of the document, which may be a word, a phrase or even a sentence.

Definition 3. A *document d* is the set of terms occurred in *d*. In this paper, we assume that the term is independent with each other and not take the sequence into account, so the *d* can be considered as the *bag-of-terms*.

Definition 4. Given a document collection $\boldsymbol{D}$, the *vocabulary* $\boldsymbol{V}$ is a term set, $\boldsymbol{V} = \{x | x \in d, \forall\, d \in \boldsymbol{D}\}$

Definition 5. Given a document $d \in \boldsymbol{D}$, we define *document language model* (*DLM*) for *d* as a joint distribution function $P(x, d)$, where $x \in \boldsymbol{V}$.

Given a *DLM* $P(x, d)$, then it can be denoted equivalently as Formula 1.

$$P(x, d) = P(x \mid d)\, P(d) \tag{1}$$

$P(d)$ is prior probability of document *d*, in the simplest case, assumed to be uniform, and so does not affect document language model, then Formula 1 can be expressed approximately by Formula 2.

$$P(x, d) = P(x \mid d)\, P(d) \propto P(x \mid d) \tag{2}$$

We assume that every document in a collection describes only one topic, and given a topic, users will select the relevant words to this topic to describe the document content, and thus the probability of selecting a word varies with the topic. The conditional probability $P(x|d)$ can be interpreted as the probability of "generating" the term *x* given the document *d*. Obviously $P(x \mid d)$ is different with respect to document, i.e. $P(x \mid d)$ is discriminative from $P(x \mid d')$ when *d* and *d'* are about the different content. The more relevant the document contents, the closer the conditional distributions.

For example, see Figure 1, given two documents d_1 and d_2, which are about the association rules and cluster methods respectively, it is evident that $P(\text{'}frequent\ item\text{'} \mid d_1) > P(\text{'}frequent\ item\text{'} \mid d_2)$, $P(\text{'}k\text{-}means\text{'} \mid d_2) > P(\text{'}k\text{-}means\text{'} \mid d_1)$, and $P(\text{'}data\ mining\text{'} \mid d_1) \sim P(\text{'}data\ mining\text{'} \mid d_2)$. So we can say in a sense that the representation based on the *DLM* reveals the document topical content.

5 Model Estimation

The simplest method of model estimation is the *maximum likelihood estimate* (*MLE*) given by relative counts (Formula 3).

$$P_{ML}(x \mid d) = \frac{c(x,d)}{\sum_{x'} c(x',d)} \tag{3}$$

where $x \in d$, $c(x', d)$ is the counts of term x' occurring in the d.

When applying the *DLM* into practices, smoothing is absolutely necessary on account of the sparseness problem. The term smoothing refers to the adjustment of maximum likelihood estimator of a language model so that it will be more accurate, and is not assigned a zero probability to unseen term that do not appear in a given document. In general, all smoothing methods are trying to discount the probabilities of the terms occurred in the text, and then to assign the extra probability mass to the unseen terms according to some fallback model. For a given document collection, it makes much sense to exploit the collection language model as the fallback model to solve the sparseness problem.

Definition 6. Given a document collection $\boldsymbol{D}$ and vocabulary $\boldsymbol{V}$, the *fallback model* is defined as $P(x \mid D) = \sum_{d \in D} c(x,d) \Big/ \sum_{x' \in V} \sum_{d' \in D} c(x',d')$, $x \in \boldsymbol{V}$, where $c(x, d)$ is the counts of term x occurring in the d. Obviously $\sum_{x \in V} P(x \mid D) = 1$.

There are many smoothing methods [4], such as *Katz smoothing* and *Good-Turing smoothing*. Because of the efficiency constraint, we consider only two popular smoothing methods: *Jelinek-Mercer smoothing* and *Bayesian smoothing*.

Jelinek-Mercer smoothing method. This method involves a linear interpolation of the maximum likelihood model with the collection model, using a coefficient λ to control the influence of each model.

$$P_\lambda(x \mid d) = \lambda P_{ML}(x \mid d) + (1-\lambda) P(x \mid D) \tag{4}$$

Bayesian smoothing using Dirichlet prior, also called *Dirichlet smoothing*. It considers a language model as a multinomial distribution, for which the conjugate prior for Bayesian analysis is the *Dirichlet* distribution with parameters $\vec{\mu} = \{\mu P(x_1 \mid D), \cdots, \mu P(x_n \mid D)\}$. Thus, the model is given by

$$P_\mu(x \mid d) = \frac{c(x,d) + \mu p(x \mid D)}{\sum_{x' \in d} c(x',d) + \mu} \tag{5}$$

6 Matching Measure

Given the *DLMs* of two documents, d and d', the relevance evaluation can be done through computing their distributional similarity directly. We denote it by *distributional similarity measure*, *DSM*. In this paper, *Kullback-Leibler divergence* and *Bhattacharyya Bound* [5] are applied to comparing the distributional similarity.

Kullback-Leibler divergence (*KL divergence*), also known as the relative entropy, is a measure to show the difference between two probability distributions. Since *KL divergence* is not symmetric, we use the symmetric *KL divergence*, as shown in Formula 6.

$$FT_{MKL}(d,d') = \sum_{x \in V} (P(x \mid d) - P(x \mid d')) \log \frac{P(x \mid d)}{P(x \mid d')} \tag{6}$$

Bhattacharyya Bound, a special case of the *Chernoff* distance, is another measure to define the error upper bound between the probability distribution, also considered as the similarity. The computation of the distributional similarity between d and d' in terms of *Bhattacharyya Bound* is

$$FT_{J_B}(d,d') = -\ln \sum_{x \in V} [P(x \mid d) P(x \mid d')]^{1/2} \tag{7}$$

Note that there are four cases when computing the similarity. The subscript "*jm*" and "*dl*" represent *Jelinek-Mercer* and *Dirichlet* smoothing methods respectively, $l = \sum_{x \in d} c(x,d)$, $l' = \sum_{x \in d'} c(x,d')$ and $P_c(x) = P(x \mid D)$.

1. $(x \in d) \wedge (x \in d')$, i.e. the term x appears in the document d and d';

$$p^1_{jm} = (\lambda c(x,d)/l + (1-\lambda) p_c(x)) \qquad p^1_{jm}{}' = (\lambda c(x,d')/l' + (1-\lambda) p_c(x))$$
$$p^1_{dl} = \frac{c(x,d) + \mu p_c(x)}{l + \mu} \qquad p^1_{dl}{}' = \frac{c(x,d') + \mu p_c(x)}{l' + \mu}$$

2. $(x \in d) \wedge (x \notin d')$, i.e. the term x appears in the document d but not in the document d';

$$p^2_{jm} = (\lambda c(x,d)/l + (1-\lambda) p_c(x)) \qquad p^2_{jm}{}' = (1-\lambda) p_c(x)$$
$$p^2_{dl} = \frac{c(x,d) + \mu p_c(x)}{l + \mu} \qquad p^2_{dl}{}' = \frac{\mu p_c(x)}{l' + \mu}$$

3. $(x \notin d) \wedge (x \in d')$, i.e. the term x appears in the document d' but not in the document d;

$$p^3_{jm} = (1-\lambda) p_c(x) \qquad p^3_{jm}{}' = (\lambda c(x,d')/l' + (1-\lambda) p_c(x))$$
$$p^3_{dl} = \frac{\mu p_c(x)}{l + \mu} \qquad p^3_{dl}{}' = \frac{c(x,d') + \mu p_c(x)}{l' + \mu}$$

4. $(x \notin d) \wedge (x \notin d')$, i.e. the term x does not appears in the document d and d'.

$$p^4_{jm} = p^4_{jm}{}' = (1-\lambda) p_c(x) \qquad p^4_{dl} = p^4_{dl}{}' = \frac{\mu p_c(x)}{l + \mu}$$

In a conclusion, we give the formulas of computing the distributional similarity corresponding to smoothing methods and similarity measures, see Table 1.

Table 1. Summary of Similarity Formulas

Measure	Smooth	Formulas
MKL	JM	$\sum_{(t\in d)\wedge(t\in d')}(p^1_{jm}-p^1_{jm}')\log\frac{p^1_{jm}}{p^1_{jm}'}+\sum_{(t\in d)\wedge(t\notin d')}(p^2_{jm}-p^2_{jm}')\log\frac{p^2_{jm}}{p^2_{jm}'}$ $+\sum_{(t\notin d)\wedge(t\in d')}(p^3_{jm}-p^3_{jm}')\log\frac{p^3_{jm}}{p^3_{jm}'}$
	DL	$\sum_{(t\in d)\wedge(t\in d')}(p^1_{jm}-p^1_{jm}')\log\frac{p^1_{jm}}{p^1_{jm}'}+\sum_{(t\in d)\wedge(t\notin d')}(p^2_{jm}-p^2_{jm}')\log\frac{p^2_{jm}}{p^2_{jm}'}$ $+\sum_{(t\notin d)\wedge(t\in d')}(p^3_{jm}-p^3_{jm}')\log\frac{p^3_{jm}}{p^3_{jm}'}+(1-\sum_{(t\in d)\vee(t\in d')}p_c(t))(\frac{\mu}{dl+\mu}-\frac{\mu}{dl'+\mu})\log\left(\frac{dl'+\mu}{dl+\mu}\right)$
J_B	JM	$-\ln\sum_{(t\in d)\wedge(t\in d')}(p^1_{jm}p^1_{jm}')^{1/2}+\sum_{(t\in d)\wedge(t\notin d')}(p^2_{jm}p^2_{jm}')^{1/2}$ $+\sum_{(t\notin d)\wedge(t\in d')}(p^3_{jm}p^3_{jm}')^{1/2}+(1-\lambda)(1-\sum_{(t\in d)\vee(t\in d')}p_c(t))$
	DL	$-\ln\sum_{(t\in d)\wedge(t\in d')}(p^1_{jm}p^1_{jm}')^{1/2}+\sum_{(t\in d)\wedge(t\notin d')}(p^2_{jm}p^2_{jm}')^{1/2}$ $+\sum_{(t\notin d)\wedge(t\in d')}(p^3_{jm}p^3_{jm}')^{1/2}+\mu/((dl+\mu)(dl'+\mu))^{1/2}(1-\sum_{(t\in d)\vee(t\in d')}p_c(t))$

7 Performance Comparison and Analysis

In this section we study the behavior of *DLM*. Because the *VSM* is the most popular document relevance model up to now, we compare the *VSM* with the distributional similarity measures over the test collection, and then examine the sensitivity of *DLM* to the different model parameters. Here in VSM the term's weight is *tf log*(*N/df*).

7.1 Corpus

The Text REtrieval Conference (TREC) was started in 1992. Its purpose was to support research within the information retrieval community by providing the infrastructure necessary for large-scale evaluation of information retrieval methodologies. Up to now it has been an acknowledged testbed for information retrieval.

We select the OHSUMED87 (http://trec.nist.gov) corpus of TREC-9 as the testing collection. OHSUMED87 corpus is a set of 54709 references from MEDLINE, the on-line medical information database, consisting of titles and/or abstracts from 270 medical journals in 1987. The available fields are title, abstract, MeSH, indexing terms, author, source, and publication type. Title and abstract fields are used in our experiment. There are 63 topics that consist of title and description, and the relevance judgments to each topic. In order to evaluate the model performance in the different corpus, we construct the OHSU87P corpus that contain only the references corre-

sponding to 63 topics and OHSU87W corpus that contain the whole OHSUMED87 references. In our experiment, the term in the vocabulary is stemmed with a porter stemmer, and then we remove the stop word according to an artificial dictionary. We select randomly 5 documents from OHSUMED87 as the testing samples and the class label of document retrieved can make the relevance judgment to the sample.

7.2 Evaluation Criteria

The retrieved set consists of a ranked list of documents according to the relevant degree to the sample, which implies standard criteria used in the evaluation of information retrieval, *recall* (Formula 8) and *precision* (Formula 9).

$$recall = \frac{\text{number of relevant document retrieved}}{\text{number of relevant document in corpus}} \tag{8}$$

$$precision = \frac{\text{number of relevant document retrieved}}{\text{number of document retrieved}} \tag{9}$$

In this paper, we select the precision at 11 points (Formula 10), and the precision at 5, 10 and 20 documents retrieved (Formula 11) as evaluation criteria.

$$precision \text{ at } 11 \text{ points} = \frac{\sum_{i=0}^{num} p_\lambda}{num},\ \lambda=\{0\ldots1.0\} \tag{10}$$

where P_λ is the sum of precision below the λ-recall, and *num* is the number.

$$precision \text{ at } k \text{ documents retrieved} = number \text{ of relevant document retrieved} / k,\ k = \{5, 10, 20\} \tag{11}$$

7.3 Experiment Results

7.3.1 *DSM* Versus *VSM*

The parameters of *Jelinek-Mercer* and *Dirichlet* smoothing methods are set experimentally as 0.95 and 2000 on each testing collection. Due to the space limitation, we give only a part of results (Figure 2 and 3).

From the results the symmetric KL measure with *Jelinek-Mercer* smoothing perform best. On the whole, it outperform significantly the *VSM* over the 0.0~0.6 points of recall on the OHSU87P and approximately over all the points on the OHSU87W. The precision of symmetric KL measure at 5, 10 and 20 documents retrieved on the OHSU87P, see Table 2, is worse than that of the *VSM* except the precision at 5 documents retrieved. However, it achieves a significant performance improvement, respectively 89%, 57.1% and 32% on the OHSU87W on average. The reason that symmetric KL divergence measure perform better on the larger testing collection lies mainly in that the model estimation is more accurate on the larger collection, in another word, larger the size of collection is, more appropriate the estimation of collection distribution $p_c(t)$ is.

Table 2. Precision at 5, 10 and 20 documents retrieved on the ohsu87p and ohsu87w

DocID	Measure	Ohsu87P			Ohsu87W		
		P-5	P-10	P-20	P-5	P-10	P-20
14039	VSM	1	1	0.9	0.4	0.5	0.6
	MKL-JM-0.95	1	1	0.75	0.8	0.5	0.5
4701	VSM	1	0.98	0.8	0.4	0.2	0.2
	MKL-JM-0.95	1	0.8	0.65	0.6	0.4	0.35
380	VSM	1	0.9	0.75	0.2	0.1	0.1
	MKL-JM-0.95	1	0.8	0.65	0.4	0.3	0.15
2913	VSM	0.8	0.9	0.65	0.4	0.3	0.2
	MKL-JM-0.95	1	0.8	0.6	0.6	0.5	0.3
283	VSM	0.6	0.6	0.45	0.4	0.3	0.15
	MKL-JM-0.95	1	0.9	0.6	1	0.5	0.35
Avg.	VSM	0.88	0.876	0.71	0.36	0.28	0.25
	MKL-JM-0.95	1	0.86	0.65	0.68	0.44	0.33
	Ratio	+14%	-1.8%	-8.5%	+89%	+57.1%	+32%

The performance of *Bhattacharyya* measure is close to the *VSM* on average, but worse than the symmetric KL divergence. It is interesting that the distributional similarity measures with *Jelinek-Mercer* smoothing are much better than that with *Dirichlet* smoothing. However, it is reverse for query-to-document relevance. This result is in accord with that of *Zhai* and *Lafferty* [4] who applied the language model to the long queries that are more verbose. The reason is that smoothing actually plays two different roles. One role is to improve the accuracy of the estimated documents language model and the other is to explain the common and non-informative terms. The discrimination between informative and non-informative terms is more important for document-to-document relevance, because a document often contains many terms irrelevant to its topic, and *Jelinek-Mercer* smoothing is better for the second role than *Dirichlet* smoothing.

7.3.2 The Sensitivity to Parameters

Because of the best performance achieved by the symmetric KL divergence measure, we examine only the sensitivity under the symmetric KL divergence measure. Due to the space limitation, here only a part of results are given. Figure 4 shows the 11-point precision for different setting for the parameters λ and μ on the ohsu87p testing collection. It is evident that the performance is much more sensitive to the setting of *Jelinek-Mercer* smoothing parameter λ than *Dirichlet* smoothing parameter μ. This result in a sense confirms two roles of smooth as mentioned above.

8 Conclusions

In this paper, document language model (*DLM*) is adopted to represent document topical content. On the basis of *DLM*, we use the distributional similarity measure to

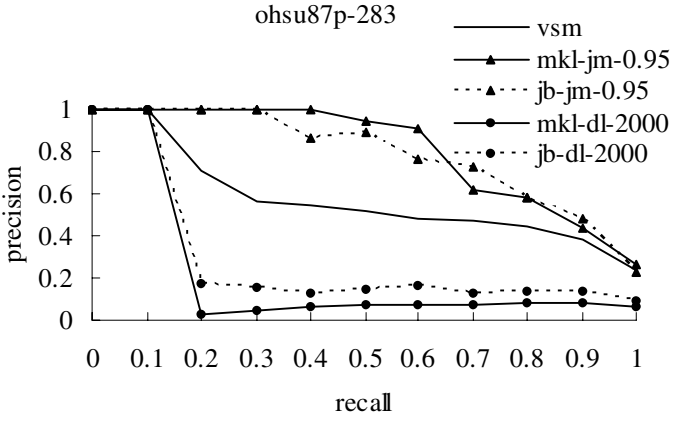


Fig. 2. 11-point precision of 283 in ohsu87p

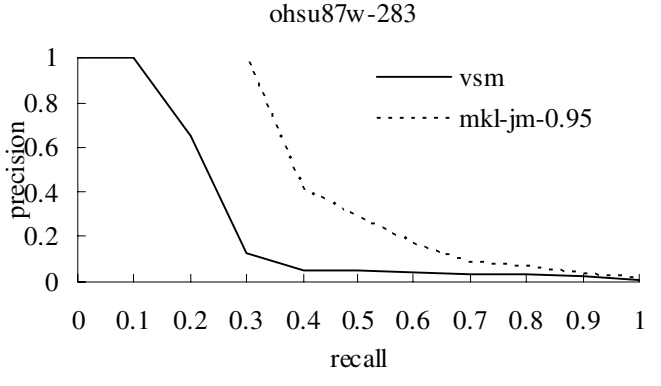


Fig. 3. 11-point precision of 283 in ohsu87w

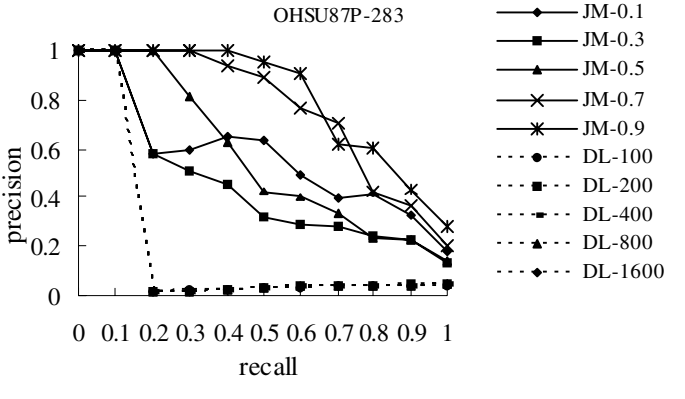


Fig. 4. Sensitivity to parameter

evaluate the document-to-document relevance, and compare it with the *VSM* on the TREC testing collection. With the results of experiment we can conclude as follows:

- Compared with the *VSM*, the *DLM* not only explains explicitly the model representation, which can be understood easily by users, but also captures the document topic in terms of the conditional distribution, and moreover, it has a solid statistic foundation.
- The symmetric KL divergence measure with *Jelinek-Mercer* smoothing outperforms significantly the *VSM* and the other measures based on *DLM* by experiment. This result means that the *Jelinek-Mercer* smoothing is better for document-to-document relevance than other methods proposed in this paper, but it is more sensitive to the model's parameter.
- Moreover, it is evident that the *DLM* has the same time-complexity as *VSM*.

Because of the sensitivity to parameters of *Jelinek-Mercer* smoothing, our future works include how to determine efficiently its parameters with respect to the data collection. One of the solutions might be to model the *DLM* by the two-state HMM mentioned in [7].

References

1. A. Abdollahzadeh: Information Retrieval on the World Wide Web and Active Logic: A Survey and Problem Definition. Computer Science Dept., University of Maryland Technical Reports TR-CS-4291 (2002)
2. Saracevic, T: Relevance Reconsidered. In P. Ingwersen and N. O. Pors. Information Science: Integration in Perspective (1996)
3. Jiayue Wang: The Relevance in Information Retrieval. Modern Foreign Languages Vol.24 No.2. (2001)
4. Zhai, C. and Lafferty, J: A Study of Smoothing Methods for Language Models Applied to Ad Hoc Information Retrieval. In: Proc. of SIGIR'01 (2001)
5. William Gibson: Pattern Recognition. Academic Press (2003)
6. Jay M. Ponte and W. Bruce Croft: A Language Modeling Approach to Information Retrieval, In: Proceedings of SIGIR'98 (1998)
7. D. Miller, T. Leek and R. M. Schwartz: A hidden Markov Model Information Retrieval System. In: Proc. of SIGIR'99 (1999)
8. Hugo Zaragoza, Djoerd Hiemstra and Michael Tipping: Bayesian Extension to The Language Model for Ad Hoc Information Retrieval. In: Proc. of SIGIR'03 (2003)
9. Lafferty, J. and Zhai, C: Document Language Models, Query Models, and Risk Minimization for Information Retrieval. In: Proc. of SIGIR'01 (2001)
10. Adam Berger and John Lafferty: Information Retrieval as Statistical Translation. In: Proc. of SIGIR'99 (1999)
11. V. I. Levenshtein; Binary Codes Capable of Correcting Spurious Insertions and Deletions of Ones. Problems of Information Transmission (1965)
12. P. Yianilos: Data Structures and Algorithms for Nearest Neighbor Search in General Metric Spaces. In: Proc. of the 4th ACM-SIAM Symposium on Discrete Algorithms (1993)
13. P. Yianilos: The Likeit Intelligent String Comparison Facility. NEC Institute Tech Report 97-093 (1997)
14. Salton, G: The SMART Retrieval System: Experiments in Automatic Document Processing. Prentice-Hall Inc., Englewood Cliffs, NL (1971)

Retrieval Efficiency of Normalized Query Expansion

Sofia Stamou and Dimitris Christodoulakis

Computer Engineering and Informatics Department, Patras University,
Research Academic Computer Technology Institute, 26221, Greece
{stamou, dxri}@cti.gr

Abstract. In this paper we experimentally study the impact of normalized query expansion on Web Information Retrieval. In this respect, we have implemented a query expansion module, which firstly normalizes the user submitted queries and subsequently attempts to enrich them with semantically related terms that are obtained from WordNet. Experimental results demonstrate that for certain query types our module has a potential in giving improved search results in terms of relevance, compared to the results retrieved for the same queries by other retrieval methods.

1 Introduction

To support information seekers in overcoming terminological problems when searching for information on the Web, several approaches have been addressed in the literature, the most prominent of which imply the expansion of the issued queries with semantically related terms. In this paper we seek to get an improved insight on how normalized query expansion can effectively cope with vocabulary mismatches, in an attempt to improve retrieval relevance when querying the Web. To challenge that, we built a prototype query expansion module that interacts with normalization techniques, applied to the subjected queries. The query expansion module we introduce explores the semantic information encoded in WordNet and determines which terms are the most suitable to be used for enriching a given query.

We evaluated our system's performance in successfully enriching queries by having humans judge the relevance of the results retrieved in response to a set of queries after these have been expanded by our system, and compare them to the relevance of the results retrieved for the same queries after employing other searching techniques. Our findings indicate that our expansion approach improves retrieval performance for certain query types, compared to the performance of other retrieval techniques. Retrieval improvements are pronounced for long queries (i.e. multiword and phrase queries) due to our system's effectiveness in disambiguating them.

In the remaining of this paper, we describe how our system proceeds in generating expanded queries (Section 2) and we report on our empirical findings (Section 3) that proof the impact of our approach in helping information seekers find alternative wordings for expressing their information needs. We conclude our work (Section 4) with a discussion on our system's contribution in retrieval performance. Before presenting our system, we should note that although we experimented with Greek data, we believe that our approach can be useful to other languages as well.

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 604 – 607, 2005.

2 Normalized Query Expansion

A core module in our query expansion system is a normalizer introduced in [3]. Normalization is a computational process, which assisted by morphological lexicons, identifies (query) term inflectional and derivational variants and reduces them to a single canonical form, i.e. lemma. Lemmatized (query) terms pass though our expansion module, which maps them to Greek WordNet (GWN) [1] synsets and retrieves their semantically equivalent terms (i.e. synonyms). Retrieved synonyms are used as supplementary terms for enriching queries, following the approach described below.

A critical factor for the successful expansion of a query requires the prior successful match between the sense of the query and the senses of the GWN synsets that map the given query. For determining the appropriate sense of a query that matches multiple GWN synsets, we rely on the matching synsets' correlation in GWN, represented by a correlation factor that is determined by the synsets' position in the WordNet graph. More specifically, synsets that belong to a lower level in the WordNet hierarchy and that also share a minimal distance from each other are deemed as highly correlated. The *Correlation* of a word pair (w_1, w_2) is formally determined as the product of the words' *Depth* [5] and their conceptual similarity (*Sim*) [4] in WordNet. Depth of a word pair (w_1, w_2) is defined as:

$$DepthScore\ (w_1, w_2) = Depth\ (w_1)^2 \bullet Depth\ (w_2)^2. \tag{1}$$

and words' similarity $Sim(w_1, w_2)$ is determined by the set of WordNet synsets (c) that subsume both w_1 and w_2, in any sense of either word with a probability Pr, i.e.:

$$Sim\ (w_1, w_2) = \max\ c \in subsumers\ (w_1, w_2) \quad [\ -\log \Pr\ (c)\ ]. \tag{2}$$

Finally, the *Correlation* between w_1 and w_2 is defined as:

$$Correlation\ (w_1, w_2) = Depth\ (w_1, w_2)\ \bullet\ Sim\ (w_1, w_2). \tag{3}$$

Using the above formulas, query terms that match multiple GWN synsets are expanded with the synonyms of the synset that has the greatest *Correlation* score to them, of all its correlation scores. While synset selection is straightforward for multiword queries, for which there is enough data to resolve ambiguities, synset selection for single term queries is more complicated, essentially due to the lack of enough query terms that would help disambiguate the query. In selecting a GWN synset for a single term query expansion, we follow the assumption of [2] that terms occurring together in the same document are related to the same theme, and we attempt query sense resolution as follows: the top 10 ranked documents retrieved as relevant to a single term query are merged together forming a sample corpus, from which we extract query co-occurrence data. We first define the optimum window size within which we consider co-occurrence to 15 words, we then merge together these context windows and we assign frequency weights to their terms according to the normalized TF.IDF scheme. Terms with frequency values above a given threshold and the initial query are mapped to GWN synset. Selection of the best matching synset relies on the synsets' *Correlation* scores and takes place as described above.

3 Evaluating Expansion Performance

To evaluate the efficiency of our module in successfully enriching queries, we conducted an experiment in which we compared the relevance of the results retrieved for a set of 18 queries after these have been expanded by our system to the results retrieved for the same queries after employing other searching techniques. Experimental queries were manually selected and they correspond to three query type formulations, i.e. single term, multiword and phrase queries. The other retrieval methods against which we compared our system's efficiency are: keyword-based searching, referred to as "baseline retrieval" and searching with query morphological variants, known as "normalized" retrieval. For our comparison, we relied on the relevance judgments of external evaluators, who analyzed the top 30 retrieved documents for each of the queries across the three searching techniques. Relevance judgments were scored on a 10-point scale, ranging between 0.1 for marginally relevant results and 1 for highly relevant results. At the end of the experiment, we collected human relevance judgments, grouped them in three categories, i.e. relevance of the top-10, top-20 and top-30 retrieved documents respectively, and we calculated the average relevance for each of the categories across the three searching methods (i.e. baseline, normalized and expansion retrieval). Average relevance was defined as the ratio of the sum of relevance scores at the specified ranking points to the total number of the documents considered, and is defined as:

$$R(r) = \left| \sum_{i=1}^{Nd} R_i(r) \right| / \left| Nd \right|. \tag{4}$$

where R(r) is the query average relevance at ranking level (r), Nd is the number of pages considered, and $R_i(r)$ is the relevance at ranking level (r) for query i. Having computed separately average relevance scores for the results retrieved across the different ranking points, we merged together relevance judgments and we computed the overall relevance of each retrieval method by summing the average relevance scores at the specified ranking points and then dividing by the total number of queries. Overall relevance of each retrieval technique is computed as:

$$\sum_{i=1}^{NUM} R_r / NUM. \tag{5}$$

where R(r) is the relevance at ranking level (r), (in our case the top 30 retrieved records) and NUM is the number of queries issued. We assessed the retrieval efficiency of our system by comparing the relevance scores that it delivered at each of the specified rankings, to the relevance of the results returned for the same queries by other searching methods, at the same ranking cut-offs. Obtained results are briefly discussed next. The interested reader can obtain a more elaborate view on the results retrieved for each of the queries across the three searching methods by referring to a longer version of the paper available at [6].

4 Discussion

An analysis of the experimental results indicates that for multiword and phrase queries, our expansion module outperforms keyword-based and morphology-based retrieval. In detail, our findings demonstrate that our system, when supplied with a sufficient number of query terms (i.e. typically 2 or 3), it can effectively resolve the senses of the underlying queries and hence enrich them with their corresponding synonyms found in WordNet. On the other side, the successful expansion of single term queries is mainly dependent on the lexical cohesion of the sample corpus that our system employs for obtaining query co-occurring words. Due to our system's dependence on corpus co-occurrence data for resolving query senses, it becomes evident that upon selection of the wrong co-occurring terms, expansion fails. Expansion failure was implied in our work by the deteriorated relevance scores that expansion gave for some single term queries. Moreover, expansion performance depends on the normalizer's ability to resolve query terms' part-of-speech ambiguities and thus induce them to their correct lemmas. Again, normalization failure leads to the selection of semantically irrelevant terms for query expansion, which results to the retrieval of irrelevant documents among the results. Summarizing, we have found that normalized query expansion has a potential in improving retrieval efficiency when dealing with multi-term queries, whose senses can be effectively inferred by their WordNet semantic correlations. For other query types, our findings suggest that expansion has a promising potential as long as it uses a balanced corpus of topics for disambiguating queries.

References

1. Grigoriadou, M., Kornilakis, H., Galiotou E., Stamou, S., Papakitsos, E.: The Software Infrastructure for the Development and Validation of the Greek WordNet. In Romanian Journal of Information Science and Technology, Vol. 7(1-2) (2004)
2. Mandala, R., Takenobu, T., Tanaka, H.: Combining Multiple Evidence from Different Types of Thesaurus for Query Expansion. In Proceedings of the 22nd ACM SIGIR Conference (1999)
3. Ntoulas, A., Stamou, S., Tzagarakis, M.: Using a WWW Search Engine to Evaluate Normalization Performance for a Highly Inflectional Language. In Proceedings of the ACL Student Workshop, France (2001)
4. Resnik, Ph.: Disambiguating Noun Groupings with Respect to WordNet Senses. In Proceedings of the 3rd Workshop on Very Large Corpora, MIT (1995)
5. Song, Y.I, Han, K.S., Rim, H.C.: A Term Weighting Method Based on Lexical Chain for Automatic Summarization. In Proceedings of the 5th CICLing Conference (2004)
6. Normalized Query Expansion Overview and Results: ftp://150.140.4.154/ftproot/

Selecting Interesting Articles Using Their Similarity Based Only on Positive Examples

Jiří Hroza and Jan Žižka

Faculty of Informatics, Department of Information Technologies,
Masaryk University, Botanická 68a, 602 00 Brno, Czech Republic
{xhroza1, zizka}@informatics.muni.cz

Abstract. The task of automated searching for interesting text documents frequently suffers from a very poor balance among documents representing both positive and negative examples or from one completely missing class. This paper suggests the *ranking approach* based on the k-NN algorithm adapted for determining the *similarity degree* of new documents just to the representative *positive* collection. From the viewpoint of the precision-recall relation, a user can decide in advance how many and how similar articles should be released through a filter.

1 Introduction

When selecting from unstructured natural language text documents, a pragmatic trouble can aggravate the design of a filter: many users collect articles that represent (almost) only the interesting ones, and the required *relevant negative examples* for training an algorithm are missing. Typically physicians, having only positive examples of articles, need to automatically single out very specific medical documents within a narrow expert area—yet, containing too many articles around very similar topics [1]; here is the inspiration for the described research. The problem with synthetical filling in the missing examples is that *arbitrary* text documents different from the positive ones cannot be generally used: how to define effectively the dissimilarity? This paper describes the *ranking approach* based on the k-NN (k-nearest neighbors) algorithm adapted for determining the similarity of articles to the representative *positive* examples. For the comparison, outcomes of the SVM (*support vector machines*) algorithm are also shown.

2 Text Documents and Their Preprocessing

To test performance of the one-class k-NN and SVM, one of the standard benchmarks 20Newsgroups dataset was used[1]. Then, the one-class k-NN was also applied to a specific set of real expert medical documents[2] from MEDLINE [1].

[1] http://www.ai.mit.edu/∼jrennie/20Newsgroups/
[2] http://www.fi.muni.cz/∼xhroza1/datasets/glall/

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 608–611, 2005.

Porter's algorithm [4] was applied to obtain a stem of each word. The dictionary was created as a set of all distinct words in the exemplary articles (*bag of words*), and 100 of the most common English words plus words occurring less than three times were removed. Each document was encoded into a *feature vector* where every position in the vector corresponded to one word in the dictionary (the number at the position was a *relative frequency* of the word). For SVM, the *binary representation* [a word is/is not (1/0) at a given position] and other parameter settings were used as recommended in [2].

3 Applied Document-Filtering Algorithms

One-Class Ranking by k-NN: The k-nearest neighbors algorithm (k-NN) [3] has a simple training phase based just on storing of training text document examples. During the classification phase, the algorithm finds the k most similar training examples for an unclassified article. Then the article's class is the most common class of the k found training examples weighted by their similarity (the cosine measure, i.e., the document similarity obtained by the vector-representation comparison). For the one-class problem, this paper suggests a modified, *one-class* k-NN ranking version. The training set consists only from instances of available interesting articles. The ranking phase computes similarities to the k nearest neighbors. When the similarities (playing a role of weights) of all new unclassified articles are known, the documents are sorted according to sums of these values as shown in Table 1.

Table 1. The ranking algorithm used by the modified k-NN

1. Represent m new unclassified documents as vectors using a *bag of words* from the training phase.
2. For each new vector $\mathbf{u_i}$, compute its cosine similarity measure $s(\mathbf{u_i}, \mathbf{v_j})$ to all training vectors $\mathbf{v_j}$, $i = 1, \ldots, n$, $j = 1, \ldots, m$:

$$s(\mathbf{u_i}, \mathbf{v_j}) = \frac{\mathbf{u_i}^T \mathbf{v_j}}{\|\mathbf{u_i}\| \ \|\mathbf{v_j}\|}, \tag{1}$$

3. For each $\mathbf{u_i}$, select its k nearest neighbors $\mathbf{v_j}$, $j = 1, \ldots, k$, where a higher similarity s means a closer distance. Using the k highest similarities s_{kNN}, compute the resulting $\mathbf{u_i}$'s value $w(\mathbf{u_i})$ used for setting up its ranking position:

$$w(\mathbf{u_i}) = \sum_{j=1}^{k} s_{kNN}(\mathbf{u_i}, \mathbf{v_j}) \tag{2}$$

4. According to the w's obtained in the previous step, create the rank of all investigated text documents: higher w's mean higher positions in the rank.
5. Within the acquired rank, classify the first r vectors as *positive* ones and release them through the filter as *interesting articles*, where r is a user's parameter.

Comparative Filtering by SVM: The one-class SVM enables to learn a concept of classification into a relevant/irrelevant class while it learns only from relevant instances; the linear kernel for the one-class SVM does not seem to be very sensitive to the choice of parameters [2]. The BSVM[3] 2.6 implementation with linear kernel and default parameters was employed. To determine the difference between training SVM by one class and—if available—two classes, the same two-class SVM software was used.

4 Experiments and Their Results

For each of 20 newsgroups, experiments were carried out. Each of the experiments consisted of the 10-fold cross-validation of a dataset created from one newsgroup as a positive class and the rest of 19 newsgroups as a negative class to establish a situation similar to a real user's one.[4] The evaluation uses *precision, recall,* and the F_1 measure [5].

5-NN Ranking: The ranking algorithm was used to rearrange filtered examples. Experiments revealed that comparisons with 5 nearest training articles provided the best results. Figure 1 shows the precision of the 5-NN when browsing through the rearranged examples from the most relevant ones to the less relevant ones. Table 2 compares results with the SVM methods. It is necessary to emphasize that a real user typically can exploit only a very small part of suggested documents, so there is a high chance to find them at the top of the rank with the 70%–80% likelihood.

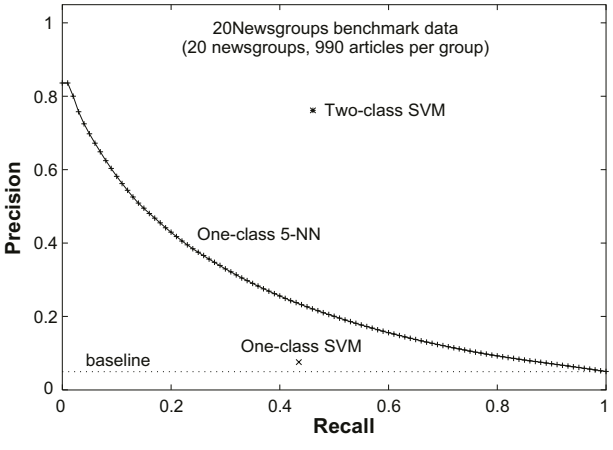


Fig. 1. 5-NN ranking, one- and two-class SVM

[3] http://www.csie.ntu.edu.tw/∼cjlin/bsvm/

[4] in the one-class and ranking cases, the splits were the same as in the two-class training, except that in the former cases the negative training examples were unused for building a classifier/ranker

Table 2. Results of the algorithms' success for the benchmark data

Algorithm	Precision	Recall	F_1 measure
2-class SVM	76.2%	46,1%	52.5%
1-class SVM	7.6%	43.5%	12.9%
1-class 5-NN	29.8%	34.0%	31.8%
baseline	5.0%	100%	9.5%

One- and Two-Class SVM: The one-class SVM results were poor, see Table 2. Predictably, the two-class SVM achieved very good results; however, it *does require* instances from both classes.

MEDLINE Documents: After verifying functionality of the one-class k-NN, it was successfully applied to the medical data: $F_1 = 0.73$, however, the best results were obtained for 1-NN and cannot be directly compared with the benchmark outcomes because of different document distribution and a higher baseline. In the tested cases, the initial problem with filtering articles using only positive examples quite acceptably handles the suggested one-class k-NN.

5 Conclusions

The ranking k-nearest neighbor algorithm trained only from the available positive examples was able to correctly arrange new text articles. Such an arrangement allows acquiring a reasonable portion of interesting documents with a higher precision, up to 70%–80%. The one-class SVM had difficult problems—its results were only slightly above the F_1 baseline; the two-class SVM algorithm cannot be used when only one class is available, otherwise it would be superior.

Acknowledgments

This research was partly supported by the Grant *Human-Computer Interaction, Dialog Systems and Assistive Technologies, MSM*-143300003.

References

1. Hroza, J., Žižka, J., and Bourek, A.: Filtering Very Similar Text Documents: A Case Study. In: Gelbukh, A. (Ed.): Proc. of the Fifth Intl. Conf. on Intelligent Text Processing and Computational Linguistics CICLing-2004, February 15-21, 2004, Seoul, South Korea, Springer Verlag, 2004, LNCS 2945, pp. 511-520
2. Manevitz, L. R. and Yousef, M.: One-Class SVMs for Document Classification. In: Journal of Machine Learning Research 2 (2001), December 2001, pp. 139-154
3. Mitchell, T. M. (1997): Machine Learning. McGraw Hill, 1997
4. Porter, M. F.: An Algorithm For Suffix Stripping. Program 14(3), 1980, pp. 130-137
5. Van Rijsbergen, C. J.: Information Retrieval, 2nd Edition. Department of Computer Science, University of Glasgow, 1979

Question Classification in Spanish and Portuguese

Thamar Solorio[1], Manuel Pérez-Coutiño[1], Manuel Montes-y-Gómez[1,2], Luis Villaseñor-Pineda[1], and Aurelio López-López[1]

[1] Laboratorio de Tecnologías del Lenguaje,
Instituto Nacional de Astrofísica Óptica y Electrónica,
Santa María Tonantzintla, Puebla, México 72840
[2] Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia, España
{thamy, mapco, mmontesg, villasen, allopez}@inaoep.mx

Abstract. We present in this work a method for question classification in Spanish and Portuguese. The method relies on lexical features and attributes extracted from the Web. A machine learning algorithm, namely Support Vector Machines is successfully trained on these features. Our experimental results show that this method performs consistently well over two different languages.

1 Introduction

Question Classification (QC) is concerned with assigning a semantic category to questions posed in natural language. This semantic category corresponds to the type of answer needed for satisfying the user query. For instance, the question *In which European city is the Eiffel Tower?* belongs to the semantic class of "LOCATION". Most approaches to Question Answering systems perform some type of question classification given that the search space of possible answers is greatly reduced, also it has been shown that a poor performance in this stage of the system can provoke over one third of the errors [1]. However, most of these approaches are targeted to specific languages, this is because they use complex linguistic tools that are language dependent. Unfortunately for most languages these resources, such as part-of-speech taggers, named entity extractors, parsers, and so on, are not very well developed. Then, the adaptability of these methods to a different language is limited to those languages for which the linguistic tools are readily available.

In previous work we presented a language independent method for question classification were evaluation was performed on three languages: English, Spanish and Italian [2]. Although we achieved high accuracies we believe that considerable improvements can be attained by modifying some of the weakest features of this method, namely the set of heuristics chosen in order to construct the Internet queries. In this paper we present results of some modifications to this approach applied to questions written in Portuguese and Spanish. Our motivation is to provide a method for question classification that can be applied to

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 612–619, 2005.

different languages without requiring additional linguistic tools, such as parsers, named entity extractors and the like.

We first summarize some of the previous related work and describe the data sets used in our evaluation. Then we introduce the problem of question classification, we describe the lexical features used in the learning process and how the Web can be successfully used in this problem. We present some evaluation results and conclude this article with the findings of this work and interesting directions of future research.

2 Related Work

Li and Roth reported a hierarchical approach for question classification in English based on the SNoW (Sparse Network of Winnows) learning architecture [3]. This hierarchical classifier discriminates among 5 coarse classes, which are then refined into 50 more specific classes. The learners are trained using lexical and syntactic features such as part-of-speech tags, chunks and head chunks together with two semantic features: named entities and semantically related words. They reported question classification accuracy of 98.80% for a coarse classification, using 5,500 instances for training.

A different approach, used for Japanese question classification, is that of Suzuki *et al.* [4]. They used SVM with a new kernel function, called Hierarchical Directed Acyclic Graph, which allows the use of structured data. They experimented with 68 question types and compared performance of using bag-of-words against using more elaborated combinations of attributes, namely named entities and semantic information. Their best results, an accuracy of 94.8% at the first level of the hierarchy, were obtained when using SVM trained on bag-of-words together with named entities and semantic information.

In [5] Zhang and Sun Lee present a new method for question classification using Support Vector Machines targeted to English. They compared accuracy of SVM against Nearest Neighbors, Naive Bayes, Decision Trees and SNoW, with SVM producing the best results. In their work, accuracy is improved by introducing a tree kernel function that allows to represent the syntactic structure of questions. Their experimental results show that SVM using this tree kernel function achieves an accuracy of 90%, however, a parser is needed in order to acquire the syntactic information.

The idea of using the Internet in a QA system is not new. What is new, however, is that we are using the Internet to obtain values for features in our question classification process, as opposed to previous approaches where the redundancy of information available on the Internet has been used in the answer extraction process [6, 7, 8, 9].

3 Data sets

The data set used in this work consists of the questions provided in the DISEQuA Corpus [10]. Such corpus was made up of simple, mostly short, straightforward

and factual queries that sound naturally spontaneous, and arisen from a real desire to know something about a particular event or situation. The DISEQuA Corpus contains 450 questions, each one formulated in four languages: Dutch, English, Italian and Spanish. The questions are classified into seven categories: *Person*, *Organization*, *Measure*, *Date*, *Object*, *Other* and *Place*. The experiments performed in this work used the Spanish versions of these questions.

For Portuguese questions we use a data set consisting of 180 questions taken from the data sets used in CLEF 2004, the categories of the questions are the same as for the Spanish corpus.

4 Learning Question Classifiers

4.1 Lexical Features

With the aim of developing a flexible method we decided to use for learning only lexical features that can be automatically extracted from the questions. The most frequently used lexical features are bag-of-words and n-grams. Since we are using a machine learning technique, the n-grams approach seems the less desirable one, given that our training examples are limited and n-grams require a large training set. Then we opted for the bag-of-words approach, we also made a comparison of results between bag-of-words and prefixes.

As mentioned before, the lexical features are used as attributes for training a classifier. In this work, we used Support Vector Machines (SVM) as they have proved to perform well over natural language related problems such as text classification [11].

SVM use geometrical properties in order to compute the hyperplane that best separates a set of training examples [12]. When the input space is not linearly separable SVM can map, by using a kernel function, the original input space to a high-dimensional feature space where the optimal separable hyperplane can be easily calculated. This is a very powerful feature, because it allows SVM to overcome the limitations of linear boundaries. They also can avoid the over-fitting problems of neural networks as they are based on the structural risk minimization principle. The foundations of these machines were developed by Vapnik, for more information about this algorithm we refer the reader to [13, 14].

In Table 1 we show experimental results of using SVM trained on three different sets of attributes: bag-of-words, prefixes of size 4 and prefixes of size 5. As we can see accuracies are very similar, with prefixes of size 5 achieving

Table 1. Question classification accuracies when training SVM with words and prefixes of size 5 and 4

Language	Words	Prefix-5	Prefix-4
PORTUGUESE	75.14%	**75.73%**	74.55%
SPANISH	76.44%	**78.44%**	71.55%

the best results for both languages. All the results reported here are the overall average of several runs of 10-fold cross-validation.

4.2 Using the Web

Previous results are encouraging considering that the only information needed to achieve these accuracies can be automatically extracted from the questions. However, we wanted to see if we can further improve these results by making use of the Web. The Web has become the greatest information source available worldwide, and although English is the dominant language represented on it, it is very likely that one can find information in almost any desired language. Considering this, and the fact that the texts are written in natural language, it is immediate to develop new methods exploring the use of the Web to solve natural language related problems [15]. Following this trend, we propose using the Web in order to acquire information that can be used as attributes in our classification problem. This attribute information can be extracted automatically from the web and the goal is to provide an estimate about the possible semantic class of the question.

The procedure for gathering this information from the web is as follows: we use a set of heuristics to extract from the question a word w, or set of words, that will complement the queries submitted for the search. We then go to a search engine, in this case Google, and submit queries using the word w in combination with all the semantic classes of interest for our purpose. For instance, for the question *Who is the President of the French Republic?* we extract the word *President* using our heuristics, and submit 5 queries in the search engine, one for each possible class. These queries take the following form:

- "President is a person"
- "President is a place"
- "President is a date"
- "President is a measure"
- "President is an organization"

We count the number of results returned by Google for each query and normalize them by their sum. The resultant numbers are the values for the attributes used by the learning algorithm. As can be seen, it is a very straightforward approach, but as the experimental results show, this information gathered from the Web is quite useful. In Table 2 we present the figures obtained from Google for the example question above, column *Results* show the number of hits returned by the search engine and in column *Normalized* we present the number of hits normalized by the total of all results returned for the different queries. It can be seen that Google returned hits for all the categories except for the "DATE" category, but the highest number of hits were returned for the category "PERSON", which is the real class of the question in our example.

An additional advantage of using the Internet is that by approximating the values of attributes in this way, we take into account words or entities belonging to more than one class (polysemy).

Table 2. Example of using the Web to extract features for question classification

Query	Results	Normalized
"President is a person"	259	0.8662
"President is a place"	9	0.0301
"President is an organization"	11	0.0368
"President is a measure"	20	0.0669
"President is a date"	0	0

Now that we have introduced the use of the Internet in this work, we continue describing the set of heuristics that we use in order to perform the web search.

Heuristics. We begin by eliminating from the questions all words that appear in our stop lists. These stop lists contain the usual items: articles, prepositions and conjunctions plus all the interrogative adverbs and all lexical forms of the verb "to be". The remaining words are sent to the search engine in combination with the possible semantic classes, as described above. If no results are returned for any of the semantic classes we then start eliminating words from right to left until the search engine returns results for at least one of the semantic categories. As an example consider the question posed previously: *Who is the President of the French Republic?* we eliminate the words from the stop list and then formulate queries for the remaining words. These queries are of the following form: *"President French Republic is a* s_i*"* where $s \in \{Person, Organization, Place, Date, Measure\}$. The search engine did not return any results for this query, so we start eliminating words from right to left. The query is now like this: *"President French is a* s_i*"* and given that again we have no results returned we finally formulate the last possible query: *"President is a* s_i*"* which returns results for all the semantic classes except for *Date*.

These were the heuristics used in previous experiments [2], in addition to these, in this work we run queries eliminating words in the reverse direction. That is, if no hits are returned after eliminating the stop words, we eliminate the first word to the left and continue repeating this process until we have results.

Being heuristics, we are aware that in some cases they do not work well. Nevertheless, for the vast majority of the cases they presented surprisingly good results, in the two languages, as shown in Table 3. What we did on this experiments was to compare results of training an SVM on the attributes from the Web. In column *Web RL* the heuristics used are those from the previous work, eliminating words from right to left. Column *Web LR* shows results eliminating words in the reverse order. These two columns seem to show that eliminating words from right to left yield more informative queries. In column *Web RL+LR* we present results of using both sets of attributes. That is, we combine the information from eliminating the words in both directions. These combination of attributes achieved the best results. These results also show that for Portuguese the accuracies are much lower than for Spanish. We believe that this is due to

Table 3. Experimental results of accuracy when training SVM with attributes extracted form the Web

Language	Web RL	Web LR	Web RL+LR
PORTUGUESE	59%	52.07%	**60.35%**
SPANISH	65.77%	44.66%	**67.11%**

Table 4. Accuracies of combining the Web-extracted attributes (Web) with lexical features. The web extracted attributes are the combination of Left-to-Right and Right-to-Left presented in section 4.2

Language	Words+Web	Prefix-5+Web	Prefix-4+Web
PORTUGUESE	76.33%	75.53%	**78.1%**
SPANISH	78.33%	**79.11%**	78.22%

the lower availability of Portuguese documents on the Web than for Spanish. The number of words available in Portuguese was near 1.3 billions whereas for Spanish was 2.6 billions [15].

4.3 Combining Web-Extracted Attributes with Lexical Features

So far we have shown that, on one hand, lexical features can provide enough information to build an automated classifier. On the other hand, information from the Web is not sufficient to provide accurate classifiers, the lack of language representation might be one reason for this. Yet, another possibility that we have to explore is a combination of these two types of features. Then, we performed

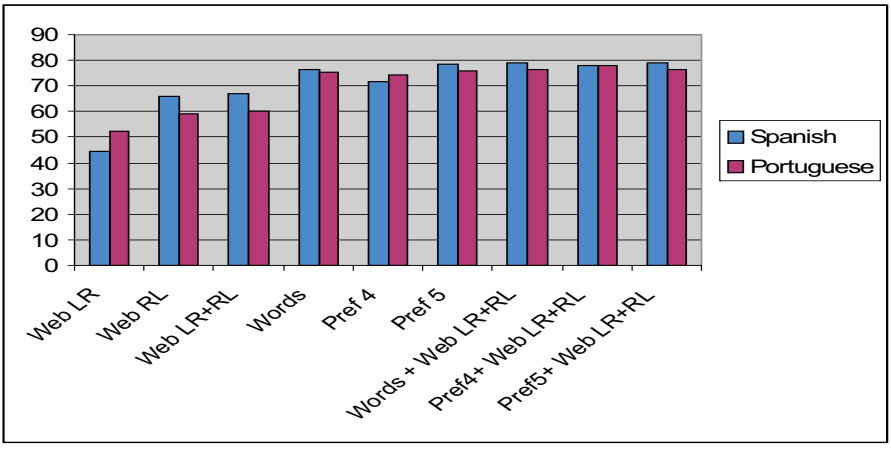


Fig. 1. Graphical comparison of question classification accuracies

new experiments combining the lexical attributes with the Web information in order to discover if we can further improve accuracy. Table 4 shows experimental results of this attribute combination and Figure 1 shows a graphical representation of these results. By comparing results presented in Tables 1 and 4 we can see that the best results are acquired using a combination of features. Even though the Web-based attributes did not seem to provide very interesting results at first, combining them with the lexical features did yield higher classification accuracy.

5 Conclusions

We have presented here experimental results of a very flexible method for question classification. The method is claimed to be language independent to a good degree since the features used as attributes in the learning task can be extracted from the questions in a fully automated manner; we do not use semantic or syntactic information because otherwise we will be restricted to work on languages for which we do have parsers that can extract this information. We believe that this method can be successfully applied to other languages, such as Romanian, French and Catalan.

We are currently working on improving the heuristics used, we believe that better queries, formulated in a more careful manner, will help increase classification accuracy.

Another interesting line for future work is exploring the advantage of using mixed languages corpora lo learn question classification. The Romance languages, for instance, such as Italian, French and Spanish have stems in common. Then it is feasible that questions for several languages may help to train a classifier for a different language. The advantage of this idea will be the availability of larger corpora for languages for which a large enough corpus is not available, counting in favor of languages that are under-represented on the Web.

Acknowledgements

We would like to thank CONACyT for partially supporting this work under grants 166934, 166876, U39957-Y and 43990A-1, and Secretaría de Estado de Educación y Universidades de España.

References

1. D. Moldovan, M. Paşca, S. Harabagiu, and M. Surdeanu. Performance issues and error analysis in an open-domain question answering system. *ACM Trans. Inf. Syst.*, 21(2):133–154, 2003.
2. T. Solorio, M. Pérez-Coutiño, M. Montes y Gómez, L. Villaseñor-Pineda, and A. López-López. A language independent method for question classification. In *The 20th International Conference on Computational Linguistics, COLING-04*, Geneva, Switzerland, 2004.

3. X. Li and D. Roth. Learning question classifiers. In *19th International Conference on Computational Linguistics, COLING'02*, Taipei, Taiwan, 2002.
4. J. Suzuki, H. Taira, Y. Sasaki, and E. Maeda. Question classification using HDAG kernel. In *Workshop on Multilingual Summarization and Question Answering 2003*, pages 61–68, 2003.
5. D. Zhang and W. Sun Lee. Question classification using support vector machines. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 26–32, Toronto, Canada, 2003. ACM Press.
6. E. Brill, S. Dumais, and M. Banko. An analysis of the AskMSR question-answering system. In *2002 Conference on Empirical Methods in Natural Language Processing*, 2002.
7. J. Lin, A. Fernandes, B. Katz, G. Marton, and S. Tellex. Extracting answers from the web using knowledge annotation and knowledge mining techniques. In *Eleventh Text REtrieval Conference (TREC 2002)*, Gaithersburg, Maryland, November 2002.
8. B. Katz, J. Lin, D. Loreto, W. Hildebrandt, M. Bilotti, S. Felshin, A. Fernandes, G. Marton, and F. Mora. Integrating web-based and corpus-based techniques for question answering. In *Twelfth Text REtrieval Conference (TREC 2003)*, Gaithersburg, Maryland, November 2003.
9. A. Del-Castillo, M. Montes y Gómez, and L. Villaseñor-Pineda. QA on the web: A preliminary study for spanish language. In *Encuentro Internacional de Ciencias de la Computación (ENC'04)*, Colima, Mexico, September 2004.
10. B. Magnini, S. Romagnoli, A. Vallin, J. Herrera, A. Peñas, V. Peinado, F. Verdejo, and M. de Rijke. Creating the DISEQuA corpus: a test set for multilingual question answering. In Carol Peters, editor, *Working Notes for the CLEF 2003 Workshop*, Trondheim, Norway, August 2003.
11. T. Joachims. *Learning to Classify Text using Support Vector Machines: Methods Theory and Algorithms*, volume 668 of *The Kluwer International Series in Engineering and Computer Science*. Kluwer Academic Publishers, 2002.
12. M. O. Stitson, J. A. E. Wetson, A. Gammerman, V. Vovk, and V. Vapnik. Theory of support vector machines. Technical Report CSD-TR-96-17, Royal Holloway University of London, England, December 1996.
13. V. Vapnik. *The Nature of Statistical Learning Theory*. Number ISBN 0-387-94559-8. Springer, N.Y., 1995.
14. B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond*. MIT Press, 2002.
15. A. Kilgarriff and G. Grefenstette. Introduction to the special issue on the web as corpus. *Computational Linguistics*, 29(3):333–347, 2003.

Learning the Query Generation Patterns

Marcin Skowron and Kenji Araki

Graduate School of Information Science and Technology,
Hokkaido University, Kita-ku, Kita 14,
Nishi 9, Sapporo, Japan 060–0814
{ms, araki}@media.eng.hokudai.ac.jp

Abstract. With the current method of query formation, a Question Answering system retrieves a set of documents that are similar to a question, while what is mostly required is a set where an answer occurs frequently. This paper addresses this problem by presenting the Query Generation Pattern method. The aim of the method is to automatically learn an optimal combination, modifications of question words and possible extension of a query with non-question words, which for a given question category and syntax, form reliable queries that retrieve an answer-rich set of documents.

1 Introduction

In a Question Answering (QA) system, locating answers requires text analysis at a level of details that cannot be performed at a satisfactory retrieval time for large text collections[1]. As a result, most of the current QA systems employ a two-stage approach. The aim of the first stage is to select a set of documents relevant to a query from the whole document collection. In the second stage, a detailed analysis of a selected set is performed to find answers. Although, the performance of the second stage, and consequently of a whole QA system depends heavily on the quality of documents retrieved in the first stage, to date the research in this area drew relatively little attention.

The commonly used keyword based and similar approaches to a query formation do not retrieve an optimal set of documents. With this approach, a QA system retrieves a set of documents that are similar to a question, while what a user requests is an answer. In order to provide an answer, a QA system needs to retrieve a set of documents, where such answers occur frequently. In our opinion, for a given question category and question syntax, patterns that transform a given question into a reliable query can be automatically learned in a training process using question-answer pairs. Below, we introduce the idea and learning process of the Query Generation Pattern method. The preliminary test demonstrated a significant improvement in the results, compared to the commonly used keyword based and similar methods of query formation.

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 620–623, 2005.

2 Basic Idea of Query Generation Pattern

In the current QA systems, a query is often generated by removing the functional and stop words. For example, for the question "What does BBC stand for?", the query (BBC stand) or (BBC stands) would be formed. Once submitted to a search engine, this query retrieves a distorted set of documents where the correct answer - "British Broadcasting Corporation" occurs relatively infrequently. However, for the same question a more reliable query ("BBC stands for"), can be formed to retrieve a less distorted set of documents. To generate such a query, the preposition (excluded in the current method), and knowledge of which words to use, as well as how to modify (stand → stands) and order them to form an "exact phrase", is required. In the current approach, these means are not available. Moreover, the queries generated by the current QA systems do not provide any information on where to expect an answer candidate to appear, thus complicating the candidates' extraction process. Such information can be associated with the latter query ("BBC stands for" <answer candidate>). Additional improvement of query reliability can be achieved by extending it with a word or other non-letter characters, which frequently connect a given query with a correct answer. For example, for the question "When was Al Pacino born?", (question category: NUM:date) the query ("Al Pacino was born on") can be generated by the addition of the preposition "on", which is frequently found with the answer, like in the phrase "Al Pacino was born on 25 April 1940 [..]".

We think that for the questions from the same question category and with similar syntax, reliable queries are formed in a similar manner. These transformation patterns can be represented using the POS tags assigned to question words and by providing information on possible query extension with words that do not appear in an original question. For example, for the question "When was Queen Victoria born?" (syntax: /WRB1/VBD1/NNP1/NNP2/VBN1/?) (question category: NUM:date) a reliable query can be generated using the pattern: ("/NNP1 /NNP2 /VBD1 /VBN1 on") learned from the previous example from the same question category and with similar syntax " When was Al Pacino born?". This pattern forms the query ("Queen Victoria was born on"). The idea of a Query Generation Pattern (QGP) method[4] is to automatically learn an optimal combination, and modifications of question and non-question words, which for a given question category and syntax form a set of reliable queries. The aim of such a query is to retrieve a set of documents, where answers occur frequently and to indicate the possible localization of an answer candidate, which further simplifies the answer candidates extraction process.

The effectiveness of surface patterns was demonstrated in the system that best performed in TREC10 QA track[5]. This achievement resulted in further researches that described methods for automatic acquisitions of surface patterns and provided further evaluation of this method[1][3][6]. These works also revealed several shortcomings and limitations, like the fact that the patterns could include only one question key phrase; inability to handle more complicated question syntax, and very limited scope of question types. The QGP method described in this paper provides the means to learn question patterns automatically for the

wide range of question types[1]. It differs also from the previous researches in extensive usage of information of syntax structure of questions and by combining several query formation techniques like "exact match", question words modification and query extension with non-question words, into one complex query generation method.

3 Learning and Testing Query Generation Patterns

In the training process we used a set of 50 question-answer pairs from the TREC QA Collection, from various question categories. The process was started with a query that consisted of an answer and question words including nouns, adjectives, adverbs, and verbs that were not on the stop-word list. Additionally, an initial query was extended with question related words, including various verb and noun forms derived from the main verb found in a question. For example, for the question "When did the Vesuvius last erupt?", the query (1944+Vesuvius+last+erupt OR erupts OR erupted OR eruption OR erupting) was generated. From the set of 100 documents accessed with this query, sentences that contain an answer and at least one question or question related word were extracted. Using these sentences as training data, the list of most frequent n-grams that contained only the question and question related words was generated. For the example question, the list of the n-grams with an occurrence greater than the set threshold included strings like: "last erupted", "last eruption", and "the last eruption". In the next step, discovered n-grams were extended with the remaining question words that did not appear in a given n-gram string, either directly or as one of the derivative forms. These constituted a set of the Query Generation Pattern (QGP) candidates. The reliability of a given candidate was calculated as a number of answers (the multiple occurrences of an answer in one snippet is counted only once) to the number of accessed snippets (a number between 1-100). The QGPs with the highest reliability score were selected and stored. Table 1 presents the results for some of the discovered patterns. The most reliable query found for this question is approximately 40% more reliable than one that could be generated by a current QA System (position 3) by extracting the keywords and transforming the verb form ((did) erupt → erupted). For all the questions from the training set, QGP method was able to discover queries more reliably than those formed using a keyword based approach.

In the same process, using all snippets containing a correct answer, the words that frequently link a question word with an answer - Connection Patterns (CP) - were discovered. The most frequent CPs were joined to the corresponding QGP. Such extended QGPs are verified using the method described above. If found to form a reliable query, it was added as an additional pattern to be stored along with a particular question category and question syntax. Table 1, position 1 presents the QGPs extended with the CP.

[1] Question types used in the training process included 50 fine-grained categories. For the details see http://l2r.cs.uiuc.edu/cogcomp/Data/QA/QC/definition.html.

Table 1. Examples of QGP found for the question "When did the Vesuvius last erupt", (question category: NUM:date) (syntax: /WRB1/VBD1/DT1/NNP1/JJ1/VB1/?)

No.	Query	Query Pattern	Reliability Score
1	"last erupted in" Vesuvius	"/JJ1 /VB1_ed in" /NNP1	60
2	"last erupted" Vesuvius	"/JJ1 /VB1_ed" /NNP1	46
3	last erupted Vesuvius	/JJ1 /VB1_ed /NNP1	42
4	last eruption Vesuvius	/JJ1 /VB1_tion /NNP1	40
5	"the last eruption" Vesuvius	"/DT1 /JJ1 /VB1_tion" /NNP1	38

The application of the discovered QGPs for the set of 50 test questions (various question categories, syntax similar to the questions used in the training process) confirmed that using the learned patterns, the system could automatically generate a set of highly reliable queries that retrieved a set of documents where an answer occurred more frequently, compared to the currently used keyword based approach. For the test set, the improvement rate varied depending on the question, between 17%-76%.

4 Conclusions and Future Work

The Query Generation Pattern method demonstrates that the QA system can automatically acquire knowledge on how to form a set of reliable queries for a given question category and question syntax. Using this method, the system also obtains information on where an answer candidate is likely to occur and what words or non-letter characters frequently connect it to a given query, even if these elements were not present in a question. The preliminary results are promising, showing a significant improvement over the currently used method. Our future work includes extensive evaluation of the proposed method and providing the means to extend a query with words semantically related to a question.

References

1. Greenwood M. (2002) Question Answering. PhD Progress Report, http:// www.dcs.shef.ac.uk/ mark/phd/work.
2. Hovy E. Hermajakob U., Ravichandran D. (2002) Proceedings of the Human Language Technology (HLT) Conference.
3. Ravichandran D., Hovy E. (2002) Learning Surface Text Patterns for a Question Answering System. In Proc. of 40^{th} Annual Meeting of the Association for Computational Linguistics (ACL).
4. Skowron M., Araki K. (2003) Basic Idea of Corpus-Supported Approach to Question Answering. Convention Record of the Hokkaido Chapters of the IEEE.
5. Soubbotin M.M., Soubbotin S.M. (2002) Patterns of Potential Answer Expressions as Clues to the Right Answer. In Proc. of 10^{th} Text retrieval Conference (TREC10).
6. Zhang D., Lee W. (2002) Web Based Pattern Mining and Matching Approach to Question Answering. In Proceedings of the 11th Text REtrieval Conference (TREC).

Exploiting Question Concepts for Query Expansion

Hae-Jung Kim, Ki-Dong Bu, Junghyun Kim, and Sang-Jo Lee

Department of Computer Engineering, Kyungpook National University,
Sangyuk-dong, Puk-gu, Daegu, 702-701, Korea
hjkim325@hanmail.net

Abstract. In this paper, we present an efficient semantic query expansion methodology based on a question concept list comprised of terms that are semantically close to concepts represented in a query. The proposed system first constructs a concept list for each question concept and then learns the concept list for each question concept. When a new query is given, the question is classified into the question concept, and the query is expanded using the concept list of the classified concept. In the question answering experiments on 42,654 Wall Street Journal documents of the TREC collection, the traditional system showed in 0.223 in MRR and the proposed system showed 0.50 superior to the traditional question answering system.

1 Introduction

Question answering (QA) systems assign relevance degrees to words, paragraphs or clauses based on a given query, and then provide answers ranked according to relevance. However, the efficacy of such systems is limited by the fact that the terms used in a query may be in a syntactic form different to that of the same words in a document. Consider, for example, the following query and sentences:

– Who is the inventor of a paper?
– S1: C is the inventor of knives
– S2: a devised paper in China…

When analyzing this query, the traditional QA system would classify the sample query into "NAME" as a subcategory of "PERSON", and then keywords such as "inventor" and "paper" would be extracted. In this example, however, S1 contains the keyword "inventor" and S2 contains the keyword "paper", and hence their relevance degrees for the query will be the same. Moreover, even if we expand the keywords to "inventor", "discoverer", and "paper", the ranking of the sample sentences will remain unchanged because the term "devise" in S2 belongs to a syntactic category different to that of "inventor" in the query. However, if we were to expand the keyword "inventor" to include related words such as "discoverer", "devise", "invent", "develop", and "creator", then we could represent the same concept over a range of syntactic and semantic categories, and thereby reduce the number of answer candidates and extract more exact answers.

In this paper, we present an efficient semantic query expansion methodology based on a question concept list comprised of terms that are semantically close to concepts

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 624 – 627, 2005.

represented in a query. The concept list associated with a particular query includes most possible representations of the concept of the question.

2 Previous Work

Answer type of QA system can be called semantic category of the query that a user requested, and it had an influence on a QA system performance enhancement to express answer type as the small classification of semantic category [1-4]. Cardie *et al*. [1] modified the traditional approach to question type classification by dividing the answer type into 13 subcategories, thereby creating more specific question categories. This modification significantly improved the performance of the traditional QA system. Prager *et al*. [4] proposed an alternative methodology for finding the semantic class that covering all possible semantic classes used in a query; specifically, they determined a synset of question terms by using an inventory such as a hypernym tree from WordNet. However, their method entails the derivation of the synset-class mapping, which is a labor-intensive task that results in incomplete coverage. In contrast to the above methods, our method contains the concept list of each question concept that can be used to expand query terms into conceptually close terms.

3 Query Expansion Based on a Question Concept List

3.1 System Description

Figure 1 shows a flow diagram of the overall system configuration. The system contains three main components: In the question concept list construction module, the concept list of each question concept is constructed for query expansion. First, the concepts of question categories are established according to important question concepts by the question concept classification module. Then, the concept list for each question concept is constructed by query pattern recognition. In the concept pattern learning module, the system learns the constructed concept list of each question concept using a learning algorithm. Finally, in the question analysis component, the system classifies the given query into the corresponding question concept node based on learned data, and then expands the query into the semantically close terms using the concept list of a classified concept node.

3.2 Question Concept List Construction

We assume that the important concept of a question will be embodied in the terms that are most frequently used in the question; hence, we categorize the question type based on the term frequency (TF) of two categories, nouns and verbs. We regard terms occupying the upper 30% of the total TF, and the question concepts of 117 Who queries from TREC-9 collection can be categorized into 8 concepts such as “inventor, killer, writer, leader, player, founder, owner, others”.

To construct the concept list of each question concept, we should extract the terms that represent the concepts of each question. To facilitate extraction of the concept of the query, we extract the pattern of the query as defined in Definition 1. For example,

the question pattern from the query "Who is the inventor of paper?" is <Who, null, is_BE, inventor_NN>.

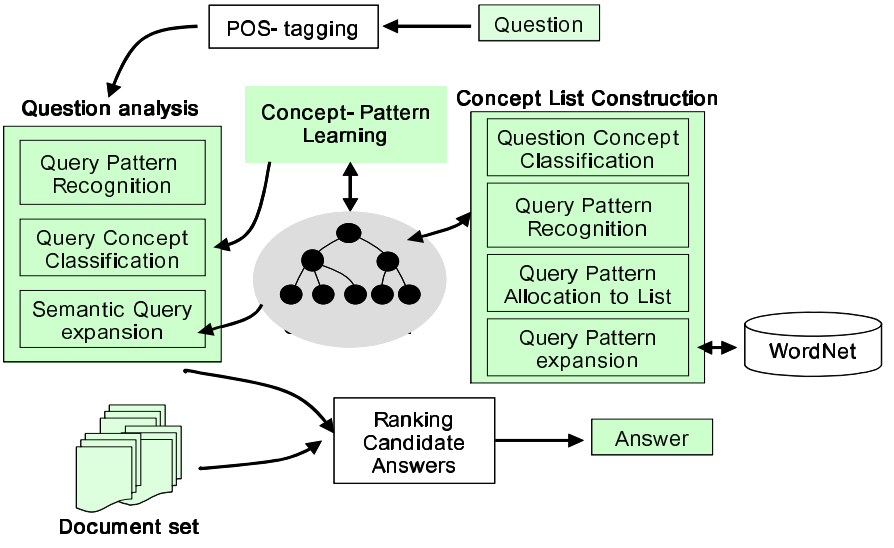


Fig. 1. Overall System Configuration

[Definition 1: Question pattern]
Question patterns are defined as the following two types based on the noun (N) and verbs (BE_V, V) around Wh_term, where, BE_V is the verb "to be" or one of its conjugated forms. Noun N1 is the first noun before verb V and noun N2 is the first noun after verb V.

Question pattern 1 = [Wh_term, N1, BE_V, N2]
Question pattern 2 = [Wh_term, N, V]

The patterns extracted from a query are assigned to the corresponding question concept and make up the "concept list" that represents the concept of a question.

3.3 Question Concept Learning and Query Expansion

For concept learning and classification, we use the Naïve Bayes theorem. For the terms v_j in a pattern, we calculate the $P(v_j|S_k)$ and $P(v_j)$ for all concepts k and select the S_k that has the highest probability as the question concept S' of the given query.

$$Decide \quad S' \quad if \quad S' = \arg\max_{S_k} [\log P(S_k) + \sum_{v_j \, in \, C} \log P(v_j \mid S_k)]$$

For a new query, the proposed system extracts the query pattern and then classifies the query into the question concept based on the learned data. The system then acquires the expansion terms from the concept list in the classified question concept.

4 Evaluation

To test the proposed method, we first tested the classification performance of the constructed question concept list, and applied the proposed query expansion method to the question answering system. We used precision as measures of the accuracy.

When the total number of patterns in the constructed concept list was 117 Who queries from TREC-9 collection learning and classification performance were 94.4% precision for the learning set and 78.7% for the test set by 10-fold cross validation.

Table 3. Precision in 10-fold cross validation for concept list learning

	Traning set	Test set
Micro avg.	0.944	0.787

We conducted retrieval test on the Wall Street Journal (WSJ) 1991 with 42,654 documents and 18 who-queries in TREC-9 collection having WSJ 1991 documents as answer set. Similarity measure between questions and documents was:

$$\mathrm{Sim}(Q, D) = \sum_i \sum_j \alpha_I \times \delta(q_i, d_j), \quad \text{where } \delta(q_i, d_j) = 1 \text{ if } q_i = d_j, \text{ otherwise } 0.$$

Table 4 shows the Mean Reciprocal Ratio (MRR) results for the comparison of the traditional QA system and the proposed. The proposed system showed 0.50 superior to the traditional system when the sentence boundary was three sentences.

Table 4. MRR of the traditional QA system and the proposed system

	Traditional	The proposed
Three sentences	0.223	0.500

5 Conclusions

In this paper, by assuming that the important concepts of a query are embodied in the most frequently used terms in the query, we constructed a question concept list that contains an expanded collection of query terms related to the concept of a query. When we evaluated the performance of the proposed method, the proposed system showed 0.50 in MRR superior to the traditional system. The results of the present experiments suggest the promise of the proposed method.

References

1. C., Cardie, V., Ng, D., Pierce and C., Buckley, Examining the Role of Statistical and Linguistic Knowledge Sources in a General-Knowledge Question-Answering System, In Proceeding of the 6th Applied Natural Language Processing Conference, 2000, 180-187
2. E. Hovy, et al., Learning Surface Text Patterns for a Question Answering system, In Proceedings of the ACL conference, 2002, 180-187.
3. H. Kazawa, T. Hirao, H. Isozaki, and E. Maeda, A machine learning approach for QA and Novelty Tracks:NTT system description, In Procs. of the 11th Text Retrieval Conf., 2003.
4. J. Prager, D. Radev, E. Brown, A. Coden, The Use of Predictive Annotation for Question-Answering in TREC8, In Proceedings of the TREC-8 Conference NIST, 2000, 309-316

Experiment on Combining Sources of Evidence for Passage Retrieval *

Alexander Gelbukh[1], NamO Kang[2], and SangYong Han[2,+]

[1] National Polytechnic Institute, Mexico
gelbukh@gelbukh.com
www.Gelbukh.com
[2] Chung-Ang University, Korea
kang@archi.cse.cau.ac.kr, hansy@cau.ac.kr

Abstract. Passage retrieval consists in identifying short but informative runs of a long text, given a specific user query. We discuss the sources of evidence that help choosing likely high-quality passages, such as relevance to the user query and self-containedness. These measures are different from the traditional information retrieval procedure due to the use of the context of the passage.

1 Introduction

Unlike full document retrieval—a traditional task of information retrieval—passage retrieval task [2, 5] consists in identifying in a long document (or collection of long documents) short text runs relevant for a specific user query, which—unlike in question answering task—do not allow a simple factual answer.

In this paper we discuss the parameters affecting selection of such passages from the text of the document. We intentionally do not give any specific formulas since our experimental result do not yet allow us to reliably argue in favor of a specific way of calculation of these parameters.

2 The Method

Our algorithm consists in the following steps:

– Preprocessing,
– Candidate passage generation,
– Assessing various properties of each candidate passage,
– Combining the obtained scores for each property in a single value overall score.

Then the passages are presented to the user in the order of obtained scores. The steps of the algorithm and the specific quality measures are described in the following subsections.

Preprocessing. Currently we only apply tokenization and stemming. In the future, anaphora resolution would be desirable, as well as resolution of other types of coreference, including hidden anaphora.

* This research was supported by ITRI of the Chung-Ang University. Work was done when the first author was on Sabbatical leave at Chung-Ang University.
+ Corresponding author.

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 628 – 631, 2005.

Generating Candidate Passage. We select as possible candidates all text windows containing from 5 to 1000 words. At this stage, we do not care of their suitability, since unfit candidates will be ruled out later on, so the lower and upper limits on the passage length were motivated only by efficiency considerations. In the future, generation of possible candidates can be made in a more intelligent manner.

In case of retrieval over a multiple-document collection, we generate all such candidate windows for all the texts in the collection.

Scoring the Candidates. Each candidate window is evaluated independently. The properties to be assessed are related with two main requirements, which often are contradicting:

– Desired passages should be *relevant*, i.e., should not contain irrelevant words,
– They should be *self-contained*, i.e., contain enough context to be understandable.

Note that the generated candidates are overlapping, so that for any passage (shorter than 1000 words) there are other passages containing it. With a proper balance between the two contradicting requirements, preference is given to slightly longer passages that contain the same relevant information but are more self-contained. However, too long passages receive too low relevance score and are ruled out.

Below we present the specific criteria used in our current experiments. Other criteria can be added in the future.

Scoring the Relevance for the Query. To score relevance, we use known information retrieval techniques, considering each passage as an independent document. In contrast to the usual information retrieval task, however, we have access to the global context of the document, which helps disambiguation as well as more accurate weighting of the importance of keywords.

To evaluate the relevance of a given passage for the user query, we use vector space similarity measure [1]. Note that this measure gives lower scores to longer passages containing more words irrelevant for the query. With this, of nested candidate windows, a shorter window containing more keywords from the query would be preferred. On the other hand, since the vector measure uses frequencies, a longer passage that contains a greater number of relevant words can receive greater score.

To allow the latter effect, the weights of the words from the query should be set to a much greater value than that of the terms not appearing in the query. This allows preferring a window with, say, 10 extra irrelevant words, to include an additional relevant word. This is a parameter that can be adjusted to control the desired size of the passages, i.e., the preference of relevance over completeness of the results.

Another factor affecting the keyword weighting is IDF weighting known from information retrieval. For usual documents, IDF weighting is measured over the whole collection, so that all documents in the collection contribute equally to the IDF weights. In case of passages, they are in linear context of the surrounding text. On the one hand, this can help in word sense disambiguation and anaphora resolution (which we do not tough upon in this paper). On the other hand, this allows for more accurate calculation of IDF value. Namely, in addition to the usual IDF, which is inversely proportional to the number of documents in the collection containing the word in

question, we use a document-related value, which is inversely proportional to the number of occurrences of the given word in the paragraphs of the same document. We scale this additional value by the distance (in paragraphs) from the given passage, so that it decreases exponentially with the distance.

The reason for this additional value is that even if a word is not very frequent in general language or in the whole collection, in can be frequent in the given (long) document, and in this case it expresses an idea that is probably already known to the user, thus not contributing to the information value of the extracted passages. However, if the word is used in the same document far from the passage in question, it can express a different idea, which does not affect the given passage.

Scoring the Self-Containedness. The requirement of self-containedness implies that desired passages, in particular, should not contain logical references or dependencies on outside of the passage. We use heuristic knowledge-poor approaches to asses the suitability of the passage. In particular, to assure the absence, or to minimize the number of, references outside the given passage, we prefer the passages that:

– Lay at the boundaries of structural units of the text, e.g., beginning of a chapter,
– Represent thematic threads of the text and do not have many thematic relations with the neighboring sentences.

Accordingly, we score higher the candidate windows that lay at the boundary of structural units of the document. Namely, we give an additional bonus to the windows lying at the beginning of a unit, and a smaller bonus to those at the end of the unit. Chapter boundaries are more important than section boundaries, which in turn are more important than paragraph boundaries, in the sense of a greater bonus. As a simplification, one can choose to consider only complete sentences, thus ensuring some degree of self-containedness, but losing short sub-sentences of complex sentences that otherwise would be good candidates.

Similarly, we boost the scores of the candidate windows that show high degree of internal interrelatedness between words and low degree of relatedness between their words to the words in surrounding context. The relatedness with the context is less dangerous at the end of the passage than at its beginning. Indeed, a passage related to the preceding context is likely to develop on the ideas explained earlier, and thus is likely not to be understandable out of context. On the other hand, if a passage is related to the following context, this may indicate that the ideas introduced in the passage are developed later on, but the passage still should be understandable without this continuation.

The idea of linguistic word relatedness is that, say, *teacher* is related to *school* and is not related to *sleep*; there exists a number of word relatedness measures suggested in literature [2, 4]. To determine the contribution of every specific pair of running words in the text to the inner interrelatedness of the paragraph or to the relatedness of the paragraph to the context, we scale the linguistic relatedness of the corresponding words by an exponentially decreasing function of the distance between them.

Combining the Scores. We combine the partial scores in a multiplicative manner, so that a candidate window that is either irrelevant (even if very comprehensible) or incomprehensible out of context (even if very relevant) is not presented to the user. The combination can be tuned to give preference to shorter (more relevant) or longer (more self-contained) candidates.

Experimental Results. We are not aware of any standard evaluation procedure for passage retrieval. We evaluated the results by manual inspection of the answers to sample questions. For example, the top three passages retrieved for the query *wars between England and France* from *A Child's History of England* by Charles Dickens, 164,772 words, are the following:

- *The Queen's husband who was now mostly abroad in his own dominions and generally made a coarse jest of her to his more familiar courtiers was at war with France and came over to seek the assistance of England. England was very unwilling to engage in a French war for his sake but it happened that the King of France at this very time aided a descent upon the English coast.*
- *As his one merry head might have been far from safe if these things had been known they were kept very quiet and war was declared by France and England against the Dutch.*
- Same as 1 plus: *Hence war was declared greatly to Philip's satisfaction and the Queen raised a sum of money with which to carry it on by every unjustifiable means in her power.*

As one can see, the lack of semantic processing (ignoring the word *between* in the query) results in some passages in fact unrelated to the query, like the second passage in the table. Elements of meaning understanding can be a topic of future work.

3 Conclusions

Linear positioning of candidate text windows in the context and their variable length allow for estimating some characteristics of passages different from those used in traditional information retrieval, e.g., topical relatedness to the context. However, specific formulas for estimating of such parameters and the ways to tune their combination to obtain optimal results are the topics of our current and future research.

References

1. R. Baeza-Yates, B. Ribeiro-Neto. *Modern Information Retrieval.* Addison Wesley, 1999.
2. C. L. A. Clarke, G. V. Cormack, T. R. Lynam, E. L. Terra. Question Answering by Passage Selection. In *Advances in Open Domain Question Answering*, Kluwer, 2004.
3. G. Hirst, D. St-Onge. Lexical chains as representations of context for the detection and correction of malapropisms. In: C. Fellbaum (Ed.), *WordNet: An electronic lexical database*, Cambridge, MA: The MIT Press, 1998.
4. S. Patwardhan, S. Banerjee, T. Pedersen. Using Measures of Semantic Relatedness for Word Sense Disambiguation. A. Gelbukh (Ed.), Computational Linguistics and Intelligent Text Processing (CICLing-2003), LNCS N 2588, Springer, 2003, p. 241–257.
5. G. Salton, J. Allan, C. Buckley. Approaches to passage retrieval in full text information systems. ACM SIGIR Research and Development in Information Retrieval, 1993, 49–58.

Summarisation Through Discourse Structure

Dan Cristea[1,2], Oana Postolache[1,3], and Ionuţ Pistol[1]

[1] Al. I. Cuza University, Iaşi, Romania
{cristea, ipistol}@infoiasi.ro
[2] Institute for Theoretical Computer Science,
Natural Language Processing Group, Iaşi, Romania
[3] Computational Linguistics, Saarland University,
Saarbrücken, Germany
oana@coli.uni-sb.de

Abstract. In this paper we describe a method to obtain summaries focussed on specific characters of a free text. Summaries are extracted from discourse structures which differ from RST structures by the fact that the trees are binary and lack relation names. The discourse tree structures are obtained by combining constraints given by cue-phrases (resembling Marcu's method) with constraints coming from the exploitation of cohesion and coherence properties of the discourse (as proved by Veins Theory). The architecture of a summarisation system is presented on which evaluations intended to evidence the contribution of each module in the final result are performed and discussed.

1 Introduction

In this paper we describe an approach to discourse parsing and summarisation that exploits cohesion and coherence properties of texts. We built discourse structures that resemble the RST (Rhetorical Structure Theory [1]) trees, although ours are binary and lack relation names. Discourse tree building resembles the cue-phrase centred approach of Marcu [2] but adds to it constraints coming from the exploitation of the relation that is proved to exist by Veins Theory (VT) [3] between discourse structure and reference chains (a manifestation of cohesion), on the one hand, and between the global discourse structure and the smoothness of centering transitions (a manifestation of coherence) [4], on the other. The output of the parsing process is used to obtain excerpt-type summaries focussed on individual characters mentioned in the text. A combined, pipe-line/parallel/incremental, type of processing is employed.

The involved modules are POS-tagging, FDG-parsing, clause segmentation of sentences in clauses, construction of elementary discourse trees, detection of noun phrases (NPs), anaphora resolution (AR), discourse parsing and summarisation. To master the combinatorial explosion yield by different sources of ambiguity, a beam-search processing is employed. We present the architecture of a discourse parsing system and discuss the evaluation methodology. The final evaluation

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 632–644, 2005.

is realised by comparing the summaries output by the system against those contributed by human subjects.

Section 2 presents the overall method and the architecture of the system. Section 3 gives a quick overview on veins theory, which stays at the basis of the focussed summarisation method. Section 4 presents the method of incremental parsing and the module that assembles elementary discourse trees corresponding to sentences. Section 5 describes how the exponential explosion induced by different sources of ambiguity is controlled. In section 6 the corpus and the evaluation method are presented and section 7 discusses the results and synthesises some conclusions, limitations, and further work.

2 The Method

We call *focussed summary* on a character/entity X, a coherent excerpt presenting how X is involved in the story that constitutes the content of the text. Such summaries are of importance in information retrieval tasks from news or scientific papers when mentions of a certain entity are traced in a document. Note that a generic summary of a discourse sometimes will not include a desired character/entity if this entity appears only collaterally in the given discourse. Suppose, for instance, a drugs company interested to track in medical journals or scientific papers all mentions of a certain drug manufactured by them; neither extraction of the contexts of the drug mentions in the articles, nor generic summaries of the articles can be of help, as the intention is to know how is the drug mentioned *within the general topics of the articles*.

We describe the architecture of a system that combines a pipe-line style of processing the text with a parallel and an incremental one, with the aim to obtain an RST-like discourse structure that marks the topology and nuclearity, while ignoring the names of the rhetorical relations. Such trees are then used to compute focussed summaries on searched discourse entities. In the process of building discourse trees, we consider properties of the relationship between reference chains and the discourse structure as well as between global discourse structure and the smoothness of centering transitions. Both reference chains and centering transitions are related with veins expressions computed following the veins theory (VT) [3].

First, the text is POS-tagged, then a syntactic parser (FDG) is run over it. Further, the process is split into two flows: one that segments the sentences into *elementary discourse units* (*edu*s) and then constructs *elementary discourse trees* (*edt*s) of each sentence, and another that detects NPs and then runs an anaphora resolution engine to detect coreferential relations. Intermediate files in the processing flow are in the XML format. When two processes join, the resulted files are merged into a single representation. An *edt* is a discourse tree whose leaf-nodes are the *edu*s of one sentence. Sentence-internal cue-words/phrases trigger the constituency of syntactically *edt*s from each sentence [2], [5]. For each sentence in the original text a set of *edt*s is obtained. At this point a process that simulates the human power of incremental discourse processing is started. At any

moment in the developing process, say after n steps corresponding to the first n sentences, a forest of trees is kept, representing the most promising structures built by combining in all possible ways all *edt*s of all n sentences. Each such tree corresponds to one possible interpretation of the text processed so far. Then, at step *n+1* of the incremental discourse parsing, the following operations are undertaken: first, all *edt*s corresponding to the next sentence are integrated in all possible ways onto all the trees of the existing forest; then the resulted trees are scored according to four independent criteria, sorted and filtered so that only a fraction of them is retained (again the most promising after *n+1* steps). From the final wave of trees, obtained after the last step, the highly scored is selected. Summaries are then computed on this tree.

In [6] a general framework to resolve anaphors is proposed. We use this framework to integrate a model of coreference resolution that deals with most types of anaphors. Centering transitions scores are computed after AR is run, therefore after all references are solved. References and transitions, as well as heuristics for the proper development of a discourse tree, contribute with scores to the overall score of a developing discourse tree. These scores are then used to control the beam-search.

3 Veins Theory and Focussed Summarisation

Veins theory (VT) [3] is used in the described process to guide the incremental tree building and to synthesize summaries. VT makes two claims: emphasizes the close relationship between discourse structure and referentiality, as an expression of text cohesion, and generalizes Centering Theory (CT) [4] to the global discourse, as an expression of text coherence. Moreover, VT adds a view on summarization (consistent with [2]) and naturally reveals how focused summaries can be produced.

The fundamental intuition underlying an integrated account on discourse structure and accessibility in VT is that the RST-specific distinction between nuclei and satellites limits the range of referents to which anaphors can be resolved; in other words, the nucleus-satellite distinction, superimposed over a tree-like structure of discourse, induces a *domain of evocative accessibility* (*dea*) for each anaphor. More precisely, for each anaphor x in a discourse unit u, VT hypothesizes that x can be resolved by examining discourse entities from a subset of the discourse units that precede u. In this way VT reveals a "hidden" structure in the discourse tree, called *vein*. The notion of vein synthesizes observations on how references interact with the discourse structure represented as an RST tree in which names of relations were ignored (we will call such a simplified representation an RST-like tree). Considering the hierarchical organization given by the tree structure and the principle of compositionality [2], which induces recursively long-distance relations between *edu*s, these observations can be stated as follows:

- a right satellite or a nucleus can refer its left nuclear sibling;
- a right nucleus can refer its left satellite;

- in a combination $n_1\ s_1\ s_2$, with s_1 and s_2 satellites of the nucleus n_1, s_1 is not accessible from s_2;
- in a combination $n_1\ s_1\ n_2$, with s_1 a satellite of the nucleus n_1 and n_2 a right nuclear sibling of n_1, s_1 is not accessible from n_2;
- a nucleus blocks the reference from a right satellite to a left satellite, therefore in a combination $s_1\ n_1\ s_2$, with s_1 and s_2 satellites of the nucleus n_1, s_1 is not accessible from s_2.

The vein expression of an *edu* u is a list of *edu*s of the discourse, including u, which is meant to express the sequence of units that are significant to understand u **in the context of the whole discourse**.

VT classifies references into three categories, in accordance with the way they align along the veins. An anaphor, belonging to an *edu* u_2, is said to issue a **direct reference**, if its linearly most recent antecedent belongs to an *edu* u_1 that is included in u_2's vein. Under the same notations, it issues an **indirect reference** if u_1 does not belong to u_2's vein, but there is a more distant antecedent, say belonging to an *edu* u_0, and u_0 is placed on u_2's vein. If the backward-looking reference chain of the anaphor does not intersect the vein of the anaphor's *edu*, we have an **inferential reference**. VT conjunctures on two types of anaphoric processes: **evocative** (or **immediate**)and **post-evocative** (or **inferential**). The evocative processes are most frequent, are rapid and can be realised by any referential means, including those as fragile as empty pronouns. They make the discourse fluid and increase the text cohesion. An evocative anaphora occurs anytime the backward-looking chain of referential links having the right-most end in the current anaphor intersects at least once the vein expression of the *edu* the anaphor belongs to (the cases of direct and indirect references). This means that the antecedent can be recuperated looking to the left only in the sub-discourse obtained by concatenating the *edu*s in the vein expression of the current anaphors *edu*. The post-evocative anaphorae are less frequent, induce more inferential load on the reader (hearer) and make use of strong referential means (like proper nouns, for instance). A post-evocative anaphora is one in which there is no *edu* of the anaphor's referential chain which belongs also to the anaphor's vein expression (the case of the inferential reference).

A corollary of VTs claims is that the text obtained by the concatenation of the spans indicated in the vein expression of an *edu* is a sub-discourse that gives a summary of the whole discourse, focused on that particular unit. Now, suppose one discourse entity is traced and a summary focused on that entity is desired. If there is only one *edu* in which the entity is mentioned, the vein expression of that *edu* gives a very well-focused summary of the entity. A problem appears if the entity is mentioned in more than just one *edu*. Because there is no a-priory reason to prefer one of the focused summaries obtained in this way to any of the others, it is clear that a combination of the vein expressions of each *edu* in which the entity is mentioned should be considered. We have proposed more methods [5] of building a final summary from the collection of particular summaries. The first method takes the vein expression of the lowest node of the tree that covers all units in which the entity is mentioned. Since the

length of a vein expression is proportional to the deepness of the node in the tree structure, this method results in shorter summaries. The second method considers that particular summary (vein expression) which sums most of the mentions of the entity. The third method simply takes the union of all vein expressions of the units that mention the entity in focus. Finally, the fourth method builds a histogram from all vein expressions of the units mentioning the focussed entity and selects all units above a certain threshold. The last two methods are not in themselves vein expressions, and therefore are more prone to incoherent summaries than the first two methods, the last one being the most exposed. In our experiments till now we have used only the first method.

4 Incremental Parsing

The basic step in an incremental discourse parser is the integration of an elementary discourse tree (*edt*), which corresponds to a sententence, int the tree representing the discourse structure of the discourse parsed so far. By doing this we will obtain discourse trees in which to each sentence corresponds one node of the discourse structure covering exactly the sentence's span ([7] have shown that in 95% of the cases this is true). The operations applied at each step during the incremental processing is *adjunction* on the right frontier [8]. Cue-words and cue-phrases (markers) are connectives having a signalling function on: the nuclearity of the *edu*s they interconnect, the form of the *edt* they belong to, and the place on the right frontier of the developing tree where an *edt* is to be adjoined. Subordinate connectives, like *just*, *as*, *although*, *as long as*, *whenever*, *because*, etc., link subordinate clauses (satellite structures) onto regent clauses (nuclear structures), while coordinate connectives, like *and*, *or*, etc., usually link sibling nuclear structures. There are also frequent cases when connectives miss completely. Different patterns of arguments for markers have been manually selected from a corpus. Fig. 1 depicts some cases (the dots suggest the nuclearities of their arguments). There are frequent cases when the same marker has more than one argument pattern.

As constraints to build syntactically correct trees we have used the rules described in [5]. Such constraints configure *edt*s in which inner nodes are labelled with markers and leaf-nodes with *edu* labels. Each node of the tree is also marked by a nuclearity function with *n* (for nuclear) or *s* (for satellite) so that at each level, between the two descendents of an inner node, at least one is marked *n*, and the root of an *edt* is always marked *s*. Since the number of inner nodes of a binary tree with *t* leaf-nodes is *t-1*, for an *edt* to be completely determined it needs a

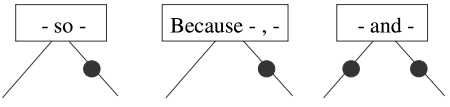


Fig. 1. Argument patterns of cue-phrases

number of cue-words, as inner *edt* nodes, with one less than the number of *edu*s. For such reasons we apply heuristics to add dummy markers where missing. Dummy markers are empty strings similar to *and* with both arguments labelled as nuclear (the implicit assumption is that a satellite is always announced by a realised marker). The incremental parsing in [5] is deterministic. Heuristics help, at each step, to adjoin the current *edt* in that place of the right frontier of the developing tree which maximizes the chances to arrive at a correct final analysis. Instead, our analysis does not go deterministically. At each step, all possible trees resulted from the application of marker argument-structure patterns and syntactic constraints are generated and then are adjoined in all possible positions of the right frontier of the developing tree. To control the exponential explosion induced by this luxurious behaviour we implemented a beam-search-like process.

5 The Beam-Search Control

Any beam-search-like process depends heavily on a scoring function able to appreciate the relevance of the objects produced at intermediate steps, and which are successively detailed or improved until a final object, supposed to satisfy the goal, is obtained. In this section we explain our scoring criteria. In [9] an empirical evaluation of VT's conjectures is described. Experiments drawn on corpora annotated to both discourse structure (RST) and coreference have shown that VT's conjectures are generally correct. The authors of VT report that 87.1% of all references they found in the investigated corpus are direct, and 8.5% are indirect. The rest of 4.4% escape the predictions of VT, some being classified as of a pragmatic type (not needing an antecedent in order to be understood) [3]. However, an important aspect is that exceptions align their frequencies per types with their evoking power, as folows: pragmatic – 56.3%, proper nouns – 22.7%, common nouns – 16.0%, pronouns – 5.0%. Following [10], the *evoking power* of each of these types of REs decreases as we move to the right in the list. Pragmatic references are those which refer to entities that can be assumed as part of general knowledge, such as *the Senate* or *our* in the phrase *our streets*. The descending order of the types of the expressions disobeying VT suggests that pragmatic references are easily understood without an antecedent while proper nouns and common noun phrases are understood less and less. At the other extreme, pronouns have very poor evoking power: a message emitter employs them only when s/he is certain that the structure of the discourse allows for an easy recuperation of the antecedent in the message receiver's memory. Except for the cases where a pronoun can be understood without an antecedent (as in the example with *our* in *our streets*), the use of a pronoun referring an antecedent that is outside the *dea* should produce an invalid message. Since the detection of pragmatic references requires knowledge that goes beyond the possibilities of our sources, we considered only proper nouns, comon nouns and pronouns for the scoring criterion based on references.

To score *references in relation with veins* we have given the values 2, 1 and 0 for the values **direct**, **indirect** and **outside vein**, respectively. Then, to score

the *anaphor type* we have given the values 3, 2 and 1 for the following categories of anaphors: **pronoun**, **common noun** and **proper noun**, respectively. Then we have multiplied these scores for each anaphor, allowing each anaphor to contribute to the general score of the tree with a value between 0 and 6, with 0 meaning that any of its antecedents are outside the *dea* of the unit of the anaphor, and 6 in case of a pronoun whose most recent antecedent is on the *dea* of the unit the anaphor belongs to. This is the s_r section of the score (see below).

The second tree-scoring criterion used the coherence conjecture of VT. Following [3], we let each unit to contribute with a score between 0 and 4, depending on the type of centering transition between the current unit and the previous unit in the vein expression, in ascending order of smoothness: **no C**$_b$, **abrupt shift**, **smooth shift**, **retaining** and **continuing** [4]. As will be shown below, the score formula is designed to keep track of the relationship between references and structure. This is the section s_c of the score (see below). The overall contribution in the score of a tree coming from VT represents the s_1 section of the score formula, and has the following form:

$$s_1 = \sum_{u \in D} \left(w_1 \sum_{x \in RE_u} \frac{s_r^x}{6} + w_2 \frac{s_c^u}{4} \right) \tag{1}$$

where u is an *edu*, D represents the whole discourse, x is an anaphor, RE_u is the set of the anaphors belonging to unit u which have antecedents outside that unit, s_r^x is the referential score contributed by the anaphor x and s_c^u is the centering score contributed by the unit u. The two weights w_1 and w_2 sum-up to 1 and are iteratively computed to accommodate optimally the score scheme to the expected results.

During the experiments we have noticed a tendency of the parsing trees to be skewed downward and to the right (a tree with this particular shape corresponds to a discourse in which each *edu* addsa detail to the preceding one, while a tree completely skewed upward and to the right corresponds roughly to a discourse in which each *edu* adds a detail to the initial *edu*. To balance this tendency we scored better an adjunction of an *edt* on the upper part of the right frontier of the developing tree than on the lower part. The contribution of this criterion represents the s_2 section in the score formula (see below).

Section s_3 of the score formula is thought to penalize too many nuclear nodes in the final tree. A tree that has only nuclear nodes is a flat structure, but between the two daughters of a node at least one should be nuclear. So, s_3 is the fraction between the number of satellites and the total number of nodes of the tree.

Finally, the last section of the score, s_4, reflects the quality of the *edt*s which are build from sentences. Each *edt* is compared against the structure returned by the FDG parser (only for English) with respect to the nuclearity of the *edu*s (0.5) and the identity of the sibling node in the structure (0.5) and then we average the sum on the number of *edu*s in the segment.

In principle, at each step of the search we have a fixed number N of developing trees and to each of them we adjoin in all possible ways all computed *edt*s. The

score of each new developing tree obtained as such is calculated as the product s_1 * s_2 * s_3 * s_4. Then we sort all these trees in the descending order of their scores and we retain for the next step again the first N best rated trees. At the end of the run, the best scored final tree gives the discourse structure.

6 Corpus and Evaluation

We have done parallel experiments on both Romanian and English. As a test we have used a fragment summing up 812 words from G. Orwell's novel "1984" in the English version and 863 words in its Romanian equivalent .

We believe that the evaluation of a complex NLP system should follow a procedure that facilitates an easy inventory of the depreciation of performance along the processing chain. This way, the identification of critical points of the system is straightforward and repairing can be focussed towards the points of maximum trouble. In this section we show how we use such a technology in order to evaluate our summarizer for both English and Romanian. The overall processing flow of the system and the points where the "temperature" is measured are depicted in Fig. 2. Early processing phases, as POS-tagging and FDG-parsing are considered included in the input in this scheme. Processing modules are indicated in light grey rectangles, evaluation results in dark squares, and files in rounded rectangles: those which are pure outputs of processing modules - in white, and those influenced in any way by a gold-standard - shadowed. The names of the files indicate their origin, so, for instance `np-seg-gold-ar-edt-tree-test` is a file that records a gold-standard (gold) of manually annotated noun-phrases (np) and *edu*s (seg), as well as the results (test) of runing the AR-module (ar), the edt-detector module (edt) and the discourse parser module (tree). Also, `sum-gold` and `all-test` are the two most distant final files, recording respectively the gold-standard of summary and the output of a complete and pure (no human intervention) processing chain.

All initial gold standards, `seg-gold`, `np-gold` and `np-ar-gold` have been created by master students in Computational Linguistics, while the `sum-gold` file was build with the help of a class of 91 terminal year undergraduate students in Computer Science, during an NLP examination. They received the initial text in which *edu*s were already marked and numbered and were asked to indicate 4 summaries by writing down sequences of discourse unit numbers: a general summary of the whole text of about 20% reduction rate and three summaries focussed on different characters mentioned in the text (*Winston's mother*, *Winston's sister* and *the girl with black hair*). For each *edu* of the original text we counted the number of times this *edu* was included in any students' summaries. As such, a histogram resulted, with the sequence of *edu* numbers on the x-axis and the frequency of mentioning on the y-axis. Then we considered a sliding horizontal threshold on this histogram, and accepted as belonging to the golden summary all units whose corresponding frequencies were above the threshold. During tests we have established the threshold to a number of hits of 20, which resulted in a gold-summary of length 30 *edu*s.

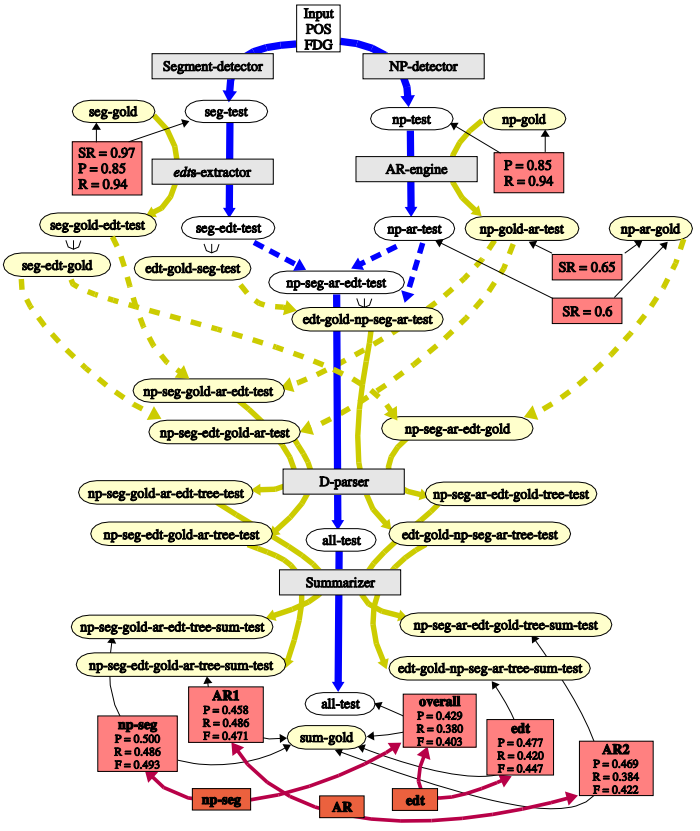


Fig. 2. Processing and evaluation points

Fig. 2 shows the processing flow and results for the implementation running English texts. In the upper part of the diagram the evaluation points are meant to determine the behaviour of the segment-detector, the NP-detector and the AR-engine, independent of the overall summarization task of the system. In the figure, P stands for precision, R for recall and SR for success rate, conforming to [11]. Precision and recall in the case of segment-detector have been computed in terms of segment borders, while success rate as the number of words correctly assigned to segments (belonging to *edu*s around the same main verb), divided by the total number of words.

As Fig. 2 shows we do not have a gold standard for discourse structure (a file `tree-gold` is absent). To evaluate our trees we used instead summaries, easier to acquire than RST-like annotations of discourse structure. If summaries extracted automatically, as by-products of a discourse parsing process, resemble those indicated by human subjects, then we should a high degree of confidence that the structures themselves reflect with enough accuracy the text content.

As baseline for our general summaries evaluation we have used the summary produced by MS Word on the same text. As baseline for the three focussed summaries we selected all sentences containing the expressions *his mother*, *his sister* and *girl*. Example 1 displays part of the text under experiment, on which the gold general summary is in boldface and the automated summary in italics.

Example 1. ***Winston was dreaming of his mother.***

He must, he thought, **have been ten or eleven years old when his mother had disappeared.** *She was a tall, statuesque, rather silent woman with slow movements and magnificent fair hair.* **His father he remembered more vaguely as dark and thin, dressed always in neat dark clothes** (Winston remembered especially the very thin soles of his father's shoes) and wearing spectacles. **The two of them must evidently have been swallowed up in one of the first great purges of the fifties.**

At this moment his mother was sitting in some place deep down beneath him, with his young sister in her arms. ***He did not remember his sister at all, except as a tiny, feeble baby, always silent, with large, watchful eyes.*** **Both of them were looking up at him. They were down in some subterranean place – the bottom of a well, for instance, or a very deep grave** – but it was a place which, already far below him, was itself moving downwards. ***They were in the saloon of a sinking ship, looking up at him through the darkening water.*** There was still air in the saloon, they could still see him and he them, but all the while they were sinking down, down into the green waters which in another moment must hide them from sight for ever. ***He was out in the light and air while they were being sucked down to death****, and they were down there* because he was up here. He knew it and they knew it, and he could see the knowledge in their faces. *There was no reproach either in their faces or in their hearts, only the knowledge* that they must die in order that he might remain alive, and that this was part of the unavoidable order of things.

He could not remember what had happened, but he knew in his dream that in some way the lives of his mother and his sister had been sacrificed to his own. *It was one of those dreams which*, while retaining the characteristic dream scenery, are a continuation of one's intellectual life, *and in which one becomes aware of facts and ideas* which still seem new and valuable after one is awake. **The thing** that now suddenly struck Winston **was that his mother's death, nearly thirty years ago, had been tragic and sorrowful in a way that was no longer possible**. *Tragedy, he perceived, belonged to the ancient time, to a time* when there was still privacy, love, and friendship, *and when the members of a family stood by one another without needing to know the reason.* His mother's memory tore at his heart because she had died loving him, when he was too young and selfish to love her in return, and because somehow, he did not remember how, she had sacrificed herself to a conception of loyalty that was private and unalterable. ***Such things****,he saw,* ***could not happen today.***

7 Discussions and Conclusion

As seen in Fig. 2 the segment-detector behaves satisfactory. A less good precision but very good recall was obtained also for the NP detector. A significant deterioration of the results are expected to occur following the AR-phase since the extreme extravagance of a free text as Orwell's novel and the need to trace

Table 1. Statistics of the edt-extractor

No of *edus*	No of sentences of this length	No of generated *edt*s per sentence
1-3	25	1-4
4-5	9	5-28
6	1	42

at once all types of anaphors made resolution of the coreferring anaphora a very difficult task. Comparing the two SR values (0.65 versus 0.6) one can perceive the influence of the NP-detector on the deterioration of the performance of the AR-engine. This behaviour is conformant to the expectations since NPs are the referential expressions that are worked out by the AR-engine. The *edt*s-extractor computed *edt*s as shown in Table 1.

We tested our discourse parser (D-parser in Fig. 2) over the set of 83 *edu*s which were grouped in 35 sentences in both `seg-gold` and `seg-test`.

To master the tree explosion we have used a slightly different threshold policy than the one described in section 5: after each step of the D-parser we have kept only the most promising trees whose combined scores range in a threshold of zero under the best score (tie-vote on the maximum). Using this policy, the maximum number of trees generated in any of the 35 steps was 320.

To learn the optimum weight values of parameters w_1 and w_2 of formula (1) we have run 10 times the whole parser modifying at each step w_1 by 0.1 (remember that w_2 = 1 - w_1). The final results of the general summaries are shown in Fig. 2. For comparison, the MS Word-baseline for the general summary was rated with a precision of 0.222, a recall of 0.176 and an F-measure of 0.197. Also, the best student general summary was rated with a precision of 1.00, a recall of 0.679 and an F-measure of 0.801. The implementation was done in Java. The interested reader can consult documentation and perform experiments with modules described in this paper at the following adresses: AR-engine at `www.coli.uni-sb.de/~oana/rare` and Discourse Parser and Summarizer at `www3.infoiasi.ro/~ipistol/parser`.

Different black boxes displaying recall (R) and precision (P) and F-measure (F) values in the lower part of Fig. 2 show different evaluations made over the summarisation system by comparing outputs in which part of the work is done manually and part of it automatically against the summary gold standard file `sum-gold`:

- `overall` – evaluates the all-automatically obtained output file `all-test`;
- `edt` – evaluates the output corresponding to an input in which elementary trees of sentences have been contributed manually;
- `np-seg` – evaluates the output corresponding to a manual detection of NPs and segmentation;

- `AR1` – evaluates the output corresponding to an input in which all the following steps have been performed manually: detection of NPs, segmentation, and elementary tree detection.
- `AR2` – supplementary to `AR1`, has also the anaphora resolution process manually annotated.

As seen, the results are above the baseline, although the values are still low. The evaluation operated at different point in the processing chain validate the expectations: the more gold components we incorporate, the more accurate are the results. We could also estimate the impact of the component modules on the summaries by counting the differences between R and P values at the edges of the thick arrows: NP-detector + segment-detector, as the difference between `np-seg` and `overall` values = 0.090; *edt*s-detector, as the difference between `edt` and `overall` values = 0.044, and AR-engine, as the difference between `AR2` and `AR1` values = 0,049. So, it seems that low level processes, as detection of NPs and segmentation influence more the summarization results than high level processes as *edt*-detection and AR resolution. The results on Romanian are still under development, but we expect to be under the ones for English because of the lack of an FDG parser.

The following aspects will make the subject of further work: retraining of the AR and segmentation processes with different heuristics, implementation of the substitution operation in incremental discourse parsing, and the improvement of the performances of the individual modules, and implementation of different focused summarisation criteria by exploiting the vein expression, as described at the end of section 3.

Acknowledgements

Our thanks go to our students who have done the manual annotations and have produced the summaries that helped to draw the final evaluation. Special thanks go to our colleagues from the Laboratory of Computational Linguistics of the University of Wolverhampton who have kindly provided the FDG annotated version of the "1984". Part of the work reported in this paper was performed while the first author was in a visiting research stage at ITC-IRST Trento.

References

1. Mann, W.C., Thompson, S.A.: Rhetorical structure theory: A theory of text organization. Text **8:3** (1988) 243–281
2. Marcu, D.: The Theory and Practice of Discourse Parsing and Summarization. The MIT Press (2000)
3. Cristea, D., Ide, N., Romary, L.: Veins theory: A model of global discourse cohesion and coherence. In: Proceedings of COLING/ACL, Montreal/Canada (1998)
4. Grosz, B.J., Joshi, A.K., Weinstein, S.: Centering: A framework for modelling the local coherence of discourse. Computational Linguistics (1995)

5. Cristea, D., Postolache, O., Puşcaşu, G., Ghetu, L.: Local and global information exploited in producing summaries. In: Proceedings of the International Symposium on Reference Resolution and Its Applications to Question Answering and Summarisation, Venice/Italy (2003)
6. Cristea, D., Dima, G.E.: An integrating framework for anaphora resolution. Information Science and Technology **4** (2001) 273–291
7. Şoricuţ, R., Marcu, D.: Sentence level discourse parsing using syntactic and lexical information. In: Proceedings of HLT-NAACL, Edmonton, Canada (2003)
8. Polanyi, L.: A formal model of the structure of discourse. Journal of Pragmatics **12** (1988) 601–638
9. Cristea, D., Ide, N., Marcu, D., Tablan, V.: An empirical investigation of the relation between discourse structure and co-reference. In: Proceedings of COLING, Saarbücken/Germany (2000) 208–214
10. Gundel, J., Herberg, N., Zacharski, R.: Cognitive status and the form of referring expressions in discourse. Language **69** (1993) 274–307
11. Mitkov, R.: Anaphora Resolution. Longman, London (2002)

LexTrim: A Lexical Cohesion Based Approach to Parse-and-Trim Style Headline Generation

Ruichao Wang, Nicola Stokes, William Doran, Eamonn Newman,
John Dunnion, and Joe Carthy

Intelligent Information Retrieval Group, Department of Computer Science,
University College Dublin, Ireland
{rachel, nicola.stokes, william.doran, eamonn.newman,
john.dunnion, joe.carthy}@ucd.ie

Abstract. In this paper we compare two parse-and-trim style headline generation systems. The Topiary system uses a statistical learning approach to finding topic labels for headlines, while our approach, the LexTrim system, identifies key summary words by analysing the lexical cohesion structure of a text. The performance of these systems is evaluated using the ROUGE evaluation suite on the DUC 2004 news stories collection.

1 Introduction

A headline is a very short summary (usually less than 10 words) describing the essential message of a piece of text. Like other types of summaries, news story headlines are used to help a reader to quickly identify information that is of interest to them in a presentation format such as a newspaper or a website. Although newspaper articles have already been assigned headlines, there are other types of news text sources, such as transcripts of radio and television broadcasts, where this type of summary information is missing. In 2003 the Document Understanding Conference (DUC) added the headline generation task to their annual summarisation evaluation. This task was also included in the 2004 evaluation plan where summary quality was automatically judged using a set of n-gram word overlap metrics called ROUGE [1]. The best performing system at this workshop was the Topiary approach [2] which generated headlines by combining a set of topic descriptors generated from the DUC 2004 corpus with a compressed version of the lead sentence, e.g. (Topic Descriptors) *BIN_LADEN EMBASSY BOMBING:* (Compressed Lead Sentence) *FBI agents this week began questioning relatives of the victims.*

Topiary-style summaries perform well in the ROUGE evaluation for a number of reasons. Firstly, summarisation researchers have observed that the lead sentence of a news story is in itself often an adequate summary of the text. However, it has also been observed that additional important information about a topic may be spread across other sentences in the text. The success of the Topiary-style summaries at DUC 2004 can be attributed to fact that this technique takes both of these observations into consideration when generating titles.

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 645 – 648, 2005.

In this paper we compare two different methods of generated topic labels and observe their effect on summary quality when combined with compressed lead sentences. The Topiary system generates topic descriptors using a statistical approach called Unsupervised Topic Discovery (UTD) [2]. This technique creates topic models with corresponding topic descriptors for different news story events in the DUC 2004 corpus. One of the problems with this approach is that it requires clusters of related documents in order to facilitate the generation of topic models and descriptors, i.e. it needs a structured corpus such as the DUC 2004 collection. In this paper we investigate the use of lexical cohesion analysis as a means of determining these event labels. The advantage of this approach is that the descriptors are gleaned from the source text being summarised, so no additional on-topic news story documents from the DUC 2004 corpus are needed to determine appropriate topic labels for a particular story headline. In Section 2, we describe how we analyse the lexical cohesive structure of news texts using our lexical chaining algorithm. In Section 3, we report on the results of our title generation experiments on the DUC 2004 collection and compare the performance of our system LexTrim with the Topiary approach to this task.

2 The Topiary and LexTrim Headline Generation Systems

In this section, we describe the Topiary system developed at the University of Maryland. This is followed by a description of our headline generation system, LexTrim, which also returns a headline consisting of topic keywords and a compressed version of the lead sentence of the source document. The Topiary system takes a two-step approach to headline generation for news stories:

1. The lead sentence is compressed using the Hedge Trimmer algorithm [2, 3]. This parse-and-trim approach to headline generation removes constituents of a parse tree representing the lead sentence that can be eliminated without affecting the factual correctness or grammaticality of the sentence. A set of linguistically motivated trimming rules is defined in [3] and [2]. These rules iteratively remove constituents until the desired sentence compression rate is reached. Firstly determiners, time expressions and other low content words are removed. More drastic compression rules are then applied to remove larger constituents like trailing prepositional phrases and preposed adjuncts until the desired length is reached.

2. The compressed sentence is then concatenated with a list of relevant topic words generated by the UTD algorithm. This unsupervised information extraction algorithm firstly identifies commonly occurring words and phrases in the DUC corpus. Then for each document in the corpus it identifies an initial set of important topic names using a modified version of the *tf.idf* metric. Topic models are then created from these topic names using the OnTopic™ software package. The list of topic labels associated with the topic model closest in content to the source document is then added to the compressed lead sentence produced in the previous step, resulting in a Topiary-style summary such as the example in Section 1.

The LexTrim system, on the other hand, uses our implementation of the Hedge Trimmer algorithm and a lexical cohesion-based approach to identifying pertinent topic labels. Lexical cohesion is the textual characteristic responsible for making the

sentences of a text appear coherent. One method of exploring lexical cohesive relationships between words in a text is to build a set of lexical chains for that text. In this context a lexical chain is a cluster of semantically related proper noun and noun phrases, e.g. {boat, ship, yacht, rudder, hull, bow}. The semantic relationships (i.e. synonymy, holonymy, hyponymy, meronymy, hypernymy) are identified using the WordNet taxonomy. Once lexical chains have been generated for a news story, topic phrases are extracted and concatenated with the condensed lead sentence to form a headline. Topic phrases are noun/proper noun phrases that occur in lexical chains with high lexical cohesion scores, i.e. they are phrases that exhibit strong semantic relationships with other important phrases in the text, and so are considered important topic labels. A more detailed description of our lexical chaining algorithm and these cohesion scores can be found in [4].

3 Evaluation Methodology and Results

In this section we present the results of our headline generation experiments on the DUC 2004 corpus using the ROUGE evaluation metrics. In this task, participants were asked to generate very short (<= 75 bytes) summaries of single documents on TDT-defined events. The DUC 2004 corpus consists of 500 Associated Press and New York Times newswire documents. The headline-style summaries created by each system were evaluated against a set of human-generated (or model) summaries using the ROUGE (Recall-Oriented Understudy for Gisting Evaluation) metrics. The format of the evaluation was based on six scoring metrics: ROUGE-1, ROUGE-2, ROUGE-3, ROUGE-4, ROUGE-LCS and ROUGE-W. The first four metrics are based on the average n-gram match, where $1 \leq n \leq 4$, between a set of model summaries and the system-generated summary for each document in the corpus. ROUGE-LCS calculates the longest common sub-string between the system summaries and the models, and ROUGE-W is a weighted version of the LCS measure. In the official DUC 2004 evaluation all summary words were stemmed before the ROUGE metrics were calculated; however stopwords were not removed.

Table 1 shows results from our headline generation experiments on the DUC 2004 collection. The aim of these experiments was two-fold: to build a linguistically motivated heuristic approach to title generation, and to look at alternative techniques for padding Topiary-style headlines with content words. As explained in Section 3, our approach, LexTrim, augments condensed lead sentences with high scoring noun phrases that exhibit strong lexical cohesive relationships with other candidate terms in a news story. The Lex system in Table 1 returns headlines consisting of lexical chain phrases only. A comparison of the LexTrim, Lex and Trim system results show that the inclusion of lexical chain topic descriptors significantly improves the 'informativeness' of the compressed lead sentence generated by the Trim system.

Comparing these system results to the performance of the Topiary and UTD DUC 2004 systems, we can see that the Lex system outperforms the UTD system for all ROUGE metrics. This indicates that our lexical chaining method identifies better topic descriptors than the UTD method. A comparison of the Topiary and LexTrim ROUGE scores also indicate that this is the case. This is an interesting result as it shows that a knowledge-based NLP approach (using WordNet) to identifying topic

labels is as good as, if not better than, the statistics-based UTD approach that requires additional word frequency and co-occurrence information from the DUC 2004 corpus before it can predict salient topic labels for a particular document. Hence, our lexical chaining approach is a useful alternative to the UTD method when a corpus containing additional documents on the topic of a particular news story is not available during headline generation.

Table 1. ROUGE scores for headline generation systems on the DUC 2004 collection

SYSTEM	ROUGE-1	ROUGE-2	ROUGE-3	ROUGE-4	ROUGE-L	ROUGE-W
LexTrim	0.25370	0.06208	0.02260	0.00870	0.20099	0.11951
Topiary	0.24914	0.06449	0.02122	0.00712	0.19951	0.11891
Trim	0.20061	0.06283	0.02266	0.00792	0.18248	0.10996
Lex	0.18224	0.02903	0.00663	0.00089	0.14676	0.08738
UTD	0.15913	0.01585	0.00087	0.00000	0.13041	0.07797

4 Conclusions

In this paper, we have compared the performance of two headline generation systems that use two distinct techniques for 'padding out' compressed lead sentences in the automatic generation of news story headlines. The results of our experiments using the ROUGE evaluation suite indicate that lexical chain phrases are more informative topic descriptors than statistically-derived topic labels for this task. We intend to proceed in future work by improving the sentence compression procedure described in this paper. Currently, we are investigating the use of lexical cohesion information as a means of improving the performance of the Hedge Trimmer algorithm by limiting the elimination of important parse tree components during sentence compression.

References

1. Lin, C-Y, Hovy E. Automatic Evaluation of Summaries using n-gram Co-occurrence Statistics. In the Proceedings of HLT/NACCL, 2003.
2. Zajic D., Dorr, B., Schwartz, R. BBN/UMD at DUC-2004: Topiary. In the Proceedings of the Document Understanding Conference (DUC), 2004.
3. Dorr, B., Zajic D., Schwartz, R. Hedge Trimmer: A Parse-and-Trim Approach to Headline Generation. In the Proceedings of the Document Understanding Conference (DUC), 2003.
4. Stokes, N. Applications of Lexical Cohesion Analysis in the Topic Detection and Tracking domain. Ph.D. thesis. Department of Computer Science, University College Dublin, 2004.

Generating Headline Summary from a Document Set

Kamal Sarkar and Sivaji Bandyopadhyay

Computer Science & Engineering Department, Jadavpur University,
Kolkata – 700 032, India
jukamal2001@yahoo.com, sivaji_ju@vsnl.com

Abstract. This paper discusses an approach to generate headline summary from a set of documents. Headline summary is basically a very short summary in the form of headline. As the amount of on-line information increases, systems that can automatically summarize multiple documents are becoming increasingly desirable. In this situation, headline summary is useful for users who only need information on the main topics in a set of documents. Headline summary from multiple documents will be very useful in the text mining applications for the generation of meaningful label (a compact identifier that allows a person to quickly see what the topic is about) for a cluster of documents.

1 Introduction

In this paper we present a system that will cluster the text documents collected from multiple online sources and generate a headline summary in one or two sentences for each cluster by identifying named entities from each document in the set to make a global list of named entities and forming a headline summary for the set.

All the previous work on headline generation [1, 2] was done on single documents but the focus in the present work is on headline summary generation from a set of documents. Moreover, instead of using only statistical approaches, we have used named entity cues and summary generation techniques for our work, since it is very difficult to have a training corpus of document set—headline pairs. In the next section we present the proposed approach. The system is evaluated in Section 3.

2 Proposed Approach

News collected from multiple sources should be clustered. Clustering technique adopted is similar to the method used by Chen and Lin [3].

The input to our system is a cluster of related documents. Based on the observations of human-produced headline summary, we have developed the following algorithm.

2.1 Algorithm

1. Prepare the local named entity list for each document in the cluster.
2. Prepare the global named entity list for the cluster.
   - Resolve the cross-document co-reference by the approach used in [4]. It uses a global translation table, changing all occurrences of each co-referred named entity to the longest version.

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 649 – 652, 2005.

- Rank the named entities according to their frequency across the local named entity lists.
- Take top *n* named entities in the global list. Value of n should be chosen in such a way that total number of words in the selected named entities should not exceed 10 words, because our objective is to generate the very short summary.

3. Pair the named entities in the global list, which co-occur in the same sentence and one of them occurs in the subject position of the sentence.
4. Identify meaningful sentence segments from the documents in the cluster. If no pairing is possible in Step 3, the sentences containing the most frequently named entity are selected. Otherwise, the following rules are used in the specified order.

   Rule1: *For each named entity pair (A,B) identified in Step 3, select the sentence segment from A to B including both A and B.*
   Rule2: *If (A,B) and (B,C) are two pairs occurring in the same sentence, and A>B>C (A>B means A occurs before B), the sentence segment from A to C is selected.*
   Rule3: *For the rest of the named entities that occur in the subject position and are not a member of any pair, pair each of the named entities with the main verb/verb group of the sentence concerned.*
   Rule4: *The named entities that do not participate in a pair and do not occur in the subject position of any sentence in the document set should simply be ignored.*

5. Headline summary generation:

   The meaningful sentence segments selected for each of the pairs would be clustered. The sentence segments selected using rule 3 in the Step 4 would be clustered separately for each such named entity. If no pairing is at all possible, only one cluster is formed.

   Each cluster will contribute one representative to the final headline summary. If a cluster contains more than one sentence segment, the most summarized one, i.e., the segment containing least number of words will be the representative from that cluster.

   Finally, all the representative sentence segments from the clusters are arranged in a particular order called majority ordering [5], which relies on the original order of sentences in the input documents where from the sentence segments have been selected.

2.2 Example

The following 5 documents have been collected from the different newspapers on US Space Shuttle Columbia crash. It has been illustrated with this example how our algorithm works on this cluster to generate a headline summary in a few sentences.

Doc-1: <Title> Columbia crashes on return <Title>
<Text Start> *Kalpana Chawla, who traveled more than any other Indian in history, met with a tragic, fiery end to her life when American space shuttle, Columbia, in which she was an astronaut, broke up and crashed over Texas* only minutes before it was to land in Florida. Israeli

first astronaut, Ilan Ramon, an air force pilot, perished in the tragedy along with Chawla and five other crew members of the ill-fated above flight.<Text End>

Doc-2: <Title> Space shuttle explodes <Title>
<Text Start> US Space shuttle Columbia with Indian-orgin astronaut *Kalpana Chawla on board burst into flames over Texas,* killing all seven astronauts, minutes before it was to land in Florida on Saturday. In North Texas, several residents reported hearing a big bang, the same time when all radio and data communication with the shuttle and its crew was lost.<Text End>

Doc-3: <Title> Columbia explodes <Title>
<Text Start>*Space shuttle Columbia broke apart in flames as it streaked over Texas* towards its scheduled landing, killing, this time too, all seven astronauts on board, six Americans including India-born Kalpana Chawla, and Ilan Ramon, the first Israeli astronaut to go into space. Nasa did not immediately declare the crew dead, but the *US* flag next to its countdown clock was lowered to half-staff. Later in the day, President Bush said US space exploration would continue despite the loss.<Text End>

Doc-4: <Title> Chawla, rest of crew killed on board space shuttle <Title>
<Text Start>US space shuttle Columbia, carrying seven astronauts including Indian-American Kalpana Chawla, disintegrated shortly before landing at *Cape Canaveral* on Saturday morning in what appeared to be a ghastly replay of the Challenger disaster 27 years ago. NASA sources have confirmed that death of all seven on board.<Text End>

Doc-5: <Title> Columbia burns up over Texas <Title>
<Text Start>*The space shuttle Columbia disintegrated over the state of Texas* today, minutes before its scheduled landing in Florida, killing all seven astronauts on board. Six of them were *Americans* including the Indian-American, Kalpana Chawla and one Israeli air force officer. Calling it as indeed a tragic day for the NASA family, the top administration of the agency, Seon Keefe told a press briefing that the terrible tragedy was not caused from the ground.<Text End>

In the documents, the local named entities have been shown by the underlined words. In the example, the cross document co-references like *US Space Shuttle, US Space Shuttle Columbia, Space Shuttle Columbia, American Space Shuttle, The Space Shuttle Columbia can be resolved to the longest version, US Space Shuttle Columbia*.

Global named entity list: *Kalpana Chawla, US Space Shuttle Columbia, Texas*.
Named Entity Pairs: (*Kalpana Chawla, Texas*), (*US Space Shuttle Columbia, Texas*).
Selected sentence segments: The italic sentence segments in the above documents are the meaningful sentence segments identified by our approach.
Resulting headline summary: Cluster#1 for the pair (*Kalpana Chawla, Texas*) and Cluster#2 for the pair (*US Space Shuttle Columbia, Texas*) will contribute the following sentences to the final summary: *The space shuttle Columbia disintegrated over the state of Texas. Kalpana Chawla on board burst into flames over Texas.*

3 Evaluation and Results

We have extracted the set of distinct words from the headlines of the documents in the document set and this set of words has been considered as a gold standard. While

comparing the output summary and the gold standard, we have considered whether they are string identical or synonymous. The precision and recall have been computed as follows.

If the reference headline summary (words in the gold standard) is of length n words, the generated headline summary is of length k words and p of n words are in the generated headline summary, Precision = p/k and recall = p/n.

For the small document set (< 10 documents) we used the above evaluation method. However, if the size of the document set is very large (say, 100 documents), the reference headline summary is likely to increase in size. So, we have restricted the size of the reference headline summary to the size of generated summary by selecting words from the list of compiled headline words by their frequencies across the headlines of the documents.

For 5 sets of documents collected from the newspapers, the average precision and the average recall which have been achieved by our system are 0.51 and 0.645, respectively.

4 Conclusion and Future Work

We have presented an algorithm for generating a headline summary from a set of related documents. The final summary can be evaluated by an extrinsic evaluation method that determines how well the machine-generated summary can classify the document sets in the test corpus. This evaluation method will be investigated in future.

References

1. Dorr, B., Zajic, D., Schwartz, R.: Hedge trimmer: a parse-and-trim approach to headline generation. In *Proceedings of Workshop on Automatic Summarization* (2003).
2. Zhou, L., Hovy, E.: Headline Summarization at ISI. In proceedings of Workshop on text summarization. Edmonton, Canada (2003).
3. Chen, H., Lin, C.: A multilingual news summarizer. In *Proceedings of the 18th International Conference on Computational Linguistics*, pages 159–165 (2000).
4. Clifton, C., Cooley, R.: TopCat: Data Mining for Topic Identification in a Text Corpus, In IEEE transactions on knowledge and data engineering. Vol.16, No.8, pages 949–964 (2004)
5. Barzilay, D., Elhadad, N., McKeown, R. K.: Sentence Ordering in Multi-document Summarization. *In Proceeding of Human Language Technology Conference (HLT), San Diego (2001).*

Extractive Summarization Based on Word Information and Sentence Position

Carlos Méndez Cruz and Alfonso Medina Urrea

GIL IINGEN UNAM, Apartado Postal 70-472,
04510 Coyoacán, DF, Mexico
{cmendezc, amedinau}@iingen.unam.mx

Abstract. This paper describes an unsupervised experiment of automatic summarization. The idea is to rate each sentence of a document according to the information content of its graphical words. Also, as a minimal measure of document structure, we added a sentence position coefficient.

1 Introduction

Plenty of research has been conducted in the field of automatic summarization. The need for such methods has motivated the exploration of many approaches which reflect the field's complexity. Many aspects of texts must be considered, such as word frequency, document, paragraph and sentence structure, topic and focus structure, information content, etc. Many of these approaches are based on the idea that the greater number of times a linguistic structure or part of it occurs (a word, phrase, sentence, etc.), the more attention the reader will pay to it except when it is a function or grammatical word. That is, a document's sentences receive different levels of attention by human readers. Current interests among researchers are multi-document summarization [1, 2], the application of artificial intelligence methods such as genetic algorithms [2], the use of lexical chains and web resources such as WordNet [3].

Since we are currently developing an open, Spanish language corpus on engineering (CLI) to be available on the Internet [4], we are exploring some summarization techniques to apply to it. The main criteria for this very first experiment was to avoid the heavy techniques that have been and can be developed if one takes into account the complexity of document, paragraph, sentence, and word structure. Thus, we opted for an unsupervised approach based on simple information content measurements that could conceivably be applied to other languages. Actually, information content estimates are typically used for a wide variety of unsupervised tasks. And in fact, some experiments have explored the notions of information content and entropy models for some aspect or another of automatic summarization — for instance, summary evaluation or reductive transformation [5, 6, 7]. In this paper, we will first define some basic concepts. Then, we will briefly describe our application and lastly, we will present results and evaluation strategy.

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 653–656, 2005.

2 Basic Working Concepts

A summary is a reductive transformation of a source text through content reduction by selection of what is important in that source [8]. It is well known that the main problem is to capture the important content of the source text. In general, two sorts of summaries can be produced: extracts and abstracts. The first ones are made by transferring part of the source text to what constitutes the summary. In the latter ones, it is necessary to modify the output to build a clear summary. Even though it is well known that extracts deliver a lower-quality output, we constrained this experiment to generating extracts.

It is also necessary to define what we mean by sentence, since we are dealing with an unsupervised method and thus, clause or sentence structure is really not considered as such. In short, we will here call *sentence* whatever occurs between two periods,[1] that is, the set of phrases surrounded by periods.

3 Method

A Python-NLTK program was developed, which ranks each document sentence according to an index estimated from the information content of its graphical words: $\log_2(p_i)$, where i refers to the graphical word, and p_i to its relative frequency in the document at hand (TF). Thus, in order to obtain a ranking index for a sentence of n words, we can simply average word information: $\frac{1}{n}\sum_{i=1}^{n}\log_2(p_i)$.

Also, we introduced a sentence position coefficient which modifies the index to give prominency to sentences occurring towards the end of the document. The idea is that the latter part of the text is more likely to present informative sentences. Thus, if we take o to be the offset or position of a sentence in a document and s to be the number of sentences in that document, then $\sqrt[25]{\frac{o}{s}}$ grows rapidly for sentences occurring at the beginning of the document and is greater for sentences occurring towards the end.

Hence, the index we used to rank each sentence combines both, word information and sentence position:

$$\frac{\sum_{i=1}^{n}\log_2(p_i)}{n} * \sqrt[25]{o/s} \tag{1}$$

Instead of using a stop list to screen out grammatical words, we used a filter which permitted us to screen function words and infrequent ones. Since function words typically contain the least information, we opted to screen out those word types with less than half of the overall information average in the document:

$$\frac{\sum_{i=1}^{t}\log_2(p_i)}{2t} \tag{2}$$

[1] We are well aware of the use of periods to signal abbreviations (in Spanish and many other languages). However, we have opted not to deal with this mainly because we are seeking unsupervisedness at this point. For our purposes, abbreviations simply cut sentences into smaller units to be ranked as eligible summary candidates.

where t is the number of word types in the document at hand. Lastly, we also filtered out words with frequency in that document of less than 3.

Based on all of this, the program selects for each targeted document the ten sentences with the highest index values in order to produce the final summary. Then, the sentences are listed according to their position in the source text and presented for evaluation.

To test this method, we selected seven documents from the CLI. Five of them were long, technical reports and two were short articles. All of them belong to different thematic areas of engineering: mechanical, electric and electronic engineering. Also, for the sake of comparison, we included two humanities texts (science-fiction and literary criticism).

4 Results and Evaluation

Summarization requires rigorous evaluation. However, the criteria for accomplishing this are so elusive, that it ends up being inevitably subjective. Our simple approach is not likely to do better than heavier approaches, but we devised a simple evaluation scheme to judge results against maximum possible scores restricted to the documents mentioned above.

In essence, we requested eight subjects to read each of the generated summaries and to write a brief text recreating the source text. Thus, they wrote a description of what they thought the source document was about. If the subject guessed the main idea of the source text, a score of 1 was registered, otherwise 0.

It is interesting to note that texts 5, 6 and 7 — which obtained the lowest scores — were the very long, technical reports with many scientific notational idiosyncracies, as well as figures and tables, whose traces appeared as part of the summaries. This made it difficult for the subjects to even read the extracts.

From these scores we can estimate the relative number of positive scores (subjects guessed the main idea of a document a total of 50 times) with respect to the possible number of positive scores (eight readers and nine documents means 72 possible positive scores): $50/72 = 0.69444$. This is a sort of precision measure, which deals with whether or not the subjects' guesses were right. However, it is also important to look at how complete the guesses were. For this, we assigned a score from 0 to 2 to each of the subjects' texts; where 0 meant much of the relevant information was missed, 1 meant some important information was omitted, and 2 no important information was missing.

The long, technical reports — texts 5, 6 and 7 — received again the lowest scores. The much shorter, technical articles and the humanities papers obtained the best scores. This second set of scores can be better appreciated if we consider the relative value of total scores (an accumulated score of 66) with respect to the maximum possible total sum of scores (twice 72). That is, $66/144 = 0.45833$. This value would be a kind of recall measure.

Although these precision and recall measures look encouraging, they constitute no appropriate criteria for comparison to other experiments. Such an

important evaluation remains to be done and will certainly require much more attention than what can be paid in this reduced space.

5 Conclusions

We have presented the results of a very basic and constrained experiment of unsupervised automatic summarization. From the evidence presented, we can conclude that information content should be further explored as a method for reductive transformation (not only summary evaluation).

This experiment can be taken further by varying the number of sentences to be included in the summary, the information content threshold for considering graphical words and sentence position coefficient. Also, we expect that the results can be much improved by including a lemmatization stage — particularly important for an inflectional language like Spanish[2] — and advancing to supervised methods which consider document, paragraph, sentence, phrase and morphological word structure.

Acknowledgments

The work reported on this paper has been supported by DGAPA PAPITT's Project IX402204.

References

1. Saggion, H., Gaizauskas, R.: Multi-document summarization by cluster/profile relevance and redundancy removal. In: DUC. (2004)
2. Jaoua, M., Ben Hamadou, A.: Automatic text summarization of scientific articles based on classification of extract's population. In Gelbukh, A., ed.: Computational Linguistics and Intelligent Text Processing Proceedings. (2003)
3. Song, Y.I., Han, K.S., Rim, H.C.: A Term Weighting Method based on Lexical Chain for Automatic Summarization. In: Lecture Note in Computer Science. Springer (2004) 636–639
4. Medina, A., Sierra, G., Garduño, G., Méndez, C., Saldaña, R.: "CLI: An Open Linguistic Corpus for Engineering". In: Iberamia. (2004) 203–205
5. Ravindra, G., Balakrishnan, N., Ramakrishnan, K.R.: "Multi-Document Automatic Text Summarization Using Entropy Estimates". In: SOFSEM. (2004)
6. Ro, Park.H., S., Han.Y., H, Kim.T.: "Heuristic algorithms for automatic summarization of Korean texts". In: Online Proceedings ICCS/JCSS99. (1999)
7. Hovy, E.: "Text Summarization". In: The Oxford Handbook of Computational Linguistics. Oxford UP (2003) 583–598
8. Spark Jones, K.: Automatic summarizing: factors and directions. In: Advances in automatic summarization. MIT (1999) 1–15
9. Gelbukh, A., Sidorov, G.: "Morphological Analysis of Inflective Languages through Generation". Procesamiento de Lenguaje Natural (2002) 105–112

[2] There already exist suitable tools for lemmatization of Spanish words [9].

Automatic Extraction and Learning of Keyphrases from Scientific Articles

Yaakov HaCohen-Kerner, Zuriel Gross, and Asaf Masa

Department of Computer Sciences,
Jerusalem College of Technology (Machon Lev)
21 Havaad Haleumi St., P.O.B. 16031,
91160 Jerusalem, Israel
{kerner, zuriel, masa}@jct.ac.il

Abstract. Many academic journals and conferences require that each article include a list of keyphrases. These keyphrases should provide general information about the contents and the topics of the article. Keyphrases may save precious time for tasks such as filtering, summarization, and categorization. In this paper, we investigate automatic extraction and learning of keyphrases from scientific articles written in English. Firstly, we introduce various baseline extraction methods. Some of them, formalized by us, are very successful for academic papers. Then, we integrate these methods using different machine learning methods. The best results have been achieved by J48, an improved variant of C4.5. These results are significantly better than those achieved by previous extraction systems, regarded as the state of the art.

1 Introduction

Summarization is a process reducing an information object to a smaller size, and to its most important points [1, 18]. Various kinds of summaries (e.g.: headlines, abstracts, keyphrases, outlines, previews, reviews, biographies and bulletins) can be read with limited effort in a shorter reading time. Therefore, people prefer to read summaries rather than the entire text, before they decide whether they are going to read the whole text or not. Keyphrases, which can be regarded as very short summaries, may help even more. For instance, keyphrases can serve as an initial filter when retrieving documents. Unfortunately, most documents do not include keyphrases.

Moreover, many academic journals and conferences require that each paper will include a list of keyphrases. Therefore, there is a real need for automatic keyphrase extraction at least for academic papers. There are a few such systems. However, their performances are rather low. In this paper, we present a system that gives results significantly better than those achieved by the previous systems.

This paper is organized as follows: Section 2 gives background concerning extraction of keyphrases. Section 3 describes a few general kinds of machine learning. Section 4 presents our baseline extraction methods. Section 5 describes our model. Section 6 presents the results of our experiments and analyzes them. Section 7 discusses the results, concludes and proposes future directions.

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 657 – 669, 2005.

2 Extraction of Keyphrases

A keyphrase is an important concept, presented either in a single word (unigram), e.g.: 'learning', or a collocation, i.e., a meaningful group of two or more words, e.g.: 'machine learning' and 'natural language processing'. Keyphrases should provide general information about the contents of the document and can be seen as an additional kind of a document abstraction.

There are two main approaches concerning keyphrase generation: keyphrase assignment and keyphrase extraction. In the first approach, keyphrases are selected from a predefined list of keyphrases (i.e., a controlled vocabulary) [4]. These keyphrases are treated as classes, and techniques from text classification are used to assign classes to a given document. The training data associates a set of documents with each phrase in the vocabulary. The given document is converted to a vector of features and machine learning methods are used to induce a mapping from the feature space to the set of keyphrases. The advantages of this approach are simplicity and consistently. Similar documents can be described using the same keyphrases. Furthermore, using a controlled vocabulary ensure the required breadth and depth of the document coverage. The disadvantages of this approach are: (1) controlled vocabularies are expensive to create and maintain, so they are not always available and (2) potentially useful keyphrases that occur in the text of a document are ignored if they are not in the vocabulary. An example for a system that implements keyphrase assignment for given documents is described in [7]. In this system, keyphrases are selected from a hierarchical dictionary of concepts. Using this dictionary, general relevant concepts that are not included in the discussed document can be selected.

In the second approach, keyphrase extraction, the approach used in this research, keyphrases are selected from the text of the input document. All words and phrases included in the document are potential keyphrases. Usually, the keyphrases are extracted using machine learning algorithms based on combinations of several baseline extraction methods. The advantages of this approach are: (1) there is no need for creation and maintenance of controlled vocabularies, and (2) important keyphrases that occur in the text can be chosen. The disadvantages of this approach are: (1) lack of consistently; i.e., similar documents might be described using different keyphrases and (2) it is difficult to choose the most suitable keyphrases; i.e., the required breadth and depth of the document coverage is not ensured. An overview on keyphrase extraction methods is given by Jones and Paynter [15]. Among their results, they show that authors do provide good quality keyphrases for their papers.

Turney [22] shows that when authors define their keyphrases without a controlled vocabulary, about 70% to 80% of their keyphrases appear in the body of their documents. This suggests the possibility of using author-assigned free-text keyphrases to train a keyphrase extraction system. In this approach, a document is treated as a set of candidate phrases and the task is to classify each candidate phrase as either a keyphrase or non-keyphrase. A feature vector is calculated for each candidate phrase and machine learning methods are used to classify each candidate phrase as a keyphrase or non-keyphrase.

Although most of the keyphrase extraction systems work on single documents, keyphrase extraction is also used for more complex tasks. Examples of such systems are: (1) automatic web site summarization [27], and (2) keyphrase extraction for a whole corpus [24]. An overview of several relevant keyphrase extraction systems that

work on single documents, which is the investigated issue in this research, is given in the following sub-sections.

Turney [22] developed a keyphrase extraction system. This system uses a few baseline extraction methods, e.g.: TF (term frequency), FA (first appearance of a phrase from the beginning of its document normalized by dividing by the number of words in the document) and TL (length of a phrase in number of words). The best results have been achieved by a genetic algorithm called GenEx. For a collection of 362 articles collected from various domains, his system achieves a precision rate of about 24%. However, subjective human evaluation suggests that about 80% of the extracted keyphrases are acceptable to human readers. In this paper, he reports that these results are much better than the results achieved by the C4.5 decision tree induction algorithm [20] applied to the same task.

Frank et al. [6] propose another keyphrase extraction system called Kea. They used only two baseline extraction methods: TF_xIDF (how important is a phrase to its document) and distance (distance of the first appearance of a phrase from the beginning of its document in number of words). In addition, they apply the naïve Bayes learning method. They show that the quality of the extracted keyphrases improves significantly when domain-specific information is exploited. For a collection of 110 technical computer science articles, their system achieves a precision rate of about 28%, similar to the precision rate of GenEx, 29%, for the same data-base. However, they show that the naïve Bayes learning method used by them is much simpler and quicker than the genetic algorithm applied in GenEx.

Turney, in a further research [23], presents enhancements to the Kea keyphrase extraction algorithm that uses the naïve Bayes algorithm. His enhancements are designed to increase the coherence of the extracted keyphrases. The approach is to use the degree of statistical association among candidate keyphrases as evidence that they may be semantically related. The statistical association is measured using web mining. Experiments demonstrate that more of the output keyphrases match with the authors' keyphrases, which is evidence that their quality has improved. Moreover, the enhancements are not domain-specific: the algorithm generalizes well when it is trained on one domain (computer science documents) and tested on another (physics documents). The main limitation of the new method is the time required to calculate the features using web mining. Evaluation measures such as: recall, precision and F-measure are not presented.

Humphreys [14] proposes a keyphrase extractor for HTML documents. Her method finds important HTML tokens and phrases, determine a weight for each word in the document (biasing in favor of words in the introductory text), and uses a harmonic mean measure called RatePhrase to rank phrases. Her system retrieves a fixed number of phrases, 9, for inclusion in the summary. Using a test bed of URLs, her conclusion is that RatePhrase performs well as GenEx. However, evaluation measures such as: recall, precision and F-measure are not presented and there is no use of any machine learning method.

Hulth [12] develops a system capable of automatic extraction of keyphrases from abstracts of journal papers. In addition to the use of basic features (such as term frequency and n-grams), she used several basic linguistic features, e.g.: NP (Noun Phrase)-chunks and Pos (Part of Speech) tag patterns. These features serve as inputs to a supervised machine learning algorithm called rule induction. She reports on better results than those of Turney and Frank. For a collection of 2000 abstracts of journal papers, the best precision result 29.7% has been achieved by a combination of the linguistic

features: NP-chunks and the Pos tag patterns. The best F-measure score, 33.9%, has been achieved by a combination of the n-gram features and the Pos tag patterns.

In a further research [13], Hulth has reduced the number of incorrectly extracted keyphrases and achieved an F-measure score of 38.1%. The improvement was obtained by: (1) taking the majority vote of the three classifiers used in her previous work [12]: n-grams, NP-chunks and Pos tag patterns and (2) removing the subsumed keywords (keywords that are substrings of other selected keywords). The classifiers were constructed by Rule Discovery Sys-tem (RDS), a system for rule induction. The applied strategy is that of recursive partitioning, where the resulting rules are hierarchically organized (i.e., decision trees).

An additional keyphrase extraction system that makes use of linguistic features has been developed by D'Avanzo et al. [3]. Their system LAKE (Learning Algorithm for Keyphrase Extraction) uses features such as: PoS tagging, multi-word recognition and named entities recognition. They have trained the naïve Bayes classifier on only two features: TF x IDF and First Occurrence. Their conclusions were: (1) PoS-tagging information proved to be far from exhaustive, introducing a lot of noise. Some candidate phrases turned out to be useless pieces of longer sentences or irrelevant, and (2) A filter containing no verbs, proved to be the most reliable one.

Automatic syntactic analysis for detection of word combinations in a given text is proposed in [8]. Using parsing, this system finds word combinations, such as: keyphrases (e.g., *machine learning*), idioms (e.g., *to kick the bucket*) and lexical functions (e.g., *to pay attention*). However, such a full-scale analysis is not usable in real world applications because of unreliable results.

3 Machine Learning Methods

Machine learning (ML) refers to a capability of a system for autonomous acquisition and integration of knowledge. ML occurs in a system that can modify some aspect of itself so that on a subsequent execution with the same input, a different (hopefully better) output is produced. There are three main kinds of learning that can occur in machine learning systems – supervised, unsupervised and reinforcement learning.

Supervised learning is a learning that is supervised by a set of examples with class assignments and the goal is to find a representation of the problem in some feature (attribute) space that is used to build up profiles of the classes. Well-known classification models are: naïve Bayes classification [26], classification by the C4.5 decision tree induction [20] and neural networks [21].

Unsupervised learning has no guidance (supervision) of known classes. Therefore, it has no training stage. Clustering is an example of unsupervised learning. In this case, data which is similar is clustered together to form groups which can be thought of as classes. New data is classified by assignment to the closest matching cluster, and is assumed to have characteristics similar to the other data in the cluster.

Reinforcement learning is one step beyond unsupervised learning. In this learning, systems are given a limited feedback concerning the utility of the input-output mappings that are made. This feedback comes in the form of a reward function. While the reward function does not reveal the correct output for a given input, it does provide the system with an answer of whether the system output was correct or incorrect.

In our model, in order to find the best combinations of the baseline methods for keyphrase extraction we decide to apply supervised machine learning methods. This

kind of learning is well-investigated and rather successful in many domains. In addition, many supervised machine learning methods are available online. Furthermore, previous systems (Turney [22], Frank et al. [6], Hulth [12, 13] and D'Avanzo et al. [3]) framed their keyphrase extraction as a supervised learning problem.

4 Baseline Methods for Selecting the Most Important Keyphrases

In this section, we introduce the baseline methods we use for keyphrase extraction. Several methods are similar to those used in summarization systems (e.g.: [16, 10]) for selecting the most important sentences. Other methods were formalized by us. Similar methods have been used for Hebrew News HTML Documents in [11].

In all methods, words and terms that have a grammatical role for the language are excluded from the key words list according to a ready-made stop list. This stop-list contains approximately 456 high frequency close class words (e.g.: we, this, and, when, in, usually, also, near).

(1) **Term Frequency (TF):** This method rates a term according to the number of its occurrences in the text [5, 17, 9]. Only the N terms with the highest TF in the document are selected.

(2) **Term length (TL):** TL rates a term according to the number of the words included in the term.

(3) **First N Terms (FN)**: Only the first N terms in the document are selected. The assumption is that the most important keyphrases are found at the beginning of the document because people tend to place important information at the beginning. This method is based on the baseline summarization method which chooses the first N sentences. This simple method provides a relatively strong baseline for the performance of any text-summarization method [2].

(4) **Last N Terms (LN)**: Only the last N terms in the document are selected. The assumption is that the most important keyphrases are found at the end of the document because people tend to place their important keyphrases in their conclusions which are usually placed near to the end.

(5) **At the Beginning of its Paragraph (PB):** This method rates a term according to its relative position in its paragraph. The assumption is that the most important keyphrases are likely to be found close to the beginning of their paragraphs.

(6) **At the End of its Paragraph (PE):** This method rates a term according to its relative position in its paragraph. The assumption is that the most important keyphrases are likely to be found close to end of their paragraphs.

(7) **Resemblance to Title (RT)**: This method rates a term according to the resemblance of its sentence to the title of the article. Sentences that resemble the title will be granted a higher score [5, 18, 19].

(8) **Maximal Section Headline Importance (MSHI):** This method rates a term according to its most important presence in a section or headline of the article. It is a known that some parts of papers are more important from the viewpoint of presence of keyphrases. Such parts can be headlines and sections as: abstract, introduction and conclusions.

(9) **Accumulative Section Headline Importance (ASHI):** This method is very similar to the previous one. However, it rates a term according to all its presences in important sections or headlines of the article.

(10) **Negative Brackets (NBR):** Phrases found in brackets are not likely to be keyphrases. Therefore, they are defined as negative phrases, and will grant negative scores.
(11) **TF x MSHI:** This method serves as an interaction between two rather successful methods TF and MSHI. This method resembles the TL*TF method, which was successful in [22].

5 Our Model

5.1 General Description

Our model, in general, is composed of the six following steps (special concepts used in this algorithm will be explained below):

For each article that is in our database:

(1) Extract keyphrases that do not contain stop-list words.
(2) Transform these keyphrases into lower case.
(3) Apply all baseline extraction methods on these keyphrases.
(4) Compare between the most highly weighted keyphrases extracted by our methods to the keyphrases composed by the authors; analyze the results and present full and partial matches.
(5) Apply several common supervised machine learning methods in order to find the best combinations of these baseline methods.
(6) Compare between the best machine learning results achieved in our system to the best machine learning results achieved in systems, which are regarded as the state of the art.

A full match for a unigram is a repetition of the same word including changes such as singular/plural or abbreviations, first letter in lower case / upper case. A partial match between two different unigrams is defined if both words have the same first five letters (explanation below). All other pairs of words are regarded as failures.

A partial match between different unigrams is defined when the first five letters of both words are the same. That is because in such a case we assume that these words have a common radical. Such a definition, on the one hand, usually identifies close words like nouns, verbs, adjectives, and adverbs. On the other hand, it does not enable most of non-similar words to be regarded as partial matches.

A positive example for this definition is as follows: all 8 following words are regarded as partial matches because they have the same 5-letter prefix "analy": the nouns "analysis", "analyst", "analyzer", the verb "analyze", and the adjectives "analytic", "analytical", "analyzable ", and the adverb "analytically". A negative example for this definition is: all 8 following words: "confection", "confab", "confectioner", "confidence", "confess", "configure", "confinement", and "confederacy" are regarded as non partial matches because they have in common only a 4-letter prefix "conf".

Concerning keyphrases which are not unigrams, a full match is a repetition of the same keyphrase. That is, a repetition of all the words included in the keyphrase. A partial match between two different keyphrases is defined when both keyphrases share at least one word. All other pairs of keyphrases are regarded as failures.

Using each one of the baseline methods (Section 4) our system chooses the N most highly weighted keyphrases. The value of N has been set at 5 and 15 in two

different experiments because these values have been used in the experiments done by Turney [22] and Frank et al. [6].

5.2 Evaluation Measures

In order to measure the success of our baseline extraction methods, we use the popular measures: recall, precision and f-measure. These measures are defined briefly below, using the following table of keyphrases' results, which is relevant to our model.

Table 1. Author's and extraction method's keyphrases

		Author's keyphrases	
		True	**False**
Extraction- method's keyphrases	**True**	a	b
	False	c	d

Precision is defined as a / (a + b). Recall is defined as: a / (a + c) and F-Measure which is an harmonic mean of Precision and Recall is defined as $\frac{(\alpha+1)\times \text{Recall}\times \text{Precision}}{\text{Recall}+(\alpha\times \text{Precision})}$, where $\alpha = 1$ gives the same importance for Recall and Precision. In this case, F-Measure is defined as $\frac{2\times \text{Recall}\times \text{Precision}}{(\text{Recall}+\text{Precision})}$.

In our research, it means that recall is defined as the number of keyphrases that appear both within the system's keyphrases and within the keyphrases composed by the authors divided by the number of keyphrases composed by the authors. Precision is defined as the number of keyphrases that appear both within the system's keyphrases and within the keyphrases composed by the authors divided by the number of keyphrases extracted by the system. F-measure is a common weighed average of the above two measures.

6 Experiments

6.1 Data Sets

We have constructed a dataset containing 161 academic papers in Physics taken from the dataset used in the experiments made by Turney [22]. Each document contains in average 1243 sentences containing 8016 words in average. Each document has its own keyphrases composed by the authors of the original documents. The total number of keyphrases is 669. That is, each document contains in average about 4.16 keyphrases. Table 2 presents various statistics concerning these documents. Table 3 presents the distribution of # words per keyphrase.

In addition, it is important to point that about 28% of the keyphrases do not appear (in an exact form) in their papers. Our baseline methods extract only keyphrases that are found in the papers. Therefore, we are limited in full matches to a maximum of 72%. Full and partial presence of keyphrases in their articles is presented in Table 4.

Table 2. Various statistics concerning our dataset

# of keyphrases	# of articles	Accum. % of articles	% of key-phrases	Accum. # of key-phrases	Accum. % of key-phrases
1	4	2.5	0.6	4	0.6
2	16	12.4	4.8	36	5.4
3	59	49.1	26.5	213	31.8
4	37	72.0	22.1	361	54.0
5	14	80.7	10.5	431	64.4
6	19	92.5	17.0	545	81.5
7	5	95.7	5.2	580	86.7
8	1	96.3	1.2	588	87.9
9	2	97.5	2.7	606	90.6
11	2	98.8	3.3	628	93.9
12	1	99.4	1.8	640	95.7
29	1	100.0	4.3	669	100.0

Table 3. Distribution of # words per keyphrase

# of words per keyphrase	# of keyphrases
1	107
2	332
3	166
4	33
5	5
6	3

Table 4. Full and partial presence of keyphrases in their articles

	# in articles	% in articles
Full presence	481	72
Partial presence	150	22
Absence	38	6

About 72% of the keyphrases appear somewhere in the body of their documents. This is similar to the finding of Turney [22] who reports that typically about 70% to 80% of the authors' keyphrases appear somewhere in the body of their documents.

About 6% of the keyphrases composed by their authors do not even exist in a partial form in their papers. Examples of such keyphrases are: (1) "non-equilibrium kinetics", (2) "protons", and (3) "gamma gamma". These keyphrases may be classified into categories of either more general or more specific keyphrases belonged to the research domain that includes the discussed paper. This kind of keyphrases might be found by the system, by for example mining the web and finding similar documents containing them or searching in dictionaries for synonyms. Another solution for finding general keyphrases is to use keyphrase assignment, as done by [7]. Using a

hierarchical dictionary of concepts, relevant concepts that are not included in the discussed paper can be selected.

About 22% of the keyphrases composed by their authors exist in their papers only partially. Examples for such keyphrases are: (1) "lattice simulation" where both "lattice" and "simulation" were included in the paper but separately, (2) "dynamical fermions" where only "fermions" was included in the paper, (3) "Hart rate" where "heart rate" was found in the paper and (4) "$1 = N$ expansion" where "$1 = Nf$ expansion" was found in the paper. The first two keyphrases can be classified into a category of more specific keyphrases belonged to the domain of discussed paper even though they are not mentioned in their papers. This kind of keyphrases might also be found by the system, by mining the web and finding similar documents containing them. The last two keyphrases do not have full matches because of syntax errors. This kind of errors might be discovered while preprocessing the papers and suggestions for correction can be given in this stage.

6.2 Results of Baseline Extraction Methods

Using each baseline method (Section 4), our system chooses the N most highly weighted keyphrases. The value of N has been set at 5 in the first experiment and 15 in the second experiment. These values of N have been chosen because these are the numbers of the retrieved keyphrases by the two previous related systems GenEx [22] and Kea [6]. Table 5 presents the recall, precision and the f-measures results, respectively, of our baseline extraction methods.

Concerning full matches, the best baseline method was found as MSHI (Maximal Section Headline Importance). That is, this method, which is based on the most important headline or section of a given paper, is very successful for academic papers. In contrast to results discovered by Frank et al. [6], in our model, TF (Term Frequency) and FN (First N) were not the best methods. However, they achieve rather good results. This finding might point that these common methods are not the best for academic papers and unique methods designed for academic papers can be better.

Concerning partial matches and up, the best baseline methods were found as TF x MSHI and ASHI (Accumulative Section Headline Importance). Two additional promising methods were PB (at the Beginning of its Paragraph) and TF.

The results presented in Table 5 are based on the keyphrases composed by the authors of the papers, although some of the keyphrases do not exist in the papers. As mentioned in Section 5.1, the result of full matches is limited to a maximum of 72%. Therefore, the results of our baseline methods are actually better.

6.3 Supervised Machine Learning Results

As mentioned in Section 3, we decide to use supervised machine learning methods. We have applied several well-known supervised classification models: naïve Bayes classification [26], classification by the C4.5 decision tree induction [20] and neural networks [21].

Table 5. Precision/recall/f-measures results for our baseline methods

#	Method	Extracted keyphrases	% of full matches			% of partial matches			% of partial matches and up		
			R	P	F	R	P	F	R	P	F
1	TF	5	**6.8**	**4.0**	**5.0**	33.9	18.4	23.9	**40.7**	**22.4**	**28.9**
		15	**13.5**	**3.6**	**5.7**	51.8	13.0	20.7	**65.3**	**16.6**	**26.4**
2	TL	5	3.6	2.0	2.6	2.0	0.0	0.0	5.5	2.0	2.9
		15	8.4	2.1	3.4	0.0	0.0	0.0	8.4	2.1	3.4
3	FN	5	**10.9**	**6.8**	**8.4**	14.3	7.6	9.9	25.2	14.4	18.3
		15	**24.5**	**5.8**	**9.4**	26.1	6.2	10.1	50.6	12.1	19.5
4	LN	5	1.7	0.9	1.1	3.9	2.2	2.8	5.5	3.1	4.0
		15	5.0	1.2	2.0	7.1	1.9	2.9	12.0	3.1	4.9
5	PB	5	7.2	4.1	5.2	38.6	21.1	27.3	**45.8**	**25.2**	**32.5**
		15	19.9	5.2	8.3	45.8	11.3	18.2	**65.7**	**16.6**	**26.5**
6	PE	5	4.3	2.7	3.3	21.7	11.6	15.1	26.0	14.3	18.4
		15	8.5	2.3	3.7	29.8	7.4	11.9	38.3	9.7	15.5
7	RT	5	8.0	4.8	6.0	31.4	17.1	22.2	39.4	22.0	28.2
		15	18.7	1.1	2.2	37.6	13.0	19.4	56.3	14.2	22.7
8	MSHI	5	**17.5**	**10.2**	**12.9**	17.9	9.8	12.7	35.3	20.0	25.5
		15	**29.4**	**7.1**	**11.4**	30.3	7.5	12.1	59.7	14.6	23.5
9	ASHI	5	8.1	4.8	6.1	**36.6**	**19.8**	**25.7**	**44.7**	**24.6**	**31.7**
		15	14.6	3.9	6.2	**54.8**	**13.7**	**21.9**	**69.4**	**17.7**	**28.1**
10	NBR	5	1.9	1.2	1.5	10.2	5.5	7.1	12.1	6.7	8.6
		15	4.4	1.1	1.8	16.9	4.4	6.9	21.3	5.5	8.8
11	TF x MSHI	5	8.9	5.2	6.6	**43.4**	**23.9**	**30.8**	**52.3**	**29.1**	**37.4**
		15	17.9	4.8	7.5	**54.9**	**13.9**	**22.2**	**72.8**	**18.7**	**29.8**

We applied these methods using the web-site of Weka [25], as done by Frank et al. [6] and D'Avanzo et al. [3]. Weka is a collection of machine learning algorithms programmed in Java for data mining tasks, such as: classification, regression, clustering, association rules, and visualization.

Table 6 presents the optimal learning results achieved by three common machine learning methods: J48[1], multilayer perceptron and naïve Bayes.

The best results in Table 6 have been achieved by J48. Therefore, this method has been selected as the best machine-learning method for our task.

Table 7 compares the precision results for extraction of 5 and 15 keyphrases between our system using J48 to the best results achieved by machine learning methods in GenEx and Kea, which are regarded as the state of the art. The reason why we compare only the precision results is because this is the only common measure used by all three systems. Kea presents only precision results. GenEx, in addition, presents a subjective human measure concerning the acceptance of the extracted keyphrases to human readers.

Our results are significantly better than those achieved by GenEx and Kea. For example, our system achieved a precision rate of 55.4% / 28.5% while GenEx achieved (on the smaller dataset) only 29% / 17% and Kea achieved only 28% / 16.5% for 5 / 15 extracted keyphrases, respectively.

In addition, our F-measure results (in Table 6) are significantly better than the best F-measure scores achieved for extraction of keyphrases from journal abstracts by Hulth [12, 13] 33.9% and 38.1%, respectively.

[1] J48 is a machine learning method in Weka [25] that actually implements a slightly improved version (Revision 8) of C4.5.

Table 6. Learning results in our system

Method	Matches	% of precision	% of recall	% of F_measure	Optimal # of extracted keyphrases
J48	Full	**84.1**	**59.46**	**69.67**	2.94
	Partial	**84.5**	**77.25**	**80.71**	3.80
Multilayer Perceptron	Full	77	45.44	57.15	2.45
	Partial	74.8	62.35	68.01	3.46
Naïve Bayes	Full	62.5	53.78	57.81	3.58
	Partial	80.4	19.90	31.91	1.03

Table 7. Comparison of precision results between learning systems

System	# of papers	# of extracted keywords	Precision
GenEx	362	5	23.9%
		15	12.8%
	110	5	29%
		15	17%
Kea	110	5	28%
		15	16.5%
Our System	161	**5**	**55.4%**
		15	**28.5%**

Explanations to these findings can be: (a) we work on academic papers only and we apply specific extraction methods for them; (b) in contrast to the related systems that used combinations of a low number (2/3) of baseline extraction methods, we have used a combination of a relatively high number (11) of baseline methods; and (c) due to J48 we have found a successful combination of our baseline extraction methods.

7 Conclusions and Future Work

Several unique baseline extraction methods, formalized by us have been found as very successful for academic papers. In contrast to previous extraction systems, we have used a combination of a relatively high number of baseline methods. Machine learning results achieved by J48 have been found significantly better than those achieved by extraction systems, which are regarded as the state of the art.

Future directions for research are: (1) Developing methods based on domain-dependant cue phrases for keyphrase extraction, (2) Applying other machine-learning techniques in order to find the most effective combination between these baseline methods, (3) Conducting more experiments using additional documents from additional domains.

Concerning research on academic papers from additional domains, there are many potential research directions. For example: (1) Which extraction methods are good for which domains? (2) What are the specific reasons for methods to perform better or

worse on different domains? (3) What are the guidelines to choose the correct methods for a certain domain? (4) Can the appropriateness of a method for a domain be estimated automatically?

Acknowledgements. The authors would like to thank Peter Turney for sharing his datasets; Ittay Stern and David Korkus for their support; and Alexander Gelbukh and anonymous reviewers for their fruitful comments.

References

1. Alterman, R.: Text Summarization. In: Shapiro, S.C. (ed.): Encyclopedia of Artificial Intelligence. John Wiley & Sons, New York (1992) 1579–1587
2. Brandow, B., Mitze, K., Rau, L.F.: Automatic Condensation of Electronic Publications by Sentence Selection. Information Processing and Management 31(5) (1994) 675–685
3. D'Avanzo, E., Magnini, B., Vallin, A.: Keyphrase Extraction for Summarization Purposes: The LAKE System at DUC-2004. Document Understanding Workshop (2004)
4. Dumais, S. T., Platt, J., Heckerman, D., Sahami, M.: Inductive learning algorithms and representations for text categorization. Proceedings of ACM-CIK International Conference on Information and Knowledge Management, ACM Press, Philadelphia (1998) 148–155
5. Edmundson, H.P.: New Methods in Automatic Extraction. Journal of the ACM. 16(2) (1969) 264–285
6. Frank, E., Paynter, G.W., Witten I.H., Gutwin C., Nevill-Manning, C.G.: Domain-Specific Key-Phrase Extraction. Proc. IJCAI. Morgan Kaufmann (1999) 668–673
7. Gelbukh, A., Sidorov, G., Guzmán-Arenas, A..: A Method of Describing Document Contents through Topic Selection. Proc. SPIRE'99, International Symposium on String Processing and Information Retrieval, Mexico, (1999), 73–80.
8. Gelbukh, A., Sidorov, G., Han, S.-Y., Hernandez-Rubio, E.: Automatic Syntactic Analysis for Detection of Word Combinations. Proc. CICLing-2004: Intelligent Text Processing and Computational Linguistics, Mexico, Lecture Notes in Computer Science 2945, Springer-Verlag, Berlin Heidelberg New York (2004) 243–247
9. HaCohen-Kerner, Y.: Automatic Extraction of Keywords from Abstracts. Proceedings of the Seventh International Conference on Knowledge-Based Intelligent Information & Engineering Systems, Vol. 1. Lecture Notes in Artificial Intelligence 2773. Springer-Verlag, Berlin Heidelberg New York (2003) 843–849
10. HaCohen-Kerner, Y., Malin, E., Chasson, I.: Summarization of Jewish Law Articles in Hebrew, Proceedings of the 16th International Conference on Computer Applications in Industry and Engineering, Las Vegas, Nevada USA, Cary, NC: International Society for Computers and Their Applications (ISCA) (2003) 172–177
11. HaCohen-Kerner, Y., Stern, I., Korkus, D.: Baseline Keyphrase Extraction Methods from Hebrew News HTML Documents, WSEAS Transactions on Information Science and Applications, 6(1) (2004) 1557–1562
12. Hulth, A.: Improved Automatic Keyword Extraction Given More Linguistic Knowledge, Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing, (2003) 216–223
13. Hulth, A.: Reducing False Positives by Expert Combination in Automatic Keyword Indexing. Proceedings of the Conference on Recent Advances in Natural Language Processing (RANLP'03), Borovets (2003) 197–203

14. Humphreys, K.J.B.: Phraserate: An HTML Keyphrase Extractor. Technical report, University of California, Riverside, Riverside, California (2002)
15. Jones, S., Paynter, G.W.: Automatic Extraction of Document Keyphrases for Use in Digital Libraries: Evaluation and Applications, Journal of the American Society for Information Science and Technology, 53(8) (2002) 653–677
16. Kupiec, J., Pederson, J., Chen, F.: A Trainable Document Summarizer. Proceedings of the 18th Annual International ACM SIGIR (1995) 68–73
17. Luhn, H.P.: The Automatic Creation of Literature Abstracts. IBM Journal of Research and Development, 2(2) (1958) 159–165
18. Mani, I., Maybury, M.T.: Advances in Automatic Text Summarization, Cambridge, MA: MIT Press (1999) ix–xv
19. Neto, J.L., Freitas, A.A., Kaestner, C.A.A.: Automatic Text Summarization Using a Machine Learning Approach. Proc. SBIA (2002) 205–215
20. Quinlan, J. R.: C4.5: Programs For Machine Learning. Morgan Kaufmann, Los Altos (1993)
21. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach, Upper Saddle River, NJ: Prentice-Hall (1995)
22. Turney, P.: Learning Algorithms for Keyphrase Extraction. Information Retrieval Journal 2(4) (2000) 303–336
23. Turney, P.: Coherent Keyphrase Extraction via Web Mining. Proceedings of IJCAI'03 (2003) 434–439.
24. Wu, J., Agogino, A. M.: Automating Keyphrase Building with Multi-Objective Genetic Algorithms, Proceedings of the 37th Annual Hawaii International Conference on System Science, HICSS (2003) 104–111
25. Weka: http://www.cs.waikato.ac.nz/~ml/weka (2004)
26. Yang, Y., Webb, G. I.: Weighted Proportional k-Interval Discretization for Naïve-Bayes Classifiers. Proceedings of the 7th Pacific-Asia Conference on Knowledge Discovery and Data Mining, (2003) 501–512
27. Zhang, Y., Milios, E., Zincir-Heywood, N.: A Comparison of Keyword- and Keyterm-based Methods for Automatic Web Site Summarization, in Technical Report WS-04-01, Papers from the on Adaptive Text Extraction and Mining, San Jose, CA, (2004) 15–20

Automatic Annotation of Corpora for Text Summarisation: A Comparative Study

Constantin Orăsan

Research Group in Computational Linguistics,
School of Humanities, Languages and Social Sciences,
University of Wolverhampton, Stafford St.,
Wolverhampton, WV1 1SB, UK
C.Orasan@wlv.ac.uk
http://www.wlv.ac.uk/~in6093/

Abstract. This paper presents two methods which automatically produce annotated corpora for text summarisation on the basis of human produced abstracts. Both methods identify a set of sentences from the document which conveys the information in the human produced abstract best. The first method relies on a greedy algorithm, whilst the second one uses a genetic algorithm. The methods allow to specify the number of sentences to be annotated, which constitutes an advantage over the existing methods. Comparison between the two approaches investigated here revealed that the genetic algorithm is appropriate in cases where the number of sentences to be annotated is less than the number of sentences in an ideal gold standard with no length restrictions, whereas the greedy algorithm should be used in other cases.

1 Introduction

Annotated corpora are essential for most branches in computational linguistics, including automatic summarisation. Within computational linguistics, annotated corpora are normally considered a gold standard, and are used to train machine learning algorithms and evaluate the performance of automatic summarisation methods. In order to be used for these purposes, the annotation usually indicates the importance of each sentence. In this paper, the term gold standard is used only to refer to sets of sentences marked as important, and not to the whole annotated document. This approach was taken in order to facilitate the explanation to follow, and it does not prevent the methods presented in this paper being used to produce gold standards where the important sentences are annotated within the document.

The decision as to whether a sentence is important enough to be annotated can be taken either by humans or by programs. When humans are employed in the process, producing such corpora becomes time consuming and expensive. Methods which automatically build annotated corpora are cheap, but have some drawbacks. Section 2 presents brief details about the existing methods for producing such corpora for text summarisation.

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 670–681, 2005.

When corpora are used to evaluate automatic summarisation methods, the sentences extracted by an automatic method are compared with the ones marked by humans as important. In order to be able to have this comparison, it is usually necessary that the size of the automatic summary is the same as the size of the gold standard. Given that automatic summarisation systems can produce summaries which can have any compression rate, in order to evaluate them, gold standards should exist for all possible lengths. Because of the high costs of having people involved in the annotation process, it is not feasible to have the same text annotated for more than two or three different lengths. Automatic annotation methods can be applied in certain conditions, but the existing ones do not offer much control on the length of the gold standard.

This paper compares two automatic methods for producing annotated corpora for text summarisation. The advantage of these two methods is that they allow the automatic production of gold standards of predefined lengths. The structure of the paper is as follows: Section 2 briefly presents existing ways to produce annotated corpora for text summarisation. A greedy method inspired by [1] is explained in Section 3. Because the second method relies on a genetic algorithm, brief background information about genetic algorithms is presented in Section 4, followed by a description of the actual algorithm. Section 6 contains a comparative evaluation, and the paper finishes with discussion and conclusions.

2 Existing Methods for Building Annotated Corpora

Annotated corpora have been employed in automatic summarisation since the late 60s when Edmundson used one in the evaluation process. In order to produce the annotated corpus, Edmundson asked humans to identify the important sentences in each text from a collection of 200 scientific documents [2]. The important sentences were considered to be those which indicated *what* the subject area is, *why* the research is necessary, *how* the problem is solved, and *which* are the findings of the research. The annotators were asked to select 25% of the text, and to mark sentences in such a way that the selected sentences are coherent.

More recently, manual annotation was used to mark the important sentences in a corpus of newswire texts [3]. In this case, the annotators were required identify the main topic of a text, and to firstly mark 15% of the most important sentences, and then mark an additional 15% of the text. In order to maximise the coherence of the selected sentences, the annotators also marked sentences which are necessary for understanding the important sentences (e.g. sentences which introduce entities mentioned in the important sentences).

Given that identification of important sentences is very subjective and difficult, [4] and [5] took advantage of human produced abstracts, and asked annotators to align sentences from the document with sentences from the human produced abstracts. This set of sentences from the document is considered to convey the information from the abstract best, and therefore can be used as a gold standard. Kupiec et. al. [4] found that 79% of the sentences in the abstracts

could be perfectly matched with sentences from the full text, whereas Teufel and Moens [5] have observed that only 31.7% of the sentences from the abstracts have a perfect match. The percent of matching clauses is even lower in the experiment presented by Marcu [1]. One reason for these very dissimilar results could be the fact that the researchers worked with various types of documents, and did not use a common definition of a perfect match.

Annotated corpora can be automatically produced by using the observation that very often humans produce abstracts through *cut-and-paste* operations. As a result, it is possible to identify sentences from the document which are the source of the abstract. Marcu [1] combines a greedy algorithm with rules to recover this set of sentences. Marcu's method was reimplemented in this research and is explained in more detail in Section 3.

Jing and McKeown [6] treat the human produced abstract as a sequence of words which appears in the document, and reformulate the problem of alignment as a problem of finding the most likely position of the words from the abstract in the full document using a Hidden Markov Model.

3 A Greedy Method for Automatic Annotation of Important Sentences in Scientific Texts

Even though, it is difficult to identify direct matches between pairs of sentences from a document and its human produced abstract, it is generally agreed that it is possible to find several sentences from the document which were combined to produce a sentence in the abstract. Brute force approaches which try all the possible combinations are out of the question given that for a text with n sentences the number of possible extracts is $C_n^1 + C_n^2 + \ldots + C_n^n = 2^n - 1$.

In order to solve this problem, Marcu [1] proposes a greedy method. His method, instead of selecting sentences which are considered to be similar to those in the abstract, eliminates sentences which do not seem like the ones in the abstract. The underlining idea is that a sentence from a document does not resemble any sentence or part of sentence from the abstract if the similarity between the document and its abstract does not decrease when the sentence is removed from the document. This elimination process continues as long as such irrelevant sentences can be identified. When no more sentences can be eliminated, Marcu proposes a set of heuristics to further reduce the set of sentences left. After the rules are applied, the remaining sentences constitute the most similar extract to the human abstract, and can be used as a gold standard.

Algorithm 1 presents a modified version of Marcu's algorithm which was used here. The similarity between a set of sentences and the human produced summary is computed using the cosine similarity formula:

$$\cos(\boldsymbol{S_e}, \boldsymbol{S_h}) = \frac{\sum_{i=1}^{n} S_e(i) S_h(i)}{\sqrt{\sum_{i=1}^{n} S_e(i)} \sqrt{\sum_{i=1}^{n} S_h(i)}} \tag{1}$$

where S_e and S_h are the vectors built from the automatic extract and human abstract respectively, n is the number of distinct words in $S_e \cup S_h$, and $S_e(i)$ and

Data: Abstract = the human produced abstract, Source = $\{S_1, S_2, \ldots, S_n\}$, ExtractLen = the desired length or 0 if no limit is imposed
Result: Extract = a set of sentences from the source which has maximum similarity

```
1  Extract = Source;
2  Eliminate from Extract all the sentences with equations and references;
3  Eliminate from Extract all the sentences with less than 5 words;
4  while (Length(Extract) > ExtractLen) do
5  |  if (∃S ∈ Extract, Sim(Extract, Abstract) < Sim(Extract \ S, Abstract))
   |  then
6  |  |  Extract = Extract \ S;
7  |  end
8  |  else
9  |  |  break;
10 |  end
11 end
12 while (Length(Extract) > ExtractLen) do
13 |  Extract = Extract \ S, where Sim(Extract \ S, Abstract) > Sim(Extract \
   |  T), ∀T ∈ Extract, T ≠ S;
14 end
```

Algorithm 1. The elimination algorithm

$S_h(i)$ are the frequencies of word i in S_e and S_h respectively. In order to make the similarity value more accurate, stopwords were filtered, but no morphological transformation was used because it was desired that the gold standard uses very similar wording to the human produced abstract.

One difference between the original algorithm and the one used here is the fact that Marcu's heuristics employed to further reduce the set of sentences were not implemented. There are two reasons for not implementing them. Firstly, some of the heuristics require knowledge of the rhetorical structure of the source to be able to apply them. This information was not available, and could not be easily obtained. In addition, for some of the heuristics, the details were insufficient to know exactly how to implement them. Instead of the original heuristics, steps *2* and *3* were introduced to remove sentences with equations and references, and those which are very short.[1]

Another change which had to be made to the algorithm was to introduce a way to control the length of the gold standard. In the algorithm proposed by Marcu, there is no way to predict or restrict this length, and as a result, the method cannot be directly used to find a gold standard of a certain length.

In order to alleviate this problem, steps *12 – 14* were introduced in the algorithm. The role of these steps is to shorten the extract until the desired length is reached. This is achieved by identifying sentences whose elimination

[1] After several experiments, it was decided that a sentence which has less than 5 words, excluding punctuation, is not worth including in the extract.

Data: Fitness = a function which evaluates the quality of a chromosome,
PopSize = the size of the population
Result: Solution
1 P = The initial population with random chromosomes;
2 Evaluate the quality of the chromosomes in P;
3 **while** *(Termination condition not met)* **do**
4 Select chromosomes for recombination and put them in P_1;
5 Combine randomly selected chromosomes from P_1 and put the result in P_2;
6 Mutate randomly selected chromosomes from P_2 and put the result in P_3;
7 Produce the new population P from P_3 and the chromosomes not selected from P;
8 Evaluate the quality of the chromosomes in P;
9 **end**
10 Return Solution = the best chromosome selected according to the chosen strategy;

Algorithm 2. A typical genetic algorithm

cause the smallest decrease in the similarity between the selected sentences and the human produced abstract. In a number of situations, it happens that the length limit is reached in the main part of the algorithm (Steps *4 – 11*), and the algorithm returns the set of sentences which has been identified so far. Evaluation of the algorithm is presented in Section 6.

4 Brief Introduction to Genetic Algorithms

Genetic algorithms (GA) provide "a learning method motivated by an analogy to biological evolution" [7]. Introduced by Holland [8], genetic algorithms proved very effective in search and optimisation problems where the search space is complex. The way GAs work mimics reproduction and selection of natural populations to find the solution that maximises a function called *fitness function.*

There is no rigorous definition of a genetic algorithm but it is generally accepted that the common elements of GAs are: a population of chromosomes, a selection method based on a fitness function and genetic operators which evolve the population. A typical genetic algorithm is presented in Algorithm 2.

Chromosomes and Population: A *chromosome* encodes a possible solution to the problem to be solved. Each chromosome contains a fixed number of *genes*, which have binary, integer or real values, depending on the problem to be solved.[2] The length of a chromosome (i.e. the number of genes) and its meaning are also dependent on the problem. The population of chromosomes maintained by the

[2] There are algorithms where a chromosome contains a mixture of binary, integer or real values, or where the length of the chromosomes is variable. Because such algorithm are rarely used, and because this section is not supposed to offer a comprehensive overview of the existing GAs, they are not discussed here.

algorithm encodes different candidate solutions, and the number of chromosomes in the population varies according to the complexity of the search space.

Selection Method: The appropriateness of a chromosome is measured by a *fitness function* which is problem dependent. In addition to measuring the quality of a chromosome, the fitness function is also used to select which chromosomes will create a new population. The most common selection operator is called *fitness proportionate selection* because the probability of selecting a chromosome is proportionate to its fitness. Alternative selection methods are *elitist selection* where only the best chromosomes are used in the next generation, and *random selection* where the chromosomes are chosen on random basis.

Genetic Operators: After a set of chromosomes is selected, *genetic operators* are applied to produce a new population. The most common operators are *crossover* and *mutation*. The role of crossover is to take a pair of chromosomes, and produce two new offsprings by swapping or combining information from them. The mutation operator takes a single chromosome, and randomly changes the value of a randomly chosen gene. It has to be pointed out that each genetic operator has a certain probability to be applied, and as a result, not all the chromosomes are changed from one generation to the next.

Selection of the Solution: The goal of a genetic algorithm is usually to find a chromosome which maximises the fitness function. In order to achieve this, the population of chromosomes is evolved, and depending on the problem which is solved, the best chromosome found in all generations, or the best chromosome from the last generation is chosen as solution to the problem.

The number of iterations also depends on the problem to be solved. In some cases, the algorithm is expected to produce a predetermined number of generations, whereas in other cases, it stops when the fitness function reaches a certain threshold.

Genetic algorithms were successfully used in a wide range of fields including computational linguistics. In computational linguistics they were employed to improve the performance of anaphora resolution methods [9, 10], resolve anaphora resolution [11], study optimal vowel and tonal systems [12], build bilingual dictionaries [13], improve queries for information retrieval [14], and learn of syntactic rules [15]. For applications in other fields than computational linguistics a good overview can be found in [16].

5 A Genetic Algorithm for Automatic Annotation of Important Sentences in Scientific Texts

As shown in Section 6, the greedy algorithm presented in Section 3 preforms poorly when a length limit is imposed due to the iterative process which is employed to eliminate the sentences. Because of this, once a sentence is eliminated, it is impossible to undo the elimination. This section presents a genetic algorithm which determines the gold standard with a specific length.

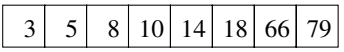


Fig. 1. A chromosome representing an extract which contains the sentences 3, 5, 8, 10, 14, 18, 66, 79 from the source

The genetic algorithm employed here does not work on a sentence by sentence basis. Instead, it encodes the whole set of sentences which might correspond to the gold standard in a chromosome. The length of the chromosome is the desired length of the gold standard, and each gene points to a sentence to be included in it. Figure 1 presents an example of a chromosome. With this encoding, caution needs to be taken whenever a new chromosome is produced so the values of the genes are distinct (i.e. the summary does not contain a sentence more than once). If a duplication is found in a chromosome, then the value of the gene which contains the duplication is incremented by one.

An alternative to this encoding is binary encoding, where the length of a chromosome is the number of sentences in the text. In this encoding, each gene corresponds to a sentence, and a value of 1 for the gene indicates that the sentence is to be included in the gold standard, whereas a 0 value designate a sentence not to be included in the gold standard. This encoding was not used here because the number of sentences in the documents is quite large, which means that the resulting chromosomes would have been too long, and in such cases the convergence of the genetic algorithm is not certain. The second difficulty that needed to be tackled with such encoding is how to produce gold standards of a certain length (i.e. to have an exact number of genes with value 1 in the chromosomes). Such encoding would have required an additional operator which checks the number of genes with value 1, but the decision on which genes should be changed to achieve the desired number of 1s seemed very arbitrary. In light of these problems, it was decided not to use binary encoding.

The fitness function employed by the genetic algorithm is cosine similarity between the set of sentences represented by a chromosome, and the human abstract. Given that the GA is trying to maximise the value of the fitness function, the set of sentences determined by the best chromosome is the most similar to the human abstract, and therefore can be used as a gold standard.

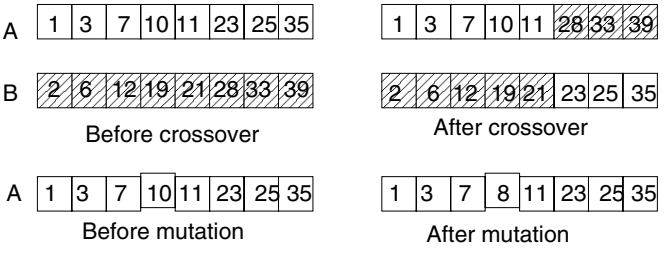


Fig. 2. The genetic operators

The genetic operators used for this particular problem are fitness proportionate selection, single point crossover and point mutation. Figure 2 shows how crossover and mutation operators work. After the crossover and mutation operators are applied, duplicate genes need to be identified. As a result, it is possible to say that two mutation operators are applied to the population.

6 Evaluation

The evaluation was performed using a corpus consisting of 65 files from the Journal of Artificial Intelligence Research (JAIR) with over 600,000 words. These texts were chosen because they contain human produced abstracts, and therefore they can be used by the methods described in this paper. From each document 2%, 3%, 5%, 6%, and 10% gold standards were produced. Using the greedy method, gold standards without a length limit were also generated. Table 1 presents the average similarity scores and their standard deviation obtained for different sets of sentences. The first column corresponds to the unrestricted gold standard, and can be determined only with the greedy algorithm. For the genetic algorithm, two different settings were tried in order to assess how the number of chromosomes influences the algorithm's convergence.

In order to have a better view of the methods' performance a baseline was also implemented. This baseline takes the first and the last sentence of each paragraph, starting with the first one, until the desired length is reached. This baseline was selected because it was noticed that important sentences in scientific articles tend to occur in these positions. A random baseline was considered too naive and easy to defeat to be employed here. Table 1 reveals that both methods perform better than the baseline.

Table 1. The average similarities between the *ideal* extract and the human produced abstracts

	Unrestricted	2%	3%	5%	6%	10%
Baseline						
Average similarity	-	0.260	0.327	0.419	0.440	0.479
Standard deviation	-	0.137	0.159	0.163	0.153	0.131
Greedy algorithm						
Similarity average	0.818	0.392	0.521	0.687	0.732	0.788
Similarity standard deviation	0.082	0.196	0.193	0.167	0.141	0.079
Genetic algorithm with a population of 500 chromosomes						
Average	-	0.720	0.734	0.738	0.738	0.733
Standard deviation	-	0.104	0.094	0.087	0.085	0.087
Genetic algorithm with a population of 2000 chromosomes						
Average	-	0.725	0.743	0.752	0.748	0.746
Standard deviation	-	0.103	0.092	0.088	0.084	0.084
The average lengths in sentences of the gold standards						
No. sentences	30.18	9.81	14.84	25.15	30.25	50.84

To assess whether there is a link between the length of the gold standards and the performance of different algorithms, the average lengths of the gold standards were computed. These values are presented in the last row of Table 1.

Investigation of the results obtained by the greedy algorithm reveals that the similarity scores obtained for the 2%, 3% and 5% gold standards are very poor in comparison with the rest of the results. As can be seen in the last row of Table 1, these gold standards are shorter than the ones with unrestricted lengths. For this reason, Steps *12 – 14* of Algorithm 1 are needed to reduce their length. As a result, a significant proportion of information is lost which suggests that Steps *12 – 14* are not the right way to control the length of the gold standard.

The solution of the genetic algorithm was chosen to be the best chromosome after 200 generations. Such a large number of generations was necessary because the search space is very large. Experiments with algorithms which iterated for 300 generation revealed that the improvement which can be obtained is negligible. Given the large search space, the number of chromosomes in the population was also large. In order to find out how this number influences the convergence of the algorithm, two experiments were run. In the first one, the population contained 500 chromosomes, and in the second one this number was increased to 2000.

As can be seen in Table 1, the results obtained with the genetic algorithm are much more homogenous than the ones of the greedy algorithm. As expected, the results for the genetic algorithm with 500 chromosomes are lower than the ones obtained with 2000. This can be explained by the fact that the latter can explore the search space more efficiently, and therefore can identify better solutions. In addition, it was noticed that the algorithm with 2000 chromosomes converges more rapidly to the solution.

Comparison between the results of the two methods reveals that the genetic algorithm is a much better option for determining the 2%, 3%, 5% and 6% gold standards, and only for 10% extracts it performs worse. The conclusion which can be drawn is that the genetic algorithm is a good way to determine the gold standard when its length is shorter than the length of an unrestricted gold standard, whereas the greedy algorithm is appropriate for gold standards which are longer than the unrestricted gold standard. The low standard deviation obtained for the genetic algorithm suggests that its results are more consistent across different texts. Figure 3 presents the sentences selected as important by the greedy algorithm and by the genetic algorithm. The human produced abstract is also displayed there.

7 Discussion and Conclusions

This paper has presented two automatic methods for building annotated corpora for text summarisation using the human produced abstracts which accompanies documents. The first method uses a greedy approach to eliminate those sentences which do not resemble the information present in the abstract, whilst the second one relies on a genetic algorithm to achieve the same goal. Because the set of sentences determined by the two methods contain similar information to the

human abstract, these sentences can be marked as important in the text, in this way obtaining a gold standard.

Given that the methods proposed in this paper can be applied only where there is a human produced abstract, one may believe that the usefulness of these methods is limited. This is not the case because the gold standards identified by the methods can be used to evaluate automatic extraction methods using precision and recall. Moreover, the gold standard can be used by machine learning algorithms to learn when to extract a sentence. None of these tasks can be done only on the basis of human abstract.

The results of the evaluation revealed that each method is appropriate for a different purpose. The greedy method is suitable in the cases where a gold standard with no length limit imposed has to be determined, and for the gold standards which have the length longer than the unrestricted one. The performance of the greedy algorithm degrades quickly with the decrease in the length of the gold standard, in all the cases where gold standards shorter than the unrestricted one need to be produced, the genetic algorithm should be used. In addition, low standard deviation noticed in the results indicates that the algorithm has a consistent performance across different documents.

As expected, the sets of sentences identified by the methods cannot match those marked by humans. This is normal given the simplicity of the process employed. However, the methods proposed in this paper can be particularly useful for long documents where it is not feasible to ask humans to mark the important sentences. Moreover, for scientific documents, it is necessary to have annotators who understand the topic of the text. In many cases this makes the annotation process more expensive because domain experts have to be employed.

One of the most common criticisms of genetic algorithms is their slow speed, especially in experiments where the population size or the number of generations are large. It is true that the genetic algorithm employed here is quite slow, but even with a population of 2000 chromosomes, it performs faster than the greedy algorithm. The explanation for this can be found in the iterative behaviour of the greedy algorithm.

An alternative use for the methods presented in this paper is to determine the upper limits of extraction methods when run on a document, provided that the evaluation method uses similarity to assess the quality of an extract. Because both methods try to maximise the similarity score between a set of sentences extracted from the text, and the human produced abstract, the maximum for the similarity score indicates the upper limit of any extraction method. It should be pointed out that given the nature of the two methods, none of them actually guarantees that the absolute maximum is reached. For both of them, it is possible that there is another set of sentences which obtains a higher similarity score, but the determined value will be very close to the absolute maximum.

The genetic algorithm presented in this paper can be used to determine not only one, but all the sets of sentences which have a maximum the similarity with the human abstract. This can be useful because, the same information can be marked in several places in the document which could pose problems

to evaluation methods. In addition, these alternative sets of sentences can be beneficial for machine learning algorithms.

Acknowledgments. The research in this paper is supported by the Arts and Humanities Research Board through the CAST project.

References

1. Marcu, D.: The automatic construction of large-scale corpora for summarization research. In: The 22nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'99), Berkeley, CA (1999) 137–144
2. Edmundson, H.P.: New methods in automatic extracting. Journal of the Association for Computing Machinery **16** (1969) 264 – 285
3. Hasler, L., Orăsan, C., Mitkov, R.: Building better corpora for summarisation. In: Proceedings of Corpus Linguistics 2003, Lancaster, UK (2003) 309 – 319
4. Kupiec, J., Pederson, J., Chen, F.: A trainable document summarizer. In: Proceedings of the 18th ACM/SIGIR Annual Conference on Research and Development in Information Retrieval, Seattle (1995) 68 – 73
5. Teufel, S., Moens, M.: Sentence extraction as a classification task. In: Proceedings of the ACL'97/EACL'97 Workshop on Intelligent Scallable Text Summarization, Madrid, Spain (1997) 58 – 59
6. Jing, H., McKeown, K.R.: The decomposition of human-written summary sentences. In: Proceedings of the 22nd International Conference on Research and Development in Information Retrieval (SIGIR'99), University of Berkeley, CA (1999) 129 – 136
7. Mitchell, T.M.: Machine learning. McGraw-Hill Series in Computer Science. McGraw-Hill (1997)
8. Holland, J.H.: Adaptation in Natural and Artificial Systems. University of Michigan Press (1975)
9. Byron, D.K., Allen, J.F.: Applying genetic algorithms to pronoun resolution. In: Proceedings of the Sixteenth National Conference on Artificial Intelligence, Orlando, Florida, USA, AAAI Press / The MIT Press (1999) 957
10. Orăsan, C., Evans, R., Mitkov, R.: Enhancing preference-based anaphora resolution with genetic algorithms. In: Proceedings of Natural Language Processing - NLP2000, Springer (2000) 185 – 195
11. Barbu, C.: Genetic algorithms in anaphora resolution. In Antonio Branco, T.M., Mitkov, R., eds.: Proceedings of the 4th Discourse Anaphora and Anaphor Resolution Colloquium (DAARC2002), Lisbon, Portugal (2002) 7 – 12
12. Ke, J., Ogura, M., Wang, W.: Modeling evolution of sound systems with genetic algorithm. Computational Linguistics **29** (2003) 1–18
13. Han, B.: Building a bilingual dictionary with scarce resources: A genetic algorithm approach. In: Proceedings of the Student Research Workshop, the Second Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL-2001), Pittsburgh, USA (2001) 14 – 19
14. Yang, J.J.: Use of Genetic Algorithms for Query Improvement in Information Retrieval Based on a Vector Space Model. PhD thesis, University of Pittsburgh, Pittsburgh, PA (1993)
15. Loose, R.M.: Learning syntactic rules and tags with genetic algorithms for information retrieval and filtering: An empirical basis for grammatical rules. Information Processing & Management **32** (1996) 185 – 197

16. Mitchell, M.: An Introduction to Genetic Algorithms. Complex Adaptive Systems. The MIT Press (1996)

Greedy algorithm. Similarity score 0.7064	
S16	Inductive Logic Programming (ILP) is a subfield of Logic Programming and Machine Learning that tries to induce clausal theories from given sets of positive and negative examples.
S24	The two main operations in ILP for modification of a theory are generalization and specialization.
S26	These operations only make sense within a generality order.
S47	Here the concept of a least generalization is important.
S76	In this paper, we give a systematic treatment of the existence and non-existence of least generalizations and greatest specializations, applied to each of these three generality orders.
S88	We survey results obtained by others and also contribute some answers of our own.
S89	For the sake of clarity, we will summarize the results of our survey right at the outset.
S112	Thirdly, we contribute a complete discussion of existence and non-existence of greatest specializations in each of the six ordered languages.
S239	These two different languages and three different quasi-orders give a total of six combinations.
S653	The two main languages are clausal languages and Horn languages.
S654	This gives a total of six different ordered sets .
S655	In this paper, we have given a systematic treatment of the existence or non-existence of least generalizations and greatest specializations in each of these six ordered sets.

Genetic algorithm. Similarity score 0.806	
S28	The three most important generality orders used in ILP are subsumption (also called subsumption), logical implication and implication relative to background knowledge.
S35	Within a generality order, there are two approaches to generalization or specialization.
S86	The combination of three generality orders and two different possible languages of clauses gives a total of six different ordered languages.
S88	We survey results obtained by others and also contribute some answers of our own.
S104	Yet least generalizations relative to function-free background knowledge do not always exist, as we will show in Section 7.
S112	Thirdly, we contribute a complete discussion of existence and non-existence of greatest specializations in each of the six ordered languages.
S113	In particular, we show that any finite set of clauses has a greatest specialization under implication.
S407	It follows from Lemma 1 that G (and hence also G0) cannot contain terms of depth greater than d, nor predicates, functions or constants other than those in S.
S454	If S is a finite set of clauses from C and S contains at least one non-tautologous function-free clause, then there exists a special LGI of S in C.
S653	The two main languages are clausal languages and Horn languages.
S655	In this paper, we have given a systematic treatment of the existence or non-existence of least generalizations and greatest specializations in each of these six ordered sets.
S657	The only remaining open question is the existence or non-existence of a least generalization under implication in C for sets of clauses which all contain function symbols.

Human produced abstract
The main operations in Inductive Logic Programming (ILP) are generalization and specialization, which only make sense in a generality order. In ILP, the three most important generality orders are subsumption, implication and implication relative to background knowledge. The two languages used most often are languages of clauses and languages of only Horn clauses. This gives a total of six different ordered languages. In this paper, we give a systematic treatment of the existence or non-existence of least generalizations and greatest specializations of finite sets of clauses in each of these six ordered sets. We survey results already obtained by others and also contribute some answers of our own. Our main new results are, firstly, the existence of a computable least generalization under implication of every finite set of clauses containing at least one non-tautologous function-free clause (among other, not necessarily function-free clauses). Secondly, we show that such a least generalization need not exist under relative implication, not even if both the set that is to be generalized and the background knowledge are function-free. Thirdly, we give a complete discussion of existence and non-existence of greatest specializations in each of the six ordered languages.

Fig. 3. 2% ideal extracts produced by the greedy algorithm and the genetic algorithm, and the human produced abstract for the text

Techniques for Improving the Performance of Naive Bayes for Text Classification

Karl-Michael Schneider

University of Passau, Department of General Linguistics,
Innstr. 40, 94032 Passau, Germany
schneide@phil.uni-passau.de
http://www.phil.uni-passau.de/linguistik/schneider/

Abstract. Naive Bayes is often used in text classification applications and experiments because of its simplicity and effectiveness. However, its performance is often degraded because it does not model text well, and by inappropriate feature selection and the lack of reliable confidence scores. We address these problems and show that they can be solved by some simple corrections. We demonstrate that our simple modifications are able to improve the performance of Naive Bayes for text classification significantly.

1 Introduction

Text classification is the assignment of predefined categories to text documents. Text classification has many applications in natural language processing tasks such as E-mail filtering [1, 2], news filtering [3], prediction of user preferences [4] and organization of documents [5]. Because of the variety of languages, applications and domains, machine learning techniques are commonly applied to infer a classification model from example documents with known class labels. The inferred model can then be used to classify new documents. A variety of machine learning paradigms have been applied to text classification, including rule induction [6], Naive Bayes [7], memory based learning [8], decision tree induction [9] and support vector machines [10].

This paper is concerned with the Naive Bayes classifier. Naive Bayes uses a simple probabilistic model that allows to infer the most likely class of an unknown document using Bayes' rule. Because of its simplicity, Naive Bayes is widely used for text classification [4, 5, 1, 2, 11].

The Naive Bayes model makes strong assumptions about the data: it assumes that words in a document are independent. This assumption is clearly violated in natural language text: there are various types of dependences between words induced by the syntactic, semantic, pragmatic and conversational structure of a text. Also, the particular form of the probabilistic model makes assumptions about the distribution of words in documents that are violated in practice [12]. Nonetheless, Naive Bayes performs quite well in practice, often comparable to more sophisticated learning methods [13, 14].

One could suspect that the performance of Naive Bayes can be further improved if the data and the classifier better fit together. There are two possible approaches: (i) modify the data, (ii) modify the classifier (or the probabilistic model).

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 682–693, 2005.

Many researchers have proposed modifications to the way documents are represented, to better fit the assumptions made by Naive Bayes. This includes extracting more complex features, such as syntactic or statistical phrases [15], and exploiting semantic relations using lexical resources [16]. These attempts have been largely unsuccessful. Another way to improve the document representation is to extract features by word clustering [17] or by transforming the feature space [18]. These methods did show some improvement of classification accuracy.

Instead of changing the document representation by using other features than words, it is also possible to manipulate the text directly, e.g. by altering the occurrence frequencies of words in documents [19]. This can help the data to better fit the distribution assumed by the model.

The most important way to better fit the classifier to the data is to choose an appropriate probabilistic model (see Sect. 2). Some researchers have also tried to improve performance by altering the way the model parameters are estimated from training data [20].

In this paper we review and explain a number of very simple techniques that can help to improve the accuracy of a Naive Bayesian text classifier dramatically. Some of them have been proposed before or are simplifications of existing methods. Many of these techniques appear to be counterintuitive but can be explained by the particular (statistical) properties of natural language text documents.

2 Naive Bayes

Bayesian text classification uses a parametric mixture model to model the generation of documents [7]. The model has the following form:

$$p(d) = \sum_{j=1}^{|C|} p(c_j)p(d|c_j)$$

where c_j are the mixture components (that correspond to the possible classes) and $p(c_j)$ are prior probabilities. Using Bayes' rule, the model can be inverted to get the posterior probability that d was generated by the mixture component c_j:

$$p(c_j|d) = \frac{p(c_j)p(d|c_j)}{p(d)}$$

To classify a document, the classifier selects the class with maximum posterior probability, given the document, where $p(d)$ is constant and can be ignored:

$$c^*(d) = \operatorname*{argmax}_j p(c_j)p(d|c_j) \tag{1}$$

The prior probabilities $p(c_j)$ are estimated from a training corpus by counting the number of training documents in each class c_j.

The distribution of documents in each class, $p(d|c_j)$, cannot be estimated directly. Rather, it is assumed that documents are composed from smaller units, usually words or

word stems. To make the estimation of parameters tractable, we make the Naive Bayes assumption: that the basic units are distributed independently.

There are several Naive Bayes models that make different assumptions about how documents are composed from the basic units. The most common models are: the binary independence model (a.k.a. multi-variate Bernoulli model), the Poisson Naive Bayes model, and the multinomial model (the latter is equivalent to the Poisson model under the assumption that the class of a document is marginally independent of its length) [7, 21]. The most apparent difference between these models is that the Poisson model and the multinomial model use word occurrence frequencies, while the binary independence model uses binary word occurrences. In this paper we consider the multinomial Naive Bayes model because it is generally superior to the binary independence model for text classification [7, 21].

In the multinomial model, a document d is modeled as the outcome of $|d|$ independent trials on a single random variable W that takes on values $w_t \in V$ with probabilities $p(w_t|c_j)$ and $\sum_{t=1}^{|V|} p(w_t|c_j) = 1$. Each trial with outcome w_t yields an independent occurrence of w_t in d. Thus a document is represented as a vector of word counts $d = \langle x_t \rangle_{t=1\ldots|V|}$ where each x_t is the number of trials with outcome w_t, i.e. the number of times w_t occurs in d. The probability of d is given by the multinomial distribution:

$$p(d|c_j) = p(|d|)|d|! \prod_{t=1}^{|V|} \frac{p(w_t|c_j)^{x_t}}{x_t!}$$

Here we assume that the length of a document is chosen according to some length distribution, independently of the class. Plugging this into (1) we get the following form (omitting parts that do not depend on the class):

$$c^*(d) = \operatorname*{argmax}_{c_j} p(c_j) \prod_{t=1}^{|V|} p(w_t|c_j)^{x_t} \tag{2}$$

The parameters $p(w_t|c_j)$ are estimated by counting the occurrences of w_t in all training documents in c_j, using a Laplacean prior:

$$p(w_t|c_j) = \frac{1 + n(c_j, w_t)}{|V| + n(c_j)}$$

where $n(c_j, w_t)$ is the number of occurrences of w_t in the training documents in c_j and $n(c_j)$ is the total number of word occurrences in c_j.

3 Word Frequency Information

It is usually claimed that the multinomial model gives higher classification accuracy than the binary independence model on text documents because it models word occurrence frequencies [7, 21]. Contrary to this belief, we show that word frequency hurts more than it helps, and that ignoring word frequency information can improve performance dramatically.

The multinomial Naive Bayes model treats each occurrence of a word in a document independently of any other occurrence of the same word. In reality, however, multiple occurrences of the same word in a document are not independent. When a word occurs once, it is likely to occur again, i.e. the probability of the second occurrence is much higher than that of the first occurrence. This is called *burstiness* [12]. The multinomial model does not account for this phenomenon. This results in a large underestimation of the probability of documents with multiple occurrences of the same word.

In [19] a transformation of the form $x'_t = \log(1 + x_t)$ was applied to the word frequency counts in a document in order to better fit the data to the probabilistic model. This does not eliminate word frequencies but has the effect of pushing down larger counts. An even simpler, yet less accurate method is to remove word frequency information altogether using the transform $x'_t = \min\{x_t, 1\}$. This can be thought of as discarding all additional occurrences of words in a document. Instead of transforming the word counts, we can change the classification rule as in (3):

$$c^*(d) = \operatorname*{argmax}_{c_j} p(c_j) \prod_{t=1}^{|V|} p(w_t|c_j)^{\min\{x_t,1\}} \tag{3}$$

and the parameter estimation as in (4), where $d(c_j, w_t)$ is the number of documents containing w_t in c_j:

$$p(w_t|c_j) = \frac{1 + d(c_j, w_t)}{|V| + \sum_{s=1}^{|V|} d(c_j, w_s)} \tag{4}$$

We compare classification accuracy with and without word frequency information on two datasets: 20 Newsgroups[1] [3] and WebKB[2] [11]. We remove all headers from the newsgroup articles. We use only the four most populous classes *course*, *faculty*, *project* and *student* of the WebKB corpus and remove all HTML markup. All non-alphanumeric characters are removed, and letters are converted to lower case. In addition, a 100 word stoplist is applied to the newsgroups articles and all numbers are mapped to a special token. Following [7] we perform feature selection using Mutual Information and vary the number of selected features between 20 and 20,000.

Figure 1 shows the results. On 20 Newsgroups, removing word frequency information improves classification accuracy on average by 7.5 percentage points (23% relative improvement). On WebKB, accuracy is improved on average by 3.8 percentage points (an average error reduction of 21%).

4 Class Prior Probabilities

In (2) one can see that for longer documents the classification scores are dominated by the word probabilities, and the prior probabilities hardly affect the classification. However, in situations where documents are usually very short *and* the class distribution is skewed, the prior probabilities may affect the classification negatively. In such cases,

[1] http://people.csail.mit.edu/people/jrennie/20Newsgroups/

[2] http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/

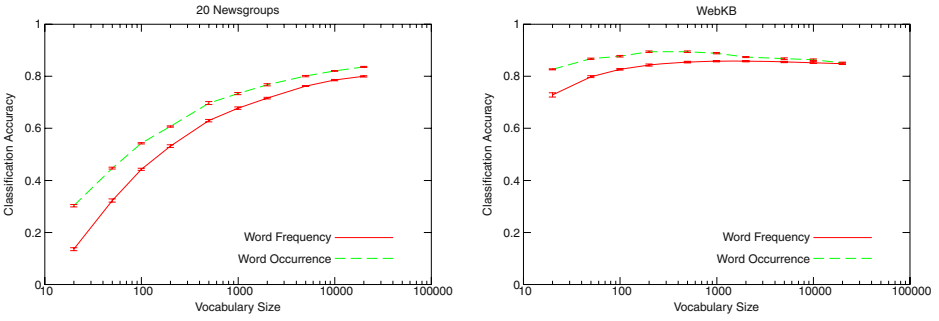


Fig. 1. Classification accuracy of multinomial Naive Bayes with and without word frequency information, on 20 Newsgroups (left) and WebKB (right). Results are averaged over five, respectively ten, cross-validation trials, with small error bars shown. The confidence level using a two-tailed paired t-test is above 99.99% for 20 Newsgroups and above 99% for up to 5,000 words on WebKB

Table 1. Comparison of classification accuracy using prior probabilities estimated from training data and uniform prior probabilities. Words are selected according to their mutual information with the class variable. Without feature selection, the average vocabulary size is 2,742 words. The confidence level using a two-tailed paired t-test is above 99.98%

Prior Probabilities	500 Words	1,000 Words	2,000 Words	Full Vocabulary
Estimated	0.7120	0.6897	0.6635	0.6544
Uniform	0.7243	0.7278	0.7146	0.7095

ignoring prior probabilities altogether (or equivalently, assuming uniform priors) can improve classification accuracy.

To examine the impact of prior probabilities on the classification of short documents, we use a corpus of 4,353 hypertext links from the Linguistics Links Database.[3] There are 16 categories with 60 documents in the smallest category and 1,583 documents in the largest category. We use only the link texts. We convert all letters to lower case, remove tokens on a stop list and replace location names, dates and numerical expressions by special tokens. The average document length after preprocessing is 9 tokens. 505 documents have less than 4 tokens, and 47 documents have only one token.

We produce 20 train/test splits using stratified random sampling with 70% of the data (3,048 documents) for training and 30% (1,305 documents) for testing. Experiments are done using 20 trials over these splits and averaging the results. Table 1 compares classification accuracy with prior probabilities estimated from the training corpus (as described in Sect. 2) and uniform priors, with various vocabulary sizes. Using uniform priors improves performance by more than 5 percentage points when all or almost all words are used. Also, with uniform priors classification accuracy depends less on the number of selected words, whereas with estimated prior probabilities there is a drop in performance when more words are used.

[3] http://www.phil.uni-passau.de/linguistik/linguistik_urls/

5 Feature Selection

Feature selection is commonly regarded as a necessary step in text classification, due to the high dimensionality of text (i.e. the large vocabulary size). Feature selection increases the efficiency of the classifier because less parameters need to be estimated and less probabilities need to be multiplied. Often, it also improves the accuracy of the classifier provided that the size of the feature subset is correctly chosen.

Feature selection for text classification uses a greedy filtering approach: A scoring function is used to assign a score to each feature independently, and the highest scored features are selected. The feature set size can be specified directly or by applying a threshold to the feature scores. The main question is how to determine the optimal number of selected features (or the optimal threshold). A common strategy is by testing the classifier with different values on a validation set.

Most feature scoring functions compute some statistics of the training data or some information theoretic measure. The following functions are very common in text categorization: Chi-square [22] uses the χ^2 statistic to estimate the dependence between a feature and the class of a document. Mutual Information (MI) [7] is an information theoretic measure that measures the amount of information a feature gives about the class. Bi-Normal Separation (BNS) [23] assumes that the occurrence of a word is modeled by the event of a random normal variable exceeding a certain threshold, and measures the separation between the thresholds for a class and its complement (i.e. the union of the other classes).

The above scoring functions are not directly related to the probabilistic framework of Naive Bayes. In the following we present a novel feature scoring function called CRQ (Cluster Representation Quality) that is derived directly from the probabilistic Naive Bayes model, rather than from some independent statistics. CRQ is based on ideas from distributional clustering [17]. Distributional clustering aims at finding a clustering of a set of elements that minimizes the distance (in some information theoretic sense) between the elements in the same cluster (tight clusters) and simultaneously maximizes the distance between the elements in different clusters (distant clusters).

In our approach to feature selection, the elements are the training documents and the clusters are the classes in the training set. The important difference is that we do not change the clustering of the training documents (i.e. their classification) but rather seek to improve the clustering quality by removing certain words from the vocabulary.

5.1 Cluster Representation Quality

First, note that a document d can be regarded as a probability distribution over words with $p(w_t|d) = n(w_t,d)/|d|$, where $n(w_t,d)$ is the number of times w_t occurs in d and $|d|$ denotes the length of d. To measure the distance of one probability distribution, p_1, from another distribution, p_2, we use Kullback-Leibler (KL) divergence [24], defined by $D(p_1\|p_2) = \sum_x p_1(x) \log \frac{p_1(x)}{p_2(x)}$. In [17] the distance between a finite set of probability distributions is measured using generalized Jensen-Shannon (JS) divergence. It can be shown that JS divergence is the weighted sum of the KL divergences of the distributions from the (weighted) mean distribution.

For any document d_i, let $p_i = p(W|d_i)$ denote the probability distribution induced by d_i, as above. The class-conditional distributions $p_j(W) = p(W|c_j)$ can be expressed as the weighted sum of the p_i (for $d_i \in c_j$): $p_j = \sum_{d_i \in c_j} \pi_i p_i$, where the weights $\pi_i = |d_i| / \sum_{d_i \in c_j} |d_i|$ are the normalized document lengths. Instead of the weighted sum of KL divergences, we measure the *within-cluster divergence* using the unweighted mean of the distributions p_i (for $d_i \in c_j$) from p_j. We do this because the standard definition of classification accuracy (which we would like to optimize) gives each document the same weight, regardless of its length. In contrast, the weighted sum of the KL divergences would give longer documents a higher weight, which corresponds to a misclassification cost that is proportional to the document length.

Similarly, the *total divergence* is defined as the unweighted mean of the distributions p_i (for all d_i in the training corpus) from the mean distribution $p = \sum_j \sum_{d_i \in c_j} \pi'_i p_i$, where $\pi'_i = |d_i| / \sum_j \sum_{d_i \in c_j} |d_i|$. Then the cluster representation quality of the training corpus is defined as the difference between total divergence and within-cluster divergence (i.e. the reduction in divergence from the mean due to clustering the training documents into their classes):

$$\begin{aligned} & \frac{1}{N} \sum_{j=1}^{|C|} \sum_{d_i \in c_j} \Big[D(p_i \| p) - D(p_i \| p_j) \Big] \\ = \quad & \frac{1}{N} \sum_{j=1}^{|C|} \sum_{d_i \in c_j} \sum_{t=1}^{|V|} p_i(w_t) \Big[\log p_j(w_t) - \log p(w_t) \Big] \end{aligned} \tag{5}$$

Note that when the total divergence is large, the documents are scattered over a large space in the document space. If, on the other hand, within-cluster divergence is small (tight clusters), the clusters are (on average) far apart.

5.2 An Information Theoretic Analysis of Naive Bayes

The connection to Naive Bayes is established via an information theoretic interpretation of the Naive Bayes classifier. (2) can be written in the following form by taking logarithms, dividing by the length of d and adding the entropy of $p(W|d)$, $H(p(W|d)) = -\sum_t p(w_t|d) \log p(w_t|d)$:

$$\begin{aligned} c^*(d) &= \operatorname*{argmax}_{c_j} \frac{1}{|d|} \log p(c_j) - \sum_{t=1}^{|V|} p(w_t|d) \log \frac{p(w_t|d)}{p(w_t|c_j)} \\ &= \operatorname*{argmin}_{c_j} D(p(W|d) \| p(W|c_j)) - \frac{1}{|d|} \log p(c_j) \end{aligned} \tag{6}$$

Note that the modifications in (6) do not change the classification of documents. (6) shows that, ignoring prior probabilities, Naive Bayes selects the class with minimum KL-divergence from the document. Therefore, classification accuracy is improved if each document is more similar to its true class than to other classes. This is achieved by maximizing the distance between the classes, i.e. (5).

5.3 Cluster Quality Based Feature Scores

Note that (5) can be written as a sum over words. Our new feature scoring function, called CRQ, is derived from (5) by using the value of the term in the sum that corresponds to the word:

$$CRQ(w_t) = \frac{1}{N} \sum_{j=1}^{|C|} \sum_{d_i \in c_j} p_i(w_t) \Big[\log p_j(w_t) - \log p(w_t) \Big] \tag{7}$$

Note that $CRQ(w_t)$ can be negative. (5) is maximized when all words w_t for which $CRQ(w_t)$ is negative are discarded.[4] Therefore, a natural threshold for CRQ is 0. In contrast, Chi-square, MI and BNS do not have natural thresholds. To find a good threshold one could guess some reasonable values and then use a validation set. However, this would have to be done for every dataset since the best threshold may depend on the data, whereas the (theoretically) optimal threshold 0 for CRQ is data independent.

We compare three different strategies for determining the number of selected words, using the four feature scoring functions:

- use a fixed threshold that depends on the scoring function but not on the dataset. To find reasonable values, we experimented with some thresholds. In addition, for CRQ we used the theoretically optimal threshold 0.
- specify the number of features directly, for each scoring function and each dataset. We tried some reasonable values and used the best one.
- use the full vocabulary (i.e. no feature selection).

In addition to 20 Newsgroups and WebKB, we use two other datasets: Ling-Spam[5] [2] and Reuters-21578.[6] Ling-Spam consists of 83.4% legitimate E-mails and 16.6% spam E-mails. For the Reuters-21578 setup, see Sect. 6.

Table 2 shows the results. We use the full vocabulary and the best number of features as a baseline against which we compare the performance of the thresholding strategy.

Chi-square with threshold 0.1 outperforms the baseline on three out of four datasets, while CRQ with threshold 0 performs better than the baseline on half of the datasets. However, the baseline for CRQ is generally higher than for Chi-square, and CRQ generally gives better performance than Chi-square. Moreover, on Reuters and Ling-Spam, CRQ with threshold 0 selects considerably less words than Chi-square with threshold 0.1. In general, CRQ is more sensitive to the complexity of the datasets (one notable exception is WebKB).

Note that all strategies except using the full vocabulary and CRQ with threshold 0 require experimentation to find good values. Often it is difficult to guess reasonable values. For example, the performance on 20 Newsgroups is best with a very large vocabulary, much larger than what one would guess. Therefore, if feature selection is required, CRQ is a priori a better scoring method because of its theoretically optimal

[4] However, removing words from the vocabulary changes the distribution of the remaining words.

[5] http://www.iit.demokritos.gr/skel/i-config/downloads/

[6] http://www.daviddlewis.com/resources/testcollections/reuters21578/

Table 2. Classifier performance and number of selected words for different feature selection strategies: thresholding on the feature score (with the same threshold for all datasets), individual selection of the vocabulary size for each dataset (best), no feature selection (full). Results printed in bold outperform both the full vocabulary and the best vocabulary size

	20 Newsgroups		WebKB		Ling-Spam		Reuters-21578	
	Words	Acc	Words	Acc	Words	F(spam)	Words	Recall
Chi^2=1	22,809	0.8012	9,671	0.8539	13,504	0.9474	12,839	0.8149
Chi^2=0.1	76,797	**0.8070**	32,712	0.8479	44,498	**0.9503**	18,861	**0.8172**
Chi^2 best	20,000	0.7999	1,000	0.8589	5,000	0.9456	20,000	0.8172
MI=10^{-6}	25,926	0.8033	11,904	0.8523	6,845	0.9445	6,802	0.8150
MI=10^{-7}	88,932	**0.8078**	32,776	0.8479	17,149	0.9455	18,014	0.8172
MI best	20,000	0.7998	1,000	0.8582	5,000	0.9434	1,000	0.8210
BNS=0.1	19,684	0.7994	20,877	0.8517	24,964	0.9516	17,449	0.8176
BNS=0.05	49,274	**0.8076**	32,550	0.8478	45,372	0.9503	20,086	0.8176
BNS best	20,000	0.8001	5,000	0.8660	700	0.9753	2,000	0.8323
CRQ=10^{-6}	38,521	**0.8231**	14,715	0.8549	7,759	0.9478	3,755	0.8294
CRQ=10^{-7}	71,493	**0.8249**	29,794	0.8504	18,300	**0.9507**	6,750	0.8252
CRQ=0	94,616	**0.8213**	32,090	0.8500	23,089	**0.9507**	7,617	0.8247
CRQ best	20,000	0.8164	1,000	0.8708	4,000	0.9429	5,000	0.8301
Full	100,874	0.8063	32,873	0.8480	56,247	0.9503	22,430	0.8161

threshold. Without experimenting to find a good vocabulary size, CRQ provides a simple way to determine the number of features by simply setting the threshold to 0. With this strategy, in most cases the classifier performance is comparable to or better than any of the other methods, and in all cases is higher than using the full vocabulary.

6 Confidence Scores

Sometimes it is desirable to have the classifier produce classification scores that reflect the confidence of the classifier that a document belongs to a class. For example, in binary classification problems where one class (the target class) contains examples that are relevant to some query, a document could be assigned to the target class only if its confidence score exceeds some threshold. In multi-label classification tasks (where each document can belong to zero, one or more classes), a document can be assigned to all classes for which the confidence is above the threshold. Such confidence scores must be independent of document length and complexity.

The posterior probabilities $p(c_j|d)$ computed by Naive Bayes are inappropriate as confidence scores because they are usually completely wrong and tend to go to zero or one exponentially with document length [25]. This is a consequence of the Naive Bayes independence assumption and the fact that the words in a document are not really independent. Note that the classification decision of Naive Bayes is not affected as long as the ranking of the classes is not changed (in fact it has been argued that the large bias can reduce classification error [14]).

We follow the approach in [11] to get better confidence scores for Naive Bayes. First, we replace the posterior scores with the KL-divergence scores in (6):

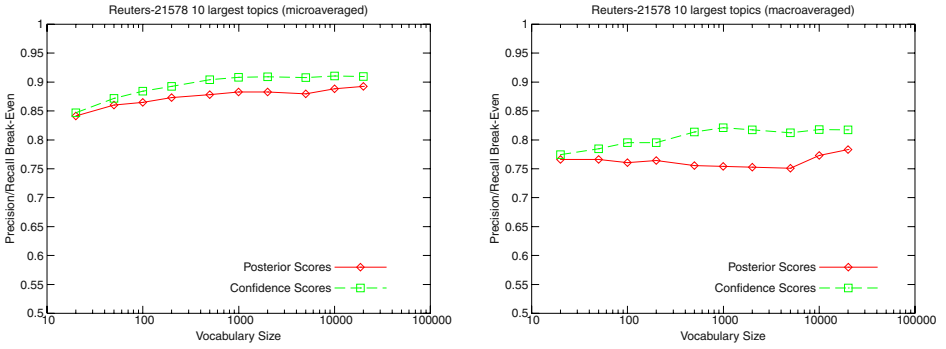


Fig. 2. Comparison of posterior scores and smoothed confidence scores on the Reuters-21578 dataset, using microaveraged (left) and macroaveraged (right) precision/recall break-even

$$score(c_j|d) = \frac{1}{|d|}\log p(c_j) - \sum_{t=1}^{|V|} p(w_t|d)\log\frac{p(w_t|d)}{p(w_t|c_j)}$$

This has two effects. By taking logarithms and dividing by the length of a document, instead of multiplying conditional probabilities (as in Eq. 2) we calculate their geometric mean and thus account for the impact of wrong independence assumptions under varying document lengths. Furthermore, by adding the entropy of (the probability distribution induced by) the document, we account for varying document complexities.

Finally, to make the scores comparable across different documents, we normalize the scores such that they form a probability distribution over classes (i.e. the scores for all classes sum to one):

$$conf(c_j|d) = \frac{score(c_j|d)}{\sum_{k=1}^{|C|} score(c_k|d)}$$

We compare the posterior scores and the confidence scores on the Reuters-21578 dataset, using the ModApte split with only the 10 largest topics [26]. We remove all non-alphabetic characters and convert all letters to lower case. In addition, we map all numbers to a special token. For each topic, we build a binary classifier using all documents in that topic as relevant examples and all other documents as non-relevant examples. The threshold is set for each classifier individually such that recall equals precision (precision/recall break-even point).

Figure 2 shows the results. Microaveraged recall is on average 2 percentage points higher using confidence scores, but macroaveraged recall is improved on average by 4.2 percentage points. This indicates that the confidence scores improve performance especially on the smaller topics (e.g. with 5,000 features the relative improvement is up to 28% on individual topics).

7 Conclusions

This paper has described some simple modifications of the Naive Bayes text classifier that address problems resulting from wrong independence assumptions. Some of the modifications have been proposed before, some are simplifications of existing methods, and some are new. In particular, we have used a simple transformation that effectively removes duplicate words in a document to account for burstiness phenomena in text; we have proposed to use uniform priors to avoid problems with skewed class distributions when the documents are very short; we have used a different but equivalent form of the Naive Bayes classification rule in an information theoretic framework to obtain more reliable confidence scores; and by viewing a training corpus as a clustering of the training documents and feature selection as a way to improve that clustering, we have obtained a new feature scoring function for Naive Bayes.

The main contribution of this paper is our novel feature scoring function, which is able to distinguish features that improve the clustering of the training documents (and thus are useful for classification) from features that degrade the clustering quality (and thus should be removed). The threshold that separates these two sets of features is data-independent. This is a big advantage over other feature scoring functions because (in theory) optimizing the threshold on a validation set becomes unnecessary. We have, however, noted that removing features may change the scores of the remaining features. Future work will have to examine the implications of that.

The modifications of Naive Bayes proposed in this paper have been applied in isolation. Future work will also consider combinations of them.

References

1. Sahami, M., Dumais, S., Heckerman, D., Horvitz, E.: A Bayesian approach to filtering junk e-mail. In: Learning for Text Categorization: Papers from the AAAI Workshop, Madison Wisconsin, AAAI Press (1998) 55–62 Technical Report WS-98-05.
2. Androutsopoulos, I., Paliouras, G., Karkaletsis, V., Sakkis, G., Spyropoulos, C.D., Stamatopoulos, P.: Learning to filter spam e-mail: A comparison of a Naive Bayesian and a memory-based approach. In Zaragoza, H., Gallinari, P., Rajman, M., eds.: Proc. Workshop on Machine Learning and Textual Information Access, 4th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD 2000), Lyon, France (2000) 1–13
3. Lang, K.: NewsWeeder: Learning to filter netnews. In: Proc. 12th International Conference on Machine Learning (ICML-95), Morgan Kaufmann (1995) 331–339
4. Pazzani, M., Billsus, D.: Learning and revising user profiles: The identification of interesting web sites. Machine Learning **27** (1997) 313–331
5. Koller, D., Sahami, M.: Hierarchically classifying documents using very few words. In: Proc. 14th International Conference on Machine Learning (ICML-97). (1997) 170–178
6. Cohen, W.W., Singer, Y.: Context-sensitive learning methods for text categorization. ACM Transactions on Information Systems **17** (1999) 141–173
7. McCallum, A., Nigam, K.: A comparison of event models for Naive Bayes text classification. In: Learning for Text Categorization: Papers from the AAAI Workshop, AAAI Press (1998) 41–48 Technical Report WS-98-05.

8. Yang, Y., Liu, X.: A re-examination of text categorization methods. In: Proc. 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'99). (1999) 42–49
9. Mitchell, T.M.: Machine Learning. McGraw-Hill, New York (1997)
10. Joachims, T.: Text categorization with support vector machines: Learning with many relevant features. In: Proc. 10th European Conference on Machine Learning (ECML98). Volume 1398 of Lecture Notes in Computer Science., Heidelberg, Springer (1998) 137–142
11. Craven, M., DiPasquo, D., Freitag, D., McCallum, A., Mitchell, T., Nigam, K., Slattery, S.: Learning to construct knowledge bases from the World Wide Web. Artificial Intelligence **118** (2000) 69–113
12. Katz, S.M.: Distribution of content words and phrases in text and language modelling. Natural Language Engineering **2** (1996) 15–59
13. Domingos, P., Pazzani, M.: On the optimality of the simple Bayesian classifier under zero-one loss. Machine Learning **29** (1997) 103–130
14. Friedman, J.H.: On bias, variance, 0/1-loss, and the curse-of-dimensionality. Data Mining and Knowledge Discovery **1** (1997) 55–77
15. Mladenić, D., Grobelnik, M.: Word sequences as features in text-learning. In: Proc. 17th Electrotechnical and Computer Science Conference (ERK98), Ljubljana, Slovenia (1998)
16. Gómez-Hidalgo, J.M., de Buenaga Rodríguez, M.: Integrating a lexical database and a training collection for text categorization. In: ACL/EACL-97 Workshop on Automatic Information Extraction and Building of Lexical Semantic Resources for NLP Applications. (1997) 39–44
17. Dhillon, I.S., Mallela, S., Kumar, R.: A divisive information-theoretic feature clustering algorithm for text classification. Journal of Machine Learning Research **3** (2003) 1265–1287
18. Torkkola, K.: Linear discriminant analysis in document classification. In: IEEE ICDM-2001 Workshop on Text Mining (TextDM'2001), San Jose, CA (2001)
19. Rennie, J.D.M., Shih, L., Teevan, J., Karger, D.: Tackling the poor assumptions of Naive Bayes text classifiers. In Fawcett, T., Mishra, N., eds.: Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003), Washington, D.C., AAAI Press (2003) 616–623
20. Kim, S.B., Rim, H.C., Yook, D., Lim, H.S.: Effective methods for improving Naive Bayes text classifiers. In Ishizuka, M., Sattar, A., eds.: Proc. 7th Pacific Rim International Conference on Artificial Intelligence. Volume 2417 of Lecture Notes in Artificial Intelligence., Heidelberg, Springer (2002) 414–423
21. Eyheramendy, S., Lewis, D.D., Madigan, D.: On the Naive Bayes model for text categorization. In Bishop, C.M., Frey, B.J., eds.: AI & Statistics 2003: Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics. (2003) 332–339
22. Yang, Y., Pedersen, J.O.: A comparative study on feature selection in text categorization. In: Proc. 14th International Conference on Machine Learning (ICML-97). (1997) 412–420
23. Forman, G.: An extensive empirical study of feature selection metrics for text classification. Journal of Machine Learning Research **3** (2003) 1289–1305
24. Cover, T.M., Thomas, J.A.: Elements of Information Theory. John Wiley, New York (1991)
25. Bennett, P.N.: Assessing the calibration of Naive Bayes' posterior estimates. Technical Report CMU-CS-00-155, School of Computer Science, Carnegie Mellon University (2000)
26. Apté, C., Damerau, F., Weiss, S.M.: Towards language independent automated learning of text categorization models. In: Proc. 17th ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '94). (1994) 23–30

Efficient Modeling of Analogy

Lars G. Johnsen and Christer Johansson

Dept. of Linguistics and Literature, University of Bergen,
N-5007 Bergen, Norway
{lars.johnsen, christer.johansson}@lili.uib.no

Abstract. Analogical modeling (AM) is a memory based model with a documented performance comparable to other types of memory based learning. Known algorithms implementing AM have a computationally complexity of $O(2^n)$. We formulate a representation theorem on analogical modeling which is used for implementing a range of approximations to AM with a complexity starting as low as $O(n)$.

1 Introduction

The algorithm for Analogical Modeling (AM) was first published in 1989 [1], and has since remained unchanged with the exception for some minor corrections [1, 2]. Known implementations of AM [2] suffer from having an exponential time complexity $O(n)$ where n is the number of features used to describe a particular example.

AM is a memory based model and constructs a classifier on the basis of a set of examples $\mathcal{D}$. The key computation for the analogical classifier is the construction of the analogical set $\mathcal{A}$ (defined below) associated with an exemplar τ in conjunction with $\mathcal{D}$. We will show that the effect of the analogical classifier is obtained by constructing a generally smaller set $\mathcal{M}$, which, together with a set of parameters C, has the same effect as the original. The aim of this article is to prove that there exists a simpler, yet (roughly) equivalent function to build an analogical classifier, which avoids building the full lattice $\mathcal{L}$ of the original model. The new function uses the set $\mathcal{M}$ and a set of parameters to compute a close approximation. Different parameter sets correspond to different approximations.

2 Background on AM

AM has been used as a simulation model of cognitive psycholinguistics, and it compares well with connectionist models [3, 4]. AM does not suffer from the problems associated with the delta-rule of connectionist learning [3–pp.62 ff], and at the same time it accounts for significant phenomena such as 'perceptual learning, latent inhibition, and extinction [...] within a single mechanism' (ibid. p.62). In fact, there are very few assumptions in AM; there are, for example, no assumption on the distribution of exemplars, nor are global weights calculated. Experience has a direct effect in AM in that only the database changes with

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 694–703, 2005.

added experience. The same unchanged mechanism is used to find support in the database for the categories of new input. Weights are not calculated, neither are there any fixed connections between any data. These ought to be very attractive features for a memory based model of categorization. In addition, AM reacts similar to noise as a naive k-NN model, and tend to fare slightly better than k-NN using information gain [5]. AM is also very robust for 'forgetting'. Randomly throwing out as much as half of the database is equivalent to standard statistical results [6–p.33], see also [3–pp.86–87] and [7–p.175].

AM has also been applied to practical tasks such as predicting morphological markings for past tense in Finnish [6, 1], Spanish gender, diminutive, and stress [4], Dutch word stress [5] and German plural [7] to mention a few studies. It has been suggested that outcomes could be found by means of analogy [8], which would be an unsupervised extension of the model.

AM typically beats a naive memory based (k-NN) model, and it performs similar to a k-NN model which uses the information gain of each feature [4, 5, 7]. Looking more closely at the results, AM estimates 'gang effects' of frequent formation patterns that are far from the input to be categorized [1, 6, 7]. This effect is harder to get in k-NN models, although clever feature design and information measures might help k-NN to discover such gang effects. AM has also inspired a clever way of database compression in k-NN models, using families of examples with homogeneous outcomes [9].

3 Preliminary Concepts and Definitions

The analogical classifier decides on the outcome for a particular instance τ based on the analogical support from a set $\mathcal{D}$ of examples. The examples in $\mathcal{D}$ have the same structure as τ, and are represented as indexed sets (see lattice definition below) over finite domains such as phonemes, words or morphemes. Each example $d \in \mathcal{D}$ has an atomic outcome classifying the example, notated $o(d)$, while the set of outcomes is notated $\mathcal{O}$. The analogical classifier now determines an outcome for τ from the set of outcomes $\mathcal{O}$. Whatever the outcomes actually are, we will use the fact that they form a finite domain of size n, so that each element of that domain can be associated with a unit vector in an n-dimensional integer space. This way the frequencies of different outcomes are computed by adding the corresponding unit vectors.

For example, consider the database in table 1 below. There are two possible outcomes, x and y. Examples belonging to the classes are listed in the columns (6 exemplars of each class). For simplicity the sets in the database are written so that 039 is $\{0_1, 3_2, 9_3\}$. The test element in this case is $\tau = \{0_1, 2_2, 6_3\}$, and the column heading $\tau \cap x$ means the cell below comes from intersecting τ with the set with outcome x from the cell's row, and likewise for y. The outcomes x and y may be encoded, in a two-dimensional space, as $(1, 0)$ and $(0, 1)$ respectively. This encoding scheme is assumed in definitions below.

Note that an alternative form of representation for τ and the examples in $\mathcal{D}$ is the vector form. Given τ as in table 1, it may be written $\tau = (1, 2, 6)$. The

Table 1. An example adapted from Skousen [1–p.40]

x	$(\tau \cap x)$	y	$(\tau \cap y)$
027	$\{0_1, 2_2\}$	126	$\{2_2, 6_3\}$
039	$\{0_1\}$	137	$\emptyset$
046	$\{0_1, 6_3\}$	147	$\emptyset$
047	$\{0_1\}$	148	$\emptyset$
048	$\{0_1\}$	157	$\emptyset$
058	$\{0_1\}$	159	$\emptyset$

vector representation is used in [1], while we choose the set representation, as sets come equipped with the all the algebraic operations needed to combine τ and elements of $\mathcal{D}$.

The lattice $\mathcal{L}$ consists of elements generalizing τ, containing both the analogical set $\mathcal{A}$ and our set $\mathcal{M}$. $\mathcal{L}$ is the powerset of τ and represents all the possible feature matches that τ may yield. Vector notation represents elements of $\mathcal{L}$ as vectors, with an underscore denoting an undefined value, e.g. $(0, \_, 6)$, while in set representation they are subsets of τ, e.g. $\{0_1, 6_3\} \subseteq \{0_1, 2_2, 6_3\}$.

The set $\mathcal{M}$ consist of elements that represent shared features between τ and elements of $\mathcal{D}$. It is constructed by the function μ, which takes the test element τ and combines it with an element from $\mathcal{D}$, thus mapping τ onto a subset of $\mathcal{L}$:

$$\mathcal{M} = \mu(\tau) = \{(\tau \cap d) | d \in \mathcal{D}\ \} \subseteq \mathcal{L}.$$

For example, applying μ to the elements of table 1 results in the cell elements:

$$\mu(\{0_1, 2_2, 6_3\}) = \{\{0_1, 2_2, 6_3\} \cap d \mid d \in \mathcal{D}\}$$

$$= \{\{0_1, 2_2\}, \{0_1\}, \{0_1, 6_3\}, \{2_2, 6_3\}, \emptyset\}.$$

It is evident from this definition that the maximum size of the match set, $\mathcal{M}$, is either $||\mathcal{D}||$ or $||\mathcal{L}||$, whichever is smallest. For cases with many features, $||\mathcal{L}||$ is typically much larger than $||\mathcal{M}||$, the former growing exponentially with the size of τ. $\mathcal{M}$ may reach the size of $||\mathcal{D}||$ when all examples of the database are disjoint. Typically, $||\mathcal{M}||$ is significantly smaller. $||\mathcal{M}||$ for the database in table 1 with $\tau = 026$ is 5, namely $\{0_1, 2_2\}$, $\{0_1\}$, $\{0_1, 6_3\}$, $\{2_2, 6_3\}$ and the empty set, $\emptyset$. Here, the theoretical maximum is determined by the $8 = 2^3$ possibilities of $\mathcal{L}$.

We define two mappings between $\mathcal{L}$ and $\mathcal{D}$ which are generalized inverses of μ. One mapping called θ induces a partition over $\mathcal{D}$, and is used in conjunction with $\mathcal{M}$ to compute a score, while the other mapping, σ, gives the support for elements of $\mathcal{L}$, and is used in conjunction with $\mathcal{A}$.

The function θ yields all the elements d of $\mathcal{D}$ such that $a \cap d = \tau$, i.e.

$$\theta(a) = \{d \in \mathcal{D} \mid a = d \cap \tau\}.$$

We will prove the partition inducing property of θ is proved in proposition 1 in section 4 below.

The support function σ is defined for a particular element of $\mathcal{L}$ as the collection of elements in $\mathcal{D}$ that share a feature with it,

$$\sigma(m) = \{d \in \mathcal{D} |\ m \subseteq d\}.$$

The actual scoring of a subset of $\mathcal{D}$ is done by adding up all the outcomes for each element of the subset. The function *score* is defined by

$$score(Sub) = \sum_{y \in Sub} o(y).$$

As noted above, each outcome is considered to be a unit vector in n-dimensional space, where n is the total number of outcomes.

For instance, with reference to table 1, if *score* is applied to

$$\theta(\{0_1\}) = \{039, 047, 048, 058\}$$

we know that since $o(a) = (1, 0)$ for each of the 4 elements $a \in \theta(\{0_1\})$ that

$$score(\theta\{0_1\}) = (4, 0)$$

hence there are 4 cases of x's in this case. Compare this with

$$\sigma(\{0_1\}) = \{039, 047, 048, 058, 027, 046\}$$

which results in

$$score(\sigma\{0_1\}) = (4, 2)$$

meaning that there are 4 votes for x and 2 for y.

Note that $\theta(a) \subseteq \sigma(a)$ for any $a \in \mathcal{L}$. Given a pattern $a \in \mathcal{L}$, the $score(\sigma(a))$ returns the frequency of each outcome from $\sigma(a)$ in a vector $(outcome_1 ... outcome_n)$. In table 1, there are two outcomes x and y, so that $n = 2$ given that particular database.

Homogeneity is a key concept in analogical modeling. A formal definition is given below, but the content of homogeneity is that elements of $\mathcal{L}$ can only contribute unique scorings, scorings on the form $n\mathbf{i}$ where $n \in \mathbb{N}$ and $\mathbf{i}$ is a unit vector in the set of outcomes $\mathcal{O}$. In this case, the element is said to be deterministically homogeneous. Non-unique scorings are taken into account if it comes from an element which contains no other element with a non-void scoring.

Homogeneity is formally captured through the number of disagreements in a pattern. Disagreements in $m \in \mathcal{L}$ is measured by counting the number of different outcomes in $\sigma(m)$, notated $\delta(m)$. Following Skousen [1], we define disagreement within $\sigma(m) \times \sigma(m)$. Each point $< r, s > \in \sigma(m) \times \sigma(m)$ is an argument to the function κ, defined as

$$\kappa(r, s) = \begin{cases} 1 & \text{if } o(r) \neq o(s) \\ 0 & \text{otherwise} \end{cases}$$

Then $\delta(m)$ is defined as the sum of κ over all pairs in $\sigma(m) \times \sigma(m)$.

Definition 1 (Differences). *For a given $m \in \mathcal{L}$, the number of differences in m is defined as*

$$\delta(m) = \sum_{r,s \in \sigma(m)} \kappa(r,s)$$

A homogeneous pattern is required to have non-empty support and, if it is more general than a pattern with non-empty support, they must have exactly the same number of differences.

Definition 2 (Homogeneity). *A pattern m is homogeneous if $\sigma(m) \neq \oslash$ and, whenever $m \subseteq n$ and $\sigma(n) \neq \oslash$, $\delta(n) = \delta(m)$*

The stricter class of deterministically homogeneous elements captures those with unique scoring.

Definition 3 (Deterministic homogeneity). *A pattern m is deterministically homogeneous if m is homogeneous and $\delta(m) = 0$*

Two simple consequences of these definitions are the following two corollaries:

1. If m is deterministically homogeneous then $\sigma(m)$ has only one outcome.
2. If m is non-deterministically homogeneous and $n \subseteq m$ is homogeneous, then $\sigma(n) = \sigma(m)$

With these definitions in place, the analogical set $\mathcal{A}$ is defined to be the homogeneous elements of $\mathcal{L}$, i.e.

$$\mathcal{A} = \{a \in \mathcal{L} |\ a \textit{ is homogeneous}\}.$$

The analogical classifier assigns the outcome with highest frequency to τ based on the score for the analogical set. One effect of this is that in order to construct an equivalent to the analogical classifier it may not be necessary to compute the whole effect. Once the resulting outcome is determined the behaviour of the classifier is captured.

We saw above, with reference to table 1, how σ and θ are calculated. Let us enumerate some of the sets defined so far using the data in the table:

$$\begin{aligned}
\mathcal{L} &= \wp(\{0_1, 2_2, 6_3\}) \text{ where } \wp \text{ is the powerset operator} \\
\mathcal{A} &= \{\{0_1, 2_2\}, \{0_1\}, \{0_1, 6_3\}, \{2_2, 6_3\}\} \\
\mathcal{M} &= \mathcal{A} \cup \{\emptyset\} \\
\theta(\{2_2\}) &= \emptyset \\
\sigma(\{2_2\}) &= \{\{0_1, 2_2, 7_3\}, \{1_1, 2_2, 6_3\}\}
\end{aligned}$$

Note that $\mathcal{A}$ is a proper subset of $\mathcal{M}$ for this particular dataset. In general, the cardinality of $\mathcal{A}$ will be a much larger than that of $\mathcal{M}$. Since homogeneity is not a defining characteristic of $\mathcal{M}$, $\mathcal{M}$ will likely have members which are not members of $\mathcal{A}$. Homogeneity enters into play via the parameters used for scoring $\mathcal{M}$, as defined below, ensuring that non-homogeneous elements do not count in the overall score.

4 Results

In this section, we characterize the analogical set $\mathcal{A}$ in terms of the match set $\mathcal{M}$. We prove that it is only necessary to consider this match set, which formally contains all the possible support from the database for the outcome of the test pattern τ.

The following proposition ensures that θ induces an equivalence class over $\mathcal{D}$. In particular, sets $\theta(x)$ and $\theta(y)$ are disjoint whenever $x \neq y$.

Proposition 1. *For any $x, y \in \mathcal{L}$, if $x \neq y$ then $\theta(x) \cap \theta(y) = \emptyset$*

Proof. Assume that $\theta(x) \cap \theta(y) \neq \emptyset$, and pick an a such that $a \in \theta(x)$ and $a \in \theta(y)$. By definition we then have that that $x = a \cap \tau$ and $y = a \cap \tau$, so that $x = y$ contradicting $x \neq y$.

Proposition 2. *For any $p \in \mathcal{L}$, the support for elements $p \in \mathcal{L}$ is characterized by elements of $\mathcal{M} \subseteq \mathcal{L}$*

$$\sigma(p) = \bigcup_{\substack{p \subseteq x \\ x \in \mathcal{M}}} \{\theta(x)\}$$

Proof. Any element $\delta \in \mathcal{D}$ which is a member of $\sigma(p)$ contains p, i.e. $p \subseteq \delta$. Since $p \subseteq \tau$ it follows that $p \subseteq \tau \cap \delta$, and since $x = \tau \cap \delta \in \mathcal{M}$ we have that $\delta \in \theta(x)$. Conversely, if $\delta \in \theta(x)$ for $x \in \mathcal{M}$, $x = \tau \cap \delta$, and by the assumption that $p \subseteq x$, it follows that $p \subseteq \tau \cap \delta \subseteq \delta$, which means that $\delta \in \sigma(p)$.

These two propositions serves as the building blocks for the representation theorem on analogical modeling.

4.1 Representing the Analogical Set

Recall the definition for scoring a set $S \subseteq \mathcal{D}$:

$$score(S) = \sum_{x \in S} o(x)$$

The total score attributed to the analogical set, *tot*, is the sum of all the scores over all elements of the analogical set:

$$tot = \sum_{p \in \mathcal{A}} score(\sigma(p)).$$

Our version of the total score, with the elements from $\mathcal{M}$, employs coefficients.

$$tot = \sum_{x \in \mathcal{M}} c_x score(\theta(x)).$$

The coefficients c_x encode for each $x \in \mathcal{M}$ the number of homogeneous elements l of $\mathcal{L}$ that are subsumed by x, i.e. $l \subseteq x$. Note that this implies that if a particular element $m \in \mathcal{M}$ is not homogeneous its coefficient $c_m = 0$, effectively erasing m's contribution from the total score.

Theorem 1. *The result of analogical modeling can be characterized in terms of the set* $\mathcal{M}$ *so that*

$$\sum_{p \in \mathcal{A}} score(\sigma(p)) = \sum_{x \in \mathcal{M}} c_x score(\theta(x))$$

where

$$c_x = \|\{p \mid p \text{ is homogeneous and } p \subseteq x\}\|$$

Proof. The scoring of the analogical set is

$$\sum_{p \in \mathcal{A}} score(\sigma(p))$$

which by proposition 2 is the same as

$$\sum_{p \in \mathcal{A}} score(\bigcup_{\substack{p \subseteq x \\ x \in \mathcal{M}}} \{\theta(x)\})$$

Proposition 1 says that this is a disjoint union, and it is therefore equal to

$$\sum_{p \in \mathcal{A}} \sum_{\substack{p \subseteq x \\ x \in \mathcal{M}}} score(\theta(x)).$$

Interchanging the two sums while retaining the condition $p \subseteq x$ on the inner sum, yields

$$\sum_{x \in \mathcal{M}} \sum_{\substack{p \subseteq x \\ p \in \mathcal{A}}} score(\theta(x))$$

From this expression we see that the number of times the term $score(\theta(x))$ is summed up is equal to the number of homogeneous elements it dominates. With c defined as above, the scoring of the analogical set is then equivalent to

$$\sum_{x \in \mathcal{M}} c_x score(\theta(x)).$$

The implication of this theorem is that $\mathcal{M}$ contains *all* the results necessary for computing the overall effect of the analogical set, without actually building the whole set. But how much is lost? Is it possible to estimate how many extensions a homogeneous pattern has?

4.2 Rough Approximation

The match set $\mathcal{M}$, as we have seen above, contains the necessary support for analogical modeling. We are primarily interested in the homogeneous subset of $\mathcal{M}$, which can be found by sorting $\mathcal{M}$ and determine which patterns have

multiple outcomes. A pattern with multiple outcomes is non-deterministically homogeneous, if there is no pattern in $\mathcal{M}$ that is closer to the test pattern.

Recall our example in table 1. The sorted match set for test pattern 026 is $\{\{0_1, 2_2\}, \{0_1, 6_3\}, \{2_2, 6_3\}, \{0_1\}, \{\}\}$. $\{0_1, 2_2\}$ and $\{0_1, 6_3\}$ match only one example each, both with outcome x. $\{2_2, 6_3\}$ matches one example with outcome o. $\{0_1\}$ matches 6 examples of outcome x. So far we have had only homogeneous outcomes. The last pattern $\{\}$ matches everything, but since there were patterns before, closer to 026, it is heterogeneous and should not be counted. In this case, the approximation gives the exact results of AM.

We could have ended here, since we have an approximation of AM, which needs to look at each example in the database only once to find the match set, and then needs a sorting of the match set (which can be done in $O(log(||\mathcal{M}||)||\mathcal{M}||)$, where $||\mathcal{M}||$ typically is much smaller than the size of the database). Finally, for each member in the match set, the algorithm goes through the examples once again. The time complexity of this algorithm in the worst case appears to be $O(log(N)N)$, though the worst case happens exactly when analogy is useless, i.e., when each example is best characterized as an atomic symbol. We expect typical time complexity to be close to linear. Furthermore, space complexity just depends on the size of the database. The search through the database is linear in the number of variables (whereas original AM is exponential, in both time and space complexity, for the number of features).

How often this approximation fails on practical language tasks is an empirical question. However, we will make an attempt at estimating the parameters c_x, which would make the approximation truly equivalent to full analogical modeling. The estimation can be done statistically by using Monte Carlo methods, presented in the next section.

4.3 Simulation of the Parameters

Monte Carlo simulation [10–pp.237–241] can be used to find the parameters c_x of theorem 7 above. When a is homogeneous, then $a \subseteq b$ implies that b is homogenous. We see that only the homogeneous subset of $\mathcal{M}$, notated $\mathcal{M}_\mathcal{A}$ is needed to estimate $\mathcal{A}$.

The collection of elements below any element $a \in \mathcal{M}_\mathcal{A}$ is $\wp(a)$. Removing the heterogeneous elements from $\wp(a)$ results in the desired collection of homogeneous elements p such that $p \subseteq a$.

The heterogeneous elements are found by counting the elements in $\wp(a)$ that are shared by other elements with a different outcome from a. The outcome, $o(p)$ for a pattern p is defined so that $o(p) = i$ if $o(x) = i$ for all $x \in \sigma(p)$. This will associate each deterministically homogenous element with a unique outcome, the outcome for the elements in its support. For non-deterministically homogenous elements the outcome is left undefined. The set $\mathcal{H}(p)$ is defined to be the intersection of p with all other elements of $\mathcal{M}_\mathcal{A}$ that have a different outcome

$$\mathcal{H}(p) = \{x \cap p \mid x \in \mathcal{M}_\mathcal{A}\}$$

under the constraint that $o(x) \neq o(p)$ if o is defined for p .

We are not aware of any algorithmically cheap (i.e. polynomial) way of determining the union of a family of powersets, but it is computationally cheap to determine whether a set belongs to that union: for a given candidate x, go through the elements of $\mathcal{H}_{limit(p)}$ one by one, and check whether x is a subset of it. This feature, together with accurate estimates of lower and upper bounds, makes $\mathcal{H}_{limit(p)}$ a candidate for Monte Carlo simulation.

The minimum number of elements in $\mathcal{H}_{limit(p)}$ is given by $\|\wp(x)\|$ where x is the largest element in $\mathcal{H}(p)$ so that $\|x\| = min(p)$. The smallest size, or lower bound $lb(p)$, of $\mathcal{H}_{limit(p)}$ is thus $2^{min(p)}$.

A candidate for the upper bound of $\mathcal{H}_{limit(p)}$ is

$$\wp(\bigcup_{x\in\mathcal{H}(p)}(x)).$$

This candidate limit can be improved by observing that no element of $\mathcal{H}_{limit(p)}$ has more members than $min(p)$.

Under the constraint that $\|\bigcup_{x\in\mathcal{H}(p)}(x))\| = max(p)$, the maximal number of sets in $\mathcal{H}_{limit(p)}$ is determined by how many ways we can construct sets from the $max(p)$ elements of p such that the sets have a cardinality smaller than or equal to $min(p)$. The upper bound, $ub(p)$ for $\mathcal{H}_{limit(p)}$ is then

$$ub(p) = \sum_{k=1}^{min(p)} \binom{max(p)}{k}$$

The size of $\mathcal{H}_{limit(p)}$ can now be estimated by sampling elements x_s from $\bigcup_{x\in\mathcal{H}(p)}(x) \subseteq p$ with cardinality less than or equal to $min(p)$.

The estimate, $\hat{h_p}$, is computed from the ratio of sampled elements over the total number of samples by the equation

$$\frac{\|\{x_s \in \mathcal{H}_{limit(p)}\}\|}{\|\{x_s\}\|} = \frac{\hat{h_p}}{ub(p)}$$

which gives

$$\hat{h_p} = \frac{ub(p)\|\{x_s \in \mathcal{H}_{limit(p)}\}\|}{\|\{x_s\}\|}$$

If every sampled x_s is in $\mathcal{H}_{limit(p)}$ the estimate equals the upper bound.

Recall

$$c_x = \|\{p \mid p \textit{ is homogeneous and } p \subseteq x\}\|$$

in terms of $\hat{h_x}$; $c_x = \|\wp(x)\| - \hat{h_x}$. Thus we have an estimate of the required parameters c_x, which we can calculate within the time bounds of the Monte Carlo simulation, which depends mainly on the size of the match set, $O(\|\mathcal{M}\|^2)$. To estimate the variation of the simulations, we are forced to repeat the simulations a limited number of times, depending on the discrepancy between upper and lower bound for each particular pattern p. This increases complexity, but we are still within polynomial time.

5 Conclusions

We have shown a rough approximation of analogical modeling, which has many beneficial properties compared to the original model. AM is often correlated with k-NN learning, but the decisions of AM are based on the entire database, and not just a neighborhood of similar instances. This makes it possible for AM to find gang effects, which may be of both practical and theoretical interest [3]. The correctness of the approximation was formally proved above (section 4). Although there might still be some support that slips away, the approximation at least gives every instance its say on the final decision. In many cases, the rough approximation exactly matches full AM.

The discussion on Monte Carlo simulation, gives an outline of how the constants c_x could be approximated without calculating the full lattice (which would be exponential in both space and time complexity). The simulations could bring the approximation even closer to the theoretical model, even though it might not be practical in many cases, and probably should only be used in cases where the decision of the rough approximation is close to being undecided.

Acknowledgement. This work was supported by a grant from the Norwegian Research Council under the KUNSTI programme (project BREDT). We thank Viktor Trón and an anonymous reviewer for helpful comments and discussions.

References

1. Skousen, R.: Analogical Modeling of Language. Kluwer Academic, Dordrecht, the Netherlands (1989)
2. Skousen, R., Lonsdale, D., Parkinson, D.: Analogical Modeling: An exemplar-based approach to language. Volume 10 of Human Cognitive Processing. John Benjamins, Amsterdam, the Netherlands (2002)
3. Chandler, S.: Skousen's analogical approach as an exemplar-based model of categorization. In: Skousen et al. (2002) 51–105
4. Eddington, D.: A comparison of two analogical models: Tilburg memory-based learner versus analogical modeling. In: Skousen et al. (2002) 141–155
5. Daelemans, W., Gillis, S., Durieux, G.: Skousen's analogical modeling algorithm: A comparison with lazy learning. In: proceedings of NEMLAP, Manchester, UK. (1994)
6. Skousen, R.: Issues in analogical modeling. In: Skousen et al. (2002) 27–48
7. Daelemans, W.: A comparison of analogical modeling to memory-based language processing. In: Skousen et al. (2002) 157–179
8. Johansson, C.: The Hope for Analogical Categories. In: Skousen et al. (2002) 301–316
9. Bosch, A.v.d.: Expanding k-nn analogy with instance families. In: Skousen et al. (2002) 209–223
10. Press, W., Teukolsky, S., Vetterling, W., Flannery, B.: Numerical Recipes in C. Second edition edn. Cambridge University Press (1992)

A Supervised Clustering Method for Text Classification

Umarani Pappuswamy, Dumisizwe Bhembe,
Pamela W. Jordan, and Kurt VanLehn

Learning Research and Development Center,
3939 0'Hara Street, University of Pittsburgh,
Pittsburgh, PA 15260, USA
umarani@pitt.edu

Abstract. This paper describes a supervised three-tier clustering method for classifying students' essays of qualitative physics in the Why2-Atlas tutoring system. Our main purpose of categorizing text in our tutoring system is to map the students' essay statements into principles and misconceptions of physics. A simple `bag-of-words' representation using a naïve-bayes algorithm to categorize text was unsatisfactory for our purposes of analyses as it exhibited many misclassifications because of the relatedness of the concepts themselves and its inability to handle misconceptions. Hence, we investigate the performance of the k-nearest neighborhood algorithm coupled with clusters of physics concepts on classifying students' essays. We use a three-tier tagging schemata (cluster, sub-cluster and class) for each document and found that this kind of supervised hierarchical clustering leads to a better understanding of the student's essay.

1 Introduction

Text Categorization (or Classification)[1] can be seen either as an Information Retrieval task or a Machine Learning task of automatically assigning one or more well-defined categories or classes to a set of documents. Starting with the work of Maron [1] in the early 60s, Text Classification (TC) has found a significant place in a variety of applications including: automatic indexing, document filtering, word sense disambiguation, and information extraction. Our main focus is on the machine learning aspect of TC with the goal to devise a learning algorithm capable of generating a classifier which can categorize text documents into a number of predefined concepts. This issue has been considered in several learning approaches both with a supervised learning scheme [2, 3] and with an unsupervised and semi-supervised learning scheme [4, 5, 6].

In its simplest form, the text classification problem can be formulated as follows: We are given a set of documents $D = \{d_1, d_2, d_3 \ldots d_n\}$ to be classified and $C = \{c_1, c_2, c_3, \ldots c_n\}$ a predefined set of classes and the values $\{0, 1\}$ interpreted as a decision to file a document d_j under c_i where 0 means that d_j is not relevant to the class defined and 1 means that d_j is relevant to the class defined. The main objective here is to devise a

[1] We prefer the term `Text Classification' to `Text Categorization' and hence use the same in the rest of our paper.

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 704 – 714, 2005.

learning algorithm that will be able to accurately classify unseen documents from D (given the training set with the desired annotations in the case of supervised learning).

In our paper, we describe a three-tier clustering method for classifying students' essay strings in the Why2-Atlas system. The students' essays are the answers to qualitative questions of physics. The task of the classifier is to map these essay strings into the corresponding principles and misconceptions of physics. A simple `Bag-Of-Words (BOW)' approach using a naïve-bayes algorithm to categorize text was unsatisfactory for our purposes of analyses as it exhibited many misclassifications because of the relatedness of the concepts themselves and its inability to handle misconceptions. Hence, we investigate the performance of k-nearest neighborhood algorithm coupled with pre-defined clusters of physics concepts on classifying students' essays. Though there have been many studies on word clustering for language modeling and word co-occurrence [7], very little work has been done on word/concept clustering for document classification.

We present the results of an empirical study conducted on a corpus of students' essay strings. The approach uses a three-tier tagging schemata (cluster, sub-cluster and class) for each document. Let C and SC refer to the Cluster and Sub-cluster respectively, and `Class (Cl)' refers to the actual principle or misconception being identified. Thus, C in the original definition now takes the form: $C = \{(C_1, SC_1, Cl_1), (C_n, SC_n, Cl_n)\}$. This kind of supervised clustering approach helps us to reduce the dimensionality of the texts and thereby leads to a better understanding of the student's essay.

The next section, namely Section 2 describes text classification in the Why2-Atlas tutoring system; Section 3 describes the three-tier clustering method and its experimental design, Section 4 presents the results and discussion of our experiment and Section 5 provides conclusions and directions for future work.

2 Text Classification in the Why2-Atlas System

The Why2-Atlas system presents students with qualitative physics problems and encourages them to write their answers along with detailed explanations to support their answers [8]. As shown in Fig. 1, the student explanation from our corpus of human-human computer-mediated tutoring sessions, illustrates the type of explanation the system strives to elicit from students. It is a form of self-explanation so it has the potential to lead students to construct knowledge [9], and to expose deep misconceptions [10].

Question: Suppose you are in a free-falling elevator and you hold your keys motionless right in front of your face and then let go. What will happen to them Explain.

Explanation (Essay): Free-fall means without gravity. The keys should stay right in front of your face since no force is acting on the keys to move them.

Fig. 1. An actual problem statement and student explanation

In the above example, there is a clear statement of misconception `Freefall means without gravity'. Unless we evaluate the answers that students type in, we would not be able to help them reconstruct their knowledge. There are a variety of ways in which a student essay can be evaluated or graded. Autotutor [11] uses Latent Semantic Analysis (LSA) to analyze student essays. AutoTutor comprehends the student input by segmenting the contributions into speech acts and matching the student's speech acts to the tutor's expectations. If the expectations are covered in the student's essay, the essay is considered to be `good'.

In Why2-Atlas system, we use a similar method. We first look for the correctness of the answer and then use a list of Principles (P) and Misconceptions (M) and look for the presence of a P or M in the student essay. We have an `ideal answer' for each problem statement which is clearly marked for the necessary principles to be mentioned in the respective essay. If the essay contains all of the Ps stated in the ideal answer, then it is considered to be a reasonably good essay and we allow the student to move on to the next problem. Thus, it is important to classify the students' essay strings into Ps and Ms in order to subject it to further processing.

Several other attempts have been made in the project to analyze students' essays in the past using TC methods. Rose et al.'s experiments [12] used `keypoints (correct answer aspects)' and `nothing' (in case of absence of a correct answer aspect) to classify essay strings; the precision and recall measures for the pumpkin problem[2] was 81% and 73% respectively. The limitation of this approach was the inability to generalize the training across problems and to identify misconceptions (if any) expressed by the student in his/her essay. There was an attempt to extend to more problems later in the project by identifying only `Principles' for each problem. The classifier's performance is measured in terms of accuracy and standard error. Accuracy is the percentage of correctly labeled documents in the test set. Standard error of the prediction is computed over the accuracy. The results are shown in Table 1. As the number of classes increased, the accuracy declined. Hand-checking of the tags assigned to these examples revealed many misclassifications. It was clear that the complexity of the problem lies in the nature of the data itself.

Table 1. Performance of NB classifier on subsets

Subset[3]	No. of classes	No. of examples	Accuracy	Std Error
Pumpkin	17	465	50.87	1.38
Packet	14	355	55.49	1.99
Keys	20	529	48.46	1.62
Sun	8	216	60.60	1.42
Truck	8	273	65.22	0.93

Furthermore, as this approach did not include training examples for misconceptions, the classifier grouped all such instances as `nothing' (false negatives) or put them under different `wrong' classes (false positives) neither of which was desirable by us. Since these problems share principles and misconceptions between them, yet

[2] Pumpkin was one of the 10 problems given to the students in the tutoring session.
[3] Subset includes data for the specific problems (pumpkin, keys, etc).

another approach was made to combine the examples from the subsets (in Table 1) into one. We included training examples for misconceptions as well. We tested this new dataset using the same NB algorithm and the results of this experiment are shown in Table 2:

Table 2. Performance of NB classifier on global data

Set	No. of classes	No. of examples	Accuracy	Std. Error
Global[4] (all problems)	38	586	56.83	0.45

Due to the similarity of the words present in the list of principles and misconceptions, there were still many misclassifications. To get a better understanding of the nature of the problem, we tested 15 documents that belong to one concept. We expected the classifier to tag all the documents for only one class `prin-only-gravity-implies-freefall' (The description of this principle is: "When gravity is the only force acting on an object, it is in freefall"). The classifier's predictions[5] reveal the following:

0 tagged for the expected principle `prin-only-gravity-implies-freefall' (Class1)
12 tagged for `prin-drop-obj-only-grav' (Class2)
1 tagged for `prin-release-freefall' (Class3)
4 tagged for `prin-freefall-same-accel' (Class4)
1 tagged for `nothing' (Class5)

Based on the training data, the classifier thus, encountered different but related principles for the above set of data. This led us to examine the probability distribution of the words used by each of these classes. Table 3 shows the probability distribution of the top 10 words.

It can be observed that the significant words `gravity, free and fall' are found in all the classes (2–4) and hence the problem of ambiguity arose. However, it should be noted that the tags obtained above are related to each other. One can say that they are partially correct and are related to the top principle in question. For instance, *`prin-drop-obj-only-grav*' is a subset of *`prin-only-gravity-implies-freefall'*. So, based on the combined probability of the key words that are common for both these principles, the classifier learned `prin-drop-obj-only-grav' as in *"*The only force acting on the man and the keys is the force of gravity*"*. Later on, we tested a few more sentences chosen randomly that contained words like `freefall' and `gravity'. Hand-checking of the predictions revealed that a sentence like `Freefall means without gravity' (a misconception) was classified as a principle. This is not surprising because `without' was

[4] This included data from all the ten problems.

[5] The mismatch in the number of tags (18) and the number of sentences (15) is due to some segmentation problems. Some of the documents were broken into more than one due to the presence of a `period'. The principles corresponding to Classes 2, 3, and 4 are related to the concept of `freefall' but do not correspond to the exact description of the concept in Class1.

Table 3. Info-gain of the top 10 words using NB

Class2		Class3		Class4	
words	**probability**	**words**	**probability**	**words**	**probability**
force	0.056206	force	0.021341	keys	0.027356
gravity	0.046838	free	0.018293	freefall	0.024316
keys	0.039813	fall	0.018293	elevator	0.024316
acting	0.035129	gravity	0.015244	free	0.018237
elevator	0.023419	acting	0.015244	person	0.015198
fall	0.018735	keys	0.015244	fall	0.015198
free	0.014052	freefall	0.012195	release	0.012158
rate	0.011710	acceleration	0.009146	accelerating	0.006079
accelerating	0.009368	elevator	0.009146	sentence	0.006079
front	0.009368	problem	0.009146	previous	0.006079

listed as a stop word (whose `info-gain' is lower than the rest of the words) in our experiment. So we decided `not to' use a stop-list in our future experiments. But, still this would not solve the problem completely because the naïve bayes algorithm ignores the relationships of significant words that do not co-occur in the document. Hence, we investigated the performance of various other classifiers on this issue and decided to use k-nearest neighborhood algorithm along with the new clustering technique[6].

3 Experimental Design

In this section, we describe our new experiment, the datasets used in the experiment and the coding scheme at length.

3.1 Dataset

All of the datasets used in this work are extracted from the WHY-Essay[7] corpus that contains 1954 sentences (of essays). A list of Principles and Misconceptions that corresponds to physics concepts of interest in the Why2-Atlas project is used as the set of classes to be assigned to these essay strings. There are 50 principles and 53 misconceptions in total.

The training and test data are representative samples of responses to physics problems drawn from the same corpus. We created tag-sets for both principles and misconceptions (a total of 103) and used these to tag the data. We carried out many trials

[6] For reasons of space, the statistical results of the various other classifiers used for this purpose are not shown here.

[7] The Why-essay corpus consists of students' essay statements mostly from Spring and Fall 2002 experiments of human-human tutoring sessions.

of classification and the performance on `old data' was used to do data-cleaning and to revise the relations between classes that we want to identify/predict. Due to scarcity of quality-data of essays containing misconceptions, we had to write some student-like statements in order to expand the corpus of training data for misconceptions. This required human expertise and a good understanding of the subject matter.

3.2. Creation of Clusters

The Principles and Misconceptions used for tagging the essay segments have similar topics (e.g. gravity-freefall and gravitational force, second law etc) and therefore share common words. The classification task is typically hard because of lack of unique terms and thus increases the feature dimensionality of these documents. Thus, it is highly desirable to reduce this space to improve the classification accuracy. The standard approach used for this kind of task is to extract a `feature subset' of single words through some kind of scoring measures (for example, using `Info-gain'). The basic idea here is to assign a score to each feature (assigned to each word that occurred in the document), sort these scores, and select a pre-defined number of the best features to form the solution feature subset (as in Latent Semantic Indexing approaches). In contrast to this standard approach, we use a method to reduce the feature dimensionality by grouping "similar" words belonging to specific concepts into a smaller number of `word-clusters' and viewing these `clusters' as features. Thus, we reduce the number of features from `hundreds' to `tens'.

Though there have been many studies (for example, [13]) that use word-clusters to improve the accuracy of unsupervised document classification, there are very few studies that have used this kind of indirect `supervised' clustering techniques for text classification. Baker and McCallum [14] showed that word-clustering reduced the feature dimensionality with a small change in classification performance. Slonim and Tishby [4] use an information-bottleneck method to find word-clusters that preserve the information about the document categories and use these clusters as features for classification. They claim that their method showed 18% improvement over the performance of using words directly (given a small training set). Our work is unique in that it uses a three-tier word-clustering method to label each student essay statement. We endorse the same claims as the other two works, that word-clustering even when done on classes instead of directly on the data improves the classification performance significantly.

3.2.1 The Three-Tier Clustering Method

Determining the `similarity' of words in these physics documents is a difficult task. Given the list of the principles and misconceptions used for tagging the students' essay strings, we examined the semantics of the descriptions of each principle and misconception and extracted those words (word clusters) that seemed to best describe a particular concept and put them together. Fig. 2 illustrates this idea.

Thus, we have a three-tier tagging schemata that we built by hand in a bottom-up fashion:

cluster, sub-cluster and class

The upper levels (cluster and sub-cluster) describe the topic of discussion and the lower level describes the specific principle or misconception. The + sign in each node means the presence of that particular `word(s)' in a concept description. For example,

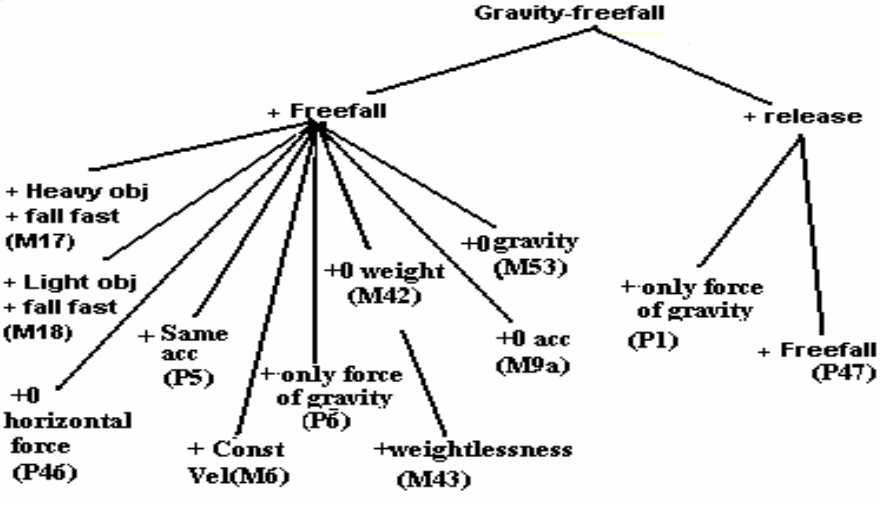


Fig. 2. Chart showing the features related to the cluster `Gravity-Freefall'

from the trees in Fig 2, we can see that **+freefall** and **+only force of gravity** describe Principle `P6' while **+freefall** and **+0gravity** describe a Misconception `M53'. Thus, words in the lower level that are shared across concepts migrate into an upper tier. The top-most level was created using the concepts described at the middle level. We created ten such clusters based on the prominent keywords for the training data (see Table 4 for specifics[8]).

This information was used to extend the original corpus annotations[9] so that the training data took the form (mapping each D to C):

C = {(clustername, subclustername, class)

as exemplified below:

Freefall acceleration is the same for all objects and the keys the person and the elevator are all accelerating downwards with the same acceleration. {*gravity-freefall, freefall-prin, prin-freefall-same-accel*}.

If the two forces were equal they would cancel each other out because they are in opposite directions. {*3rdLaw, act-react, misc-act-react-cancel*}

In addition, there was also a `nothing' class. The student statements that were neither a `P' nor a `M' are in this class.

[8] Absence of sub-clusters in some groups means that there was no ambiguity between the principles and misconceptions in that cluster. The numbers found under the third column indicate the number of principles and misconceptions that fall under the respective sub-cluster.

[9] Annotations were for the principles, misconceptions and `nothing'.

Table 4. The three-tier clusters of principles and misconceptions

Cluster	Subcluster	Classes	
		P	M
Gravity-freefall	Freefall	3	7
	Release	2	0
Gravita-tional-force	-	3	11
Secondlaw	Netforce	3	1
	Force	2	8
Thirdlaw	One-object	1	0
	2obj	4	1
	Act-react	0	5
Kinematics and vectors	Force	2	1
	Zero- netforce	4	0
One-object-second-third-law	Lightobj	0	4
	heavyobj	0	2
	objhit	0	1
Two-objects-motion	Samevel	7	2
	cons.vel- over-t	1	0
	jointmotion	3	0
Acceleration-velocity-displacement	-	4	1
Weight-mass	-	0	4
General	-	5	5

3.3 Document Modeling

Our main interest is to prove that `BOW approach with clusters' outperforms `BOW approach without clusters' on students' essay strings. Additionally, we are concerned with how this comparison is affected by the size and the nature of the training set. In this section, we discuss the various stages of our `document modeling'.

Document Indexing

Texts cannot be directly interpreted by a classifier or by a classifier-building algorithm. Therefore, it is necessary to have an indexing procedure that maps a text (dj) into a compact representation of its content. This should be uniformly applied to training, validation, and test documents. We used the bag-of-words representation to index our documents with binary weights (1 denoting presence and 0 absence of the term in the document). A document for us is a whole proposition and not a general topic (commonly used in most BOW approaches to classify web pages).

The `k- Nearest Neighborhood (kNN) Classifier
We used a simple k-Nearest Neighborhood (kNN) algorithm[10,] which is an instance based learning approach for the classification task. Fix and Hodges defined a metric to measure "closeness" between any two points and formulated a kNN rule: `Given new point x and a training set of classified points, compute the kNN to x in the training data. Classify x as Class y if more k-nearest neighbors are in class y than any other class' [15]. In the context of TC, the same rule is applied where documents are represented as `points' (vectors with term weights). We used the Euclidean distance formula to compute the measure of "nearness'.

Procedure
kNN was used for the three-tier clustering model that included the following stages:

1. Modeling the dataset (X) at the cluster level,
2. Dividing the dataset (X) into sub-datasets (Y) for sub-clusters, and bifurcating them into two (one for principles and another for misconceptions)
3. Modeling the sub-datasets (Y) at the subcluster level
4. Dividing the dataset (X) into sub-datasets (Z) for the third level (classes),
5. Modeling the subdatasets (Z) at the class level.

The classification outputs at each level were the cluster, subcluster and class tags respectively. At runtime, the output of a level is used to select a model in the next level.

Cross-Validation
We used the 2/3 and 1/3 split of training and test data for this experiment. We set the value of `k' to 1 in kNN and evaluated kNN on the test set. A stratified ten-fold cross-validation was repeated over 10 trials to get a reliable error estimate.

4 Results and Discussions

The metrics used to measure the performance of the learner are: accuracy, standard error, and precision and recall. In order to define precision and recall, we need to define `true positives, false positives, and false negatives. In our context, if a document D is related to C, it will be considered to be a `True Positive (TP)', with a value of `1'. If a document D is not related to C, it will have a value of `0' and can either be marked as `nothing' which constitutes the `False Negatives (FN)' for us or it can be misclassified (as some other C) which means that it is a `False Positive (FP)'. For example, if a student string `Freefall means without gravity, is correctly classified as misconception statement (M53), it is a TP. On the other hand, if it is categorized as `nothing' then it is a `FN' and if it is misclassified as anything else then it is `FP'. Precision and Recall can thus be defined as:

[10] We used the kNN algorithm from the RAINBOW software devised by McCallum (1996). McCallum, Andrew Kachites. "Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering", www.cs.cmu.edu/~mccallum/bow.

Precision = TP/ (TP + FP),
Recall = TP/ (TP + FN).

Using this formula, we computed the recall and precision of the bow-approach with three-tier clusters (see Table 5):

Table 5. Precision and Recall results[11] for the three-tier model

Model		Precision (%)	Recall (%)
Three-tier clustering	Cluster(one level)	80.88	92.13
	Subcluster(two levels)	74.25	88.75
	Classes (three levels)	62.58	90.75
Without clustering (using NB)		68.59	83.41

The accuracy and standard error of prediction at each level of clustering are shown in Table 6 below along with the statistics of the bow-only approach using naïve bayes classifier without clustering:

Table 6. Accuracy and Standard Error of the three-tier model

Model		Accuracy	Std. Error
Three-tier clustering	Cluster (one level)	78.01	0.016
	Subcluster(two levels)	74.50	0.020
	Classes(three levels)	64.16	0.185
Without clustering (using NB)		50.99	0.019

The above results show that the three-tier clustering indeed helped to improve the performance of the classification. Ambiguity (or noise) among classes was significantly reduced as the documents were forced to traverse the whole path (cluster → subclusters → classes). Our model significantly outperformed the bow-only approach using the naïve bayes classifier (27.02%, 23.51% and 13.17% of improvement in the classification accuracy for the levels 1, 2 and 3 respectively).

5 Conclusions and Future Directions

This paper discussed a three-tier clustering approach of classifying data pertaining to students' essay statements of qualitative physics problems in a tutoring system. We claim that `supervised three-tier clustering' outperforms the non-clustering models related to this domain. We conjecture that expansion of the training corpus for more examples for misconceptions will further improve the clustering results and thereby aid us in effective evaluation of the students' essays.

[11] The measures at each level use the output of the previous level regardless of correctness.

Acknowledgements. This work was funded by NSF grant 9720359 and ONR grant N00014-00-1-0600.

References

1. Maron, M. Automatic indexing: an experimental inquiry. *Journal of the Association for Computing Machinery* (1961) Vol. 8(3): 404–417.
2. Duda, R., and Hart, P. Pattern Classification and Scene Analysis. John Wiley & Sons. (1973) 95-99.
3. Sebastiani, Fabrizio. Machine learning in automated text categorization. *ACM Computing Surveys,* (2002) Vol.34(1):1–47.
4. Slonim, N. and Tishby, N. The power of word clusters for text classification. In *Proceedings of ECIR-01, 23rd European Colloquium on Information Retrieval Research*, Darmstadt, Germany, (2001).
5. Slonim. N. N. Friedman, and N. Tishby. Unsupervised document classification using sequential information maximization. *Proceedings of SIGIR'02, 25th ACM international Conference on Research and Development of Information Retireval,* Tampere, Finland, ACM Press, New York (2002).
6. El-Yaniv, R and Oren Souroujon. Iterative Double Clustering for Unsupervised and Semi-supervised Learning. European Conference on Machine Learning (ECML) (2001) 121-132.
7. Periera, F, N. Tishby, and L. Lee. Distributional clustering of English words. In *31st Annual Meeting of the ACL*, (1993) 183-190.
8. VanLehn, Kurt, Pamela Jordan, Carolyn Rose´, Dumisizwe Bhembe, Michael Bottner, Andy Gaydos, Maxim Makatchev, Umarani Pappuswamy, Michael Ringenberg, Antonio Roque, Stephanie Siler, and Ramesh Srivastava. The architecture of Why2-Atlas: A coach for qualitative physics essay writing. In *Proceedings of Intelligent Tutoring Systems Conference, volume 2363 of LNCS*, Springer. (2002) 158–167.
9. Chi, Michelene, Nicholas de Leeuw, Mei-Hung Chiu, and Christian LaVancher. Eliciting self explanations improves understanding. *Cognitive Science*, (1994) Vol. 18:439–477.
10. Slotta, James, Michelene T.H. Chi, and Elana Joram. Assessing students' misclassifications of physics concepts: An ontological basis for conceptual change. *Cognition and Instruction*, (1995) Vol. 13(3):373–400.
11. Graesser, A. C.; Wiemer-Hastings, P.; Wiemer-Hastings, K.; Harter, D.; Person, N.; and the Tutoring Research Group. Using Latent Semantic Analysis to Evaluate the Contributions of Students in AUTOTUTOR. *Interactive Learning Environments* (2000) Vol. 8:129–148.
12. Rosé C,P, A. Roque, D. Bhembe, K. VanLehn. A Hybrid Text Classification Approach for Analysis of Student Essays, Proceedings of the Human Language Technology conference/ North American chapter of the Association for Computational Linguistics annual meeting. *Workshop on Educational Applications of Natural Language Processing*. (2003).
13. Hotho, Andreas, Steffen Staab and Gerd Stumme. Text Clustering Based on Background Knowledge. Institute of Applied Informatics and Formal Description Methods AIFB, *Technical Report No. 425.* (2003).
14. Baker, L.D. and A. K. McCallum. Distributional Clustering of Words for Text Classification. In ACM SIGIR 98. (1998).
15. Fix, E. and J.L. Hodges. Discriminatory Analysis -- Nonparametric Discrimination: Consistency Properties, Project 21--49--004, Report No. 4, USAF School of Aviation Medicine, Randolf Field, TX (1951) 261--279.

Unsupervised Text Classification Using Kohonen's Self Organizing Network

Nirmalya Chowdhury and Diganta Saha

Department of Computer Science and Engineering,
Jadavpur University, Kolkata – 700 032, India
nir63@vsnl.net

Abstract. A text classification method using Kohonen's Self Organizing Network is presented here. The proposed method can classify a set of text documents into a number of classes depending on their contents where the number of such classes is not known a priori. Text documents from various faculties of games are considered for experimentation. The method is found to provide satisfactory results for large size of data.

1 Introduction

Text classification research and practice [2] have exploded in the past decade. There are different methods of text classification such as methods based on ontologies and key words [3] or machine learning techniques [4]. We present here an unsupervised text classification technique that uses a special type of neural network called Kohonen's Self Organizing Network. The novelty of the method is that it automatically detects the number of classes present in the given set of text documents and then it places each document in its appropriate class. The method initially uses the Kohonen's Self Organizing Network to explore the location of possible groups in the feature space. Then it checks whether some of these groups can be merged on the basis of a suitable threshold to result in desirable clustering. These clusters represent the various groups or classes of texts present in the set of given text documents. Then these groups are labeled on the basis of frequency of the class titles found in the documents of each group. The proposed method needs no *a priori* knowledge about the number of classes present in a given set of text documents.

The next section presents the steps involved in the formation of the pattern vector for each document for clustering followed by the statement of the clustering problem.

2 Statement of the Problem

Given a set of text documents, the steps adopted to extract the features and form the pattern vectors for all the documents in the given set of text documents are as follows.

Step 1: Remove all stop words such as 'the', 'a', 'an' etc., and also all functional words such as adverbs, preposition, conjunction etc, from the text of all the documents in the given set.

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 715 – 718, 2005.

Step 2: Remove the words that have the value W below a threshold 0.9. W is an elimination factor that is calculated as follows.

W_{EF} = Number of occurrence in its own context / Total number of occurrences in all contexts

We chose the value of ***W*** empirically. Thus, the words that has W below the threshold 0.9 do not participate in evaluation

Step 3: Choose the remaining words as the features for document classification.

Step 4: Create one pattern vector for each document with the features selected in step 3. The numeric value for each component of such vector would be the number of occurrence for the particular word corresponding to that component in the given document.

Note: Step 2 eliminates all those words that have almost no discriminatory significance so far classification is concerned. For instance, the word "play" can be expected to occur frequently in all the documents of various faculties of games. Thus it will have a relatively lower value of W_{EF} and accordingly it will not be considered as a feature to form the pattern vector.

Clustering is an unsupervised technique used in discovering inherent structure present in the set of objects [1]. Let the set of patterns be $S = \{x_1, x_2, ..., x_n\} \subseteq \Re^m$, where x_i is the *i*-th pattern vector corresponding to *i*-th document, *n* is the total number of documents in a given set of text documents and *m* is the dimensionality of the feature space. Note that the value of *m* for a given set of texts is determined in Step 3 as stated above. Let the number of clusters be *K*. The value of *K* may or may not be known *a priori*. In the present work, the value of *K* is computed automatically by the proposed method. If the clusters are represented by C_1, C_2, ..., C_K then we assume:

1. $C_i \neq \varnothing$ for $i = 1, 2, ..., K$
2. $C_i \cap C_j = \varnothing$ for $i \neq j$ and
3. $\cup_{i=1}^{K} C_i = S$ where $\varnothing$ represents null set.

The next section describes the proposed method which uses Kohonen's Self Organizing Network to detect the groups present in a given set of text documents. Then the groups are labeled on the basis of frequency of the class titles found in the documents of each such group.

3 The Proposed Method

We have used Kohonen's self-organizing network with *m* input nodes (since *m* is the number of features in the pattern vector) and 16 output ($p = 16$) nodes being arranged in a two-dimensional 4 x 4 grid. After the algorithm (to produce self organizing feature map) has converged, the locations of weight vectors in feature space almost correspond to mean vectors of *p* possible groups of the given data. In other words, after convergence, each node of the Kohonen's self organizing network represents a local best representative (seed) point associated with a particular high density region of the feature space.

The number of seed points p and their location in feature space are obtained by using Kohonen's self-organizing network. Then the data set is divided into p groups using the standard minimum squared Euclidean distance classifier concept. Later the groups are merged using a threshold h_n on the minimum interpoint distance between the groups. Here the threshold h_n for the cluster separation is taken to be equal to $h_n = l_n / n$, where l_n is the sum of the edge weights (edge weight is taken to be the Euclidean distance) of minimal spanning tree of S. The process of merging decreases the number of groups in the data set. The process terminates when no further merging is required and we get the desired clustering.

Then we compute the frequency of the possible class titles such as cricket, football, chess, athletics etc. in each of the groups obtained by the above algorithms. Then each group is labeled a specific class title which has the largest frequency in that particular group. For instance, a group is labeled as "cricket" if the number of occurrence of the word "cricket" in all the documents of that group is the largest than that found for other groups. Note that text classification should not be done on the basis of the frequency of class titles only, because a document of a specific class may not contain its class title at all. However, when we have a group of documents of a specific class, the group can be labeled on the basis of largest frequency of all the possible class titles.

4 Experimental Results and Conclusion

We have considered text documents from various faculties of games and sports such as cricket, football, hockey, basketball, swimming, lawn tennis, chess and atheletics. We have used three sets of data for experimentation. Each data set consists of a number of documents or articles published in three leading English daily newspaper of our country, namely *The Telegraph*, *The Times of India* and *The Statesman*. All these articles belong to any one of the said eight faculties of games. Each data set is constructed with unequal number of documents from different classes. This is done to incorporate variability in the size of the clusters. The total number of articles for each of the three data sets is also taken in the increasing order so that we can detect the effect of the size of the data on the performance of the proposed method.

Experiment 1: The data consists with 400 articles, where 115, 110, 95 and 80 articles are taken from the classes cricket, football, basketball and hockey athletics respectively. Here the proposed method has provided a success rate of 94.2 %.

Experiment 2: Here 500 news articles, out of which 128, 110, 90,98 and 74 articles are taken from cricket, football, chess, athletics and swimming respectively. In this case, the success rate provided by the proposed method is 96.8 %.

Experiment 3: The data consists with 600 articles where 132, 130, 82, 72, 94 and 90 articles are taken from cricket, football, lawn tennis, hockey, basketball and athletics respectively. The success rate of the proposed method in this experiment is 98.3%.

It is observed that the performance of the proposed method improves as the size of the given data set increases. It seems that the mentioned method would be able to provide a success rate of nearly 100 % when the size of the data set is very large.

Table 1. Results of experiments by the proposed method

Expt. No.	Size of data	No. of groups in the data	Performance of the Proposed Method		
			No. of groups detected by the proposed method	No. of correct classification	% of correct classification
1	400	4	4	377	94.2
2	500	5	5	484	96.8
3	600	6	6	590	98.3

References

1. M. R. Anderberg, *Cluster Analysis for Application*, Academic Press, Inc, New York, 1973.
2. Michael W. Berry, *Survey of Text Mining: Clustering, Classification, and Retrieval*, Amazon, 2003.
3. A. Gelbukh, G. Sidorov, A. Guzman-Arenas. Use of a weighted topic hierarchy for text retrieval and classification. *Lecture Notes in Artificial Intelligence*, N 1692, Springer, 1999.
4. P. G. J. Lisboa, *Neural Networks: Current applications* . Chapman and Hall, London, 1992.

Enhancement of DTP Feature Selection Method for Text Categorization*

Edgar Moyotl-Hernández and Héctor Jiménez-Salazar

Facultad de Ciencias de la Computación,
B. Universidad Autónoma de Puebla
emoyotl@mail.cs.buap.mx, hjimenez@fcfm.buap.mx

Abstract. This paper studies the structure of vectors obtained by using term selection methods in high-dimensional text collection. We found that the *distance to transition point* (DTP) method omits commonly occurring terms, which are poor discriminators between documents, but which convey important information about a collection. Experimental results obtained on the Reuters-21578 collection with the k-NN classifier show that feature selection by DTP combined with common terms outperforms slightly simple *document frequency*.

1 Introduction

The goal of *text categorization* (TC) is to classify documents into a set of predefined categories. In TC each document is usually represented as a vector of terms in a multidimensional space, in which each dimension in the space corresponds to a term. Typically even a moderately sized collection of text has tens or hundreds of thousands of terms. Hence, the document vectors are high-dimensional. However, most documents contain fewer terms, 1-5% or less, in comparison to the total number of terms in the entire text collection. Thus, the document vectors are *sparse* [3].

For reasons of both efficiency and efficacy, *feature selection* (FS) techniques are used when applying machine learning algorithms to text classification. In our previous experiments [6] we found that FS using DTP achieves performance superior to *document frequency*, and comparable to *information gain* and *chi-statistic*; three well known and effective FS techniques [10]. However, the vectors produced by DTP have a "sparse" behavior that is not commonly found in low-dimensional text collections.

In this paper, our first focus is to study the structure of the vectors produced by term selection methods when applied to large document collections. Such structural insight is a key step towards our second focus, which is to explore the relationships between DTP and the problem of the sparseness. We hypothesized that supplementing it with high frequency terms would improve term selection by adding important (and also common) terms; and we report experimental results

* This work was supported by VIEP-BUAP, grant III9-04/ING/G.

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 719–722, 2005.

obtained on the standard Reuters-21578 benchmark with the k-NN classification algorithm.

The paper is organized as follows. Section 2 compares the sparseness and weighting of the vectors produced from the output of the term selection techniques: *document frequency*, *information gain*, *chi-statistic*, and DTP. Furthermore, section 2 shows that vectors obtained by DTP are sparse and that vectors obtained by combining DTP with *document frequency* are dense. Section 3 presents conclusions and future research.

2 Density and Weighting of Vectors

In this section, we empirically study the structure of the vectors produced by term selection methods. As we will see density of vectors is calculated instead of sparseness.

We used the Reuters-21578 collection which consists of 12,902 news stories classified according to 115 thematic categories. The experiments used the set of the 10 most frequent categories (R10), partitioned (according to the ModApte split) into a training set of 6,490 documents and a test set of 2,545 documents. Term weighting was done using $tfidf_{ij} = tf_{ij} * \log(N/df_i)$, where tf_{ij} is the number of times t_i occurs in a document d_j, df_i is the number of documents in which t_i occurs, and N is the number of documents [7]. Also tf_i is defined as $\sum_j tf_{ij}$. We will refer to four FS methods (see [10] for more details), which can be briefly defined as follows. Document frequency (DF) is the number of documents in which a term t_i occurs (df_i). Chi-statistic (CHI) measures the lack of independence between a term and the category. CHI computed for a term takes the maximum on all categories; CHImax. Information gain (IG) of a term measures the number of bits of information obtained for category prediction by knowing the presence or absence of the term in a document. IGsum for a term represents the expected information gain on categories. As we have said, these three FS methods are effective in the TC task [10]. DTP is based on the proximity to the frequency that splits the terms of a text into low and high frequency terms; this frequency is called the *transition point* (TP). DTP is computed by the distance from frequency (tf_i) of term (t_i) to TP. Given a text, TP is easy to calculate because it only requieres the number of words t_i with $tf_i = 1$ [1][9][5]. We refer to the above technique as the *Inverse DTP* (IDTP) rule. More important terms for the TC task are those producing the lowest DTP scores. IDTP showed a comparable performance [6] to the best term selection techniques, CHI and IG, although this fact depends on the size of text collection.

Table 1 shows density (columns 2-6) and average weighting (columns 7-11) for three percentages[1] of terms selected by DF, IGsum, CHImax, and IDTP. One more FS method was included, IDTP$_{df}$*DF, which will be discussed after Density was calculated as the ratio of the number of nonzero terms in training

[1] Since FS methods give better performance in the TC task taking 1%, 5% and 10% of highest score [10], we used such percentages in the experiments.

and test vectors to the total number of selected terms. Zipf's Law implies that more than 50% of terms have frequency 1, more than 10% have frequency 2, etc. [1]. So the more terms selected, the higher the percentage of low frequency terms. Therefore, the density of vectors with terms given by any FS method decreases as the percentage of terms grows; which can be seen in columns 2 to 5 of table 1. It must be remarked that DF has the highest density and IDTP has the lowest density. This means that selecting terms using IDTP will give us sparse vectors. Besides, because term selection by CHImax or IGsum takes into account categories intended to match the right class in the TC task, they do not depend on weighting (cols. 8 and 9 in table 1); weighting is distributed among categories. A growing tendency of average weighting of vectors is observed in DF and IDTP values (cols. 7 and 10). Since IDTP is based on the importance of terms, from the vector space model point of view, then less frequent terms are more important, i.e. medium frequency terms are the weightiest.

Table 2 summarizes microaveraged F_1 values obtained for the $k-$NN classifier (using $k = 30$) with the evaluated FS techniques for different percentages of terms. Columns 2, 3, 4 and 5 correspond to DF, IGsum, CHImax and IDTP, respectively. DF, IGsum and CHImax values for k-NN are in accordance with the findings of Debole and Sebastiani [2]. The results of IDTP imply we should reinforce term selection with frequent terms. A simple way to attain this purpose is by providing a higher score to frequent terms; for example IDTP*DF as the score. Although the IDTP*DF score is better than IDTP, this score is only comparable to DF at 10%; see columns 2, 5 and 6 of table 2. Thus, the IDTP score was reformulated considering that tf_i represents the intratext frequency, while df_i represents the intertext frequency. We define IDTP_{df} for a term t_i as the inverse distance between df_i and TP_{df}, where TP_{df} is computed as the transition point using df_i instead of tf_i. In order to select important terms, we use IDTP_{df} multiplied by DF, which raises the score of frequent terms. Results of IDTP_{df}*DF are shown in columns 6 and 11 of table 1 and in column 7 of table 2. We see that F_1 for IDTP_{df}*DF is as good as for DF. Also, values for IDTP_{df}*DF show the increase of the density of the selected terms (col. 6 of table 1), and a more stable average weighting (col. 11 of table 1).

Table 1. Term selection *vs* density and average weighting of training and test vectors

Percent	**Density**					**Weight (avg.)**				
of Terms	DF	IGsum	CHImax	IDTP	IDTP_{df}*DF	DF	IGsum	CHImax	IDTP	IDTP_{df}*DF
1	0.1	0.075	0.056	0.012	0.016	4.2	5.4	5.7	5.1	5.3
5	0.035	0.030	0.021	0.009	0.036	5.5	5.5	5.8	7.2	5.2
10	0.021	0.020	0.017	0.006	0.021	5.8	5.7	5.6	8.0	5.8

Table 2. Microaveraged F_1 for several FS criteria using $k-$NN on R10

% terms	DF	IGsum	CHImax	IDTP	IDTP*DF	IDTP_{df}*DF
1	0.826	0.855	0.855	0.302	0.314	0.392
5	**0.851**	**0.860**	**0.863**	0.499	0.738	0.851
10	0.850	0.853	0.859	**0.545**	**0.845**	**0.853**

3 Conclusions

A feature selection method based on IDTP was proposed. It was motivated by remarks about density and weighting of vectors built with terms near the transition point. This feature selection method multiply IDTP by DF and thus improves on DF, one of the most effective term selection methods in TC tasks.

An advantage of IDTP is the low computational cost compared with top feature selection methods (CHI or IG). However, there are several point pending such as testing with other high-dimensional text collections, applying criteria for selecting terms which take the category into account, and to experiment with differents ways to use TP.

Acknowledgements. We would like to thank James Fidelholtz by useful comments on this work.

References

1. Booth, A.: A Law of Occurrences for Words of Low Frequency, Information and Control, (1967) 10(4) 386-93.
2. Debole, F. and Sebastiani, F.: An Analysis of the Relative Difficulty of Reuters-21578 Subsets, In Proceedings of LREC-04, 4th International Conference on Language Resources and Evaluation, Lisbon, PT, pp. 971-974.
3. Dhillon, I. S., Modha, D. S.: Concept Decompositions for Large Sparse Text Data using Clustering. Mach. Learn., Kluwer Academic Publishers, (2001) 42(1-2) 143-175.
4. Galavotti, L., Sebastiani, F., Simi, M.: Experiments on the Use of Feature Selection and Negative Evidence in Automated Text Categorization, Proc. of ECDL-00, 4th European Conference on Research and Advanced Technology for Digital Libraries, (2000) 59-68.
5. Moyotl, E., Jiménez, H.: An Analysis on Frequency of Terms for Text Categorization, Proc. of SEPLN-04, (2004).
6. Moyotl, E., Jiménez, H.: Experiments in Text Categorization using Term Selection by Distance to Transition Point, Proc. of CIC-04, (2004).
7. Salton, G., Wong, A., Yang, C.: A Vector Space Model for Automatic Indexing, Communi-cations of the ACM, (1975) 18(11) 613-620.
8. Sebastiani, F.: Machine Learning in Automated Text Categorization, ACM Computing Surveys, Vol. 34(1), (2002) 1-47.
9. Urbizagástegui-Alvarado, R.: Las posibilidades de la ley de Zipf en la indización automática, Reporte de la Universidad de California Riverside, (1999).
10. Yang, Y., Pedersen, P.: A Comparative Study on Feature Selection in Text Categorization, Proc. of ICML-97, 14th Int. Conf. on Machine Learning, (1997) 412-420.
11. Yang, Y., Liu, X.: A Re-examination of Text Categorization Methods, Proc. of SIGIR-99, 22nd ACM Int. Conf. on Research and Development in Information Retrieval, (1999) 42-49.
12. Zipf, G.K.: Human Behaviour and the Principle of Least Effort, Addison-Wesley, (1949).

FASiL Adaptive Email Categorization System

Yunqing Xia, Angelo Dalli, Yorick Wilks, and Louise Guthrie

NLP Research Group, Department of Computer Science, University of Sheffield
Regent court, 211 Portobello Street, Sheffield, S10 4DP
{y.xia, a.dalli, y.wilks, l.guthrie}@dcs.shef.ac.uk
http://nlp.shef.ac.uk/

Abstract. This paper presents an adaptive email categorization method developed for the Active Information Management component of the EU FASiL project. The categorization strategy seeks to categorize new emails by learning user preferences, with a feature-balancing algorithm that improves the data training effectiveness and with a dynamic scheduling strategy that achieves the system adaptivity. The results of our evaluation with user-centric corpora constructed automatically from email servers are presented, with around 90% precision consistently being achieved after three months of use. Adaptivity of the system is also evaluated by studying system performance within the continuous three months.

1 Introduction

Email is one of the most ubiquitous applications used on a daily basis by millions of people world-wide. Typically, emails are stored in different folders for easy access, imposing some structure on the increasingly unmanageable amount of information received by email. Our work is focused on creating better ways of categorizing email automatically in a way that adapts to the changing needs of a user.

This work has been done as part of the EU FASiL project, which aims to construct a conversationally intelligent Virtual Personal Assistant (VPA) designed to manage the user's personal and business information through a voice-based interface accessible over mobile phones. Mobile phones have achieved high penetration rates in most major EU states and around the world, with 73% of the EU population using a mobile phone, compared to 59% in Japan and 46% in the USA and 16% average worldwide(David, 2003). As the quality of life and productivity of EU citizens is to be maintained and improved in an increasingly information dominated society, people need access to information anywhere and at any time. In order to provide this capability, two fundamental issues must be addressed: information access and information overload. When trying to access email, especially over a voice-based interface, an ideal system should categorize and present timely messages in a prioritized manner.

Empirical studies show that an active email user will create between 10 to 130 subfolders to file their emails, with a value of around 73 sub-folders being typical(Fisher and Moody, 2001). Other studies show that as number of sub-folders increases, users in fact feel more in control of the situation, but more effort is required to get through their emails (Moody, 2003).

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 723 – 734, 2005.

Receiving email through a mobile phone presents several additional problems since unlike desktop or laptop computers or PDAs mobile phones have rather small screens that make it difficult to use visual cues to manage a complex folder-based system.

Voice-based systems also constrain users to access emails sequentially. Efficiency and adaptivity therefore become the two most critical features in active information management for mobile users. We envision a scenario where users are encouraged to utter requests to read new emails about a specific topic or class rather than all new emails, enabling them to focus on relevant classes quickly. A one-size-fits-all approach to enhancing the user experience cannot be effective in general since different users have different preferences and priorities in their email management.

The Active Information Management (AIM) service in the FASiL VPA seeks to prioritize and present information that is most pertinent to the mobile users and automatically adapt to different user preferences. The AIM Email Categorizer (AIMEC) assigns the most pertinent sub-folder for new emails automatically through content-based techniques.

AIMEC is designed to collect training data automatically by continually evaluating the user's email and behavior. A term-set discrimination and balancing algorithm is used to refine user preference statistics, coupled with a dynamic training scheduling strategy that will enable the new adaptive information to be utilized by the main system.

No generic manually trained corpora are needed for AIMEC. Instead, user-specific corpora are constructed automatically by retrieving email text from sub-folders while inferring possible initial user preferences for each sub-folder.

2 Related Work

Various approaches towards text and email classification have been developed in the past few years, ranging from traditional document classification algorithms to newer spam-detection techniques (Smadja and Tumblin, 2003). The most popular approaches to text classification are decision trees like the popular C4.5 algorithm (Quinlan, 1993), Naïve Bayesian and related Bayesian learning methods (Lewis, 1998; McCallum and Nigan, 1998), clustering techniques such as the k-nearest neighbor algorithm (Androutsopoulos et al., 2000) and boosting (Carrerras et al., 2001), support vector machines (Thorsten, 2001), neural networks (Wiener et al., 1995), and statistical learning and induction methods(Yang, 1999).

Several document classification methods have already been applied to email categorization with mixed success (Cohen, 1996; Payne & Edwards, 1997) and anti-spam filtering by classifying emails into folders (Androutsopoulos, 2000). In most of these methods data training depends on the availability of a manually categorized corpus used as a gold standard, with the training results being used for the rest of the categorization processes. This lead to over-general training results, having minimal effect on improving email categorization for users. Text classification algorithm in AIMEC is based on the statistical learning method (Guthrie et al., 1994), incorporating user preference learning method to tackle the data training problem.

In (Guthrie et al., 1994) the training documents, say $\{TD_i\}$ where $1 \leq i \geq m$, are assigned a particular class $\{C_i\}$. Thus number of classes defined in the user email ac-

count, m, is automatically inferred using the user's sub-folder structure as guidance. The document content is used by the training algorithm to construct disjoint word sets $\{WS_i\}$ for every class. The training algorithm also creates a word frequency vector $(p_{i1}, p_{i2}, \ldots p_{im})$ mapped to class C_i, where p_{ij} denotes word frequency of WS_i in TD_j.

Given a document on one of the possible classes, the document is classified by counting word frequencies and calculating probabilities that the document belongs to all classes. The most probable class is assigned to this document.

The method is simple but efficient. According to the experiment described in (Guthrie et al., 1994), 700 documents containing over 200 words each can provably be classified correctly around 99.9% of the time.

3 User Preference Learning

In AIMEC the data training problem is overcome by extracting training data automatically from user's existing sub-folders. The basic algorithm was enhanced to cater for more sophisticated training techniques that incorporate syntactic features of the document which are automatically integrated with a data collecting mechanism that retrieves user emails on a dynamically scheduled basis. Feature discrimination and balancing algorithms together with the dynamic training scheduling strategy ensure adequate short term and long term adaptivity of the categorizer.

Adaptivity in AIMEC is mainly achieved by combining a conventional content-driven classification approach together with automated inferences about user preferences that aim to make intelligent guesswork about how a particular user wants their email presented and classified.

One of our main assumptions is that the way that emails are stored in different sub-folders mirrors the way that users want their emails to be classified. Additionally, this method can be adapted to cater for content-driven virtual folder creation techniques such as email systems that allow sub-folders to be created dynamically depending upon the results of particular search queries.

3.1 User-Centric Data Collecting

Training documents are created for every sub-folder in every user's email store, and the name of the sub-folder itself is tentatively assigned to this document as the class. When the data-collection process is over the training document vector $\{TD_i\}$ and the class vector $\{C_i\}$ are constructed and mapped automatically.

Email is a structured document defined by Internet RFC2822(Resnick, 2001, the original document where the now ubiquitous email is defined) where many fields are not directly related to the email content. In data collection process only fields relevant to the content are considered such as "SUBJECT", "FROM" and "CC" and "BODY". Our experiments show that text within fields of "SUBJECT", "FROM" and "CC" is much more important in categorization modeling than that within "BODY" field. We improved weight for text within fields of "SUBJECT", "FROM" and "CC", i.e. 3 in our case, and kept the weigh for text within "BODY" field, i.e. 1, and surprisingly, the average performance is improved by around 20 percent.

Many emails are composed by replying to an original email, often including part or whole of the original email together with new content, thus creating a thread or chain of emails. So the first email in the thread could be repeated many times over, which misleads the data-training algorithm. A thread-detection filtering tool is used to eliminate unoriginal content in the email. Stylistic features like the presence of a greater than symbol at the beginning of each line, or the presence of the old email header are also used to determine if a particular section of the email should be filtered out or not.

The documents used for training indirectly reflect the user's current preferences for viewing and organizing their emails. This strategy basically guarantees that user preference can be learned in the data-training algorithm and therefore achieve some tangible measure of user adaptivity. The categorization model is updated by utilizing the new email data to regenerate the feature sets associated with every user's folders, while performing user preference learning every time training occurs.

3.2 Feature Set Construction

The training algorithm tries to extract user preference from a set of user created subfolders. User preference is represented by feature sets and a group of preference parameters, which are assigned to every class after the training process concludes.

The basic elements of feature are term and frequency. Term consist of distinguishing word or phrase that appears in a training document. A feature is thus a term-frequency pair. There are many term-frequency pairs in any training document, but not all term-frequency pairs are selected as features. For a term-frequency pair to be chosen as a feature for a particular class C_i, it must have a significantly bigger frequency in the class C_i than it has in the other classes $\{C_j\}$ where j is different from i. Secondly, every feature is chosen to be unique to a particular class. The feature sets for different training documents (in our case corresponding to classes) thus forms a disjoint set. The process of feature sets construction is presented in Figure 1.

AIMEC constructs the feature set by first retrieving term-frequency pairs from training documents. POS classification methods in GATE(Cunningham et al., 1996) are then applied to identify common nouns and verbs in the training documents.

Stoplist represent words that are not discriminative of message contents, such as prepositions, pronouns and conjunctions. These words are automatically filtered out of the training set at this stage. The stoplist used for English is based on the stoplist built for the SMART information retrieval system at Cornell University(Buckley et al., 1994). Besides, user-specific stoplist is automatically constructed in each training process by retrieving the first 10 terms that have the biggest frequency in the whole user corpus. User-specific stoplist varies from user to user so that the categorization model can be constructed in a user-specific manner.

User-specific stoplist give better results than generic stoplist since the generated stoplist will be specifically optimized for the user's particular collection of emails.

Non-stoplist terms are then sorted according to frequency and the terms are pruned by removing the most frequent terms and least frequent terms in the set. This is in accordance with the implication of Zipf's law – that the most frequent and the least frequent terms are usually not significant(Rijsbergen, 1979). After the term pruning process, the original feature sets are constructed.

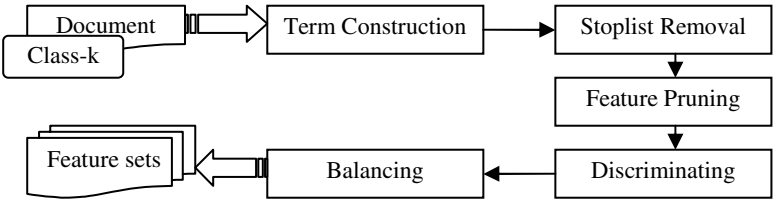


Fig. 1. Feature set construction

User-specific stoplist give better results than generic stoplist since the generated stoplist will be specifically optimized for the user's particular collection of emails.

Non-stoplist terms are then sorted according to frequency and the terms are pruned by removing the most frequent terms and least frequent terms in the set. This is in accordance with the implication of Zipf's law – that the most frequent and the least frequent terms are usually not significant(Rijsbergen, 1979). After the term pruning process, the original feature sets are constructed.

Since user's preference in email management is partly reflected by feature sets, feature sets are designed to be distinguished from each other in our system. Features are discriminated with the following rule.

```
a) IF term in one feature appears in only one class,
   THEN it is kept for this class;
b) ELSE IF the term appears in more than one class, and
c) IF the term frequency in one class is around N times
   bigger than those of the other classes (we used N to
   be 20),
d) THEN it is kept in this class, and deleted from all
   the other classes;
e) ELSE it is deleted from all these classes.
```

After the feature discriminating process, feature sets become disjoint between each other, and each of them is mapped to one class uniquely.

3.3 Feature Balancing

Feature numbers in different sub-folders vary from 812 to 1804 in our experiments. The significant variance among the number of features in different classes may lead to errors in the email categorization process.

As an example, suppose we encounter an email with class PERSONAL (manually assigned). AIMEC extracts 11 features with class CORPORA and 8 features with class PERSONAL from this email. According to the categorization theory, the new email is wrongly determined to be in the CORPORA class. After manually analyzing the two feature sets, it is evident that the mistake is caused by the significant difference amongst numbers of features within different sub-folders.

This error is counter-balanced through our feature balancing technique. The two feature sets are balanced by removing the least frequent features (in this case 586

features) from class CORPORA, making the number of features in both CORPORA and PERSONAL classes on an equal footing, and re-running the categorization process again. After balancing, only 4 features with class CORPORA are found in the new email, and the email is thus correctly categorized. In our experiments we applied the feature-balancing algorithm to all feature sets and repeated the experiments again. The experimental results verify that precision can be significantly improved by applying the feature-balancing algorithm.

The rationale for feature balancing can be proven in a more theoretic manner. Let us consider classes i and k. Suppose that the feature number of feature set i is much bigger that that of feature set k. According to the categorization theory described in section 2, p_{ij} will be much bigger than p_{kj} and *Prob(i)* will be bigger than *Prob(k)* if n_k is not much bigger than n_i. In this case, wrong decision could be easily drawn by the categorizer.

The feature-balancing algorithm in AIMEC tries to overcome this problem by pruning features from those classes whose feature numbers are more than twenty percent bigger than that of the class with the minimal number of features. The feature-balancing algorithm is as simple as comparing every two feature sets, which is described using pseudo-code as below.

```
a) FOR every i-th feature set, FS_i, i=1, 2, …, m
b)   FOR every FS_j, j=1, 2, …, m; i•j
c)     IF Num(FS_i)>(1+20%)*Num(FS_j),
d)       Prune FS_i from bottom
e)     ELSE IF Num(FS_j)>(1+20%)*Num(FS_i),
f)       Prune FS_j from bottom
g)     ELSE, do nothing
h)   NEXT FS_j
i) NEXT FS_i
```

3.4 Preference Parameter Evaluation

Feature sets for different sub-folders reveal the feature distribution in sub-folders. The probabilistic distribution of features needs to be evaluated during data-training process to calculate the probability that the new email belongs to one class.

The probabilistic distribution of the features across the classes can be represented as a vector. In AIMEC, the preference parameter is represented by the probability vector $\{p_{i1}, p_{i2} \ldots p_{im}\}$. On class C_i, the probability p_{ij} of terms in i-th feature set is determined by term counting. The probability vector $\{p_{i1}, p_{i2} \ldots p_{im}\}$ varies with every class and $p_{i1} + p_{i2} + \ldots + p_{im} \leq 1$. The probability vector is referred to as the preference parameters. The parameters are evaluated by the following formula:

$$p_{ij} = \frac{O(i,j)}{C(j)} \tag{1}$$

In formula (1) O(i, j) denotes the number of terms in feature set i that appear in training document j, and C(j) denotes number of terms in training document j. Every probability vector is mapped to one class uniquely.

3.5 Dynamic Scheduling and Adaptivity

The AIM service will generally need to cater to three different user scenarios (and various combinations of these scenarios together). In the first scenario, AIMEC is provided to a new user, meeting with a brand new user preference. In the second scenario, AIMEC will encounter a user who has various roles in his or her organization. In the third scenario, the content of an existing sub-folder might imply a different preference as time goes by. For example, in a typical research project, email in the first few months might be more focused on research activities, then the focus shifts on to development and deployment activities, and so on, following the natural progression of events as they happen.

To adapt to the new preferences, AIMEC proposes a dynamic scheduling (DS) strategy that schedules training processes in two different ways. A fixed-time scheduled process is automatically run at a predetermined time (usually at some opportune moment when the system is not being utilized heavily during the night). The other way is a more lightweight incremental training process that is triggered when sub-folder content changes.

With the DS strategy the categorization model will be refreshed iteratively with the method described in section 3 based on the latest training corpora, in which the emails received after last training are included. Extra features within the recent emails are extracted and supplemented to the feature set belonging to a sub-folder. The preference parameters are also updated with formula (1).

The DS strategy enables AIMEC to adapt to new preferences in a timely manner without degrading user's runtime performance, which is an extremely important issue in voice based systems.

4 Email Categorization

Theoretically, the email categorization can be described as following. Suppose there are n terms found in one email text, and the n terms distribute into m classes. Let nk be the number of terms in *k*-th feature set. Thus the vector of term frequencies for this email is (n1/n, n2/n … nm/n). The categorization algorithm will identify an email to be of class C_i if its frequencies most closely resemble the frequency distribution $\{p_{i1}, p_{i2} \ldots p_{im}\}$.

We are thus given k multinomial populations, with the *i*-th population having frequencies $\{p_{i1}, p_{i2} \ldots p_{im}\}$. The *i*-th population may be envisioned to be an infinite set consisting of m types of elements, with the proportion of type j being p_{ij}. Given an email with n terms from one of the populations, the categorization process seeks to determine from which of the populations it came. The probability that the email came from population i can be calculated by the following formula(Guthrie et al., 1994).

$$\Pr ob(i) = \frac{n!}{\prod_{k=1}^{m} n_k!} \times \prod_{l=1}^{m} p_{il}{}^{n_l} \tag{2}$$

When new email is input to AIMEC, term-frequency list is first constructed. By matching feature sets of all classes terms are then grouped. We sum up term frequencies for every group, which is denoted by n_k. Finally a probability that is calculated with the above formula is assigned to every class, and the most probable class is output to user as the suggested sub-folder.

5 Experiments and Results

5.1 Experiment Setup

Experiments described in this paper aim firstly to evaluate the performance of the AIMEC, i.e. recall and precision, and secondly to evaluate the adaptivity of the AIMEC.

Ten users volunteered in AIMEC evaluation. User U1, U2 and U3 are university researchers who receive about 20 emails each day. User U4~U10 are user evaluation experts in Europe who receive about 40 emails each day. All the ten users are using Microsoft Outlook as their email manager, in which the folders are able to be created and modified to manage emails with different purposes.

The experiments were carried out using a typical snapshot respectively from users' mailbox. In our experiments, emails are retrieved from folders thus the corpora can be constructed automatically. Every email is associated to a class which is named with the name of the folder. Emails that received before 31/11/2003 were used as training corpus, and those received in December 2003 and January 2004 were used as test corpus. Number of emails that used in our experiments is showed in Table 1.

To assess the adaptivity of AIMEC, the experiments were carried out with three training corpora for each user separately. The #1 corpus contains 200 random emails received before 31/9/2003, the #2 corpus contains 200 random emails received before 31/10/2003, and the #3 corpus contains 200 random emails received before 31/11/2003. The intention of this treatment is to verify whether AIMEC is able to adapt to a new user preference and achieve a satisfying performance shortly. The #4 corpus which contains 400 random emails having a receive date within December 2003 and January 2004 is setup exclusively for test and quality evaluation. Corpora for each user are showed in Table 2.

5.2 Experimental Results

The experiments were carried out separately for different users in the following way. User ran the training program with the #1 corpus, and with the generated categorization model he then ran the categorization program over all emails within the #4 corpus and recorded the results. Thereafter, he moved forward to experiments with the #2 and #3 corpora in the same way. Recall and precision were applied in AIMEC evaluation in which Recall indicates the proportion of relevant emails categorized, and Precision indicates the proportion of categorized emails that are relevant.

Table 1. Emails Involved in the Experiments

User	# of sub-folders	# of emails over time			
		Before 31-9-03	Before 31-10-03	Before 31-11-03	12-03 & 1-04
U1	4	283	531	785	461
U2	4	313	577	831	516
U3	5	530	1012	1439	865
U4	6	475	894	1425	657
U5	6	316	563	929	475
U6	5	275	413	542	423
U7	5	325	453	568	479
U8	5	345	510	741	514
U9	5	425	947	1405	763
U10	5	342	516	727	451

Table 2. Corpus setup for each user

Corpus	Purpose	Emails Received Over Time	# of Emails
#1	Training	Before 31-09-03	200
#2	Training	Before 31-10-03	200
#3	Training	Before 31-11-03	200
#4	Testing	Dec. 2003 and Jan. 2004	400

As every email in the test corpus has been associated to a certain class explicitly by the users moving emails to difference sub-folders, we calculated Recall and Precision through automatic email counting. When the categorization process finished, a folder name was assigned to the email by the categorizer if a proper class was matched. If no proper class was matched, OTHER was assigned to the email. F-Measure(Yang, 1999) was also applied to evaluate the overall performance. Experiment results from the ten users are showed in Figure 2 and Figure 3.

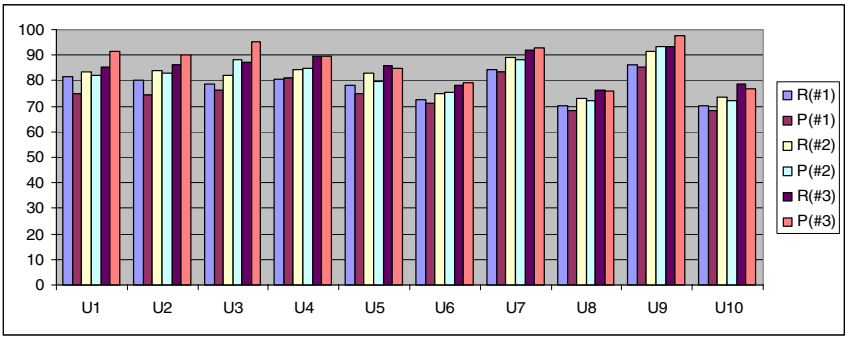


Fig. 2. Experimental results for 10 users: Recall and Precision. R(#N) refers to recall values on corpus #N, and P(#N) refers to precision values on corpus #N

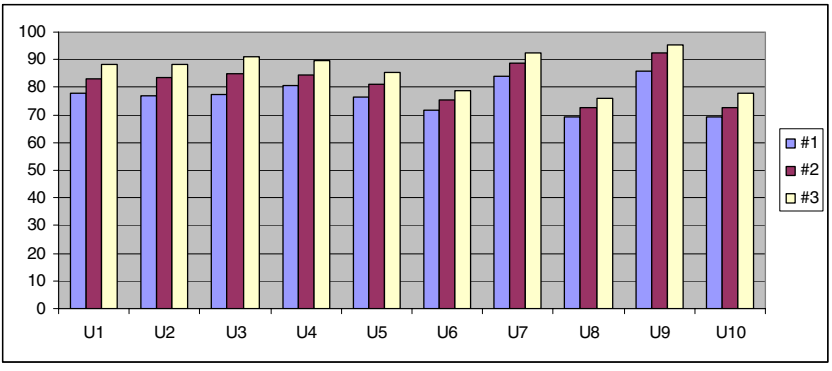


Fig. 3. Experimental results for 10 users: F-Measure values over corpus #N

5 Discussion

99.9% classification precision was achieved with 700 documents each containing over 200 words according to (Guthrie et al., 1994). However, in AIMEC we achieved a lower precision of around 90%. There are three reasons to address this. First, according to the categorization theory, the more distinguished the feature sets are, the higher performance the AIMEC is able to achieve. Experiments described in (Guthrie et al., 1994) considered two classes, i.e. TIPSTER-business and MUC-terrorist, in which two distinguished feature sets can be constructed. In our experiments, four to six classes were considered, some of which are not very distinguished due to specific user preference. Second, the training and test corpora in (Guthrie et al., 1994) were collected and refined manually, while in our experiments the training corpus is collected automatically. Third, the original algorithm in (Guthrie et al., 1994) is proposed to classify formal documents, but personal emails in AIMEC.

Email often exhibits a more informal, personal style of writing compared to formal documents, and email management implies a very strong demand to be able to manage and identify personal preferences. No general preference exists to cover all email users. From this point of view, email categorization is rather different and more difficult compared to document classification.

The experimental results in the three tables indicate that with more recent training data, both Recall and Precision become higher. In corpus #1, the Precision is lowest in the three corpora due to most lacking in recent training data. The fact that the Precision is improved significantly in three months for all ten evaluators shows that AIMEC has a robust nature to adapt to new email environments for different genres of email.

When AIMEC is applied to new users who already have been using Outlook as their email manager for a long time, AIMEC can quickly extract user preferences in email management and adapt quickly by analyzing the existing folder structure.

The worst situations occur with users who have no sub-folder in their email accounts, and who start to manage their emails with AIMEC from scratch. In this situation AIMEC is not able to work well at the very beginning. The system will have to encourage the users to classify and move emails to the appropriate sub-folder manu-

ally for the first several days, enabling AIMEC to start adapting and improving its performance continuously.

6 Conclusion and Future Work

Automated user-centric data analysis contributes significantly to adaptive email categorization. AIMEC can easily adapt to the changing priorities and preferences of particular users by combining user preference inference with statistical text learning. The incremental training capabilities in AIMEC ensure rapid adaptation to existing users after the initial period of adaptation is over. The training process also has the capability to bootstrap its learning process by analyzing new user's email and identifying its most salient characteristics automatically.

An inevitable weakness in the training algorithm occurs if there are not enough emails available for a particular user. AIMEC cannot adapt well when there is a shortage of training data in the first several days, although AIMEC has a strong capability to learn and improve itself as more emails are received. Another unaddressed weakness in the current system is that its performance generally degrades when the email contains less than 200 words. The exact minimum word count when the system performance degrades significantly is difficult to determine as the performance depends upon the actual words used in the email. This problem – of having short, content poor emails, cannot be easily resolved by content-based classifiers. We are currently working on adding a rule-based framework on top of the current system that will automatically extract rules in a user-transparent manner to improve system performance in the absence of adequate content.

The last notable issue is threads in email communication. Emails belonging to one thread are more likely to belong in the same class and research is currently being carried out to improve the performance of AIMEC using this observation.

Acknowledgements. This work has been made possible through the financial support of the EU FASiL project(IST-2001-38685, www.fasil.co.uk), which aims to create a robust and scalable voice-driven Virtual Personal Assistant(VPA) managing email, calendar and agenda through an intelligent, friendly adaptive multi-modal interaction.

References

1. Androutsopoulos, I. Koutsias, J. Chandrinos, K. V. Paliouras, G. and Spyropoulos, C. D.. 2000. An Evaluation of Naive Bayesian Anti-Spam Filtering. Proc. of the workshop on Machine Learning in the New Information Age.
2. Buckley, C. Allan, J. Salton, G., J. 1994. Automatic Routing and Ad-hoc Retrieval Using SMART : TREC 2. In Proc. of the 2nd Text Retrieval Conference(TREC 2), page 45-55.
3. Carrerras, X. Marquez, L. 2001. Boosting Trees for Anti-Spam Email Filtering. Proc. RANLP-2001.
4. Cohen, W. 1996. Learning Rules that Classify EMail. Proc. AAAI Spring Symposium on Machine Learning in Information Access, Stanford, California.

5. Cunningham, H. Wilks, Y. and Gaizauskas, R. J. 1996. GATE: A General Architecture for Text Engineering. Proc. 16th International Conference on Computational Linguistics (ACL), Copenhagen, Denmark, Vol. 2, 1057-1060.
6. David, C. 2003. Information Society Statistics: PCs, Internet and mobile phone usage in the EU. European Community, Report KS-NP-03-015-EN-N.
7. Fisher, D. Moody, P. 2001. Studies of Automated Collection of Email Records. University of California, Irvine, Technical Report UCI-ISR-02-4.
8. Guthrie, L. Walker, E. and Guthrie, J. 1994. Document Classification by machine: Theory and practice. Proc. COLING'94, 1059-1063.
9. Lewis, D. 1998. Naive Bayes at forty: The independence assumption in information retrieval. Proc. ECML-98, Chemnitz. 4.15.
10. McCallum, A. Nigam, K. 1998. A comparison of event models for naive bayes text classification. AAAI-98 Workshop on Text Categorization.
11. Moody, P. 2003. Re-inventing Email. IBM Research Report.
12. Payne, T. Edwards, P. 1997. Interface Agents that Learn: An Investigation of Learning Issues in a Mail Agent Interface. Applied Artificial Intelligence Journal, AUCS/TR9508.
13. Quinlan, J.R. 1993. C4.5: Programs for Machine Learning. Morgan Kaufmann, San Mateo.
14. Resnick P. 2001. RFC2822 Internet Message Format. 24 April 2001, QUALCOMM Incorporated.
15. Rijsbergen, C. J. 1979. Information Retrieval, 2nd edition. Butterworths, London.
16. Smadja, F. Tumblin, H. 2003. Automatic Spam Detection as a Text Classification Task. Elron Software.
17. Thorsten, J. 2001. A Statistical Learning Model of Text Classification with Support Vector Machines. Proc. SIGIR-01, New Orleans. ACM Press, New York.
18. Wiener, E. Pederson, J.O. and Weigend, A.S. 1995. A neural network approach to topic spotting. Proc. SDAIR-95, Nevada, Las Vegas. 317-332.
19. Yang, Y. 1999. An evaluation of statistical approaches to text categorization. Journal of IR. 1(1/2):67-88.

ESPClust: An Effective Skew Prevention Method for Model-Based Document Clustering*

Xiaoguang Li, Ge Yu, Daling Wang, and Yubin Bao

School of Information Science and Engineering, Northeastern University,
Shenyang 110004, P.R.China
xgli7312@mail.163.com, yuge@mail.neu.edu.cn

Abstract. Document clustering is necessary for information retrieval, Web data mining, and Web data management. To support very high dimensionality and the sparsity of document feature, the model-based clustering has been proved to be an intuitive choice for document clustering. However, the current model-based algorithms are prone to generating the skewed clusters, which influence the quality of clustering seriously. In this paper, the reasons of skew generating are examined and determined as the inappropriate initial model, and the interaction between the decentralization of estimation samples and the over-generalized cluster model. An effective clustering skew prevention method (ESPClust) is proposed to focus on the last reason. To break this interaction, for each cluster, ESPClust automatically selects a part of documents that most relevant to its corresponding class as the estimation samples to re-estimate the cluster model. Based on the ESPClust, two algorithms with respect to the quality and efficiency are provided for different kinds of applications. Compared with balanced model-based algorithms, the ESPClust method has less restrictions and more applicability. The experiments show that the ESPClust can avoid the clustering skew in a great degree and its Macro-F1 measure outperforms the previous methods' measure.

1 Introduction

In recent years, there is a tremendous growth in the volume of documents available on Web, digital libraries, and news media. This has led to an increased interest in developing methods that can help users to effectively navigate, summarize, and organize this information or to discovery the inherent knowledge underlying document collection. As an important technique towards these goals, a high-quality document clustering algorithm plays an important role for information retrieval, Web data mining, and Web data management.

In general, current clustering methods can be divided into similarity-based approaches and model-based approaches. Due to the very high dimensionality and the

* Supported by the National Natural Science Foundation of China under Grant No.60173051 and the Teaching and Research Award Program for Outstanding Young Teachers in Higher Education Institution of the Ministry of Education, China

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 735 – 745, 2005.

sparsity of document features, similarity-based clustering algorithms meet a great challenge. Strehl [1] has proved that the traditional similarity functions are not adaptable to high-dimensional space. Moreover the cluster in similarity-based clustering is represented by a medoid or mean, which is meaningless to document clustering. When it is difficult to extract good features or represent the cluster, probabilistic model-based clustering is an intuitive choice [2]. In the model-based methods, the cluster is described by a representative probabilistic model, which provides a probabilistic interpretation. Probabilistic model-based clustering has been studied extensively in the last two decades, and has shown the promising results [3~15]. Typically, model-based clustering can be divided into partitioning approaches and hierarchical approaches. In this paper we focus on the partitioning clustering for very high dimensional data.

In practice, model-based clustering, as well as similarity-based clustering, quite often generates some skewed clusters that are empty or extremely small, especially when the data is in high-dimensional (>100) space [15]. The skewed clusters influence the quality of clustering seriously. Even though the feature selection techniques applied, a document has approximately 10000 dimensions so that it is more prone to generating skewed clusters in document clustering. To prevent the skewed clusters, balanced clustering methods were proposed in the past [2, 14, 15, 16, 17]. Generally, these methods avoid the skewed clusters by setting the proportion of each cluster to the whole data as the algorithm's constraint, and are applied into the situations that the clusters have the comparable size. But it is difficult to set this constraint in most cases. E.g., in the case of IR, when clustering the retrieved results of a given query, the number of data irrelevant to the query is much larger than that of the relevant ones, and it is nearly impossible to estimate the proportions of each cluster in advance. Moreover current methods do not study further how the skewed clusters generate, but just consider it as a constraint-based optimization problem.

In this paper, our basic idea is to design a clustering algorithm that can group the documents into the clusters of inherent size without any balancing constraint. We first analyze the reasons how the skewed clusters generate and consider that it is due to two factors: the inappropriate initial model and the interaction between the decentralization of estimation samples and the over-generalized cluster model. Because there are no initialization techniques performing well by far, we focus on the second factor and propose an effective skew prevention method (ESPClust) for model-based clustering. In ESPClust, for each cluster, not all the documents, but a part of documents that most relevant to the corresponding class, are selected automatically as the samples to re-estimate the cluster model, which reduces the decentralization of estimation samples, and then prevents the model estimated over-generally. Based on ESPClust, two algorithms with respect to the quality and efficiency are provided for different applications. Compared with the balanced model-based methods, the ESPClust method doesn't need the prior knowledge about the proportion of each cluster, and is more feasible in the practical usages. The experiment shows that the ESPClust can prevent the clustering skew in a great degree, and the Macro-F1 measure outperforms that of the previous' methods.

The rest of the paper is organized as follows. Section 2 introduces briefly the model-based partitioning clustering method. Section 3 gives the definition of clustering skew and examines the reasons how the skew generates. The ESPClust method and the corresponding algorithms are proposed in Section 4 and Section 5. The experiment to

evaluate the performance of ESPClust is presented in section 6. Section 7 discusses the related works. Section 8 is our conclusions.

2 Overview of Model-Based Partitioning Clustering Method

In this section, we introduce briefly the model-based partitioning clustering. Given a m-dimension data $x_i=(x_{i1}, x_{i2}\ldots x_{im})$, and a data collection $X = \{x_1, x_2\ldots x_n\}$, for model-based clustering, the data $x_i \in X$ is considered to be a sample independently drawn from a mixture model [18] $\theta=\{\theta_1, \theta_2\ldots\theta_k\}$. The main assumption is that data points are generated by, firstly, randomly picking a model θ_j with probability $P(\theta_j)$, and secondly, by drawing a data x_i from a corresponding distribution. Each cluster j is associated with the corresponding distribution model θ_j, called *cluster model*, and each data point carries not only its observable attributes, but also a hidden cluster. The overall likelihood of the data collection X is its probability to be drawn from a given mixture model θ, and then model-based clustering boils down to finding the maximum likelihood estimation of θ. Expectation-Maximization (EM) is applied to compute the MLE of θ. EM is a two-step iterative optimization. E-step estimates probabilities $P(j \mid x_i)$, j=1~k, which is equivalent to the data assignment. M-step finds an approximation to a mixture model, given current assignments, which is equivalent to the cluster model re-estimation.

In general, partitioning clustering can be divided into three categories: hard clustering, soft clustering, and stochastic clustering [2]. Due to its simplicity, hard clustering has been applied widely in data clustering [2, 10, 11, 12]. The most popular probabilistic models in document clustering are multivariate Bernoulli model [2], multinomial model [2, 12], and von Mises-Fisher (vMF) model [10, 11]. Of the three types of models, vMF leads to best performance and multivariate Bernoulli to worst; the multinomial model is a bit worse than vMF. However the estimation of parameters in vMF model is computationally much more expensive [2], the multinomial distribution is used widely as the underlying cluster model for data clustering. In this paper, the multinomial model-based partitioning hard clustering, denoted by multiK-means, is selected as the baseline algorithm [2].

3 Analysis of Clustering Skew

We begin with some concepts. Given a document collection X and a inherent criteria R of X to evaluate the relevance between the documents, suppose that there are k inherent classes in X with respect to the R, the class i is denoted by l_i, i =1~k and L is the class set, $L=\{l_1, l_2\ldots l_k\}$. For the model-based clustering, we associate $L = \{l_1, l_2\ldots l_k\}$ with the class model $\theta=\{\theta_1,\theta_2\ldots\theta_k\}$. A partitioning clustering algorithm constructs k partitions of X, where each partition represents a cluster, denoted as c_i, i =1~k. C is the cluster set, $C = \{c_1, c_2\ldots c_k\}$. The documents assigned to cluster c_i construct the *sample set* X_i, which satisfies $\cup X_i = X \Lambda X_i \cap X_j = \Phi$, i =1~k, $i \neq j$. Here $C = \{c_1, c_2\ldots c_k\}$ is associated with the cluster models $\theta'=\{\theta'_1, \theta'_2\ldots\theta'_k\}$ obtained by the algorithm. Without the loss of generality, let each $c_i \in C$ associated correspondingly with $l_i \in L$, i =1~k. For example in Fig. 1, there are three classes, $L=\{l_1, l_2, l_3\}$, where the documents of class l_1, l_2 and l_3 are

represented by the symbol "+", "-"and "o" respectively, and there are three clusters, $C=\{c_1, c_2, c_3\}$, whose samples are the documents in corresponding ellipse.

Definition 1. (***Estimation sample***) For a cluster, the *estimation sample* is the document used to re-estimate the cluster model's parameters.

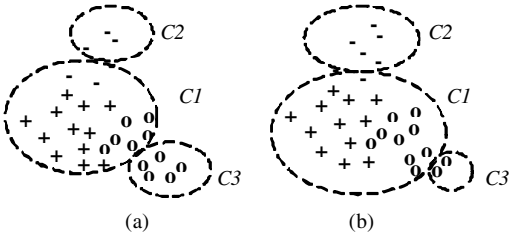


Fig. 1. Illustration of clustering skew. (b) represents the clustering result after (a)

Definition 2. (***Decentralization of estimation sample***) For a *cluster* c_i, if its estimation samples contain many documents $x \in l_j, j \neq i$, we call it the decentralization of estimation sample.

For example, in Fig. 1, the samples of each cluster also are the estimation samples. the estimation samples of c_1 contain many documents belonging to not only l_1 but also l_3, so it is decentralized.

Definition 3. (***Over-generalization of cluster model***) If a cluster model θ'_j reflects not only the characteristic of l_j, but also that of other l_i, $i \neq j$, especially when $P(x|\theta'_j) > P(x|\theta'_i)$, $x \in l_i$, it is called as over-generalization of cluster model θ'_j .

Definition 4. (***Clustering skew***) Given X, L and C, if there is a subset $C' \subset C$, where the cluster $c_i \in C'$ contains so many documents satisfying $x \in X_i \wedge x \notin l_i$ as to generate empty or extremely small cluster $c_j \notin C' \wedge c_j \in C$, we call it as *Clustering skew*.

For example, in Fig. 1 the clustering is skewed. Most of *documents* belonging to l_1 and l_3 incline to be assigned to the cluster c_1, whereas the cluster c_3 only contains a few documents belonging to l_3.

In general, there exists the "winner-take-all" behavior for high *dimensional* space. For the similarity-based clustering algorithm, it has been argued in [19, 20] that given two targets, one target is prone to winning most of data and few of data are assigned to the other, since the contrast between the distances of different data points does not exist. This behavior also appears in model-based clustering so that if an inappropriate initial model applied, most of data are prone to being grouped into a few wrong clusters.

Ideally, the model θ'_j of cluster c_j will approach gradually to the distributional characteristic of class l_j. However, because of the inappropriate cluster model mentioned above, at the assignment stage, especially the first assignment stage, most of data are prone to being assigned to a few wrong clusters, denoted by C', and then it results in the decentralized samples X_c for each cluster $c \in C'$. If all the decentralized samples X_c are used as the estimation samples, as the current model-based methods do, it will be probable to make the cluster model θ'_c estimated over-generally, and then with the θ'_c, there will be more data $x \notin l_c$ assigned to c at the next assignment step. With the interaction between the decentralization of estimation samples and the over-generalization of cluster model, the skewed clusters are generated ultimately.

	<1>	<2>	<3>
#1	75	15	22
#2	17	121	12
#3	**104**	**56**	**163**

1st-iteration

	<1>	<2>	<3>
#1	38	2	4
#2	10	126	6
#3	**148**	**64**	**187**

3rd-iteration

	<1>	<2>	<3>
#1	17	2	3
#2	8	122	6
#3	**171**	**68**	**188**

Clustering result

Fig. 2. Clustering process of multiK-means in BG3C600

A detailed example given in Fig. 2 depicts the clustering process of multiK-means in the BG3C600 dataset (the details of dataset is given in Section 6). In the 1st-iteration the cluster #3 associated with <3> class contains many documents of <1> class, and multiK-means uses all the samples of each cluster to re-estimate the cluster model, as a result the model of #3 fits not only <1> class but also <3> class, i.e. the model of #3 is re-estimated over-generally, so at the assignment stage more documents of <1> class are assigned to the cluster #3 in the 3rd-iteration. Finally most of data belonging to <1> class are skewed to the cluster #3.

4 Selection of Estimation Samples

We think that there are two ways to prevent the skewed clusters. One is to try to select an appropriate initial model to reduce the decentralized samples at the first assignment stage. The other is that at the model re-estimation stage a part of documents that are most relevant to the corresponding class for each cluster as the estimation samples to break the interaction between the decentralization and the over-generalization, and then to prevent skewed results. There are many initialization techniques proposed in the past, but none of them perform well [12]. Here we aim at the last solution. Actually, even though the decentralized samples occur in some clusters on account of the inappropriate initial model, with the estimation samples selection, this influence will not expand further.

Definition 5. For a cluster c_j and X_j, if a document $x \in X_j$ belongs to class l_j, we call it as that *x matches* the cluster c_j, otherwise, *x mismatches* c_j and is called as a *noise* data of cluster c_j.

We define the *matching function* $\Delta: X \rightarrow \boldsymbol{R}^+$ as: Given the cluster set $C = \{c_1, c_2 \ldots c_k\}$, for $x \in X$,

$$\Delta(x) = Max\{P(x \mid c_i) \mid i = 1 \sim k\} - \frac{1}{k} \sum_{i=1 \sim k} P(x \mid c_i) \tag{1}$$

Given cluster c_j and X_j, the function $\Delta(x)$ measures the matching degree between $x \in X_j$ and c_j. In general, given $x_1, x_2 \in X_j$, if x_1 belongs to l_j, but x_2 not, then $\Delta(x_1) > \Delta(x_2)$. If the Δ of document is near to zero, it is difficult to judge which cluster it belongs to. Note that, this conclusion is not true for all cases, i.e. given $x_1, x_2 \in X_j$ and $x_1 \in l_j$, $x_2 \notin l_j$, $\Delta(x_1)$ may be smaller than $\Delta(x_2)$, but as discussed as follow, according to the matching degree we could select the estimation samples whose the probability of matching the cluster is higher than that of mismatching.

We define the variable z that takes two values: r and $\bar{r}$, which represent "match" and "mismatch" respectively. From a lot of experiments we found that for each cluster c_j, the Δ distribution of the documents that match c_j can be modeled as a normal distribution (Formula 2), while the Δ distribution of the documents that mismatch c_j can be modeled as an exponential distribution (Formula 3). The Fig. 3 illustrates the Δ distribution in one of our experiments.

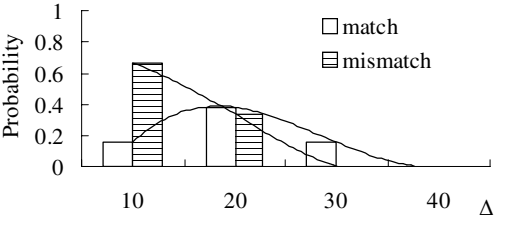


Fig. 3. Δ distribution of matching and mismatching

$$P(\Delta \mid z = r; c_j) = \frac{1}{\sqrt{2\pi}\delta_j} \exp\left(-\frac{(\Delta - \mu_j)^2}{2\delta_j^2}\right) \tag{2}$$

$$P(\Delta \mid z = \bar{r}; c_j) = \lambda_j \exp\left(-\lambda_j \Delta\right) \tag{3}$$

The EM algorithm is used to estimate the parameters $\hat{\mu}_j$, $\hat{\delta}_j$ and $\hat{\lambda}_j$.

Theorem 1. Given $\hat{\mu}_j$, $\hat{\delta}_j$, $\hat{\lambda}_j$, $P_j(r)$ and $P_j(\bar{r})$, for $x \in X_j$, if $\frac{1}{2}(a - \sqrt{a^2 - 4b}) < \Delta(x) < \frac{1}{2}(a + \sqrt{a^2 - 4b})$, then $P(r \mid \Delta; c_j) > P(\bar{r} \mid \Delta; c_j)$, where $a = 2(\hat{\mu}_j + \hat{\lambda}_j \hat{\delta}_j^2)$, $b = \hat{\mu}_j^2 + 2\hat{\delta}_j^2 \ln \sqrt{2\pi} \hat{\lambda}_j \hat{\delta}_j P_j(\bar{r}) / P_j(r)$.

Proof

$$P(r \mid \Delta; c_j) > P(\bar{r} \mid \Delta; c_j) \overset{(1)}{\Rightarrow} P(\Delta \mid r) P(r) > P(\Delta \mid \bar{r}) P(\bar{r})$$

$$\Rightarrow \frac{1}{\sqrt{2\pi}\hat{\delta}_j} \exp\left(-\frac{(\Delta - \hat{\mu}_j)^2}{2\hat{\delta}_j^2}\right) P_j(r) > \hat{\lambda}_j \exp\left(-\hat{\lambda}_j \Delta\right) P_j(\bar{r}) \overset{(2)}{\Rightarrow} \Delta^2 - a\Delta + b < 0 \tag{4}$$

where (1) applies the Bayes rule; (2) takes the logarithm for the inequation, and then $a = 2(\hat{\mu}_j + \hat{\lambda}_j \hat{\delta}_j^2)$, $b = \hat{\mu}_j^2 + 2\hat{\delta}_j^2 \ln \sqrt{2\pi} \hat{\lambda}_j \hat{\delta}_j P_j(\bar{r}) / P_j(r)$. The inequation 4 is a one variable quadratic inequation, its solution is $\frac{1}{2}(a - \sqrt{a^2 - 4b}) < \Delta < \frac{1}{2}(a + \sqrt{a^2 - 4b})$. ■

When $\Delta > \frac{1}{2}(a + \sqrt{a^2 - 4b})$, though it doesn't satisfy Theorem 1, $P(\bar{r} \mid \Delta; c_j)$ is usually so small as to be close to zero, we can ignore it and set the threshold for the cluster ε_j as

$$\varepsilon_j = \tfrac{1}{2}(a - \sqrt{a^2 - 4b}) \tag{5}$$

The algorithm of computing the selection threshold is given as follows. Note that in the step 1, all $\Delta(x)$, $\forall x \in X_j$ are sorted descending, and the 2/3 highest Δ in X_j are selected to compute the mean and variance as the initial value of μ_j and δ_j. The initial value of λ_j is set to the mean over the whole X_j. In the step 2, $N = |X_j|$.

Algorithm 1. Compute the selection threshold

Input: cluster model $\theta=\{\theta_1, \theta_2,\ldots, \theta_k\}$ and X_j for cluster c_j

Output: threshold ε for X_j

1. Initialize the μ^0, δ^0 and λ^0;
2. Do until convergence {

   // E-step

   $$\hat{P}^i(r \mid \Delta) = \frac{P^{i-1}(\Delta \mid r)P^{i-1}(r)}{P^{i-1}(\Delta \mid r)P^{i-1}(r) + P^{i-1}(\Delta \mid \bar{r})P^{i-1}(\bar{r})}$$

   $$\hat{P}^i(\bar{r} \mid \Delta) = \frac{P^{i-1}(\Delta \mid \bar{r})P^{i-1}(\bar{r})}{P^{i-1}(\Delta \mid r)P^{i-1}(r) + P^{i-1}(\Delta \mid \bar{r})P^{i-1}(\bar{r})}$$

   // M-step

   $$\mu^i = \frac{\sum_\Delta \hat{P}^i(r \mid \Delta)\Delta}{\sum_{\Delta'} \hat{P}^i(r \mid \Delta')} \qquad \delta^i = \frac{\sum_\Delta \hat{P}^i(r \mid \Delta)(\Delta - \mu^i)^2}{\sum_{\Delta'} \hat{P}^i(r \mid \Delta')} \qquad \lambda^i = \frac{\sum_\Delta \hat{P}^i(\bar{r} \mid \Delta)}{\sum_{\Delta'} \hat{P}^i(\bar{r} \mid \Delta')\Delta'}$$

   $$P^i(r) = \tfrac{1}{N}\sum_\Delta P^i(r \mid \Delta) \quad P^i(\bar{r}) = 1 - P^i(r)\ \}$$

3. Using Formula 5 to compute ε;
4. Return ε;

5 ESPClust Algorithm

For each cluster c_j, j=1~k, we compute a threshold ε_j, and then select the documents that satisfy $\Delta(x)>\varepsilon_j$ as the samples to estimate θ'_j. According to the theorem 1, for each selected sample, its probability of matching the cluster is higher than that of mismatching. In such a case, the risk of cluster model estimated over-generally could be reduced to a certain extent. In this section we design two algorithms based on estimation samples selection. One is the ESPClust-I algorithm focusing more on the clustering quality, as shown as follows. Note that in this paper we select the multinomial distribution as cluster model because of its popularity and effectiveness in practice, actually the algorithms proposed here can be applied into any distribution. In Algorithm 2, M is a *Laplace* parameter to avoid zero probability and $c(x^l, X'_j)$ represents the count of dimension l appearing in the X'_j.

Algorithm 2. ESPClust-I

Input: $X = \{x_1, x_2 \ldots x_n\}$.

Output: Trained cluster model $\theta=\{\theta_1, \theta_2 \ldots \theta_k\}$, and a partition of X given by the cluster identity vector $Y = \{y_1, y_2 \ldots y_n\}$, $y_i \in \{1,\ldots,k\}$.

1. Initialize $\theta^0=\{\theta^0_1, \theta^0_2 \ldots \theta^0_k\}$;
2. Do until convergence {
   // Assignment step
3. For $x_i \in X$, $i = 1 \sim n$, do $y_i = \arg\max_k \sum_{l=1\sim m} \log P(x_i^l \mid \theta_k^{i'})$;
   // Re-estimation step
4. For $j = 1 \sim k$, do {
5. Let $X_j = \{x_i \mid y_i = j\}$;
6. Using the algorithm 1 to compute ε_j;
7. Let X_j '$= \{x_i \mid y_i = j \Lambda \Delta(x_i) > \varepsilon_j\}$;
8. For $l = 1\sim m$, do $P(x^l \mid \theta_j^{i'+1}) = \dfrac{1 + c(x^l, X'_j)}{M + \sum_{\substack{l'=1\sim m \\ j=1\sim k}} c(x^{l'}, X'_j)}$;}}
9. Return θ and Y;

In some cases, the efficiency is more important than the quality, e.g. in IR, when clustering the retrieved results, users couldn't bear the long waiting after submitting the query, so we design the other algorithm, ESPClust-II, to improve the clustering efficiency. In ESPClust-II the re-estimation samples are selected at the first iteration only, because in most cases the decentralization occurs at the initial phase of clustering as mentioned in section 3. Compared with the ESPClust-I algorithm, while its clustering quality is degraded in some sense, as shown in section 6, ESPClust-II still achieves much better performance than current methods.

6 Performance Experiment and Analysis

We first introduce the testing datasets, and then give the experiment method and the criteria used to evaluate the performance.

20NG [21] and BINGLE [22] are selected as testing corpus. In order to evaluate the algorithm's performance on the different datasets, 7 datasets are constructed from the two corpuses. All the words are stemmed, and then the stop words are removed according to an artificial dictionary. The summary of datasets is shown in the Table 1.

Table 1. Summary of datasets

Dataset	*K*	*Number of document*	*Number of word*
20NG4C4000	4	3995	14077
TALK4C4000	4	3997	12186
SCI4C4000	4	4000	13427
BG2C1000	2	987	14256
BG3C600	3	585	10380
BG6C120	6	120	2559
BG10C1000	10	1000	17142

Because of the constraints on the cluster size, the results obtained by the balanced methods aren't skewed certainly. So ESPClust is not compared with the balanced methods, but with multiK-means. All the algorithms are initialized with the same models selected at random. In this paper we use the confusion matrix and Macro-F1 as evaluation criteria. The confusion matrix can reflect intuitively whether the clustering skew or not, and Macro-F1 can evaluate the overall quality of clustering.

6.1 Analysis of Skew Prevention

Due to the space limitation, the clustering on the BG3C600 dataset is only used to show the process of skew prevention in detailed. Note that since we just want to show the method proposed in this paper how to prevent the skew generating regardless of the detailed algorithms, here multiK-means is compared only with ESPClust-I. As ESPClust-I shown in Fig. 4 and multiK-means in Fig. 2, in the 1st-iteration, there is the decentralization occurred in the cluster #3 with the two algorithms because of the initialization, but in the 3rd-iteration, the decentralization in the cluster #3 is reduced greatly with the partial estimation. As a result, ESPClust-I avoids the clustering skew.

	<1>	<2>	<3>
#1	92	20	21
#2	17	126	12
#3	**87**	**46**	**164**

1st-iteration

	<1>	<2>	<3>
#1	126	18	14
#2	3	139	2
#3	**67**	**35**	**181**

3rd-iteration

	<1>	<2>	<3>
#1	162	5	13
#2	4	155	3
#3	**30**	**32**	**181**

Clustering results

Fig. 4. Clustering process in BG3C600 of ESPClust-I

For multiK-means, the clustering skew appears in 20NG4C4000, TALK4C4000, SCI4C4000, BG3C600, BG6C120 and BG10C1000 respectively, but not in BG2C1000. BG2C1000 dataset includes two classes (art and networks) with high inter-similarity and low intra-similarity, so multiK-means performs better on the BG2C1000 dataset than the others. ESPClust-I prevents the clustering skew in all the 7 datasets. The average Macro-F1 of ESPClust-I, as shown in figure 5, is 0.732, 63.76% higher than that of multiK-means. Especially on the SCI4C4000 dataset, ESPClust-I avoids the clustering skew and achieves significant performance improvement, 253.8% higher than multiK-means.

6.2 Comparison Between ESPClust-I and ESPClust-II

The Fig. 5 and 6 depict the Marco-F1 and runtime of the three algorithms in all the datasets. On the whole ESPClust-II is more efficient than ESPClust-I, and its runtime is 39.7% lower than that of ESPClust-I on average. The Macro-F1 of ESPClust-II is 5.3% lower than that of ESPClust-I, but 51.1% higher than that of multiK-means. It is surprising that except for BG2C1000 and BG6C120, the runtime of ESPClust-II is only 6.9% higher on average than that of multiK-means, especially in BG3C600, 45.4% lower than that of multiK-means. The reason lies in that while the estimation samples

selection is time-consuming at the initial phase, the accurate cluster model can be determined more early than multiK-means, and the less iteration are required to converge so that the whole runtime reduces. E.g. in BG3C600, ESPClust-II requires two iterations only, but multiK-means requires six iterations.

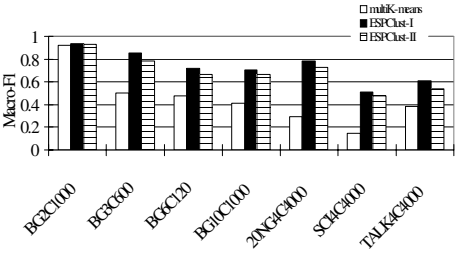


Fig. 5. Comparison of Macro-F1

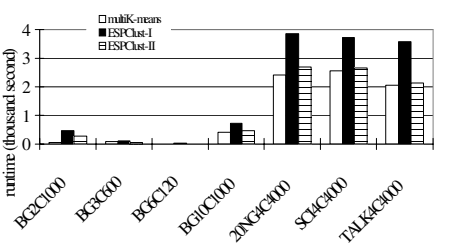


Fig. 6. Comparison of runtime

7 Related Work

Although the clustering skew often occurs in high-dimensional data clustering and influences the quality of clustering seriously, it is surprising that there are few studies focusing on it in the past. To prevent the clustering skew, current solutions is to set the size constraints on each cluster and to consider the clustering as constraint-based optimization problem. Their algorithms focus on the efficiency and scalability, and are only applicable for some applications where the constraints can be obtained in advance, such as segmenting customers into the groups of rough equal size in market analysis.

In [15] a balanced clustering method assumes that the data points satisfy the balancing constraint. Reference [14] also constrains each cluster to be assigned at least a minimum number m ($<N/k$) of data points. The data assignment problem is formulated as a minimum cost flow problem. Zhong [2] propose a balanced model-based hard clustering framework that is applied to any distribution. They [16] also propose a soft balancing strategy built on a general soft model-based clustering framework. Instead of constraining the actual number of data objects in each cluster to be equal, they constrain the expected number of data objects in each cluster to be equal. In [17] the approach to obtain balanced clusters is to convert the clustering problem into a graph-partitioning problem, and proposed the "min-cut" algorithms that incorporate a balancing constraint.

Different from above methods, the ESPClust method needs not the prior knowledge about the proportion of each cluster, but with the automatic selection of samples, to prevent the skewed clusters. This feature makes it more feasible in the practical applications.

8 Conclusions

In this paper we analyze the reason of skew generating in high-dimensional data clustering, and propose the ESPClust method to prevent the skewed results for documents. In order to avoid the influence of estimation samples decentralization we automatically select samples within a cluster to re-estimate model, which needn't set the parameters

manually and achieves a significant improvement on the clustering results. Compared with the balanced methods in previous work, the ESPClust method has less restrictions and more applicability for document clustering.

References

1. Strehl, A., Ghosh, J., and Mooney, R.: Impact of Similarity Measures on Web Page Clustering. In Proc. 17th National Conf. of Artificial Intelligence for Web Search, 2000.
2. S. Zhong: Probabilistic Model-Based Clustering of Complex Data. PhD thesis, The University of Texas at Austin, 2003.
3. S. Zhong and J. Ghosh: A Unified Framework for Model-based Clustering. Machine Learning Research, 2003.
4. S. Kamvar, D. Klein, and C. Manning: Interpreting and Extending Classical Agglomerative Clustering Algorithms Using A Model-based Approach. In Proc. 19th Int. Conf. Machine Learning, 2002.
5. J. D. Banfield and A. E. Raftery: Model-based Gaussian and non-Gaussian Clustering. Biometrics, 1993.
6. C. Fraley: Algorithms for Model-based Gaussian Hierarchical Clustering. SIAM Journal on Scientific Computing, 1999.
7. S. Vaithyanathan and B. Dom: Model-based Hierarchical Clustering. In Proc. 16th Conf. Uncertainty in Artificial Intelligence, 2000.
8. M. Ramoni, P. Sebastiani, and P. Cohen: Bayesian Clustering by Dynamics. Machine Learning, 2002.
9. I. V. Cadez, S. Ganey, and P. Smyth: A General Probabilistic Framework for Clustering Individuals and Objects. In Proc. 6th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining, 2000.
10. I. S. Dhillon and D. S. Modha: Concept Decompositions for Large Sparse Text Data Using Clustering. Machine Learning, 2001.
11. A. Banerjee and J. Ghosh: Frequency Sensitive Competitive Learning for Clustering on High-Dimensional Hyperspheres. In Proc. IEEE Int. Joint Conf. Neural Networks, 2002.
12. M. Meila and D. Heckerman: An Experimental Comparison of Model-Based Clustering Methods. Machine Learning, 2001.
13. A. K. Jain, M. N. Murty, and P. J. Flynn: Data Clustering: A review. ACM Computing Surveys, 1999.
14. P. S. Bradley, K. P. Bennett, and A. Demiriz: Constrained k-means Clustering. Technical Report MSR-TR-2000-65, Microsoft Research, Redmond, WA, 2000.
15. A. Banerjee and J. Ghosh: On Scaling Up Balanced Clustering Algorithms. In Proc.2nd SIAM Int. Conf. Data Mining, 2002.
16. S. Zhong and J. Ghosh: Model-based Clustering with Soft Balancing. In Proc. of ICDM 2003.
17. G. Karypis and V. Kumar: A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. SIAM Journal on Scientific Computing, 1998.
18. G. McLachlan and D. Peel: Finite Mixture Models. John Wiley & Sons, 2000.
19. Beyer K., Goldstein J., Ramakrishnan R., Shaft U.: When is Nearest Neighbors Meaningful? In Proc. of ICDT Conf., 1999.
20. C. C. Aggarwal, A. Hinneburg, D. A. Keim: On the Surprising Behavior of Distance Metrics in High Dimensional Spaces. In Proc. of ICDT, 2001.
21. http://kdd.ics.uci.edu/databases/20newsgroups/20newsgroups.html
22. http://net.pku.edu.cn/~yanqiong/

A Method of Rapid Prototyping of Evolving Ontologies *

Pavel Makagonov and Alejandro Ruiz Figueroa

Mixteca University of Technology, Huajuapan de León, Oaxaca, 69000, México
mpp@mixteco.utm.mx, figueroa@nuyoo.utm.mx

Abstract. A text corpus analysis method is proposed for analyzing a branch of knowledge evolving in time. We apply this method to the case of parallel computing, which has shown a great progress during the last 15 years. For this, we construct a rapid prototype of ontology (RPO) of this domain, based on a large collection of different types of documents. We demonstrate the possibility of embodying the RPO idea and its usefulness for revealing prospects for research in different areas of parallel computing.

1 Introduction and Problem Formulation

The principal goal of our study is to create a method for analyzing the development of a given branch of knowledge [1]. We take as an example parallel computing.

We propose a rapid prototyping of its ontology. By ontology we understand a hierarchical scheme of concepts and words [2]. The bottom levels of the "pyramid" are more detailed concepts of a given branch of scientific knowledge, and every next upper level contains more abstract concepts. Creation of an ontology usually implies manual work involving experts in a given domain and knowledge engineers.

2 Method and Results of the Study

For rapid prototyping of an ontology (RPO) we use a corpus of documents belonging to the selected branch of knowledge, with different level of abstractedness. In this study we used texts of four levels of abstractedness:

- Titles of conferences, monographs, and manuals; it is possible to include titles of journals or Proceedings;
- Tables of contents of monographs and manuals; it is possible to include the headings of conference sections;
- Titles of papers (but not their sections);
- Bodies of the abstracts and papers.

Our text corpus contained 16 books on parallel computing and 556 abstracts of IEEE journal papers [3]. The papers are from different conferences on parallel, distributed, concurrent, and simultaneous computing. That is why these four adjectives are the high level concepts of our ontology prototype.

* Work partially supported by National Council for Science and Technology (CONACYT), Mexico under project N-39011-A.

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 746 – 749, 2005.

From this collection of books and abstracts we extracted all textual materials of the levels of abstractedness mentioned above. Then we grouped the textual materials into groups of files (T_i). Every file contained textual material of the same types (titles of conferences, titles of articles without data about their authors, tables of contents of books, etc.) and for the same period of time, i.e., joined by year or some years.

Every portion of text materials (except for bodes of papers) is very short and does not have enough specific words for clustering. That is why we need to enrich links between textual units by concatenating them. We use measures for enriching knowledge-poor texts developed in our pervious works [4, 5]. For concatenating short texts into one we use external attribute of every texts: is time of issue. Joining of the tables of content of books is possible by topic coincidence (what we did) as well as by year of publication. Without such concatenation, short texts are clustered around a small amount of words for which it is difficult to reveal temporal tendencies. The files $\{T_i\}$ formed in this way are used for the next stages of our study, as described below:

First stage. We used the toolkit *Visual Heuristic Cluster analysis for texts* (VHCA for texts) [4] as an elemental step of proposed method. With VHCA for texts, we obtained a Domain oriented dictionary (DOD) for the textual corpus and an image of every text as vector of frequencies of DOD's words in the text; note that the texts are stemmed [6]; in what follows for simplicity we refer to the stems as words. These textual images were used to form three matrices: *texts-by-words matrix* with frequencies of the words from the DOD in each text; *words-by-words matrix* with frequencies of word co-occurrence in a text (number of texts containing both words); and *texts-by-texts matrix* with the number of DOD words in common in each pair of texts.

Second stage. We clustered the words in texts-by-words matrices for sets of texts, using VHCA. Every set contained the texts of different level of abstractedness. The first matrix contained texts of conference and paper titles and tables of contents of books; second matrix contained the bodies of abstracts and titles of texts; third matrix contains the bodies of abstracts (it is possible to include the bodies of papers).

In the first matrix we separate first of all the clusters of words that are most numerous for conference titles from the words which are not representing conference titles. The cluster ***A*** is shown in Figure 1 in the left part of the matrix. Every non-zero element of the matrix is represented by a rectangle with corresponding grey scale color in according to the scale in the left part of the figure.

The clusters of words which are not connected with conference titles can be used for the next (lower) levels of abstractedness. We exclude conference titles from the processes of construction of the next level of RPO.

For the last step of this stage we used the matrix of texts of abstracts. The second level of PRO hierarchy in Figure 1 is represented by clusters ***B*** and ***C***, which have rich representation in clusters of book tables of contents but poor representation in titles of abstracts. All other clusters are observed at lower levels of hierarchy.

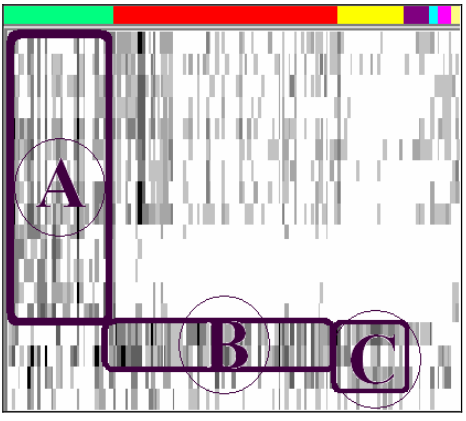


Fig. 1. Distribution of clusters of words by types of documents (titles of abstracts, titles of congresses, and contents of books)

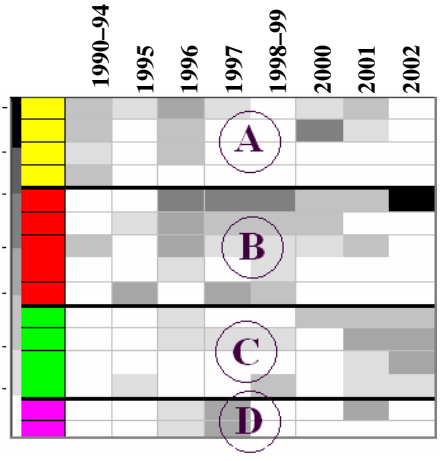


Fig. 2. Distribution of clusters of words from abstracts by years

Here are examples of stems in cluster ***A*** of abstract and conference titles:

- HIGH PERFORMANCE **VLSI** TECHNOLOG MAINTENANCE
- APPLICATION COMPUTER ENGINEER MICRO
- NETWORK ALGORITHM TIME MODEL DEPENDABL SPECIFICAT EUROMICRO PROGRAMM FAULT ARCHITECTURE REAL SYNTHESIS
- SIMULAT **OBJECT ORIENT**
- CONCURRENCE RELIABILITY CLUSTER TOLERANCE DEFECT TEST
- COMPREHENSE VOLUME NEURO

For the bottom level, we have clusters of words of abstracts. These clusters contain sub-clusters of words that have correlation more than 0.9. For example, typical clusters ***A*** and ***B*** in one of our experiments are as follows ({...} denotes sub-clusters):

Cluster ***A***. Most numerous, peak in 1997 with gradual rising and decaying:

VLSI, {VLIW ARITHMET ADAPTABL}, {IRREGULAR BULK EMULAT OPTIMIZATION WHILST LABEL VALIANT EFFICAC ARMSTRONG IMPART}, CORBA, ADDITIONAL, SCHMIDT, {MASSIVE RELI}, THEORETIC, LOCAT, SPEED, {PARALLELIZAT COMPIL HIGHLIGHT WORKSTATION DEPENDENC SLIC DEBUGG SEQUENC INPUT DOMAIN METRIC}, SYNCHRONIZ, {ASIC ARBITRAT FAIRNES}, {LONG CENTRALIZ}, VOLATIL, {SHOULD PARALLELIZ OPTIMIZ OUTPERFORM DUAL}, {DATAFLOW RELIABL DESIGNER}, EXPLICIT, {ARRA BANDWIDTH FACILITAT}, TOPOLOG, {DIGIT CHECK MUCH VERIFICAT}, {IMPLEMENTATION LIMIT INVESTIGAT LAYER CLOCK}, ENTITI, {LIBRAR ENVIRONMENT TEMPORAL}, {PRAM PEND FOCUSS COMA}, SESSION, VHDL, VERIF, SPECIFICATION, DECOD, XILINX MOBIL INTRINS POTENTIAL MULTIPLIER, COMPILER.

Cluster ***B***. Growth from 1997, good in 1998–98, peak in 2001, slight decay in 2002–3:

MULTIMEDIA, {PROGRAMMABL JAVA REALIS EXTRACT PLATFORM KEYWORD}, {DUPLICAT PIPELIN MICROPROCESSOR MULTIPLEX SPECULAT}, {FEASIBILIT OPTIMIST}, {MANIPULAT LOOP DETECTOR REDUNDANC TRANSPARENT SHELF UBIQUIT ATTAINABL FACTORIZAT SIMD INTEGRIT PROPAGAT REGIST MESH}, {REGISTER TESTABILIT}, {ROBUST MULTICAST}.

Third stage. Using texts-by-words matrix (which could be called time-by-words or years-by-sub-clusters of words), we analyze the word distribution into clusters and temporal evolution of every cluster. For example, in Figure 2 topics related to cluster ***A*** decay in time; topics of cluster ***B*** are most numerous in 1997, with gradual rising and decaying, and drop their position in the last years. Topics related to cluster ***C***, growing from 1997, received great attention in 2003. Topics related to cluster ***D*** are local in time.

In our previous work [5] we have developed an approach to analysis of temporal evolution for the documents of one type. This approach can be added to the third stage of this study.

3 Conclusion

We propose a method for evaluation of prospect of different branches of knowledge, for long periods of time. It is useful to construct a RPO for a set of time periods, and construct a measure of distance between different RPO by a method that is close to the one presented in [7].

Our method of rapid prototyping can be adapted for other applications, such as constructing more objective ontology, updating and correcting existing ontology.

References

1. M. Montes y Gómez, A. López López, A. Gelbukh. Text mining as a social thermometer. In: *Text Mining workshop at 16th International Joint Conference on Artificial Intelligence* (IJCAI'99), Stockholm, 1999, pp. 103–107.
2. A. Gelbukh, G. Sidorov, A. Guzman-Arenas. Use of a weighted topic hierarchy for text retrieval and classification. In: *TSD-99*. Lecture Notes in Artificial Intelligence, No. 1692, Springer, 1999, p. 130–135.
3. www.computer.org/publications/dlib/.
4. P. Makagonov P., M. Alexandrov, K. Sboychakov. A toolkit for development of the domain-oriented dictionaries for structuring document flows. In: H.A. Kiers *et al*. (Eds.), *Data Analysis, Classification, and Related Methods*, Springer, 2000. Studies in classification, data analysis, and knowledge organization, pp. 83–88.
5. P. Makagonov, A. Ruiz Figueroa. Study of Knowledge Evolution in Parallel Computing by Short Texts Analysis. In: *Progress in Pattern Recognition, Image Analysis and Applications*, CIARP2004, Lecture Notes in Computer Science N3287, Springer, 2004.
6. A. Gelbukh, Mikhail Alexandrov, S.Y. Han. Detecting Inflection Patterns in Natural Language by Minimization of Morphological Model. In: *CIARP-2004*. Lecture Notes in Computer Science, N 3287, Springer-Verlag, 2004, p. 432–438.
7. A. Lozano Tello. Métrica de Idoneidad de Ontologías. PhD thesis. Feb. 2002. Dep. Informática, Escuela politécnica de Cáceres. Univ. de Extremadura, Spain.

Resolution of Data Sparseness in Named Entity Recognition Using Hierarchical Features and Feature Relaxation Principle

Guodong Zhou, Jian Su, and Lingpeng Yang

Institute for Infocomm Research, 21 Heng Mui Keng Terrace, Singapore 119613
{zhougd, sujian, lpyang}@i2r.a-star.edu.sg

Abstract. This paper introduces a Mutual Information Independence Model (MIIM) and proposes a feature relaxation principle to resolve the data sparseness problem in MIIM-based named entity recognition via hierarchical features. In this way, a named entity recognition system with better performance and better portability can be achieved. Evaluation of our system on MUC-6 and MUC-7 English named entity tasks achieves F-measures of 96.1% and 93.7% respectively. It also shows that 20K words of training data would have given the performance of 90 percent with the hierarchical structure in the features compared with 30K words without the hierarchical structure in the features. This suggests that the hierarchical features provide a potential for much better portability.

1 Introduction

Named entity recognition is to identify and classify the entity names that occur in each sentence of a document. Generally, the overall algorithm works by considering sentences one at a time as they occur in the document. It is a critical component for information extraction and an important step for other natural language processing applications, e.g. information retrieval, machine translation and language understanding.

During last decade, named entity recognition has drawn more and more attention from the MUC named entity tasks [1, 2]. Previous approaches are mainly rule-based [3, 4, 5, 6]. However, rule-based approaches lack the ability of coping with the problems of robustness and portability.

The current trend is to use the machine-learning approach, which is more attractive in that it is trainable and adaptable. Representative machine-learning approaches include HMM [7, 8, 9], Maximum Entropy [10, 11, 12], Decision Tree [13], Winnow [14], MEMM [15] and Conditional Random Fields [16]. Among these approaches, the evaluation performance of HMMs, MEMMs and CRFs is higher than those of others. The main reason may be due to its better ability of capturing the locality of phenomena, which indicates names in text. Moreover, HMMs, MEMMs and CRFs seem more and more used in named entity recognition because of the efficiency of the decoding algorithms, e.g. the Viterbi algorithm [17] in HMM. Therefore, this paper will focus in this direction.

This paper introduces a Mutual Information Independence Model (MIIM) and a feature relaxation principle to resolve the data sparseness problem in MIIM-based

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 750 – 761, 2005.

named entity recognition. Moreover, various features are structured hierarchically to facilitate the feature relaxation process. In this way, the data sparseness problem in named entity recognition is resolved effectively and a named entity recognition system with better performance and better portability is achieved.

The layout of this paper is as follows. Section 2 introduces the Mutual Information Independence Model (MIIM). Section 3 describes its application in named entity recognition and various hierarchical features. Section 4 presents the feature relaxation principle to resolve the data sparseness problem. Section 5 gives the experimental results of our system. Section 6 concludes our work.

2 Mutual Information Independence Model

Given an observation sequence $o_1^n = o_1 o_2 \cdots o_n$, the goal of a conditional probability model is to find a stochastic optimal state (tag) sequence $s_1^n = s_1 s_2 \cdots s_n$ that maximizes $\log p(s_1^n \mid o_1^n)$ [9]:

$$s^* = \arg\max_{s_1^n} \{\log p(s_1^n \mid o_1^n)\} \tag{1}$$

By applying the Bayes' rule, we can rewrite the equation (1) as:

$$\begin{aligned} s^* &= \arg\max_{s_1^n} \{\log p(s_1^n \mid o_1^n)\} \\ &= \arg\max_{s_1^n} \{\log P(s_1^n) + \log \frac{P(s_1^n, s_1^n)}{P(s_1^n) \cdot P(s_1^n)}\} \\ &= \arg\max_{s_1^n} \{\log p(s_1^n) + PMI(s_1^n, o_1^n)\} \end{aligned} \tag{2}$$

Obviously, the second term $PMI(s_1^n, o_1^n)$ captures the pairwise mutual information between the state sequence s_1^n and the observation sequence o_1^n. To compute $PMI(s_1^n, o_1^n)$ efficiently, we propose a novel mutual information independence assumption:

$$PMI(s_1^n, o_1^n) = \sum_{i=1}^{n} PMI(s_i, o_1^n) \text{ or } \log \frac{p(s_1^n, o_1^n)}{p(s_1^n) \cdot p(o_1^n)} = \sum_{i=1}^{n} \log \frac{p(s_i, o_1^n)}{p(s_i) \cdot p(o_1^n)} \tag{3}$$

That is, we assume a state is only dependent on the observation sequence o_1^n and independent on other states in the state sequence s_1^n. This assumption is reasonable because the dependence among the states in the state sequence s_1^n has been directly captured by the first term $\log p(s_1^n)$ in equation (2).

By applying the assumption (3) to the equation (2) and using the chain rule, we have:

$$
\begin{aligned}
s^* &= \arg\max_{s_1^n}\{\sum_{i=2}^{n}\log p(s_i \mid s_1^{i-1}) + \log p(s_1) - \sum_{i=1}^{n}\log p(s_i) + \sum_{i=1}^{n}\log p(s_i \mid o_1^n)\} \\
&= \arg\max_{s_1^n}\{\sum_{i=2}^{n}\log p(s_i \mid s_1^{i-1}) - \sum_{i=2}^{n}\log p(s_i) + \sum_{i=1}^{n}\log p(s_i \mid o_1^n)\} \\
&= \arg\max_{s_1^n}\{\sum_{i=2}^{n} PMI(s_i, s_1^{i-1}) + \sum_{i=1}^{n}\log p(s_i \mid o_1^n)\}
\end{aligned} \tag{4}
$$

The above model consists of two models: the state transition model $\sum_{i=2}^{n} PMI(s_i, s_1^{i-1})$ measuring the state dependence of a state given the previous states in a generative way, and the output model $\sum_{i=1}^{n}\log p(s_i \mid o_1^n)$ measuring the observation dependence of a state given the observation sequence in a discriminative way. This is done by assuming a novel mutual information independence as shown in equation (3). Therefore, we call the above model as shown in equation (4) a Mutual Information Independence Model (MIIM). The MIIM separates the dependence of a state on the previous states and the observation sequence. On the one hand, the state transition model of a MIIM takes advantage of the HMM on modeling sequential states. On the other hand, the output model of a MIIM directly captures the context dependence between successive observations in determining the states. In this sense, the output model also takes advantage of a discriminative Markov Model, such as MEMM [18] and CRF [19], on incorporating arbitrary overlapping features.

3 Named Entity Recognition

Given an observation sequence $o_1^n = o_1 o_2 ... o_n$, the MIIM finds the most likely tag (state) sequence $s_1^n = s_1 s_2 ... s_n$ that maximizes equation (4). Here, $o_i = < f_i, w_i >$, $w_1^n = w_1 w_2 \cdots w_n$ is the word sequence and $f_1^n = f_1 f_2 \cdots f_n$ is the feature sequence. s_i is a structural NE-chunk tag and consists of:

- Boundary Category: BC = {O, B, M, E}. Here O means that current word is a whole entity and B/M/E means that current word is at the Beginning/in the Middle/at the End of an entity.
- Entity Category: EC. This is used to denote the class of the entity name.
- Word Formation Pattern Feature: WF. This feature captures capitalization, digitalization and other word formation information (Please see section 3.1 for more details). Because of the limited number of boundary and entity categories, the word formation pattern feature is added into the structural tag to represent a more accurate state transition model in the MIIM. That is, the structural tag is factored by the word formation pattern feature to achieve a more detailed and powerful state transition model in the MIIM.

The idea behind our model is that we try to assign each observation an appropriate tag, which contains boundary and class information. For example, "*Bill Gates is the chief engineer of Microsoft Corp.*". The tag assigned to the token "*Bill*" should indicate that it is at the beginning of an entity name and it belongs to the "*Person*" class;

and the tag assigned to the token "is" should indicate that it does not belong to an entity name. Here, a variant of the Viterbi algorithm [17] is implemented to find the most likely tag sequence with the new state transition model and the new output model as in equation (4).

As stated above, any observation o_i is denoted as an ordered pair of word w_i itself and its related feature set f_i: $o_i =< f_i, w_i >$. f_i consists of several features, which are either found within the word and/or word string to capture internal evidence or derived within the whole document context of an entity name to capture external evidence. Moreover, each of the features is classified hierarchically to deal with the data sparseness problem and can be represented by any node in its hierarchy.

3.1 f^1 : Word Formation Pattern

The purpose of this feature is to capture capitalization, digitalization and other word formation information. f^1 is the basic feature and organized into two levels. Table 1 shows it with the descending order of priority, For example, in the case of non-disjoint feature classes such as "FirstWord" and "InitialCap", the former will take precedence. This kind of feature has been widely used in machine-learning systems, such as BBN's IdendiFinder and New York Univ.'s MENE. The rationale behind this feature is clear:

- Capitalization gives good evidence of entity names in Roman languages;
- Numeric symbols can be grouped into categories.

Table 1. Feature f^1 word formation pattern

Hierarchical structure		Example	Explanation
1st level	2nd level		
ContainDigitAndAlpha		A8956-67	Product Code
YearFormat	TwoDigits	90	Two-Digit year
	FourDigits	1990	Four-Digit year
	YearDecade	90s, 1990s	Year Decade
DateFormat	ContainDigitDash	09-99	Date
	ContainDigitSlash	19/09/99	Date
NumberFormat	ContainDigitComma	19,000	Money
	ContainDigitPeriod	1.00	Money, Percentage
	ContainDigitOthers	123	Other Number
AllCaps		IBM	Organization
ContainCapPeriod	CapPeriod	M.	Person Name Initial
	CapPlusPeriod	St.	Abbreviation
	CapPeriodPlus	N.Y.	Abbreviation
FirstWord		The	First word of sentence
InitialCap		Microsoft	Capitalized Word
LowerCase		will	Un-capitalized Word
Other		$	All other words

3.2 f^2: Semantic Trigger

f^2, as shown in Table 2, is based on the rationale that important trigger words are useful for named entity recognition and can be classified according to their semantics. This feature applies to both single word and multiple words. For example, in the semantic trigger "chief executive official", all the three words have the semantic trigger feature *PrefixPerson2*. In this paper, f^2 is organized into two levels. This set of semantic triggers is collected semi-automatically from inside entity names themselves and their local context of the training data.

Table 2. Feature f^2 semantic trigger

Hierarchical structure		Example	Explanation
1st level	2nd level		
SuffixPERCENT		%	Percentage Suffix
TriggerMONEY	PrefixMONEY	$	Money Prefix
	SuffixMONEY	Dollars	Money Suffix
SpecialDATE	SuffixDATE	Day	Date Suffix
	WeekDATE	Monday	Week Date
	MonthDATE	July	Month Date
	SeasonDATE	Summer	Season Date
PeriodDATE	PeriodDATE1	Month	Period Date
	PeriodDATE2	Quarter	Quarter/Half of Year
	EndDATE	Weekend	Date End
TriggerTime	SuffixTIME	a.m.	Time Suffix
	PeriodTime	Morning	Time Period
PrefixPerson	PrefixPERSON1	Mr.	Person Title
	PrefixPERSON2	President	Person Designation
NamePerson	FirstNamePERSON	Michael	Person First Name
	LastNamePERSON	Wong	Person Last Name
	OthersPERSON	Jr.	Person Name Initial
SuffixLOC		River	Location Suffix
SuffixORG	SuffixORGCom	Ltd	Company Name Suffix
	SuffixORGOthers	Univ.	Other Org Name Suffix
TriggerNumber	Cardinal	Six	Cardinal Numbers
	Ordinal	Sixth	Ordinal Numbers

3.3 f^3: Gazetteer Feature

This feature determines whether and how an entity name candidate occurs in the gazetteers (entity name dictionaries). As shown in Table 3, f^3 is organized into two levels. When the system encounters an entity name candidate (e.g. a word or sequence

of words with initial letter capitalized), the entity name candidate is looked up in the gazetteers to determine if it exists and its entity class. The gazetteer feature is represented as *ENTITY**G**n,* where ***G*** indicates the gazetteer feature; *ENTITY* indicates the class of the matched entity name in the gazetteers and *n* indicates the number of the words in the matched entity name.

Table 3. Feature f^3 : gazetteer feature *ENTITY**G**n*

Hierarchical structure		Example
1st level	2nd level	
DATE***G***	DATE***G****n*	Christmas Day: DATE***G***2
PERSON***G***	PERSON***G****n*	Bill Gates: PERSON***G***2
LOC***G***	LOC***G****n*	Beijing: LOC***G***1
ORG***G***	ORG***G****n*	United Nation: ORG***G***2

Notes: The letter ***G*** indicates the gazetteer feature; *ENTITY* indicates the class of the matched entity name in the gazetteers and *n* indicates the number of the words in the matched entity name.

Instead of collecting entity name lists from the training data, we collect them from public resources: a list of 20 public holidays in several countries, a list of about 5,000 locations from websites such as geohive.com, a list of about 10,000 organization names from websites such as yahoo.com and a list of about 10,000 famous people from websites such as scopesys.com. In literature, various gazetteers have been widely used in named entity recognition systems to improve performance.

3.4 f^4 : Macro Context Feature

This feature determines whether and how an entity name candidate is occurred in the list of entity names already recognized from the document. As shown in Table 4, f^4 is organized into three levels. The rationale behind this feature is the name alias phenomenon that application-relevant entities will be referred to in many ways throughout a given text and thus success of named entity recognition task is conditional on success at determining when one noun phrase refers to the very same entity as another noun phrase.

During decoding, the entity names are recognized sentence by sentence and those already recognized from previous sentences of the document are stored in a list. When the system encounters an entity name candidate (e.g. a word or sequence of words with initial letter capitalized), a name alias algorithm is invoked to dynamically determine if the entity name candidate might be an alias for a previously recognized name in the recognized list and the relationship between them. The macro context feature is represented as *ENTITY**LT**nm,* where ***L*** indicates the locality of the name alias phenomenon; ***T*** indicates the name alias type, e.g. Ident (identity) and LastName (person last name); *ENTITY* indicates the class of the matched entity name in the recognized entity name list; and *n* (optional) indicates the number of the words in the matched entity name and *m* (optional) indicates the number of the words in the entity

name candidate. For example, when the decoding process encounters the word "UN", the word "UN" is proposed as an entity name candidate and the name alias algorithm is invoked to check if the word "UN" is an alias of a recognized entity name by taking the initial letters of a recognized entity name. If "United Nation" is an organization entity name recognized earlier in the document, the word "UN" is determined as an alias of "United Nation" with the macro context feature *ORG**L**Acro2* by taking the two initial letters of the two-word "organization" name "United Nation".

Table 4. Feature f^4 : the external discourse feature *ENTITY**GT**nm*

Hierarchical structure			Example	Explanation
1st level	2nd level	3rd level		
PER***L***	PER***L***FullMatch	PER***L***Ident***n***	Bill Gates: PER***L***Ident2	Full identity person name
		PER***L***Acro***n***	G. D. ZHOU: PER***L***Acro3	Person acronym for "Guo Dong ZHOU"
	PER***L***PartialMatch	PER***L***LastNam***nm***	Jordan: PER***L***LastNam21	Personal last name for "Michael Jordan"
		PER***L***FirstNam***nm***	Michael: PER***L***-FirstNam21	Personal first name for "Michael Jordan"
ORG***L***	ORG***L***FullMatch	ORG***L***Ident***n***	Dell Corp.: ORG***L***Ident2	Full identity org name
		ORG***L***Acro***n***	NUS: ORG***L***Acro3	Org acronym for "National Univ. of Singapore"
	ORG***L***PartialMatch	ORG***L***Partial***nm***	Harvard: ORG***L***tPartial21	Partial match for org "Harvard Univ."
LOC***L***	LOC***L***FullMatch	LOC***L***Ident***n***	New York: LOC***L***Ident2	Full identity location name
		LOC***L***Acro***n***	N.Y: LOC***L***Acro2	Location acronym for "New York"
	LOC***L***PartialMatch	LOC***L***Partial***nm***	Washington: LOC***L***Partial31	Partial match for location "Washington D.C. "

Notes: The letter ***L*** indicates the locality of the name alias phenomenon; the letter ***T*** indicates the name alias type, e.g. Ident (identity) and LastName (person last name); *ENTITY* indicates the class of the matched entity name in the recognized entity name list; and *n* (optional) indicates the number of the words in the matched entity name and *m* (optional) indicates the number of the words in the entity name candidate.

While the above method is useful to detect the inter-sentential name alias phenomenon (i.e. name alias and its full form occurs in the different sentences), it is unable to identify the inner-sentential name alias phenomenon (i.e. name alias and its full form occurs in the same sentence): the inner-sentential abbreviation. In our system, we present an effective and efficient algorithm to recognize the inner-sentential abbreviations more accurately by mapping them to their full forms. We observe that the full form and its abbreviation often occur together via parentheses. Generally, there are two patterns: "full form (abbreviation)" and "abbreviation (full form)". We also observe that the first pattern dominates except the case that the expression inside the parentheses includes at least two words, since an abbreviation normally includes

only one word. Our algorithm is based on the fact that it is much harder to classify an abbreviation than its full form. Generally, the full form is more evidential than its abbreviation to determine its class. The algorithm works as follows: When an abbreviation with parentheses is detected in a sentence, we remove the abbreviation and the parentheses from the sentence. After applying the HIIM-based named entity recognizer to the sentence, we restore the abbreviation with parentheses to its original position in the sentence. Then, the abbreviation is classified as the same class of the full form, if the full form is recognized as an entity name. Finally, the full form and its abbreviation are stored in the recognized list of entity names from the document to help the resolution of forthcoming occurrences of the same full form and abbreviation in the document.

4 Feature Relaxation Principle

While the first item in equation (4) can be estimated using ngram (e.g. bigram, as used in this paper) modeling [20], the main problem in the above equation is how to effectively and efficiently estimate the second item $\sum_{i=1}^{n} \log P(s_i \mid O_1^n)$. Ideally, we would have sufficient training data for every event whose conditional probability we wish to calculate. Unfortunately, there is rarely enough training data to compute accurate probabilities when decoding on new data, especially when a complex feature set is considered. Generally, two smoothing approaches [20] are applied to resolve this problem: linear interpolation [21] and back-off [22, 23, 24, 8]. However, these two approaches only work well when the number of different information sources is limited. When a few features and/or a long context are considered, the number of different information sources is exponential. To resolve this problem, this paper proposes a dynamic back-off modeling algorithm using a feature relaxation principle. Here, a feature to be relaxed can be any of f^1, f^2, f^3, f^4 and w (the subscripts are omitted as described in Section 2) in the observation sequence O_1^n. However, faced with the large number of ways in which the features could be relaxed, the challenge is how to avoid intractability and keep efficiency. In this paper, three restrictions are proposed to keep the feature relaxation process tractable and manageable:

– Feature relaxation is done through iteratively moving up the hierarchy of the feature. The feature is dropped entirely from the pattern when reaching the root of the hierarchy.
– We assume $P(s_i \mid O_1^n) \approx P(s_i \mid E_i)$, where the pattern $E_i = o_{i-2}o_{i-1}o_i o_{i+1}o_{i+2}$. That is, we only consider the context in a window of 5 words. Here, an observation $o_i =< f_i, w_i >$, w_i is the current word itself and $f_i =< f_i^1, f_i^2, f_i^3, f_i^4 >$ is the set of the four features as described in Section 3. For convenience, we denote $P(\bullet \mid E_i)$ as the probability distribution over various NE-chunk tags given the pattern E_i and $P(s_i \mid E_i)$ as the probability of tag s_i given E_i.

– A pattern E_i should conform to a valid form set of feature conjunctions, which is decided by *ValidEntryForm* and *ValidFeatureForm*. Here, *ValidEntryForm* defines possible forms of feature conjunctions over the observation context while *ValidFeatureForm* defines possible forms of feature conjunctions over a given observation. In this paper, *ValidEntryForm* is set as { $f_{i-2}f_{i-1}f_iw_i$, $f_{i-1}f_iw_if_{i+1}$, $f_iw_if_{i+1}f_{i+2}$, $f_{i-1}f_iw_i$, $f_iw_if_{i+1}$, $f_{i-1}w_{i-1}f_i$, $f_if_{i+1}w_{i+1}$, $f_{i-2}f_{i-1}f_i$, $f_{i-1}f_if_{i+1}$, $f_if_{i+1}f_{i+2}$, f_iw_i, $f_{i-1}f_i$, f_if_{i+1}, f_i } while *ValidFeatureForm* is set as $\{<f_k^1,f_k^2,f_k^3,f_k^4>,<f_k^1,\Theta,f_k^3,\Theta\}>$, $<f_k^1,\Theta,\Theta,f_k^4>$, $<f_k^1,f_k^2,\Theta,\Theta>$, $<f_k^1,\Theta,\Theta,\Theta>$ }. Θ means empty (dropped or not available). Obviously, *ValidEntryForm* and *ValidFeatureForm* define the valid forms of feature conjunctions.

Given the above feature relaxation principle, we recast the problem of estimating $P(\bullet \mid E_i)$ as finding an optimal frequently occurred pattern E_i^0 which can be used to reliably replace $P(\bullet \mid E_i)$ with $P(\bullet \mid E_i^0)$. This is done by a dynamic back-off modeling algorithm using the feature relaxation principle.

The dynamic back-off modeling algorithm solves the problem by iteratively relaxing a feature in the initial pattern E_i until a near optimal frequently occurred pattern E_i^0 is reached. Here, all the frequently occurred patterns are extracted exclusively from the training data and stored in *FrequentEntryDictionary*. If E_i occurs in *FrequentEntryDictionary*, we just return E_i as E_i^0. Otherwise, a valid set of patterns $C^1(E_i)$ can be generated by relaxing one of the features in the initial pattern E_i. If no pattern in $C^1(E_i)$ occurs in *FrequentEntryDictionary*, a further valid set of patterns $C^2(E_i)$ can be generated by relaxing one of the features in each pattern of $C^1(E_i)$. The process continues until E_i^0 is found. If yes, we can choose the one which maximizes the likelihood measure, and return it as E_i^0. Here, the likelihood of a pattern is determined heuristically by the number of features f^2, f^3 and f^4 in it. The rationale comes from that f^2, f^3 and f^4 are more informative in determining named entities than f^1 and w.

5 Experimental Results

In this section, we will report the experimental results of our system for English named entity recognition on MUC-6 and MUC-7 shared tasks. For each experiment, we have the MUC formal training data as the only training data, the MUC dry-run data as the held-out development data and the MUC formal test data as the held-out test data.

Table 5 shows the performance of our system using MUC evaluation while Figure 1 measures the portability of our system and its possible performance improvement if a much larger training corpus is available. Table 5 shows that our system achieves the precision/recall/F-measure of 96.1%/96.2%/96.1 and 93.7%/93.8%/93.7 on MUC-6 and MUC-7 respectively. Figure 1 shows that further increasing the size of the training data beyond 120K words only slightly improves the performance. It also shows that 20K words of training data would have given the performance of 90% while reducing to 10K words would have had a significant decrease in the performance. Figure 1 also shows the contribution of the hierarchical structure in the features. It shows that 30K words of training data would have given the performance of 90 percent without the hierarchical structure in the features (only considering the flat features in the lowest levels). Although the systems with or without the hierarchical structure in the features perform similarly given a large training data, the system with the hierarchical feature structure performs much better than the system with the flat feature structure given a small training data. This means that the hierarchical features provide a potential for much better portability.

Table 5. Performance of our System on MUC-6 and MUC-7 Named Entity Tasks

	F-measure	Precision	Recall
MUC-6	96.1	96.1	96.2
MUC-7	93.7	93.7	93.8

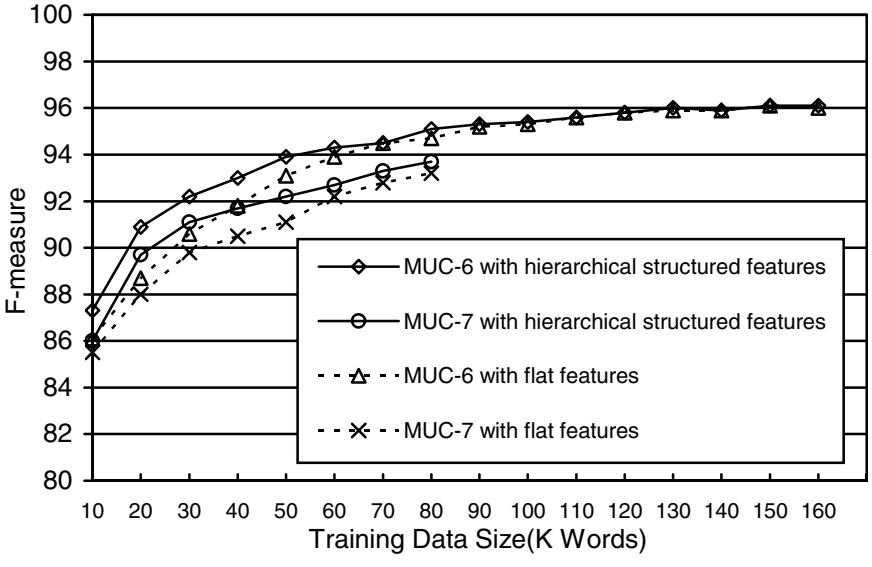


Fig. 1. Impact of training data size on MUC-6 and MUC-7 tasks

7 Conclusion

This paper proposes a feature relaxation principle to resolve the data sparseness problem in MIIM-based named entity recognition. Using the feature relaxation principle, a dynamic back-off modeling algorithm is developed. The idea of using feature relaxation for dealing with data sparseness is not new. However, to our knowledge, our system is the first to integrate it in named entity recognition. Moreover, various features are structured hierarchically to facilitate the feature relaxation process. It shows that the system with the hierarchical feature structure performs much better than the system with the flat feature structure given a small training data. This means that the hierarchical features provide a potential for much better portability. In this way, the data sparseness problem in named entity recognition is resolved effectively and a named entity recognition system with better performance and better portability is achieved.

In the future work, we will explore the dynamic back-off modeling facility via the feature relaxation principle in other applications, e.g. a more challenging problem in extraction of templates or events.

References

1. Chinchor N. 1995a. MUC-6 Named Entity Task Definition (Version 2.1). *Proceedings of the Sixth Message Understanding Conference (MUC-6)*. Columbia, Maryland.
2. Chinchor N. 1998a. MUC-7 Named Entity Task Definition (Version 3.5). *Proceedings of the Seventh Message Understanding Conference (MUC-7)*. Fairfax, Virginia.
3. Aone C., Halverson L., Hampton T. and Ramos-Santacruz M. 1998. SRA: Description of the IE2 System Used for MUC-7. *Proceedings of the Seventh Message Understanding Conference (MUC-7)*. Fairfax, Virginia.
4. Krupka G. R. and Hausman K. 1998. IsoQuest Inc.: Description of the NetOwl™ Extractor System as Used for MUC-7. *Proceedings of the Seventh Message Understanding Conference (MUC-7)*. Fairfax, Virginia.
5. Mikheev A., Grover C. and Moens M. 1998. Description of the LTG System Used for MUC-7. *Proceedings of the Seventh Message Understanding Conference (MUC-7)*. Fairfax, Virginia.
6. Mikheev A., Moens M. and Grover C. 1999. Named entity recognition without gazeteers. Proceedings of the Ninth Conference the European Chapter of the Association for Computational Linguistics (EACL'1999). Pages1-8. Bergen, Norway.
7. Miller S., Crystal M., Fox H., Ramshaw L., Schwartz R., Stone R., Weischedel R. and the Annotation Group. 1998. BBN: Description of the SIFT System as Used for MUC-7. *Proceedings of the Seventh Message Understanding Conference (MUC-7)*. Fairfax, Virginia.
8. Bikel D. M., Schwartz R. and Weischedel R.M. 1999. An Algorithm that Learns What's in a Name. *Machine Learning* (Special Issue on NLP). 1999.
9. Zhou GuoDong and Su Jain. 2002. Named Entity Recognition Using a HMM-based Chunk Tagger, Proceedings of the fortieth Annual Meeting of the Association for Computational Linguistics (ACL'2002). Philadelphia.
10. Borthwick A., Sterling J., Agichtein E. and Grishman R. 1998. NYU: Description of the MENE Named Entity System as Used in MUC-7. *Proceedings of the Seventh Message Understanding Conference (MUC-7)*. Fairfax, Virginia. 1998.

11. Borthwick A. 1999. A Maximum Entropy Approach to Named Entity Recognition. *Ph.D. Thesis.* New York University.
12. Chieu Hai Leong and Ng Hwee Tou. Named Entity Recognition: A Maximum Entropy Approach Using Global Information. *Proceedings of the 19th International Conference on Computational Linguistics (COLING 2002)*. Pages190-196. Taipei.
13. Bennett S.W., Aone C. and Lovell C. 1996. Learning to Tag Multilingual Texts Through Observation. *Proceedings of the First Conference on Empirical Methods on Natural Language Processing (EMNLP'1996)*. Pages109-116. Providence, Rhode Island.
14. Zhang T. and Johnson D., A Robust Risk Minimization based Named Entity Recognition System. *Proceedings of CoNLL-2003*, Edmonton, Canada, 2003, pp. 204-207.
15. Klein D., Smarr J., Nguyen H. and Manning C.D., Named Entity Recognition with Character-Level Models. *Proceedings of CoNLL-2003*, Edmonton, Canada, 2003, pp. 180-183.
16. McCallum A. and Li W. 2003. Early results for Named Entity Recognition with Conditional Random Fields, Feature Induction and Web-Enhanced Lexicons. *Proceedings of CoNLL-2003*, Edmonton, Canada, 2003, pp. 188-191.
17. Viterbi A.J. 1967. Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm. *IEEE Transactions on Information Theory.* IT(13). Pages260-269.
18. McCallum A. Freitag D. & Pereira F. 2000. Maximum entropy Markov models for information extraction and segmentation. ICML-19. 591-598. Stanford, California.
19. Lafferty J. McCallum A & Pereira F. 2001. Conditional random fields: probabilistic models for segmenting and labeling sequence data. *ICML-20.*
20. Chen and Goodman. 1996. An Empirical Study of Smoothing Technniques for Language Modeling. In *Proceedings of the 34th Annual Meeting of the Association of Computational Linguistics (ACL'1996).* pp310-318. Santa Cruz, California, USA.
21. Jelinek F. 1989. Self-Organized Language Modeling for Speech Recognition. in *Readings in Speech Recognition*. Alex Waibel and Kai-Fu Lee(Editors). Morgan Kaufmann. 450-506.
22. Katz S.M. 1987. Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer. *IEEE Transactions on Acoustics. Speech and Signal Processing*. 35. 400-401.
23. Collins M. and Brooks J., 1995. Prepositional Phrase Attachment through a Backed-Off Model. *Proceedings of the Third Workshop on Very Large Corpora*
24. Roth D. and Zelenko D. 1998. Part of Speech Tagging Using a Network of Linear Separators. COLING-ACL'98, pp.1136-1142. Montreal, Canada.

Learning Named Entity Recognition in Portuguese from Spanish

Thamar Solorio and Aurelio López López

Instituto Nacional de Astrofísica Óptica y Electrónica,
Luis Enrique Erro # 1, Santa María Tonantzintla, Puebla, México 72840
{thamy, allopez}@inaoep.mx

Abstract. We present here a practical method for adapting a NER system for Spanish to Portuguese. The method is based on training a machine learning algorithm, namely a C4.5, using internal and external features. The external features are provided by a NER system for Spanish, while the internal features are automatically extracted from the documents. The experimental results show that the method performs well in both languages Spanish and Portuguese.

1 Introduction

Named entities are sequences of words that refer to a concrete entity such as person, organization, location, date and measure [1]. Named Entity Recognition (NER) consists in determining the boundaries of named entities, and even though this task is trivial for a human, the same cannot be said about making computer programs to perform this task. However, it is important to have accurate methods as Named Entities (NEs) can be valuable in several natural language applications. For instance, automatic text summarization systems can be enriched by using NEs, as they provide important cues for identifying relevant segments in text. Other uses of NE taggers are in the fields of information retrieval (i.e. more accurate Internet search engines), automatic speech recognition, question answering and machine translation.

There has been a lot of work in NER, but most approaches are targeted to specific languages, moreover, some are suitable only to narrow domains within that language. We believe this is an important disadvantage, specially considering the fact that all efforts are aimed at developing tools for a handful of languages. In this paper we present results of adapting a NE extractor for Spanish to Portuguese. Our method is based on training a machine learning classifier with the output of the NE extractor and additional lexical attributes. The experimental results are promising and represent an important advance towards cross language NER.

We begin by describing our learning scenario in section 2. We continue presenting some experimental results in section 2.3, where we compare performance of our method when applied to Spanish and Portuguese corpora. In section 3 we describe some related work and then we conclude in section 4.

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 762–768, 2005.

2 Named Entity Recognition

One of our goals is to develop a method that facilitates the adaptability of a NER system to a new domain or language. In this setting, we assume that we have available one NER system (in this case one that is targeted for Spanish) and we want to adapt it so it can be capable of performing NER accurately in documents from a different language, namely Portuguese. It is important to note here that we try to avoid the use of complex and costly linguistic tools or techniques, apart from the existing NER system, given the language restrictions they pose. Although, we do need a corpus of the target language. However, we consider the task of gathering a corpus much easier and faster than that of developing linguistic tools such as parsers, part-of-speech taggers, grammars and the like.

In our approach NER is tackled as a learning task. The features used as attributes are automatically extracted from the documents. For each word we combined two types of features: internal and external; we consider as internal features the following: the word itself, orthographic information and the position in the sentence; additionally we used some external features that are provided by the NER system for Spanish, and these are the POS tag and the NE tag (where these tags use the BIO scheme explained below). Then, the attributes for a given word w are extracted using a window of three words anchored in the word w, each word described by the internal and external features mentioned previously.

Within the orthographic information we consider 6 possible states of a word. A value of 1 in this attribute means that the letters in the word are all capitalized. A value of 2 means the opposite: all letters are lower case. The value 3 is for words that have the initial letter capitalized. 4 means the word has digits, 5 is for punctuation marks and 6 refers to marks representing the beginning and end of sentences.

The most important feature however, is the output of the NER system for Spanish. This system was developed by the TALP research center [2]. They have developed a set of NLP analyzers for Spanish, English and Catalan that include practical tools such as POS taggers, semantic analyzers and NE extractors. This NER system is based on hand-coded grammars, lists of trigger words and gazetteer information.

As in most approaches to NER we used the BIO classification scheme. Here every word in the document must be labeled with one of three tags. The B tag is assigned to words believed to be the beginning of a NE, the I tag is for words that belong to an entity but that are not at the beginning, and the O tag is for all words that do not satisfy any of the previous two conditions. Different from other methods we do not perform binary classifications, and we do not build specialized classifiers for each of the tags. Our classifier learns to discriminate between the three classes and assigns labels to all the words in a single step. In Table 1 we present an example taken from the data used in the experiments were internal and external features are extracted for each word in a sentence. As we can see there are some misclassifications from the NER system for Spanish.

Table 1. An example of the attributes used in the learning setting for NER in Portuguese

	Internal Features			External Features		
Word	**Word**	**Caps**	**Position**	**POS tag**	**BIO tag**	**Class**
Somente	Somente	3	1	NP	B	O
em	en	2	2	VM	O	O
algumas	algumas	2	3	AQ	O	O
localidades	localidades	2	4	NC	O	O
de	de	2	5	SP	O	O
o	o	2	6	CC	O	O
Vale	Vale	3	7	VM	O	B
do	do	2	8	NC	O	I
Paranapanema	Paranapanema	3	9	NP	B	I

This was expected, as Portuguese was not the target language of the system. We still believe that there is useful information provided by the existing system that will help the recognition of named entities in Portuguese.

2.1 The C4.5 Algorithm

C4.5 is an extension to the decision-tree learning algorithm ID3 [3]. Only a brief description of the method is given here, more information can be found in [4]. The algorithm consists of the following steps:

1. Build the decision tree from the training set (conventional ID3).
2. Convert the resulting tree into an equivalent set of rules. The number of rules is equivalent to the number of possible paths from the root to a leaf node.
3. Prune each rule by removing any preconditions that result in improving its accuracy, according to a validation set.
4. Sort the pruned rules in descending order according to their accuracy, and consider them in this order when classifying subsequent instances.

We used the version of C4.5 implemented in the WEKA environment, it is called J48 and is claimed to have some minor improvements over the C4.5 algorithm [5].

2.2 The Data sets

We used two data sets in our experiments. For evaluating NER on Portuguese we downloaded the corpus provided by the Lácio-Web project[1]. This corpus contains newspaper articles published by Folha de São Paulo in 1994. It consists of 1,174,206 words. The data are provided with POS tags, where named entities are labeled as proper names. From these tags we were able to measure accuracy of our system.

[1] This corpus is freely available at http://www.nilc.icmc.usp.br/lacioweb/

The other corpus is that used in the CoNLL 2002 competitions for the Spanish NE extraction task. This corpus is divided in three sets: a training set consisting of 20,308 NEs and two different sets for testing, *testa* which has 4,634 NEs and *testb* with 3,948 NEs, the former was designated to tune the parameters of the classifiers (development set), while *testb* was designated to compare the results of the competitors. As in our setting there is no parameter tuning we performed experiments with the testb set.

These available corpora are considerably large, so we were forced to cut down the size for both languages to probe them in a short time. We selected for experimentation smaller subsets consisting of 20,000 words for each language. In the Portuguese corpus the distribution of classes is as follows: for class O there were 18,012 instances, for class B there were 983 and for class I there were 1,005 instances. In the case of the Spanish corpora there were 17,114 instances of class O while for classes B and I there were 1,602 and 1,283 respectively.

2.3 Evaluation

We present here our experimental results. For all results reported here we show the overall average of several runs of 10-fold cross-validation. We used common measures from information retrieval: precision, recall and F_1 and we present results from individual classes as we believe it is important in learning settings such as this, where nearly 90% of the instances belong to one class.

Table 3 presents results of using our method for the Spanish corpus. That is, all the training and testing are targeted to the Spanish language. We can see that even though the NER system performs very well by itself, by training the C4.5 algorithm on its outputs we improve performance in all the cases, with the exception of precision for class B. In Table 3 we show results of applying our method to the Portuguese corpus. In this case the improvements are much more notorious, particularly for class B, in all the cases the best results are obtained from our technique.

From the results presented above, it is clear that the method can perform NER in Portuguese with very high accuracy, however those results give no indication of the usefulness of the Spanish system for NER in Portuguese. In Tables 4 and 5 we can see comparative results of training C4.5 with only the internal features against using only the external features. As mentioned in Section 2, internal features are those extracted automatically from the documents (the original word, orthographic information, position in the word, etc.) while the external features are the ones provided by the Spanish NER system (POS and BIO tags). In the case of Spanish NER, the features from the NER system (external features) did much better than the internal features. While for Portuguese the opposite occurs, improved results are achieved when using the internal features only. This may be due to the fact that the external features for Spanish are more accurate given that the existing NER system was designed for this language. On the contrary for Portuguese the tags assigned by the NER system are not very accurate, as it is shown in Tableẽft:attr. However, for both languages, the best results are attained when combining internal and external

Table 2. Experimental results for Spanish NER

	NER system for Spanish			Our method		
Class	**Precision**	**Recall**	F_1	**Precision**	**Recall**	F_1
B	92.89%	89.33%	91.07%	92.0%	93.6%	92.8%
I	84.36%	85.22%	84.78%	91.6%	86.5%	89.0%
O	98.67%	98.97%	98.83%	99.1%	99.3%	99.2%

Table 3. Experimental results for Portuguese NER

	NER system for Spanish			Our method		
Class	**Precision**	**Recall**	F_1	**Precision**	**Recall**	F_1
B	58.68%	91.45%	71.48%	87.7%	94.0%	90.8%
I	89.71%	72.93%	80.45%	94.9%	91.6%	93.3%
O	99.03%	97.05%	98.03%	99.5%	99.3%	99.4%

Table 4. Comparison of results for Spanish NER using only the internal features, such as the word and orthographic information, against using features from the NER system for Spanish (external features)

	Internal features			External features		
Class	**Precision**	**Recall**	F_1	**Precision**	**Recall**	F_1
B	71%	87.5%	78.4%	92.9%	89.3%	91.1%
I	90.8%	35.7%	51.3%	84.4%	85.2%	84.8%
O	96.7%	99.2%	97.9%	98.7%	99%	98.8%

Table 5. Comparison of results for NER in Portuguese using only the internal features, such as the word and orthographic information, against using features from the NER system for Spanish (external features)

	Internal features			External features		
Class	**Precision**	**Recall**	F_1	**Precision**	**Recall**	F_1
B	89.2%	85.9%	87.5%	82.2%	83.2%	82.7%
I	90.3%	91.3%	90.8%	86.2%	86.6%	86.4%
O	99.3%	99.5%	99.4%	99.3%	99.2%	99.3%

features, as shown in Tables 2 and 3. Thus, the results suggest that in order to perform NER in Portuguese we do not need an existing NER system for Spanish, but if we have one available, we can use the information provided by it and improve accuracy.

3 Related Work

Spanish resources for NER have been used previously to perform NER on a different language. Carreras et al. presented results of a NER for Catalan us-

ing Spanish resources [1]. They explored several methods for building NER for Catalan. Their best results are achieved using cross-linguistic features. In this method the NER system is trained on mixed corpora and performs reasonably well on both languages. Our work follows Carreras et al. approach, but differs from theirs since we apply directly the NER system for Spanish to Portuguese and train a classifier using the output and the real classes.

There has been a lot of work on NER and classification, and there is a remarkable trend towards the use of machine learning algorithms. For instance, Zhou and Su use Hidden Markov Models where the attributes are a combination of internal features such as gazetteer information, and external features such as the context of other NE already recognized [6].

In [7] a new method for automating the task of extending a proper noun dictionary is presented. The method combines two learning approaches: an inductive decision-tree classifier and unsupervised probabilistic learning of syntactic and semantic context. The attributes selected for the experiments include POS tags as well as morphological information whenever available.

One work focused in NE recognition for Spanish is based on discriminating among different kinds of named entities: core NEs, which contain a trigger word as nucleus, syntactically simple weak NEs, formed by single noun phrases, and syntactically complex named entities, comprised of complex noun phrases. Arévalo et al. focused on the first two kinds of NEs [8]. The method is a sequence of processes that uses simple attributes combined with external information provided by gazetteers and lists of trigger words. A context free grammar, manually coded, is used for recognizing syntactic patterns.

4 Conclusions

We present here a fast and easy method for adapting a NER system for Spanish to Portuguese. Our findings are the following:

- This proposal is a good alternative to develop tools for languages for which linguistic resources are underdeveloped. We believe that similar results can be obtained with other languages, such as Italian.
- Our method can also be applied to adapt the NER system to a different domain of the same language. As the results showed, our method outperformed the existing NER system regardless that the target documents were the same for which the NER system was created. So it is very likely that the same will happen when working in a different domain.
- The internal features suggested in this work are sufficient to train a machine learning algorithm to perform NER in Portuguese. However, the features from the NER system for Spanish in combination with the internal features proved to be very useful for enhancing results.

As work in progress, we are exploring the use of this method to named entity classification, which is a more challenging problem, given that orthographic and lexical features are less helpful. Another research direction is the adaptation of

this method to cross language NER. We are very interested in exploring if, by training a classifier with mixed language corpora, we can perform NER in more than one language simultaneously.

Acknowledgements

We would like to thank CONACyT for partially supporting this work under grants 166934 and U39957-Y.

References

1. X. Carreras, L. Márquez, and L. Padró. Named entity recognition for catalan using spanish resources. In *10th Conference of the European Chapter of the Association for Computational Linguistics (EACL'03)*, Budapest, Hungary, April 2003.
2. X. Carreras and L. Padró. A flexible distributed architecture for natural language analyzers. In *Proceedings of LREC'02*, Las Palmas de Gran Canaria, Spain, 2002.
3. J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.
4. J. R. Quinlan. C4.5: Programs for machine learning. 1993. San Mateo, CA: Morgan Kaufmann.
5. I. H. Witten and E. Frank. *Data Mining, Practical Machine Learning Tools and Techniques with Java Implementations.* The Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, 1999.
6. G. Zhou and J. Su. Named entity recognition using an hmm-based chunk tagger. In *Proceedings of ACL'02*, pages 473–480, 2002.
7. G. Petasis, A. Cucchiarelli, P. Velardi, G. Paliouras, V. Karkaletsis, and C. D. Spyropoulos. Automatic adaptation of proper noun dictionaries through cooperation of machine learning and probabilistic methods. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 128–135. ACM Press, 2000.
8. M. Arévalo, L. Márquez, M.A. Martí, L. Padró, and M. J. Simón. A proposal for wide-coverage spanish named entity recognition. *Sociedad Española para el Procesamiento del Lenguaje Natural*, (28):63–80, May 2002.

A Simple Rule-Based Approach to Organization Name Recognition in Chinese Text*

Wang Houfeng and Shi Wuguang

Institute of Computational Linguistics
School of Electronic Engineering and Computer Science
Peking University, Beijing, 100871, China
{wanghf, shiwuguang}@pku.edu.cn

Abstract. This paper presents a simple rule based approach to organization name recognition in Chinese text. Based on Chinese knowledge sources, our approach detects potential left and right boundaries in a text, and then determines whether a left-right boundary pair encloses an organization name by using a length constraint and non-organization name words/POS-tag constraints. Organization names with nested structure are also processed. This approach is easy to implement and the evaluation results are satisfactory.

1 Introduction

Named entity recognition (NER) is a fundamental component for many NLP applications, such as Information Extraction, Topic Detection and Tracking, Machine Translation and so forth. It is also a subtask of Message Understanding Conference. In recent years, this research has attracted much attention. Various approaches, have been proposed, e.g. maximum entropy model [1, 2], EM bootstrapping [3] and hidden Markov model [4]. While research work has mainly focused on NER in English, the study of NER in non-English has also made great advance [4, 5, 6]. However, organization name recognition in the Chinese text presents a special difficulty in NER, and no good performance in this field has been achieved yet.

In general, an integrated tool based on word segmentation and POS-tagging (Seg-Pos) is used to process many kinds of names, e.g. personal names, location names, date names and so on. Organization names, however, usually need to be processed independently. In [5] the author presented a role-based method by which all words are classified into roles, and as a result, the recognition of an organization name is simply reduced to role tagging. In [4], the author used both rule-based and statistics-based methods to identify organization names.

This paper focuses on multi-word organization name recognition. A one-word name, like 国务院 'State Council', is usually processed by a Seg-Pos tool.

* This work is funded by National Natural Science Foundation of Chinese (No. 60473138).

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 769–772, 2005.

2 Features and Heuristic Rules

Considering the huge number of organization names and the ever-changing list of new organizations, it is not realistic to maintain a fixed list of organization names. A feasible way is to make use of Chinese features to help recognize organization names in Chinese.

2.1 Features

The key to organization name recognition is to identify left and right boundary words of the name. Some features of such words are identified by analyzing 40205 names from *People Daily*. Table1 shows the features of left boundary words. Both tags {ns, nz, nx, nt, m} and word lists can be used to help identify potential left boundary words. Yu explained each tag in detail [7].

Table1. Left boundary words of organization names and their tags, where f, s stand for an orientation (eg. east, north), m for a numerical value (e.g. 18)

	Tag	Ratio(%)	Example
Location name	ns	70.60	中国/ns 科学院/n '**Chinese** Academy of Sciences'
Special name	nz / nx	8.45	丰田/nz 公司/n '**Toyota** Corp.' ＮＥＣ/nx 公司/n '**NEC** Corp.'
Common noun	n	8.19	国家/n 林业局/n '**State** Forestry Bureau'
Abbreviation	j	6.95	北大/j 法律系/n 'Department of Law, **PKU**'
Organization name	nt	2.56	北京大学/nt 数学/n 学院/n 'School of Mathematical Science, **Peking University**'
Others	f, s, m	3.25	南方/f 航空/n 公司/n '**Southern** Airlines Company Limited, China'

Right boundary words can be identified using special POS-tags and Chinese character lists. The character lists are from the last character of organization names, e.g. 署 'office' from 警察署 'Police office'. Such characters are called cue-char, the total number of which is smaller than 100 among the 40205 names.

A multi-word enclosed by a left and a right boundary word is not necessarily an organization name. Some words and POS-tags never occur in an organization name, for example, the words with tag e (exclamation), therefore, we can use such words to filter out candidates for organization names.

In addition, length of an organization name can also be used as a constraint. The length of an organization name varies usually between 2 and 12 words.

2.2 Linguistic Knowledge

Our heuristic rules are based on a set of word lists, cue-character lists and POS-tag lists, as follows:

1. left_n_list: Left boundary words with tag n.
2. left_f_s_list: Left boundary words with tag f or s.
3. left_j_list: Left boundary words with tag j.
4. Right_j_list: cue-chars of Right boundary words with tag j.
5. Right_l_list: cue-chars of Right boundary words with tag l.
6. Right_n_list: cue-chars of Right boundary words with tag n.
7. Right_Ng_list: cue-chars of Right boundary words with tag Ng.
8. Right_nx_list: Right boundary words with tag nx.
9. Right_noun_list: Right boundary words with tag n.
10. Right_Ex_noun_list: Exclusive words with tag n.
11. Inner_word_list: Special inner words with exclusive tag.
12. Inner_Ex_pos: Exclusive tag used to filter out candidate.
13. Right_stop_list: words immediately following right boundary words.

The last characters of words in Right_noun_list are not contained in Right_n_list because they will cause serious ambiguity if treated as cue-chars.

2.3 Heuristic Rules

Hrule-1: **Left boundary.** For a word W, if its tag is ns, nz, nx ($W \notin$ Right_nx_list), n ($W \in$ left_n_list), j($W \in$ left_j_list), or f, s ($W \in$ left_f_s_list), then W is recognized as a potential left boundary word.

Hrule-2: **Right boundary.** For a word W whose last character is C, if its tag is j ($C \in$ Right_j_list), l ($C \in$ Right_l_list), nx($W \in$ Right_nx_list), or, Ng ($C \in$ Right_Ng_list) , then W is recognized as a potential right boundary word.

Hrule-3: **Left or Right boundary for noun.** For a word W whose last character is C, if its tag is n (($C \in$ Right_n_list)$\vee$($W \in$ Right_noun_list)) $\wedge$($W \notin$ Right_Ex_noun)), Then W is recognized as a potential right boundary word. If ($W \in$ left_n_list), then it is a potential left boundary word.

Hrule-4: **Constraint.** For a potential left boundary word W_i and a potential right boundary word W_k which is the closest to W_i, if there exists a W_j (i<j<k) and (($W_j \in$ Inner_Ex_pos$\wedge$ $W_j \notin$ Inner_word_list) $\vee$(k-i>12)), then [W_i W_{i+1} … W_j … W_k] will not be an organization name; otherwise, the sequence of word is tagged as nt, i.e. [W_i W_{i+1} … W_j … W_k]nt.

Hrule-5: **nt tag.** If there is a word W_i with tag nt and there is no left boundary word preceding W_i without matching any right boundary word, then W_i is identified as left boundary word; otherwise, it will be as right boundary word.

Hrule-6: **Nested structure.** If a multi-word [W_i W_{i+1} …W_j …W_k] is recognized as an organization name and the immediate following word $W' \notin$ Right_stop_list, then this multi-word will be taken as a potential left boundary.

The nested structure is very common in organization names. For example, for 北京大学/ 计算机科学技术系/计算语言学研究所 'Institute of Computational Linguistics, Dept. of Computer Science and Technology, Peking Univ.', the structure is **[[** 北京大学/nt 计算机/n 科学/n 技术/n 系/n]**nt** 计算/n 语言学/n 研究所 **]nt**.

3 Evaluation and Discussion

Our approach is mainly based on the rules given above, but more details are considered in our experimental system. For example, the words that stand for number are sometimes left boundary word, but in most cases, they are not. Therefore, they are processed in a special way in our algorithm.

To evaluate our approach, news texts are selected from *People Daily* at random and MET-2 data[1] downloaded as our test-data. 476 organization names are contained in the test-data. MET-2 data was firstly transformed into a plain text by removing XML tags, and all texts were preprocessed using our Seg-Pos tool. After that, our algorithm is applied to recognize multi-word organization names.

Each recognized multi-word name was marked. An organization name is thought of as being correctly recognized if the entire multi-word name is marked.

Our test shows a precision of 80.5%, a recall of 92.2% and F-measure of 86.0%. The result reported by Yu, Zhang & Liu is: precision 89.6%, recall 87.1% and F-measure 88.3% [5]. Both our dataset and theirs are open. Our results are comparable to theirs although our dataset is relatively small as compared to the large set they used. In [4] the author conducted a similar test on a smaller set and the F-measure is 76.6%. Obviously, our approach is the easiest to implement and has the most inexpensive complexity.

References

1. Andrew Borthwick. A Maximum Entropy Approach to Named Entity Recognition. Ph.D. Thesis, New York University, 1999.
2. Chieu Hai Leong, Hwee Tuo Ng. Named Entity Recognition: A Maximum Entropy Approach Using Global Information. In Proceedings of Coling-2002, Taipei, 2002, 190–197.
3. Cucerzan Silviu, David Yarowsky. Language Independent Named Entity Recognition Combining Morphological and Contextual Evidence. In Proc of 1999 Joint SIGDAT Conference on Empirical Methods in NLP & Very Large Corpora, 1999, 90–99.
4. Yanli Zhang, Degen Huang, Lijing Zhang, Yuansheng Yang. Identification of Chinese Organization Names based on Statistics and Rules. In Proceedings of JSCL-2001(Natural Language Understanding and Machine Translation). China, 2001, 233–239.
5. Hongkui Yu, Huaping Zhang, Quan Liu. Recognition of Chinese Organization Name based Role Tagging. In Proceedings of Advances in Computation of Oriental Languages. Beijing, 2003, 79–87.
6. Zhiyong Luo, Rou Song. An integrated and fast Chinese Word Segmentation. In Proceedings of International Chinese Computing Conference, Singapore, 2001, 323-328.
7. Shiwen Yu *et al*. The Grammatical Knowledge-base of Contemporary Chinese—A Complete Specification 2nd Edition. TsingHua University Press, 2003.

[1] See www.itl.nist.gov/iaui/894.02/related_projects/muc/muc_data/muc_data_index.html.

Disentangling from Babylonian Confusion – Unsupervised Language Identification

Chris Biemann and Sven Teresniak

Leipzig University,
Computer Science Institute, NLP Dept.,
Augustusplatz 10/11
04109 Leipzig, Germany
biem@informatik.uni-leipzig.de, knorke@zehnvierzig.org

Abstract. This work presents an unsupervised solution to language identification. The method sorts multilingual text corpora on the basis of sentences into the different languages that are contained and makes no assumptions on the number or size of the monolingual fractions. Evaluation on 7-lingual corpora and bilingual corpora show that the quality of classification is comparable to supervised approaches and works almost error-free from 100 sentences per language on.

1 Introduction

With a growing need for text corpora of various languages in mind, we address the question of how to build monolingual corpora from multilingual sources without providing training data for the different languages.

According to [Pantel et al. 2004], shallow methods of text processing can yield comparable results to deep methods when allowing them to operate on large corpora. The larger the corpus, however, the more difficult it is to ensure sufficient quality. Most approaches in Computational Linguistics work on monolingual resources and will be disturbed or even fail if a considerable amount of 'dirt' (sublanguages or different languages) are contained. Viewing the Internet as the world's largest text corpus, it is difficult to extract monolingual parts of it, even when restricting downloading to country domains or some domain servers.

While some languages can be identified easily due to their unique coding ranges in ASCII or UNICODE (like Greek, Thai, Korean, Japanese and Chinese), the main difficulty arises in the discrimination of languages that use the same coding and some common words, as most of the European languages do.

In the past, a variety of tools have been developed to classify text with respect to its language. The most popular system, the *TextCat Language Guesser* as described in [Cavnar & Trenkle 1994], makes use of the language-specific letter N-gram distribution and can determine 69 different natural languages. According to [Dunning 1994], letter trigrams can identify the language almost error-freely from a text-length of 500 bytes on. Other language identification approaches use short words and common words as features, e.g. [Johnson 1993], or combine both approaches (cf. [Schulze 2000]). For a comparison, see [Grefenstette 1995].

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 773–784, 2005.

All these approaches work in a supervised way: given a sample of each language, the model parameters are estimated and texts are classified according to their similarity to the training texts. But supervised training has a major drawback: The language identifier will fail on languages that are not contained in its training and, even worse, it will mostly have no clue about that and assign some arbitrary language[1].

This work proposes a method that operates on words as features and finds the number of languages as well as the sentences that belong to each language in a fully unsupervised way. Of course, the method is not able to tell the names of the involved languages, but rather groups sentences of similar languages together.

2 Methodological Approach

In this section we describe our methodology. With the use of co-occurrence statistics we construct weighted lexeme graphs built from words as nodes and their associations as edges. A graph algorithm determines sub-graphs with high connectivity (clusters) and assigns class labels to each word. Under the assumption that two words of the same language exhibit more frequent co-occurrence than word pairs of different languages, the graph algorithm will find one cluster for each language. The words in the clusters serve as features to identify the languages of the text collection by using a sentence-based language identifier.

2.1 Sentence-Based Co-occurrence Graphs

The joint occurrence of two or more words within a well-defined unit of information (sentence, document or word window) is called a co-occurrence. For the selection of significant co-occurrences, an adequate measure has to be defined: Our significance measure is based on the Poisson distribution: Given two words *A*, *B*, each occurring *a*, *b* times in sentences, and *k* times together, we calculate the significance *sig(A, B)* of their occurrence in a sentence as follows:

$$sig(A,B) = \frac{x - k\log x + \log k!}{\log n}$$

$$\text{with } n = \text{number of sentences in the corpus,}$$

$$x = \frac{ab}{n}$$

Roughly speaking, this significance measure is the negative logarithm of the probability for seeing at least k joint occurrences if A and B would be statistically independent.

If the significance value is above a certain threshold (we use 0.4), the co-occurrence of *A* and *B* is considered significant. How significant co-occurrences can serve as a data basis for a variety of applications is described in [Biemann et al. 2004a] and [Biemann et al. 2004b].

---

[1] e.g. TextCat (*http://odur.let.rug.nl/~vannoord/TextCat/Demo/*) assigns “Nepali” to texts like “xx xxx x xxx …” and “Persian” to “öö ö öö öööö …”

The entirety of all words having significant co-occurrences can be viewed as nodes in a graph. An edge between two words exists, if their co-occurrence is significant in the text, and the weight of the edge is given by the significance value.

Figure 1 shows the co-occurrence graph calculated from the text of this paper from section 1 until the significance measure formula. For texts that short, significant co-occurrences do not reflect semantic relations as they do in large corpora (see [Quasthoff & Wolff 2002]).

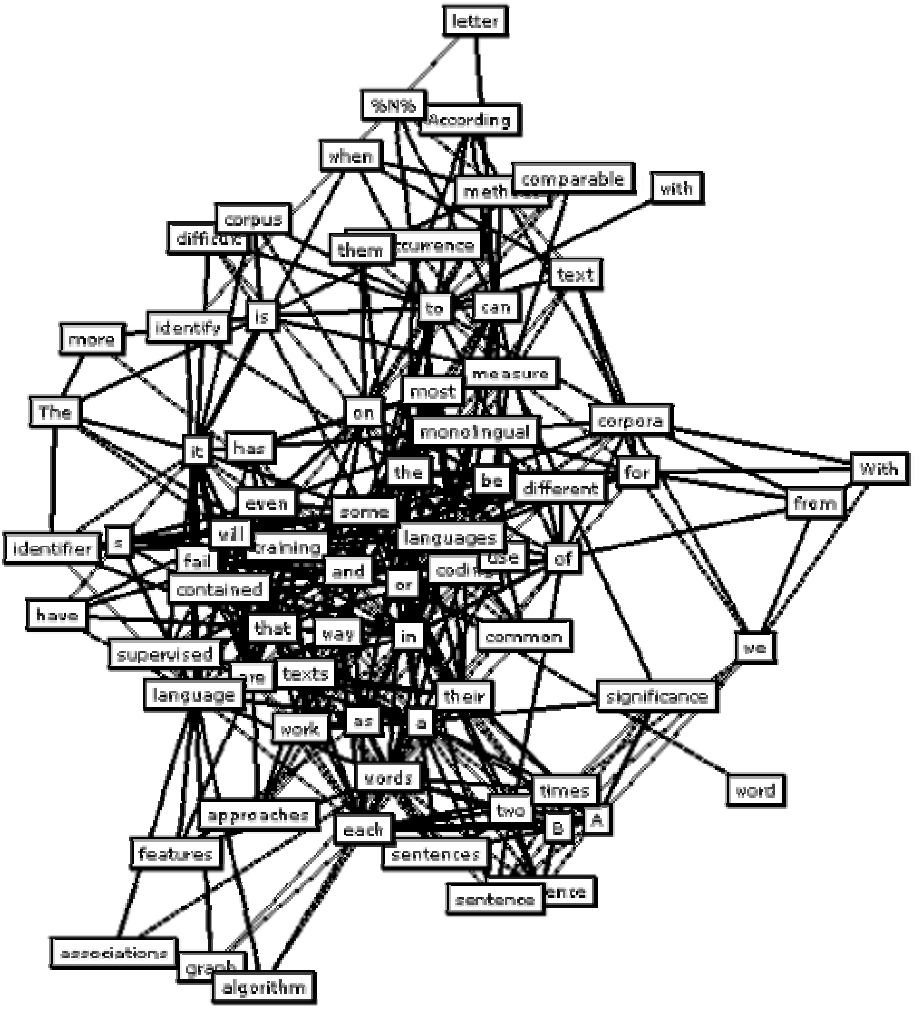


Fig. 1. Co-occurrence graph for the beginning of this document. The visualization software positions associated words close to each other. Note the cluster containing *two, times, A* and *B*. Edge weight is not reflected in the visualization

The structure of co-occurrence graphs can be characterized by the small-world property: a high clustering coefficient paired with short path lengths between nodes and a number of scale-free properties (cf. [Ferrer-i-Cancho & Sole 2001], [Barabási et al. 2000].) As a consequence, there exist sub-graphs that are almost completely connected (clusters), and some hubs (nodes with a high connection degree) that connect those clusters with each other.

When calculating co-occurrence graphs for multilingual corpora, one cluster for each language can be expected (that has in itself again the small world property), while frequent words belonging to different languages serve as hubs.

2.2 Finding Clusters in Co-occurrence Graphs

In the following we describe a graph algorithm called Chinese Whispers,[2] which finds clusters in (co-occurrence) graphs by assigning class labels for each cluster. The name is motivated by the children's game where a message is passed by whispering amongst several children, transforming the message into something funny. In this case, the nodes of the graph whisper their label to each other, until every node agrees with its adjacent nodes on some label.

Figure 2 gives an outline of the algorithm:

```
Assign different labels to every node in the graph;
For iteration i from 1 to total_iterations {
          mutation_rate= 1/ (i^2);
          For each word w in the graph {
            label of w = highest ranked label in
                              neighbourhood of w;
            with probability mutation_rate:
                  label of w = new class label;
          }
}
```

Fig. 2. Outline of the Chinese Whispers algorithm, which finds monolingual clusters in co-occurrence graphs of multilingual corpora

The algorithm begins by assigning different class labels to all words in the graph. In the main loop, the class labels are subject to change: Every word inherits the highest ranked class label in its neighbourhood, which consists of all nodes that are connected to the word in question. Ranking is done using the weights of the edges that are given by the significance value of the co-occurrence: For each class label in the neighbourhood, the sum of the weights of the edges to the word in question is taken as score for ranking. Figure 3 illustrates the change of a class label. Note that all changes are not immediately applied, but take effect in the next iteration.

With increasing iterations, clusters become self-preserving: If a strongly connected cluster happens to be homogenious with respect to class labels, it will never be infected by a few connections from other clusters.

[2] Thanks goes to Vincenzo Moscati for the name of the algorithm.

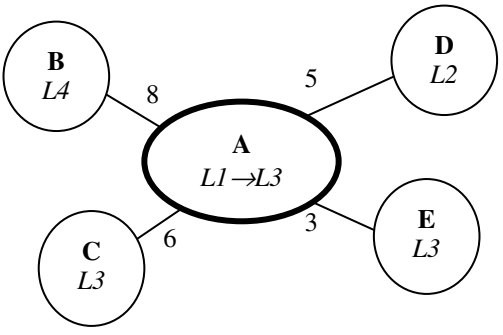


Fig. 3. The class label of word A changes from *L1* to *L3* due to the following scores in the neighbourhood: *L3*:9, *L4*:8 and *L2*:5

To avoid premature convergence and scenarios, where several clusters end up with the same class labels due to strong influence of hubs, a mutation step is included that assigns new class labels with decreasing probability dependent on the iteration. In principle, the algorithm works without mutation on large graphs as well but its performance decreases on small graphs. Although random influence is introduced, only a very small part of the nodes are labeled differently in different runs on the same graph.

The number of total iterations was set to 20 for all experiments, because after 20 iterations the changes in the class labels were only minimal, even in graphs of 1 million nodes and more.

2.3 Sentence-Based Language Identification

For sentence-based language identification we use a tool called "LanI" („Language Identifier") which we developed with the aim to cleanse the sentences and corpora downloaded from the WWW from alien material like source code, sublanguages, foreign languages and so on as described above. LanI was written in Python. Here sentence-based means that LanI returns acceptable results on strings containing at least four or five words. Thereby only statistical data are used, so no semantic or syntactic information is utilized and the well-formedness of sentences is not essential. By designing LanI we assumed, that the probability for a sentence to belong to a given language is computable with the knowledge of the relative frequency of the words of the given sentence in different languages. Considering Zipf's Law [Zipf 1929], LanI gets by with the 250 to 10,000 high-frequency words of given languages. So a word that only appears in one language *L* (accordingly in one language-wordlist), of course increases the probability for a given sentence to be written in that language *L*. Needless to say, word lists are not disjoint, for example in borrowings like the English "kindergarten" and German "Kindergarten" or like "[to] die" in English and "die" (an article) in German. If so, the relative frequency for in both languages matters. For example: the relative frequency for "die" in English is approx. 0.0000367 (0.004% of all words in written English) but around 0.029 (2.9%) in German. Thus, a sentence with "die" contributes more for German (and for English only a little bit).

For the relative frequency for words in different languages we used corpora collected in "Projekt Deutscher Wortschatz", *http://wortschatz.uni-leipzig.de/*). The relative frequencies are regarded as word probabilities $P_L(w)$ for every word w in language L.

After normalisation and smoothing (to avoid multiplication with zero, every word not occurring in a wordlist but in another, gets a very small fixed value) we compute the sentence-probability by multiplying the relative probability of each word. For a sentence S we compute for every language L:

$$P_L(S) = P_L(w_1) \cdot P_L(w_2) \cdot \ldots \cdot P_L(w_s)$$

The result of this is, that we have a probability for every language and now we can easily select the best language. Does the winning language have a likelihood (at least) as twice as large than the second best language, the language for the given sentence is determined. If only ten percent or less of the words of a sentence contained in the wordlists, the calculation will be discarded and LanI reports "unknown".

However, for the algorithm described in this paper we modify the approach a little: for every word in each wordlist the probability is set to a given constant value, thus there is no attempt to use distribution information. For each cluster we got, the word lists consist of the items of each of it. Because every word in the graph is assigned to at most one cluster by the graph algorithm, there are no words contributing to the assignment of two or more languages.

3 Experiments and Evaluation

In order to evaluate our approach, we applied the method to two different settings. A corpus compiled from equal fractions of seven European languages served to determine the lower threshold of how many sentences of each language should be at least contained to make the method work. To find out, how much the amount of monolingual material can differ in biased corpora, experiments on bilingual corpora with different size factors between the two contained languages were performed.

We used the standard Information Retrieval measures for evaluation: *Precision* (P), which is the number of true positives divided by the sum of true positives and false positives, and *recall* (R), which is calculated by dividing the number of true positives through the number of total target items. For some experiments, we provide the harmonic mean of P and R, the *F-value* (F), which is given by (2PR)/(P+R).

In all experiments, clusters were assumed to represent languages if they contained at least 1.8% of all words in the co-occurrence graph. All smaller clusters were regarded as noise. This value was determined empirically by observations on monolingual corpora.

3.1 Multilingual Test Corpora

From a sample of 100'000 sentences each of the languages Dutch, Estonian, English, French, German, Icelandic and Italian we compiled multilingual corpora that contained 100, 200, 500, 1'000, 5'000, 10'000, 50'000 and 100'000 sentences of each language and performed the calculation of co-occurrences. Smaller versions are completely contained in larger versions, average sentence length is 127 characters.

Figure 4 illustrates the number of nodes and edges in the co-occurrence graphs for each 7-lingual corpus.

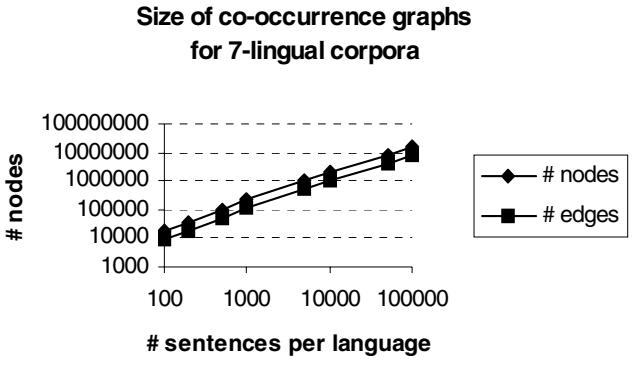


Fig. 4. The sizes of co-occurrence graphs dependent on corpus size. Both axes in logarithmic scale

After applying the graph algorithm, all experiments resulted in seven large clusters of words that ended up with the same class label, as well as many very small clusters that never exceeded 1.7% of the total number of nodes in the graph. Figure A1 and A2 in the appendix illustrate the effect of the graph algorithm on the co-occurrence graph of the smallest 7-lingual corpus.

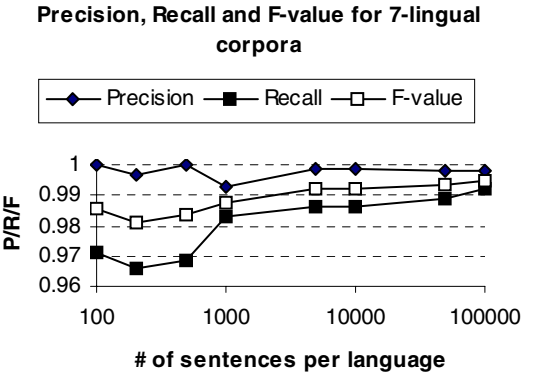
Precision and Recall varied slightly for the different languages. Whereas English was the easiest language to identify (Precision > 99.9%, Recall > 98.1% for all experiments), Estonian was the most difficult (Precision > 99.3% for all experiments, Recall 86.7% for 100 sentences, 92.7% for 200 sentences and > 93.7 for all other experiments). All other languages were classified close to the quality of English.

Figure 5 presents the overall results from the 7-lingual corpora experiments.

For error analysis, we looked at the reasons for low recall in small corpora and checked misclassified items. The main recall flaw was caused by the Estonian parts, which is rooted in the nature of the Estonian corpus. Whereas the other monolingual fractions are taken from newswire sources, our Estonian originates from legal texts. A fraction of about 3% consists of 'sentences' indicating dates or law paragraph ciphers, like "*10.12.96 jõust.01.01.97 - RT I 1996 , 89 , 1590.*" Other unclassified sentences in all languages were mostly enumerations of sport teams or short headlines.

The few misclassified sentences mainly contained proper names, in many cases company names that were usually classified as English. In some cases, the language identifier picked the wrong language for bilingual sentences, like French for "*Frönsku orðin "cinéma vérité" þýða "kvikmyndasannleikur"*." in the Icelandic section.

Experiments with fewer than 100 sentences for each of the seven languages resulted in more than seven (mostly about eleven) clusters that could not group the sentences of one language together via the language identifier. The significant co-occurrences were too small in numbers and too noisy in quality.

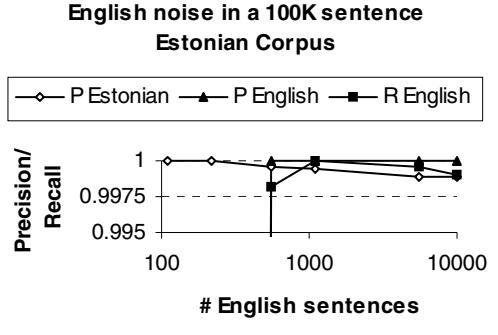


Size	Prec	Rec	Fval
100	1	0.9714	0.9855
200	0.9969	0.9657	0.9810
500	0.9997	0.9684	0.9838
1K	0.9927	0.9828	0.9877
5K	0.9985	0.9859	0.9922
10K	0.9985	0.9865	0.9924
50K	0.9979	0.9889	0.9922
100K	0.9978	0.9922	0.9950

Fig. 5. Results for 7-lingual corpora of different sizes

3.2 Bilingual Corpora

While the experiments in the previous section focussed on equal fractions of many languages, we now describe how the method behaves on bilingual corpora with monolingual fractions of differing sizes. This setting is somewhat more realistic when identifying languages in web data, for a domain usually provides most of its content in one language and translates only a few parts. In order to examine the minimum size of 'dirt' language our method can notice, we performed experiments with biased mixtures of Estonian and English, German and Dutch, Italian and French. The first language contributed the main part comprising of 100'000 sentences, the amount of the second language was varied from 100, 200, 500, 1'000, and in the English-Estonian case 5'000 and 10'000 sentences. It turns out that a second language is identified if it contributes 500 or more sentences. Figure 6 depicts the results for the English-Estonian mixture.

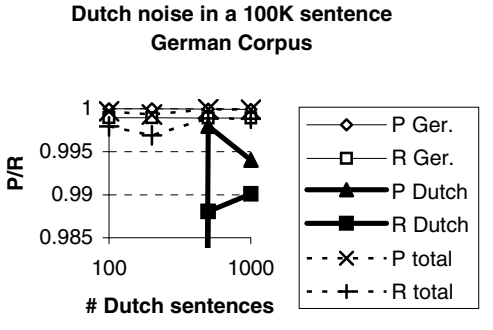


Eng. sent.	Eng P	Eng R	Est P	Est R
100	0	0	1	0.9678
200	0	0	1	0.9674
500	1	0.9982	0.9996	0.9664
1K	1	1	0.9994	0.9674
5K	1	0.9996	0.9989	0.9674
10K	1	0.9991	0.9989	0.9674

Fig. 6. Results for different noise levels of English in a 100'000-sentence Estonian corpus. Estonian recall is not depicted for scaling reasons

The low recall on Estonian was caused by the same reasons as mentioned in the previous section. As we expected, languages from as different families as English and Estonian are very well separable, at least if the size factor is below 200. For less than 500 English sentences, the cluster for English became too small to distinguish it from other small Estonian clusters. Nevertheless, 80% of the English sentences were classified as unknown.

More difficulties were to be expected dealing with language of the same family, like the two Germanic languages German and Dutch. Figure 7 presents the results:

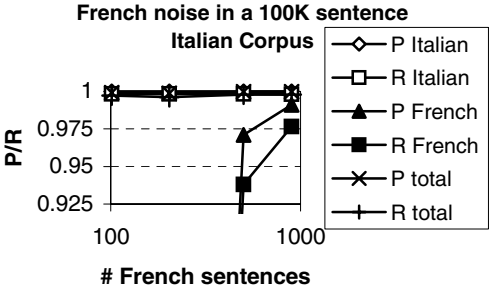


Dut.. sent.	100	200	500	1000
Dut. P	0	0	0.9980	0.9940
Dut. R	0	0	0.9880	0.9901
Ger. P	1	1	0.9999	0.9999
Ger. R	0.9990	0.9990	0.9989	0.9989
Total P	0.9997	0.9993	0.9999	0.9999
Total R	0.9979	0.9969	0.9989	0.9988

Fig. 7. Results for Dutch noise in a 100'000-sentence German corpus

In the two experiments where Dutch could not be identified, 66% of the Dutch sentences were regarded unknown, one third was classified as German.

Two languages that have even more common words were examined in the same setting: the Romanic languages Italian and French, see figure 8. Results differ slightly from the other language pairs: already at 500 sentences of French, figures are decreasing. The similarity of these two languages is also reflected in that about 60% of French sentences were considered to be Italian in the two small-fraction experiments.



Fre. sent.	100	200	500	1000
Fre. P	0	0	0.9710	0.9910
Fre. R	0	0	0.9380	0.9767
Ita. P	1	1	1	0.9999
Ita. R	0.9983	0.9983	0.9983	0.9982
Total P	0.9972	0.9988	0.9999	0.9999
Total R	0.9973	0.9962	0.9980	0.9980

Fig. 8. Results for French noise in a 100'000-sentence Italian corpus

Experiments show that similar languages are only slightly more difficult to separate than languages of different families, as long as the bias towards the main language in the corpus is not larger than 200.

In further experiments, no language pairs were found that could not be separated under the conditions described. Even German dialects could be identified in large German corpora.

4 Conclusion and Further Work

The main contribution of our work is that we show that unsupervised language identification is possible and works with the same high accuracy as the well-known supervised approaches mentioned in the introduction. The only requirement on the language data is the possibility to recognise word boundaries (which even might be replaced by using chunks of fixed length for e.g. Chinese) and sentences (which also might be replaced by using chunks of e.g. 20 words) for the calculation of significant co-occurrences. The method is robust with respect to the number and the mass distribution of the involved languages and produces reliable results from 100 sentences per language on.

When sorting document collections by splitting the documents into sentences, applying our approach and assigning the majority of sentence-languages to the document, there should be virtually no errors.

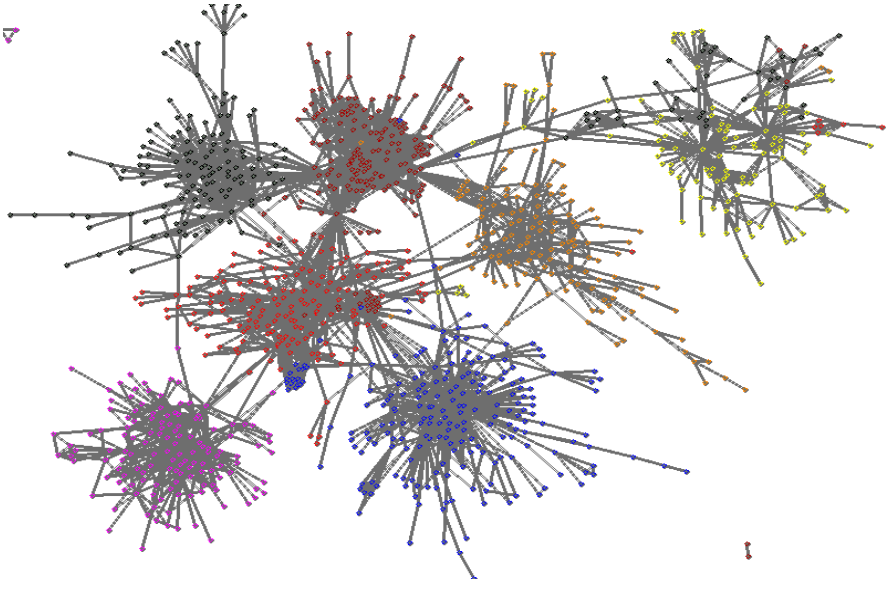


Fig. A1. The co-occurrence graph of the 7-lingual corpus with 100 sentences per language, coloured according to the class labels. Black nodes in the right upper corner do not belong to one of the seven large clusters obtained by Chinese Whispers

In further work, we will examine how the method performs on other document classification tasks, like identification of web genres (cf. [Rehm 2002]) or text classification on standard resources like [Reuters 2000]. For this, we will have to modify the graph algorithm in order to find small clusters that reflect genre-specific wording.

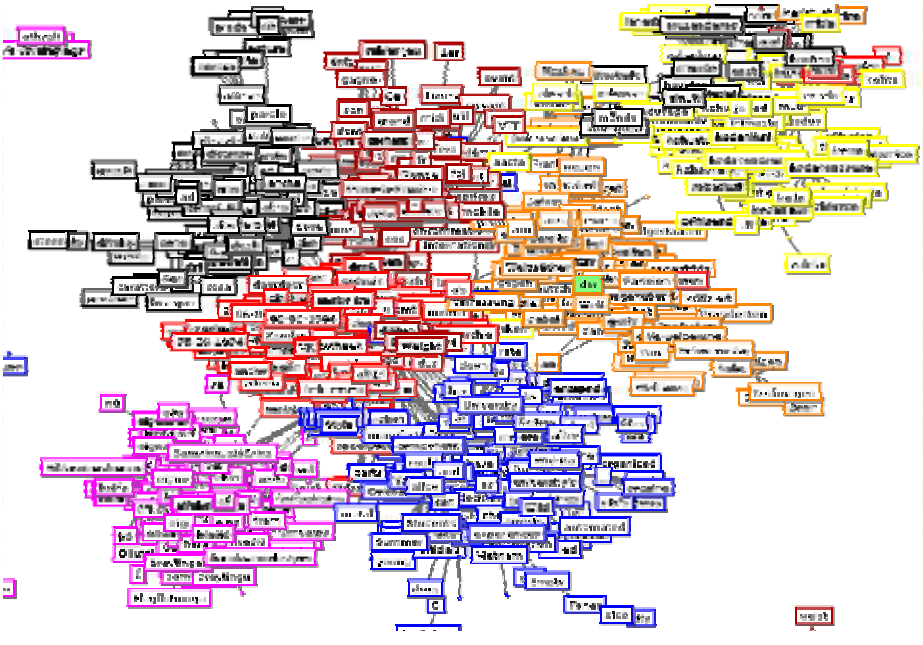


Fig. A2. Lexicalized version of figure A1. The regions of Dutch, English, Estonian, French, German, Icelandic and Italian are clearly visible

References

1. [Barabási et al. 2000] Barabási, A.L., Albert, R., Jeong, H. (2000): Scale-free characteristics of random networks: the topology of the World-wide web, Physica A (281)70-77
2. [Biemann et al. 2004a] Biemann, C., Bordag, S., Heyer, G., Quasthoff, U., Wolff, Chr. (2004): Language-independent Methods for Compiling Monolingual Lexical Data, Proceedings of CicLING 2004, Seoul, Korea and Springer LNCS 2945, pp. 215-228, Springer Verlag Berlin Heidelberg
3. [Biemann et al. 2004b] Biemann, C., Böhm, K., Heyer, G., Melz, R. (2004): Automatically Building Concept Structures and Displaying Concept Trails for the Use in Brainstorming Sessions and Content Management Systems, Proceedings of I2CS, Guadalajara, Mexico
4. [Cavnar & Trenkle 1994] Cavnar, W. B., J. M. Trenkle (1994): N-Gram-Based Text Categorization. In Proceedings of Third Annual Symposium on Document Analysis and Information Retrieval, Las Vegas, NV, UNLV Publications/Reprographics, pp. 161-175

5. [Dunning 94] Dunning, T. (1994): Statistical Identification of Language. in Technical report CRL MCCS-94-273, Computing Research Lab, New Mexico State University, March 1994.
6. [Ferrer-i-Cancho & Sole 2001] Ferrer-i-Cancho, R. and Sole, R. V. (2001) The small world of human language. Proceedings of The Royal Society of London. Series B, Biological Sciences, 268(1482):2261--2265
7. [Grefenstette 1995] Grefenstette, G. (1995): Comparing Two Language Identification Schemes. The proceedings of 3rd International Conference on Statistical Analysis of Textual Data (JADT 95), Rome, Italy, Dec. 1995.
8. [Johnson 1993] Johnson, S. (1993): Solving the problem of language recognition. Technical Report, School of Computer Studies, University of Leeds
9. [Quasthoff & Wolff 2002] Quasthoff, U.; Wolff, C. (2002): The Poisson Collocation Measure and its Applications. In: Proc. Second International Workshop on Computational Approaches to Collocations, Wien.
10. [Pantel et al. 2004] Pantel, P., Ravichandran, D., Hovy, E. (2004) : Towards Terascale Semantic Acquisition, Proceedings of the 20th International Conference on Computational Linguistics (COLING-04), Geneva, Switzerland
11. [Rehm 2002] Rehm, G. (2002): Towards Automatic Web Genre Identification. Proceedings of the 35th Hawaii International Conference on System Sciences, Hawaii.
12. [Reuters 2000] Reuters Corpus (2000). Volume 1, English language, http://about.reuters.com/researchandstandards/corpus
13. [Schulze 2000] Schulze, B.M. (2000): Automatic language identification using both N-gram and word information. US Patent No. 6,167,369.
14. [Zipf 1929] Zipf, G. K. (1929): Relative Frequency as a Determinant of Phonetic Change, Reprinted in Harvard Studies in Classical Philology, Volume XI.

On the Syllabic Similarities of Romance Languages

Anca Dinu[1] and Liviu P. Dinu[2]

[1] University of Bucharest, Faculty of Foreign Languages,
5-7 Edgar Quinet, 70106, Bucharest, Romania
anca_radulescu@yahoo.com
[2] University of Bucharest, Faculty of Mathematics and Computer Science,
14 Academiei, 70109, Bucharest, Romania
ldinu@funinf.cs.unibuc.ro

Abstract. In this paper we study the syllabic similarity between Romance languages via rank distance. The results confirm the linguistical theories, bringing a plus of quantification and rigor.

1 The Syllabic Similarity of Romance Languages

The problem of classifying Romance languages is an intensely studied issue. Unfortunately, in many studies of this kind, the data referring to Romanian are incomplete or even missing (as it happens, for example, in Ziegler, 2000). Here we study the "syllabic" similarity of Romance languages. The work corpus is formed by the representative vocabularies of Romance languages (Latin, Romanian, Italian, Spanish, Catalan, French and Portuguese languages) (Sala, 1988). We syllabified the vocabularies. For each vocabulary we constructed a classification of syllables: on the first position we put the most frequent syllable of the vocabulary, on the second position the next frequent syllable, and so on.

The method we applied in investigating the syllabic similarity of Romance languages is the following: each of the seven Romance languages is compared to the other six (using rank distance (Dinu, 2003)), for each comparison having a graphic as a result. We apply the normalized rank distance between all pairs of such classifications and we obtain a series of results which express the "syllabic" similarity between Romance languages. We also investigate the distances between partial classifications. Each graphic represents the behavior of the function $f_{\Delta}(i)$ with i varying between 1 and 561 (the minimum number of syllables correspondent to the Latin language). The function f_{Δ} expresses the variation of the normalized rank distance between two classifications (see Appendix).

We chose this method for the following reasons: when a listener hears for the first time a language, it is difficult to believe that he is able to distinguish syntactic constructions or even words. In fact, it is more plausible that he can distinguish and individualize syllables; due to this fact, he is able to say to which language (or family of languages) the language he hears is similar.

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 785–788, 2005.

Table 1. The syllable number of Romance languages

Language	The percentage covered by the first ··· syllables						No. syllables	
	100	200	300	400	500	561	type	token
Latin	72%	86%	92%	95%	98%	100%	561	3922
Romanian	63%	74%	80%	84%	87%	90%	1243	6591
Italian	75%	85%	91%	94%	96%	97%	803	7937
Portuguese	69%	84%	91%	95%	97%	98%	693	6152
Spanish	73%	87%	93%	96%	98%	99%	672	7477
Catalan	62%	77%	84%	88%	92%	93%	967	5624
French	48%	61%	67%	72%	76%	78%	1738	5691

In Table 1, we present the number of distinct syllables (*types*) and the number of all the syllables (*tokens*) from every language analyzed. The frequency of the syllables from every language is not uniformly distributed. Table 1 shows the fact that the syllables are distributed according to some principles of the minimum effort type (Zipf 1932, Herdan 1966); thus, a relatively small number of distinct syllables will cover a large part of the corpus. Generally, the first 300 syllables (ordered according to their frequency) cover over 80% (even 90% for some languages) of the number of all the syllables of the corpus. After this number, the percentage increases slowly. The analyze of the graphics in Fig. 1 and Fig. 2 enables us to make some observations, inaccessible otherwise. If we look at the first 300 syllables, Romanian is closest to Italian, followed by Spanish, Portuguese and Catalan. We observe, that the more syllables, the further is Italian from Romanian, whereas Portuguese is closer (however, at the level of all the syllables taking into consideration, Portuguese is the closest to Romanian). At the same time, if we look at the graphic that exhibits the similarities between Italian and the other Romance languages, we observe that Romanian is the furthest; the closest Romance language to Italian is Spanish, situated at a very short distance. This fact is in accordance with the generalized observation that Rumanians understand and learn more quickly Italian, than Italians understand and learn Romanian. In figures 1-2 each graphic represents the behavior of a given language compared to the other 6 languages. However, if we analyze the graphics, we observe that almost every time Romanian finds itself at the biggest distance from the other languages. This fact proves that the evolution of Romanian in a distanced space from the Latin nucleus leaded to bigger differences between Romanian and the rest of the Romance languages, then the differences between any other two Romance languages (at least at the phonological level). Therefore, our study reveals the fact that the syllabic distances between Latin and each of the Romance languages analyzed have very near values. Obviously, our study could be further improved. It would be interesting to study if, in the context of some representative texts belonging to Romance languages, the values of these distances remain the same.

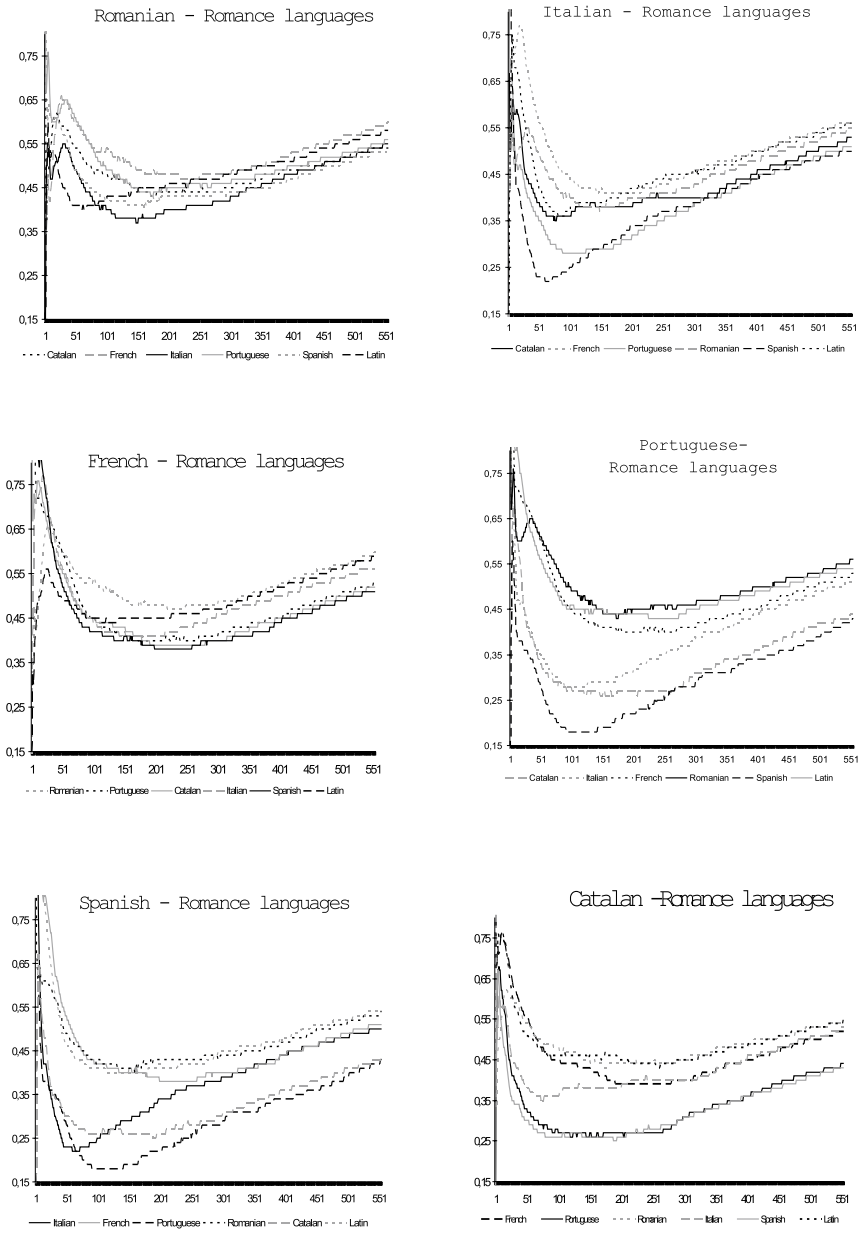


Fig. 1. The similarity of Romance Languages

2 Appendix: Rank Distance

In natural languages, in the framework of lexical units, the most important information is carried by the first part of the unit (Marcus, 1971). By analogy,

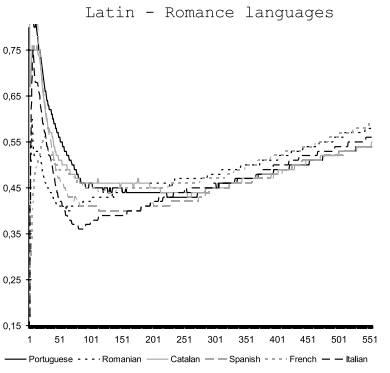


Fig. 2. The Latin - Romance languages similarity

the difference on the first positions between two classifications is more important than the difference on the last positions. This was the starting point in the construction of the rank distance (Dinu, 2003):

Definition 1. *The rank-distance between two classifications L_1 and L_2 is:*

$$\Delta(L_1, L_2) = \sum_{x \in L_1 \cap L_2} |ord(x|L_1) - ord(x|L_2)| + \sum_{x \in L_1 \setminus L_2} ord(x|L_1) + \sum_{x \in L_2 \setminus L_1} ord(x|L_2).$$

where we denoted by $ord(x|L)$ the rank of the element x in classification L.

Let L_1 and L_2 be two classifications having the same length, n. For each $i \in \{1, 2, ..., n\}$ we define the function f_Δ by:

$$f_\Delta(i) \stackrel{\text{def}}{=} \overline{\Delta}(L_1^i, L_2^i) = \frac{\Delta(L_1^i, L_2^i)}{i(i+1)},$$

where L_1^i and L_2^i are classifications of length i obtained from the previous classifications by deleting the elements below position i.

References

1. Dinu, L.P. On the classification and aggregation of hierarchies with different constitutive elements. *Fundamenta Informaticae*, 55, 1, 39-50, 2003.
2. Herdan, G. *The Advanced Theory of Language as Choice and Chance*, Springer, New-York, 1966
3. Marcus, S. Linguistic structures and generative devices in molecular genetics. *Cahiers Ling. Theor. Appl.*, **11**, 77-104, 1974
4. Sala, M. (coord.) *Vocabularul reprezentativ al limbilor romanice*, Bucureşti, 1982.
5. Ziegler, A. Word Length in Romance Languages. A Complemental Contribution. *Journal of Quantitative Linguistics*, 7, 1, 65-68, 2000.
6. Zipf, G.K. *Selected Studies of the Principle of Relative Frequencies in Language*. Cambridge, Mass. 1932.

Automatic Language Identification Using Multivariate Analysis

Vinosh Babu J. and Baskaran S.

AU-KBC Research Centre
{vinosbabu, baskaran}@au-kbc.org

Abstract. Identifying the language of an e-text is complicated by the existence of a number of character sets for a single language. We present a language identification system that uses the Multivariate Analysis (MVA) for dimensionality reduction and classification. We compare its performance with existing schemes viz., the N-grams and compression.

1 Introduction

The rapid growth of the lesser-known languages in the Internet has created a need of language identification for applications like multilingual information retrieval, machine translation, spell checking etc. This task is complicated by three factors viz., the varying sizes of the character sets used to encode different languages, the usage of a variety of character sets for a single language and the same script being shared by more than a language.

The predominant technique used in written language identification is that of identifying short sequences of letters, characterizing a language (N-grams) [6]. These sequences may roughly be thought of as encoding the common, yet unique, character sets of a language. The Canvar's algorithm [6], chiefly aimed for text categorization and language identification is viewed as a task of text categorization with the different languages corresponding to different domains. In this method the accuracy directly varies with the increase in the number of N-gram statistics considered.

Dunning [5] uses Markov models in combination with Bayesian decision rules to develop language models for each language in the training data. These language models are then used to determine the likelihood of a test data generated by a particular model.

Benedetto et. al. [2] developed Shannon's ideas on the entropy of a language by suggesting that the compressibility of a given text depends on the source language. This method uses Lempel-Ziv [3] compression algorithm for identification.

We[1] used a combination of techniques for language identification with the N-grams (using Bayes' rule), Compression and the MVA using Principal Component Analysis (PCA). To our knowledge, this is the first work to use MVA technique for language identification. We show that the MVA method consistently outperformed the N-gram and the compression method in the penultimate section of this paper.

[1] The authors acknowledge contributions of Ramesh Kumar and Viswanathan.

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 789–792, 2005.

2 System Implementation

We have implemented a system based on the N-Grams, Compression and PCA techniques. The system is trained on the training data, separately using these three techniques and the language models are generated. Each of these techniques are then used to individually rank a test document for the different language models in the training corpora. These ranks are then combined into a global rank, based on the positions in each method and the language with the highest global rank is identified as the resultant language. As details on the implementation of N-grams and compression are available from literature, we present the PCA procedure alone in here.

PCA is a multivariate procedure [3, 4], which rotates the data such that the maximum variability is projected onto the axes. Sets of correlated variables get transformed into uncorrelated variables, ordered by the reducing variability. These are linear combinations of the original variables, and the last of these variables can be removed with minimum loss of real data. PCA is mainly used to reduce the dimensionality of a data set, while retaining as much information as possible. In the earlier methods, Bayesian classification was performed on the languages, as the sum of the probabilities in the spectrum of the random variables (in the N-grams) summed up to one. PCA captures information from them all and does a better discrimination by classifying them with respect to the principal components.

Let us consider that there are l languages and N dimensions in which they are represented. Let x_i, i=1, …, l denote the different language vectors formed over the bi-grams and X is the matrix formulated by arranging them as column vectors.

$$\begin{aligned} X &= [x_1^T\, x_2^T \ldots x_l^T] \\ L &= X - x_m \end{aligned} \tag{1}$$

The mean of these vectors are computed using the relation $x_m=\Sigma x_i \,/\, l$. The independent language vectors are separated from this mean and the new matrix L is formulated as can be seen in eqn (1). The eigenvectors for the covariance matrix XX^T is calculated and is termed V_{NxN}. Choose those eigenvectors corresponding to the top few eigenvalues and call the corresponding matrix as $V'_{Nxl'}$. The choice of the value l' depends on the correlation seen between the languages and usually $l' << N$. Using this matrix V', transform the matrix L to get a new matrix E as in eqn (2).

$$\begin{aligned} V_{NxN} &= \text{eigenvector}\,(XX^T) \\ E &= V'^T \text{ x } L \end{aligned} \tag{2}$$

This being an orthogonal transformation, results in the language vector L represented with spatial separation. All the language coordinates are now mapped into a lower dimensional space with a wider separation. A similar transformation can be carried out with the mean separated test language vector l_t (where $l_t = x_t - x_m$). The new coordinate corresponding to l_t after transformation is given by eqn (3).

$$e_t = V'^T \text{ x } l_t \tag{3}$$

Now, to identify the language find the Euclidean distance between the test vector and the other vectors in E and the result will be in favor of the one with the least distance between them, i.e. find,

$$\text{Language identified, } l_i = \arg\min (\text{Euclidean } (e_i, e_t)), \forall\ 1 \le i \le l \quad (4)$$

By using the PCA, the search can be carried on a larger volume of known languages, with a shorter duration, as the dimension on which the search is carried over is far less than the original dimension over which the data exists.

3 Experiment Setup

The languages to be used were carefully chosen to represent a wide variety of the language families, influenced by the availability of the online material in the specific language. The following languages / character-sets are currently supported by the system: Afrikaans, Arabic, Bengali, Bulgarian, Catalan, English, German, French, Hebrew, Hindi, Kannada, Malay, Malayalam, Oriya, Russian, Sanskrit, Sinhala, Slovakian, Spanish, Swahili, Tamil (Iscii, Shree-Tam, Tab, Tam, Tscii, Vikatan), Telugu, Urdu and Vietnamese, categorized into seven different language families.

The data for the different languages were collected from the web using a web-robot from a wide variety of sites. About 2.5 MB to 20MB text data in several files were collected for each language. The data thus obtained were cleaned up to remove unnecessary blank spaces and repeated hyperlinked texts. Around 30 KB of this cleaned data was used for training. The rest of the data was used for testing the system, accounting to around 25 files in each of the different languages. Some of the files used for testing were very small in size and the others had white spaces, which affected the performance of the system accounting to 1-2% of the total test data.

4 Results and Discussion

Table 1 compares the performance of the PCA with the N-gram and compression techniques. The algorithm has been tested for 29 languages / encoding schemes belonging to seven different language families. Failures were identified across the different languages and are tabulated concisely across the language families.

Table 1. Results for the various language families

Techniques	*Language Families*						
	IE	*ID*	*IA*	*AfA*	*AuA*	*AN*	*NC*
N-Gram	88.5	99.59	100	94	100	86	90.12
Compression	82.87	94.6	100	90	85.71	80	86.58
MVA/PCA	96.97	100	100	93.61	100	100	100
Overall	89.45	98.06	100	92.54	95.24	88.67	92.23

Legend:- IE – Indo-European, ID – Indo-Dravidian, IA – Indo-Aryan, AfA – Afro-Asiatic, AuA – Austro-Asiatic, AN - Austronesian, NC – Niger-Congo

The table reports an overall performance close to 99% for the Indian languages. This might be due to the clean corpus that has been used for these languages. The performance is around the 90% for the Indo European, Afro Asiatic, Austro Asiatic

and the Niger Congo family of languages, around 95% for the remaining, which is comparable with the results in the existing language identification techniques. Our analysis shows that for most of the wrongly identified cases the size of the test data was less than 500 bytes, which is too small. Even in cases, where the system makes a mistake, the system always identifies a language within the same language family, like German for a French text. The overall system performance is around 94.5%.

It can be inferred from table-1, that the performance of the PCA technique outperforms the combined performance of the N-gram and the compression methods. With PCA as the single language identification scheme, the system offered a performance of around 98.5%, which is far above the combined performance of the remaining schemes, which account for 91.5%. This is because, PCA removes the statistical dependence between the languages and then classifies as per the unique features in a language. This dual advantage rates the performance of our system better than the other schemes.

5 Conclusion

The language identification system that we have built uses a number of techniques including N-grams, compression, and multivariate analysis over the bi-gram spectral distribution. The current system can identify 24 languages and 29 character sets in 7 language families, with an average accuracy of 94.5%. We observe that the MVA / PCA technique outperforms the N-gram and compression technique with an average accuracy of 98.5 %. This we believe is due to the fact that the language texts are first de-correlated and then the language specific features are identified. We are planning to improve the performance of the system by using other statistical methods and applying the PCA over the tri-grams. We also hope to fuse the individual rank by a better approach to compute the global rank of the system.

References

1. Dario Benedetto, Emanuele Caglioti, and Vittorio Loreto. Language trees and zipping. Physical Review Letters, 88(4), January 2002
2. Jacob Ziv, Abraham Lempel. A Universal Algorithm for Sequential Data Compression. IEEE Transactions on Information Theory 23(3): 337-343 (1977)
3. Bryan F. J. Manly. Multivariate Statistical Methods. A Primer, Chapman & Hall.
4. Singular value decomposition and principal component analysis. In: D.P. Berrar, W. Dubitzky, M. Granzow (eds.) A Practical Approach to Microarray Data Analysis. Kluwer: Norwell, MA, 2003. pp. 91-109. LANL LA-UR-02-4001.
5. Ted Dunning. 1994. Statistical identification of language. Computing Research Laboratory Technical Memo MCCS 94-273, New Mexico State University, Las Cruces, NM
6. W. Canvar and J. Trenkle. N-gram based text categorization, In Proceedings of 3rd Annual Symposium on Document Analysis and Information Retrieval pp 161-176, 1994.

Design and Development of a System for the Detection of Agreement Errors in Basque

Arantza Díaz de Ilarraza, Koldo Gojenola, and Maite Oronoz

Department of Computer Languages and Systems,
University of the Basque Country,
P.O. box 649, E-20080 Donostia
{jipdisaa, jibgogak, jiporanm}@si.ehu.es

Abstract. This paper presents the design and development of a system for the detection and correction of syntactic errors in free texts. The system is composed of three main modules: a) a robust syntactic analyser, b) a compiler that will translate error processing rules, and c) a module that coordinates the results of the analyser, applying different combinations of the already compiled error rules. The use of the syntactic analyser (a) and the rule processor (b) is independent and not necessarily sequential. The specification language used for the description of the error detection/correction rules is abstract, general, declarative, and based on linguistic information.

1 Introduction

The problem of the detection and correction of syntactic errors has been addressed since the early years of natural language processing. Different techniques (Vandeventer, [2003]) have been proposed for the treatment of the significant portion of errors (typographic, phonetic, cognitive and grammatical) that result in valid words (Kukich, [1992]). Although many commercial grammar checkers have been developed (Paggio and Music, [1998]), there is little published work on their implementation or evaluation. This is due in part to the fact that the mechanisms used for the implementation have not been very sophisticated (as in some systems that use a large set of regular expressions) and also that commercial companies are not willing to reveal implementation details about their tools. The aim of the present work is to examine the feasibility of corpus-based syntactic error detection focusing in detecting agreement errors.

The system will be applied to Basque, an agglutinative language with relative free order among sentence elements. In our research group, work in error detection at morphological level has already been accomplished and a spelling checker-corrector (Aldezabal *et al.* [1999]), - called XUXEN - was marketed 10 years ago. Error detection at syntactic level needs of a robust syntactic analyser and we will use the Basque syntactic analyser (Aduriz and Díaz de Ilarraza, [2003]) that was developed using Constraint Grammar (CG) (Karlsson *et al.*, [1995]).

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 793–802, 2005.

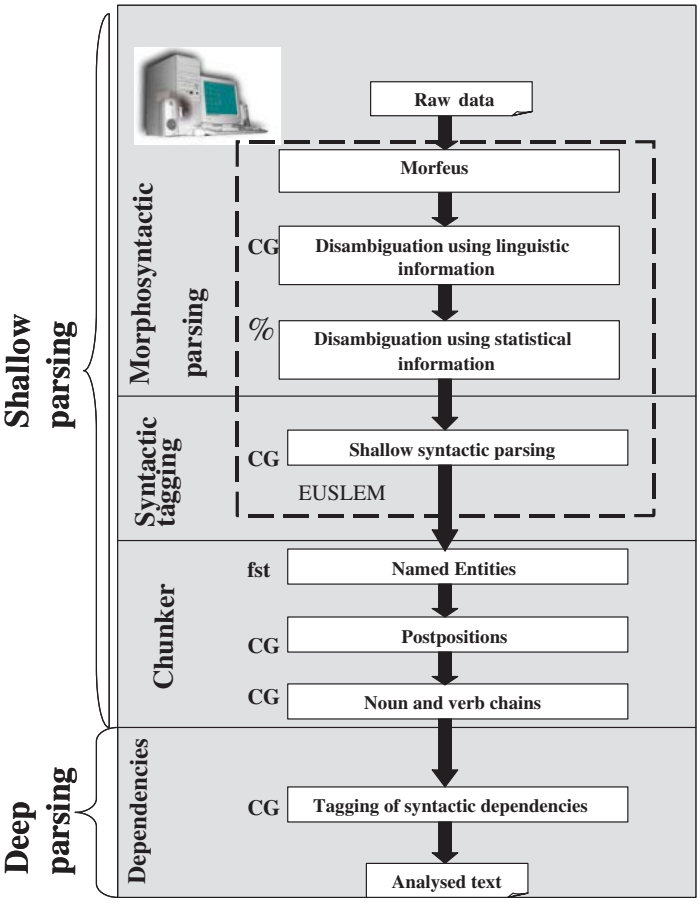


Fig. 1. The multi-layered syntactic analyser for Basque

Figure 1 shows the syntactic analysis chain in which sequential rule layers, most of them materialised in Constraint Grammars, enrich the output of the previous layer with the respective information.

The parsing process starts with the outcome of the morphosyntactic analyser (Morfeus), which was created following the two-level morphology (Koskeniemi, [1983]) and deals with all the lexical units of a text, both simple words and multi-word units. The tagger/lemmatiser EUSLEM, not only obtains the lemma and category of each form but also includes a module for disambiguation. The disambiguation process is carried out by means of linguistic rules (CG) and stochastic rules based on Markovian models (Ezeiza, [2003]). Once we have the morphosyntactic information by means of EUSLEM, the recognition of named entities and post-positional phrases is carried out. The subsequent level of chunking identifies verb and noun chains. The last step in this analysis chain is the identification of dependency relations among the components of the sentence in order to obtain

a dependency syntactic tree. As a result of using the CG formalism, a limited amount of ambiguity remains, which reduces the number of parsing errors. However, this ambiguity is much lower than in other approaches such as CFG-based parsing systems, in which it is usual to encounter hundreds of parses for each sentence that are later discriminated using statistical information (Briscoe and Carroll, [2004]). All the information in the analysis chain is interchanged by means of stardarised XML files (Artola *et al.*, [2004]) and a class library for the management of all the linguistic information.

Not all the information provided by this analysis chain is necessary for the detection of some syntactic errors. For example, we can detect errors in ill composed postpositional phrases using pattern rules defined in CG and information only at morphosyntactic (word) level. Other kinds of errors such as agreement errors need all the available information, and specially the information related to the morphosyntactic analysis of each lexical unit and to the dependencies among the elements of the sentence.

The remainder of this paper is organised as follows. Section 2 describes the general architecture of the error processing system. In section 3 we will analyse the agreement errors found in a set of incorrect sentences. Section 4 describes the state of development of the application and we mention the main problems we will have to tackle. Finally some conclusions are outlined in section 5.

2 Architecture of the System

We have divided our system for agreement error detection in 4 independent modules (see figure 2). Each of the modules is explained in the following paragraphs.

2.1 Syntactic Analysis

The syntactic analysis module that we already presented in figure 1 produces, for each input sentence, one or more dependency trees (see figure 3). Each dependency tree contains in each node information about the morphosyntactic analysis of each lexical unit, as well as the dependency relation with its parent and child nodes (e.g. subject, noun modifier, ...). Figure 3 shows the dependency tree of an incorrect sentence we took as an example. The subject *zentral nuklearrak* (nuclear power station), in the absolutive case, and the auxiliary verb, *dute*, which needs the subject to be in ergative case, do not agree. In the next section we will explain how we can represent this agreement error by means of a rule.

2.2 Error-Rule Compiler

We have defined a general specification language to be used to search for any linguistic structure, correct or not, in a dependency tree and to transform the obtained structure into a different one. Although this language allows to search for/transform sentences that fulfil a specific requirement, in this application we have used it to write error detection/correction rules.

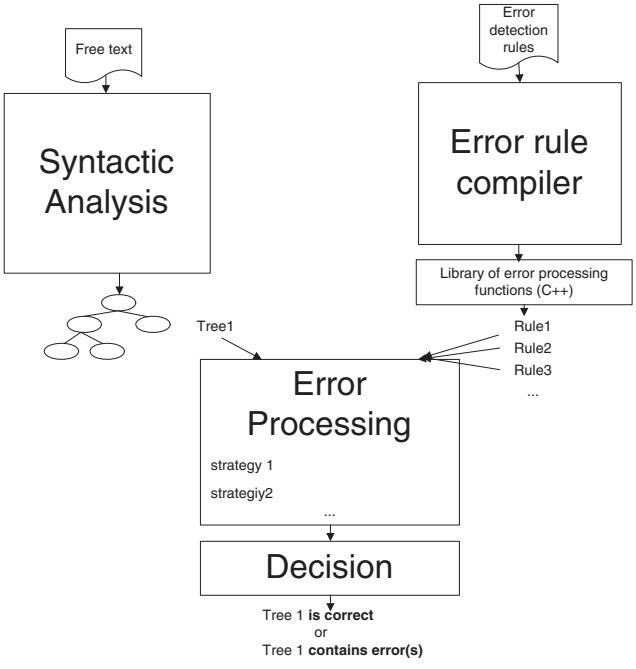


Fig. 2. Architecture of the system

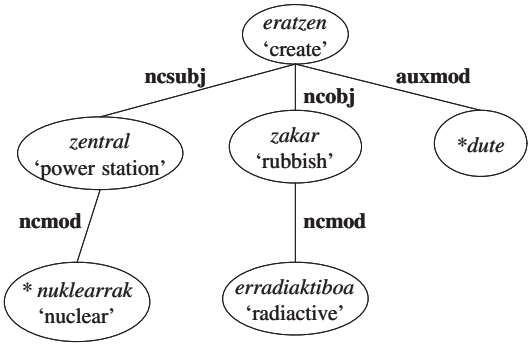


Fig. 3. Dependency tree for the sentence **Zentral nuklearrak zakar erradiaktiboa eratzen dute* (*Nuclear power station create radioactive rubbish)

The use of an abstract specification language has several advantages: a) declarativeness, b) maintainability and, c) efficiency, as the abstract rules will be compiled to an object language (C++).

Each rule contains four different sections (see figure 4): i) *Detect*: detection of the error in the dependency tree, ii) *Correct*: one or more possible ways to correct the error, iii) *Mark*: branches in the tree to be marked to represent the error and, iv) *Info*: message explaining the error.

```
RULE AGREEMENT_SUBJ_CASE_NOR_NORK
(
      Detect (
            @!ncsubj!ncmod∼ &
            @!auxmod.type == 'nor-nork' &
            @!ncsubj!ncmod.case != @!auxmod.nork.case
            )
      Correct (
            (@!auxmod.nork.case := @!ncsubj!ncmod.case)
            # Zentral nuklearrAK zakar erradiaktiboa eratzen DU.
            # 'The nuclear power station creates radioactive rubbish'
            |
            (@!ncsubj!ncmod.case := @!auxmod.nork.case)
            # Zentral nuklearrEK zakar erradiaktiboa eratzen DUTE.
            # 'Nuclear power stations create radioactive rubbish'
            )
      Mark ((ncsubj & auxmod))
      Info (The subject and the auxiliary verb do not agree in case.)
            )
```

Fig. 4. Example of a rule

This type of rule is very similar to the ones described in related systems (Knutsson *et al.*, [2001]). In the following paragraphs we will describe each component in detail.

Error Detection. The error processing rules allow the traversing of the dependency tree while at the same time checking syntactic constraints.

In the description of the errors we use linguistic information such as tags that define dependency relations between the elements of the sentence (e.g. ncsubj, ncobj,...), as well as tags defining features of the syntactic elements (number, case, ...). Apart from this, some operators have been defined, i) to describe the traversal across the branches of the dependency tree (e.g. '@' indicates the current tree node the program is inspecting, '!' followed by a dependency tag (e.g. ncsubj) crosses a dependency link down the current node, '¡' ascends a dependency link, ...) and, ii) to inspect linguistic features (e.g. '∼' looks for the existence of a feature, '#' asks for the number of dependency tags,...).

Thus, for example, the *Detect* part of the example in figure 4 can be paraphrased as: starting from the current node ('@'), descend in the tree ('!') across the branch tagged with the *'ncsubj'* dependency relation and descend again ('!') across the branch tagged with *'ncmod'*. An agreement error between the subject and the verb occurs if the case of the subject and the 'case' of the agreement marker in the auxiliary verb are different (*'!='*).

Error Correction. The *Correct* part of the rule contains two possible corrections: it can be corrected by assigning the case of the subject to the auxiliary verb so that we obtain the acceptable sentence *'Zentral nuklearrAK zakar erra-*

diaktiboa eratzen DU' (THE nuclear power station createS radioactive rubbish). Another option ('|') could be to assign the 'case' of the auxiliary verb to the subject and obtain the correct sentence *'Zentral nuclearrEK zakar erradiaktiboa eratzen DUTE'* (Nuclear power stationS create radioactive rubbish).

Morphological generation will be used to obtain these two correct sentences. One of them will fit in better than the other one in the text. For the process of discriminating among candidate corrections, several methods have been proposed. Some of them are based on heuristics regarding the number of changes required at the morphosyntactic level (Menzel, [1988]) or at the semantic and phonetic levels (Genthial *et al.*, [1994]). Some other methods take into account the syntactic/semantic context of the incorrect element (Golding and Roth, [1996], Carlsson *et al.* [2001]). In this last paper *'context sensitive text correction'* is used when dealing with spelling errors in order to choose the best candidate. We propose a similar technique but at a syntactic level to choose the best option among all the candidate corrections. It is important to remark that after applying morphological generation to create all the possible corrections, a reparsing of the resulting dependency trees would be necessary to test whether the introduction of a correction does not produce any other errors.

At the moment, a number of rules have been designed, and only the section relative to *detection* has been completely implemented. As the rules are written in an abstract language, they cannot be directly applied to a dependency tree because they must first be translated into executable statements. We defined and implemented a syntax-directed translation scheme (Aho *et al.*, [1985]) for that purpose. A lexical analyser recognises lexical units in the rules and a syntactic analyser analyses the correct syntax of the rule and generates the code in C++. Once the code is created, the error processing module will apply the executable rules to the trees.

2.3 Error Processing Module

The next step in the process of error detection is the application of the rules to the trees. Some strategies will be defined for that purpose. The simplest strategy could be to 'apply all the rules to all the nodes in the tree'. We have manually analysed 64 sentences containing agreement errors and have noticed that the structures in which the error detection rules are applied are repeated. This phenomenon could be reflected in a strategy by applying first the rules that match up these structures. In a near future, different strategies will be defined and evaluated taking into consideration aspects such as the minimisation of morphological and structural changes, or the introduction of new errors.

2.4 Decision Module

The problem when dealing with syntactic ambiguity lies in deciding which of the different analyses is the correct one. In case of ambiguity, the error detection system will have more than one dependency tree for a sentence. When the error detection rules are applied to them, we should study what happens when an error is detected. It may happen that, in an ambiguous correct sentence, the

error detection rules could mark one of the incorrect analyses as a syntactic error, giving a false alarm. In these cases, we think that an error should only be marked when all the analyses contain an error. The decision module will be in charge of this task.

3 Analysis of Agreement Errors

In order to define the error rules to be used for agreement error detection, first we manually analysed a set of 64 sentences with some type of agreement error. These sentences were manually extracted from texts of students that have Basque as their mother language. Most of the sentences are related to scientific issues (e.g. computer science students' final year projects,...).

In that corpus, we found 66 errors in 64 sentences (2 sentences with 2 errors). We want to notice that other 2 sentences have been rejected since the agreement error was difficult to detect. One of them is a relative clause with ellipsis that we want to analyse in depth. The second one is a sentence with a non-finite verb, which do not carry any mark to indicate tense or person. As the number of non-finite verbs in this corpus is small, we have decided to start with the finite verbs and try using verb subcategorisation for non-finite verbs later.

Regarding agreement errors, we have made a distinction taking into account the linguistic context in which they occur. Thus, we differentiate between:

- Agreement inside noun or verb chains (5 errors from 64, 7.8 %).
- Agreement inside clauses (59 errors from 64, 92.2 %).
- Agreement between subordinate and main clauses in the sentence (0 errors).

We can see that the number of agreement errors inside clauses is high, so we have focused our analysis in that kind of error.

With the aim of better understanding the high number of errors into clauses, we will briefly explain this kind of agreement. In finite verbs, the agreement elements are marked explicitly in the following way: the verb agrees with the subject, object or indirect object of the sentence. These elements can appear in any order in the sentence, and each of them must agree with their corresponding agreement markers in the verb morphemes in number and person. This is a source of many syntactic errors, considerably higher than in languages with a more reduced kind of agreement, as English or Spanish. Basque is a morphologically ergative language with accusative syntax. Morphological ergativity implies that the subject of a transitive verb is realized in the ergative case, as opposed to the object of a transitive verb, and the subject of an intransitive verb, which are realized in the absolutive case. Besides, the indirect object is realized in the dative case. Some of the rules we have defined for agreement error detection are based on these characteristics. They look for disagreement in case or number between the agreement markers of the auxiliary verbs and finite lexical verbs and the subject, object and indirect object.

Table 1 indicates the number of errors found in each of the mentioned categories.

Table 1. Number of errors in each category

	Elements of the sentence		
Verb	Subject	Object	Indirect object
In agreement markers in 'case'	30	0	1
In agreement markers in number	17	4	3
Total	55		

As we can see in the table above, 30 errors (54.6% of the total) are due to disagreement between the verb and the subject. The reason for this phenomena is that in Basque the morpheme for the absolutive plural and the one for the ergative singular are identically written.

The remaining 4 errors are divided as follows: 3 are due to the fact that if the object of the sentence is declined in the partitive case, the correspondent agreement mark in the verb must be singular. In the other error, an object appears in the sentence with an intransitive verb.

28 rules have been already designed for detecting these errors. Most of them check general structures and detect a high number of errors (e.g. 24 errors are detected with 3 rules) while others are for very specific structures.

At the moment, we are analysing different possibilities of dealing with several phenomena such as some types of ellipsis, relative clauses and coordination.

4 The State of Development

At this point, the state of the work we present in the paper is the following:

1. The syntactic parser is almost complete.
2. The error detection rule compiler is finished and running while the error correction compiler is still in the design phase.
3. The rule application module is operational.
4. 64 sentences with agreement errors have been analysed, and the corresponding rules designed.
5. The first experiments have been manually developed with good results.

Regarding evaluation, we have in mind a type of assessment process similar to that of Starlander *et al.* ([2002]). It is to be expected that some of the errors in the evaluation will be due to the problems described in the following paragraphs.

When developing a system to cope with syntactic errors in real texts, we think that one of the main problems we will have to deal with is the lack of coverage of the syntactic analyser.

Since the parsing system we are using contains several modules, each of them could add a limited amount of errors, which would be accumulated through each phase. As each module can introduce new errors, this could increase the number of false alarms, where a correct sentence is marked as incorrect due to

errors in any of the previous phases. In fact, it could be paradoxical to have the fact that the analysed sentences contain more errors due to correct sentences incorrectly analysed than to real syntactic errors. This fact makes compulsory a corpus-based evaluation (Gojenola and Oronoz, [2000]), in order to obtain high precision and minimise the number of false alarms. We also expect that this will be done at the cost of lowering recall, only marking errors that can be detected with very high precision.

5 Conclusions

We have presented a system for the treatment of complex syntactic errors based on three main modules: a robust syntactic analyser, an error processing rule compiler and a coordination module that will give way to experiment with different strategies to error detection and correction. The goal is to process real texts with high precision error detection/correction, minimising false alarms, which are the main bottleneck in current grammar checking systems. The system will be mainly declarative, as all the modules are based on abstract views of the syntactic processes involved (syntactic analysis and error rules), and extensible, being based on an object oriented approach. At the moment, the syntactic analysis module and the error rule application module are operational. Regarding the error rule compiler, the error detection part is already implemented, and a number of rules have been devised and tested on real sentences.

Acknowledgments

This research is supported by the University of the Basque Country(9UPV00141.226-146012002) and the Ministry of Industry of the Basque Government (XUXENG project, 0D02UN52). Thanks to Ruben Urizar for his help writing the final version of the paper.

References

[2003]Aduriz, I., Díaz de Ilarraza, A.: Morphosyntactic Disambiguation and Shallow Parsing in Computational Processing of Basque. Inquiries into the Lexicon-syntax Relations in Basque (2003) 1–21 Bernard Oyharçabal Ed. University of the Basque Country, Bilbao.

[1985]Aho, A.V., Sethi, R., Ullman, J.D.: Compilers: Principles, Techniques, and Tools. Reading, Mass.: Addison-Wesley (1985)

[1999]Aldezabal I., Alegria I., Ansa O., Arriola J., Ezeiza N.: Designing Spelling Correctors for Inflected Languages Using Lexical Transducers In: Proceedings of EACL'99, 265-266. Bergen, Norway. 8-12 June 1999.

[2004]Artola X., Díaz de Ilarraza A., Ezeiza N., Gojenola K., Sologaistoa A., Soroa, A.: Eulia: a Graphical Web Interface for Creating, Browsing and Editing Linguistically Annotated Corpora. In: Proceedings of the Fourth International Conference on Language Resources and Evaluation, Lisbon, Portugal (2004)

[2004]Briscoe, E., Carroll, J.: Robust Accurate Statistical Annotation of General Text. In: Proceedings of the 3rd International Conference on Language Resources and Evaluation, Las Palmas, Gran Canaria (2002) 1499–1504

[2001]Carlson, A.J., Rosen, J., Roth, D.: Scaling Up Context-sensitive Text Correction. In: Proceedings of the Thirteenth Conference on Innovative Applications of Artificial Intelligence Conference, AAAI Press (2001) 45–50

[2003]Ezeiza, N.: Corpusak Ustiatzeko Tresna Linguistikoak. Euskararen Etiketatzaile Sintaktiko Sendo eta Malgua. PhD thesis. University of the Basque Country, Donostia (2003)

[1994]Genthial, D., Courtin, J., Menezo J.: Towards A More User-friendly Correction. In: COLING-94, Tokyo. (1994)

[2000]Gojenola, K., Oronoz, M.: Corpus-based Syntactic Error Detection Using Syntactic Patterns. In: NAACL-ANLP00,Student Research Workshop. (2000)

[1996]Golding, A.R., Roth, D.: Applying Winnow to Context-sensitive Spelling Correction. In: Proc. 13th International Conference on Machine Learning, Morgan Kaufmann (1996) 182–190

[1995]Karlsson, F., Voutilainen, A., Heikkila, J., Anttila, A.: Constraint Grammar: Language-independent System for Parsing Unrestricted Text. Prentice-Hall, Berlin (1995)

[2001]Knutsson, O., Carlberger, J., Kann, V.: An Object-oriented Rule Language for High-level Text Processing. In: Poster, NoDaLiDa '01 - 13th Nordic Conference on Computational Linguistics. (2001)

[1983]Koskenniemi, K.: Two-level Morphology: a General Computational Model for Word-form Recognition and Production. University of Helsinki, Helsinki (1983)

[1992]Kukich, K.: Techniques for Automatically Correcting Words in Text. ACM Computing Surveys **24** (1992) 377–439

[1988]Menzel, W.: Error Diagnosing and Selection in a Training System for Second Language Learning. COLING-88 (1988) 414–419

[1998]Paggio, P., Music, B.: Evaluation in the Scarrie Project. In: Proceedings of the First International Conference on Language Resources & Evaluation, Granada, Spain (1998) 277–282

[2002]Starlander, M., Popescu-Belis, A.: Corpus-based Evaluation of a French Spelling and Grammar Checker. In: Proceedings of the Third International Conference on Language Resources and Evaluation. (2002) 268–274

[2003]Vandeventer, A.: Syntactic Error Diagnosis in the Context of Computer Assisted Language Learning. PhD thesis. Universite de Geneve, Geneve (2003)

An Experiment in Detection and Correction of Malapropisms Through the Web*

Igor A. Bolshakov

Center for Computing Research (CIC)
National Polytechnic Institute (IPN), Mexico City, Mexico
igor@cic.ipn.mx

Abstract. Malapropism is a type of semantic errors. It replaces one content word by another content word similar in sound but semantically incompatible with the context and thus destructing text cohesion. We propose to signal a malapropism when a pair of syntactically linked content words in a text exhibits the value of a specially defined Semantic Compatibility Index (SCI) lower than a predetermined threshold. SCI is computed through the web statistics of occurrences of words got together and apart. A malapropism detected, all possible candidates for correction of both words are taken from precompiled dictionaries of paronyms, i.e. words distant a letter or a few prefixes or suffixes from one another. Heuristic rules are proposed to retain only a few highly SCI-ranked candidates for the user's decision. The experiment on mala-propism detection and correction is done for a hundred Russian text fragments—mainly from the web newswire—in both correct and falsified form, as well as for several hundreds of correction candidates. The raw statistics of occurrences is taken from the web searcher Yandex. Within certain limitations, the experiment gave very promising results.

1 Introduction

One of the most important applications of computers is automated checking of text accuracy. The problem of out-of-context orthographic correction of letter strings not existing in the language is practically solved.

Syntactic errors leave correct all separate words but violate the sentence structure by using words with wrong morpho-syntactic features (POS, number, gender, case, person, etc.) or violating habitual word order. There is some advance in grammar checkers, though they still need more powerful and robust parsers.

In raw texts, semantic errors also occur. They are of various types and usually violate neither orthography nor grammar. Being expressed by correct words inappropriate in a given context or by grammatically correct phrases contradicting common sense and knowledge, these errors break text understanding. A particular type of semantic errors is malapropism. Encyclopedia Britannica [8] defines malapropism as a

* Work done under partial support of Mexican Government (CONACyT, SNI) and CGEPI-IPN, Mexico. Many thanks to Denis Filatov for help with manuscript preparation.

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 803–815, 2005.

verbal blunder in which one word is replaced by another similar in sound but different in meaning, e.g., *travel around the* ***word*** (stands for *world*). The similarity in sounds or letters singles out malapropisms from the global class of real-word errors replacing one word existing in the language by another existing one—in the same syntactic role.

It is impossible nowadays to automatically detect real-word errors by the total morphological, syntactic, semantic, and pragmatic analysis of text. Within the limitations of the state of the art, we are aware of rather few papers on the problem of malapropism detection and correction [2, 4, 9, 10].

In [9, 10] a method of the malapropism detection and correction is proposed that relies on semantic anomalies in a text indicated by words distant from all contextual ones in WordNet terms; much closer corrections are searched as editing variants of the anomalous word. The distance is determined through paradigmatic relations (synonyms, hyponyms, hyperonyms), mainly between nouns. The syntactic links between words are ignored; words are usually from different sentences or paragraphs.

General idea in [2] is similar: an anomaly is a word that does not match context words and the editing variants of the anomalous word are searched for corrections matching the context. However, the anomaly detection is based on different relations between words. The syntactically correct and semantically compatible content word combinations (= collocations) are considered, and it is presumed that malapropisms violate text cohesion, conserving syntactic correctness of word combinations but destroying them as collocations. Much smaller context—one sentence—is needed for the detection, and words of four principal POS—nouns, verbs, adjective, and adverbs—are considered as collocation components (=collocatives). To test whether a content word pair is collocation, three types of linguistic resources are presumed: a precompiled collocation DB like CrossLexica [1], a text corpus, or the web searcher. However, the experimental part of [2] is scarce and inadequate.

This paper continues the work [2] with stress on a solid experiment with the web searcher for collocation testing. The web is widely considered now as a huge but very noisy resource [5, 6, 7]. It proved necessary to revise the algorithm of malapropism detection & correction and to create new threshold procedures for these two operations, balancing between decision errors of various types.

More specifically our objectives are:

- To clarify the notion of collocation adopted in this paper;
- To define various type of paronyms and corresponding dictionaries for them;
- To compile a set of Russian malapropism samples and of all available paronymy candidates for their correction;
- To modify the algorithm for malapropism detection & correction and to propose a new type of numeric estimate for semantic compatibility of two content words—by Semantic Compatibility Index (SCI);
- To extract raw statistics from Russian web searcher Yandex for malapropism samples and their primary correction candidates;
- To use Yandex statistics transformed to SCI values for detection malapropisms and selecting only elite correction candidates.

We use English examples to explain language independent issues. Russian text fragments somewhere necessitate translations and grammatical tags. Not expecting these fragments be spelled, we rest them in Cyrillic.

2 Collocations

To clarify what we mean by a collocation, let us recall that each text in natural language is a sequence of *word forms*. Commonly, these are strings of letters from one delimiter to the next (e.g., *links*, *are*, *very*, *short*). Word forms pertaining to the morpho-paradigm with common meaning are associated into lexemes. One word form from a paradigm is taken as the lexeme title for the corresponding dictionary entry, e.g. ***pen*** is taken for {*pen, pens*}; ***go*** for {*go, going, gone, went*}; and ***next*** for {*next*}. In languages with rich morphology (e.g., in Russian), paradigms are broader. We divide all word forms into three categories:

- ***Content words***: nouns; adjectivals, i.e. adjectives or participles; adverbials, i.e. adverbs or gerunds; verbs except for auxiliary and modal ones;
- ***Functional words***: prepositions; auxiliary and modal verbs;
- ***Stop words***: pronouns; proper names except of the well known geographic or economic objects or personalities reflected in encyclopedias; any other POS.

According to dependency grammars [12], each sentence can be represented at the syntactic level as a dependency tree with directed links "head → its dependent" between word-form labeled nodes. Following these links in the same direction of the arrows, from one content node through any linking functional nodes up to another content node, we obtain labeled subtree structure corresponding to a word combination. If this is a sensible text, we consider the revealed combination as a collocation. For example, in the sentence *she hurriedly went through the big forest*, we see the collocations ***went*** → *through* → ***forest, hurriedly*** ← ***went*** and ***big*** ← ***forest,*** but not *she* ← ***went*** and *the* ← ***forest*** with stop words at the extremes. Thus, the syntactic links between collocatives can be immediate or realized through functional words.

The given operational definition of collocation guarantees that collocatives are syntactically linked, whereas their belonging to a semantically correct (sensible, conceptualized) text guarantees their semantic compatibility. It is valid for collocations (in our definition) of any idiomaticity, including both complete idioms and totally free combinations. As to collocation stability, the advance of the web shows that any semantically correct word combination eventually realizes several times, and we can consider as collocations all those exceeding a rather low threshold.

To fully describe a collocation, it is necessary to specify its collocatives and the type of the link, including functional word(s), obligatory morpho-characteristics of collocatives (number, gender, case, person, etc., depending on their POS and specific language), and their admissible linear orders. A specific link is determined by syntactical type of the collocation. The most frequent types in European languages are: "the modified → its modifier" (*strong tea*); "verb → its noun complement" (*go to cinema*); "adjective → its noun complement" (*easy for girls*); "verb predicate → its subject" (*light failed*); and "noun → its noun complement" (*struggle against terrorism*).

In texts, collocatives can be linearly separated not only by their own functional word(s) but by many others usually dependent on the same head. To put it otherwise, a close context in a dependency tree is in no way a close linear context. This makes difference with intensively studied bigrams [6, 13].

Any collocation has its normalized (to store in a DB or dictionaries) and textual forms. The latter compose morpho-paradigm of a collocation. E.g., the collocation *go to cinema* comprises the paradigm with three members: {*go/going/went to cinema*}.

3 Paronyms and Their Dictionaries

For the malapropism correction, it is necessary to quickly find words similar to other words suspected erroneous. We call similar words paronyms. Depending on the type of similarity, paronyms can be literal (differ in a few letters), morphemic (differ in a few auxiliary morphs), and acoustic (differ in a few sounds). In Russian with rather immediate correspondence between letters and sounds, we ignore purely acoustic similarity. Below only literal and morphemic paronyms are concerned.

One literal string can be formed from another with a series of editing operations. Elementary editing operations are: replacement of a letter with any other letter in any position; omission of a letter; insertion of a letter; permutation of two adjacent letters. A string obtained with one such operation is 1-distant from the source; another elementary step forms a 2-distant string, etc. Thus, *word* is 1-distant to *world* and *hysterical* is 2-distant to *historical*.

Let us collect all 1-distant word forms for each word form, e.g. English *sign*: {*sigh, sin, sing*}; *sin*: {*bin, gin, kin, pin, sign, sing, son, sun, tin*}. Such groups contain only forms of the same POS (hereby nouns) that conserve morpho-syntactic features after replacement by any paronym. The groups are rather small, although the total number of 1-distant strings for a string of, say, four letters equals 237.

Gathering such groups for multiplicity of word forms of a language, we obtain ***dictionary of literal paronyms***. More distant literal paronyms should be included to it with discretion, since 2-or-more-letter errors are much rarer and multiple errors in short words usually ruin their subjectively conceived similarity.

For languages with rich morphology, to operate with word-form paronyms is tedious because of large size of paronymy dictionaries. E.g., Russian has more than 100,000 word forms with paronyms and thus such a dictionary will contain totally, say, a million forms. A solution suggests itself to take for the dictionary entries some larger objects, e.g. lexemes.

Let us take an element $\lambda(\chi)$ of the morpho-paradigm of the lexeme titled $\lambda(\chi_0)$, where χ is the set of values of morpho-characteristics (specific number, gender, case, person, etc.) that selects the form $\lambda(\chi)$; the set χ_0 selects the lexeme title. Let introduce the string editing operator $R_i()$, where i is its cardinal number in an enumeration system. E.g., the operator $R_{43}()$ changes u to v in the first position of any Latin letter string. We define two paradigm parallel by the logical proportion

$$\forall\chi \;\; \exists i \; \{\lambda(\chi) : \lambda(\chi_0) = R_i(\lambda(\chi)) : R_i(\lambda(\chi_0))\}.$$

It means that for each χ selecting the form $\lambda(\chi)$ of the lexeme $\lambda(\chi_0)$ there exists an editing operator R_i that being applied to $\lambda(\chi)$ gives the correct form $R_i(\lambda(\chi))$—with

the same χ—of the lexeme $R_i(\lambda(\chi_0))$. E.g., the paradigms ***sign***: {*sign, signs, signing, signed*} and ***sigh***: {*sigh, sighs, sighing, sighed*} are parallel with $R()$ replacing *n* to *h* in position 4.

Trying to correct a malapropism, we extract a suspicious form $\lambda(\chi)$ from a text, restore its dictionary form $\lambda(\chi_0)$, find in the dictionary its paronym $R_i(\lambda(\chi_0))$, inflect it to $R_i(\lambda(\chi))$ with the χ equal to that of $\lambda(\chi)$, and then replace $\lambda(\chi)$ by the result. If the semantic compatibility of the text is restored, the candidate is acceptable.

Unfortunately, Russian lexemes, especially verbs and nouns, rarely possess strictly parallel paradigms. The situation improves if to take subparadigms named grammemes. By definition, grammemes are subparadigms with fixed values of some morpho-characteristics χ. E.g., the whole paradigm of Russian noun is split to grammemes of singular and plural, each with six case forms. The verb paradigm is split to personal forms with infinitive as their representative, participles of various tense (each having 24 case-number forms, is similar to adjectives), and gerund (each having one form, it is similar to adverbs). Splitting to grammemes is also suggestible from the viewpoint of combinability of collocatives [1]. For adjectives and adverbs, grammemes cover whole lexemes.

Hence, the best unit for paronymy dictionaries is grammeme. In Russian, we name ***literal paronyms*** any two grammemes that:

- play syntactic role of the same POS;
- have equally sized sets of parallel forms; and—only for nouns—
- have the same gender in singular or are both plural.

It these terms, a Russian literal paronymy dictionary was reported in [3]. Now it has 15,500 group entries with the mean group size 2.65. The gain of its use compared with a series of blind editing operations is ca. 300, just as for its Spanish analog [4].

Some persons, especially foreigners, make errors of a different nature. They could apparently change the collocation *language sensitive* to the malapropos *language sensible* or *massive migration* to *massy migration*. Using the same POS and word radixes, they do not feel the difference. We name the corresponding similarity morphemic paronymy. In Russian, ***morphemic paronyms*** are any two grammemes that:

- play syntactic role of the same POS;
- have the same radix;
- differ in any auxiliary morphs, i.e. suffixes (with the reflexive particle *ся*/*сь* ‘self’ but without inflexional endings) and/or prefixes (including the negation *не* ‘non’);
- may have homonymous radix, like in adjectives ***бур****ный* ‘roaring,’ ***бур****овой* ‘boring,’ and ***бур****ый* ‘brown.’

It these terms, a Russian morphemic paronymy dictionary was reported in [4]. Now it contains 1307 paronymy groups with the mean length 6.71. A group contains on an average 3.4 paronyms at the morphemic distance not more than 2, nouns taken of the same gender in singular or all plural. The mean number of links irrespective of morphemic distance, number and gender is 77 per group.

4 Compiling a Set of Malapropisms

In this paper, only malapropisms conserving the syntactic link between two content words but eliminating their semantic compatibility are considered. Hereinafter, malapropisms are the entire mutilated pairs.

Sometimes an error converts one collocation to another, as a rule, rarer and contradicting the outer context. E.g. *ценовой коридор* 'price margin' changes to *цеховой коридор* 'shop passageway' or *селевой поток* 'mountain torrent' changes to *сетевой поток* 'network flow'. We name such errors quasi-malapropisms. Their detection (if possible) sometimes permits to restore intended words, just as for malapropisms proper.

The formal proof that a given pair is collocation is its presence in a collocation DB, for which we take CrossLexica (CL) [1]. Thus, CL is a peculiar baseline for us.

As many as 20 samples were included in our set of malapropism on a free choice. We put into service well-known jokes of Soviet times like *Does exist the life on* ***Marx****? – Well, it's just a scientific* ***hypotenuse*** *by now!*" We also take several blunders concerning terms from professional jargons. E.g., *мальчишеские размеры* (*брюк*) 'boyish sizes (of trousers)' is wrongly used for *мальчиковые размеры* 'boy sizes,' now quite literary. Our cases only concern literal (1-distant) or morphemic (not-more-that-2-distant) paronyms.

The rest 80 sample were formed based on the newswire of Russian site *Gazeta.ru*. Specifically, we extracted from the news messages contiguous fragments, an element of which could be easily falsified using one of paronymy dictionaries—with strict retention of morphological features of the changed word. E.g., if the original collocation contained the word form *точками* '$\text{dot}_{\text{PL,INSTR}}$' in instrumental case of plural, the falsifying paronym was *тычками* '$\text{poke}_{\text{PL,INSTR}}$.' A complete collocation was extracted humanly. Both malapropos and true versions were included in the set.

This gave 100 malapropisms and their true corrections. Since literal errors are much more frequent in any language than morphemic ones, the proportion between them was taken 86 to 14. All quasi-malapropisms arisen were kept in the set.

Then all combinations with paronyms of both types—for both changed and changeless collocative—were gathered. Indeed, on the correction stage it is unknown what collocative is erroneous and how it changed. To properly algorithmize the problem and to get statistics, we tagged all true corrections. The total number of primary correction candidates reaches 645, i.e. 6.45 candidates per malapropism.

The beginning fragment of our experimental set is in Fig. 1. The set consists of enumerated sample sections with malapropos headlines that begin with the number of the changed collocative (1 or 2) and the symbol of the used paronymy dictionary (**L**iteral or **M**orphemic). Paronyms pertaining to the both dictionaries are marked with **L**. The next is code $n_1.n_2$ of syntactic type of collocation (see Table 1). Then goes the malapropism string, maybe with a short context (in parentheses) helping mental correction of the error. The translations of malapropisms and their wrong corrections exhibit their nonsense.

The lines with primary corrections contain the number of the changed word (1 or 2), symbol of the involved dictionary (L or M), and maybe the symbol '!!' of true correction or '!' of existing collocation differing from the true correction.

```
1)2L 2.2 жизнь на Марксе          'life on Marx'
  2L!! жизнь на Марсе             'life on Mars'
2)1L 1.1 (совершать) полые акты   '(to perform) hollow acts'
  1L   голые акты                 'naked acts'
  1L!! подлые акты                'mean acts'
  1L!  полные акты                'complete acts'
  1L   пошлые акты                'vulgar acts'
  2L   полые акры                 'hollow acres'
  2L   полые пакты                'hollow pacts'
  2L   полые такты                'hollow tacts'
  2L   полые факты                'hollow facts'
3)1L 1.1 истерический центр       'hysterical center'
  1M   истеричный центр           'hysterical center'
  1L!! исторический центр         'historical center'
  1L   стерический центр          'steric center'
  2L   истерический цент          'hysterical cent'
4)1L 2.1 (без) призраков жизни    '(without) specters of life'
  1L!! признаков жизни            'signs of life'
5)2L 3.1 добиваться суждения (вора)'to aim at an opinion (of the thief)'
  2L!! добиваться осуждения       'to aim at the conviction'
  2L!  добиваться сужения         'to aim at narrowing'
  1L   добираться суждения        'to reach of an opinion'
  1L   добываться суждения        'to be mined of an opinion'
  1L   доживаться суждения        'to live until the opinion'
  1L   доливаться суждения        'to be filled up of the opinion'
  1L   дошиваться суждения        'to be tailored of an opinion'
6)1L!1.1 (смыло) сетевым потоком  '(washed away) by the network flow'
  1L!! селевым потоком            'by the mountain torrent'
  1M   сеточным потоком           'by the grid flow'
  1M   сетчатым потоком           'by the netted flow'
  2L   сетевым пороком            'by the network flaw'
  2M   сетевым притоком           'by the network influx'
  2L   сетевым протоком           'by the network duct'
  2L!  сетевым током                  'by the network current'
```

Fig. 1. Some malapropisms and their correction candidates

Table 1. Types and structures of collocations

Type title	Type code	Dependency subtree	English analogue	% in set	% in CL
modified → its modifier	1.1	Adj ← N	*strong tea*	39	36.8
	1.2	Adv ← Adj	*very good*		
noun → its noun complement	2.1	$N \rightarrow N_{comp}$	n/a	26	12.2
	2.2	$N \rightarrow Prep \rightarrow N_{comp}$	*signs of life*		
verb → its noun complement	3.1	$V \rightarrow N_{comp}$	*give books*	20	19.7
	3.2	$V \rightarrow Prep \rightarrow N_{comp}$	*go to cinema*		
	3.3	$N_{comp} \leftarrow V$	n/a		
verb predicate → its subject	4.1	$N_{sub} \leftarrow V$	*light failed*	7	12.9
	4.2	$V \rightarrow N_{sub}$	(*there*) *exist people*		
	4.3	$Adj_{pred} \rightarrow N_{sub}$	n/a		
adjective → its noun complement	5.1	$Adj \rightarrow Prep \rightarrow N_{comp}$	*easy for girls*	6	18.3
	5.2	$Prep \rightarrow N_{comp}$ Adj	n/a		

In Table 1, the type titles (column 1) ignore functional words and the order of collocatives. The specific order is given explicitly in dependency subtrees (column 3) operating with **N**ouns, **Adj**ectivals, **V**erbs, and **Adv**erbials; the subindex *comp* means the noun complement with the case depending on the ruling preposition or the management pattern of the head collocative; subindex *sub* means the noun subject in nominative case; the subindex *pred* means specifically Russian predicative form of adjectival. The columns 5 and 6 give percentage of each collocation type—in our set and in CrossLexica [1].

5 Algorithm for Malapropism Detection and Correction

The main idea of our algorithm is to look through all pairs of content words within a sentence under revision, testing each pair on its syntactic combinability and semantic compatibility. If the pair is syntactically combinable but semantically incompatible, a malapropism is signaled. Then all pairs formed by a collocative and its counterpart's paronym are tested on semantic compatibility. If the pair fails, it is discarded, else it is included into a list of secondary candidates. The list is ranked and only elite candidates are left. We propose the following procedure for revision of a sentence:

```
Detect&Correct_Malapropisms
  for each W(i) in sentence repeat
    for each W(j)such that j < i repeat
      if ContentWord(W(j)) & ContentWord(W(i))
        & SyntCombinable(W(j),W(i))
        & not SemCompatible(W(j),W(i)) then
        { ListOfPairs = ∅
          repeat      % for all literal paronyms of the 1st collocative
            TakeNextLiteralParonym(P,W(j))
            if SemAdmissible(P,W(i))then
              InsertToListOfPairs(P,W(i))
          until NoMoreLiteralParonymFor(W(j))
          repeat      % for all morphemic paronyms of the 1st collocative
            TakeNextMorphemicParonym(P,W(j))
            if SemAdmissible(P,W(i))then
              InsertToListOfPairs(P,W(i))
          until NoMoreMorphemicParonymFor(W(j))
          repeat      % for all literal paronyms of the 2nd collocative
            TakeNextLiteralParonym(P,W(i))
            if SemAdmissible(W(j),P))then
              InsertToListOfPairs(W(j),P)
          until NoMoreLiteralParonymFor(W(j))
          repeat      % for all morphemic paronyms of the 2nd collocative
            TakeNextMorphemicParonym(P,W(j))
            if SemAdmissible(W(j),P)then
              InsertToListOfPairs(W(j),P)
          until NoMoreMorphemicParonymFor(W(j))
          Filter(ListOfPairs)
          LetUserTests(ListOfPairs) }
```

Boolean function **SyntCombinable**(*V, W*) determines if the pair (*V, W*) form a syntactically correct word combination. Hence, it contains a partial dependency parser

searching all those conceivable dependency subtrees with V and W at the extremes that are proven with the immediate context (see subtrees examples in Table 1).

Boolean functions **SemCompatible**(V,W) and **SemAdmissible**(V,W) both determine if the pair (V,W) is semantically compatible. The procedure **Filter**(*ListOfPairs*) selects elite candidates. Operations of these three heavily depend on the available resource for collocation testing.

When the resource is a collocation DB, the Boolean functions are equal. They query the DB if the corresponding collocation is recorded in it, i.e. both collocatives are in DB dictionary and the syntactical link between them revealed by the function **SyntCombinable**(V, W) corresponds to a DB record. If collocation is in DB, the function **SemCompatible** considers (V, W) to be a true collocation, whereas **SemAdmissible** admits the pair (V, W) to be a possible candidate. If collocation is absent, **SemCompatible** signals a malapropism, while **SemAdmissible** discards a candidate. With a DB, all candidates are elite and the procedure **Filter** is excessive.

When the resource is a text corpus, **SemCompatible**(V, W) determines the number $N(V,W)$ of co-occurrences of V and W at a limited distance from one another in the whole corpus. If $N(V,W)$ equals zero, the function is *False*. If $N(V,W)$ is positive, for a definite decision it is necessary to syntactically analyze each co-occurrence, which is evidently impossible. In the case where the co-occurrences are either collocations or mere coincidences in a text span, only statistical criteria are applicable. According to one of them, the pair is compatible if the relative frequency (= empirical probability) $N(V,W)/S$ of the co-occurrence is greater that the product of the relative frequencies $N(V)/S$ and $N(W)/S$ of V and W taken apart (S is the size of the corpus). Using logarithms, we have the following threshold rule of pair compatibility:

$$\mathrm{MI}(V, W) \equiv \ln(N(V, W)) + \ln(S) - \ln(N(V)) - \ln(N(W) > 0,$$

where $\mathrm{MI}(V, W)$ is the mutual information [11].

In the web searchers, statistical approach is inevitable, since in addition to random coincidences, an abundance of errors in the web texts emulate malapropisms and their corrections. Any searcher automatically delivers statistics about a queried word or a word combination usually measured in numbers of pages. We can re-conceptualize MI with all N as numbers of relevant pages and S as the page total managed by the searcher. However, now N/S is not empirical probability; it is supposedly a value monotonously connected with the probability. The situation with Russian web searcher Yandex is happy in that $N(V,W)$, $N(V)$ and $N(W)$ are given by one query.

As a heuristic estimate of collocative pair compatibility we introduce a Semantic Compatibility Index (SCI) similar to MI:

$$\mathrm{SCI}(V, W) \equiv \begin{cases} \ln(N(V,W)) + \ln(P) - (\ln(N(V)) + \ln(N(W))) / 2, & \text{if } N(V,W) > 0, \\ NEG, & \text{if } N(V,W) = 0, \end{cases}$$

where NEG is a big negative constant; P is positive constant chosen experimentally.

A merit of SCI as compared to MI is that the total amount of pages is not estimated at all. However, SCI, as MI, does not depend on the growth of all statistics in the searcher—because of the divisor 2. For Yandex, the compensation of the size variations is essential, since its entire statistical data rise now by 0.5% a day.

Thus, **SemCompatible** gives *False* with the malapropism (V_m,W_m) signaled if $\mathrm{SCI}(V_m,W_m) < 0$, whereas **SemAdmissible** gives *True* with the primary candidate

(V,W) admitted as a secondary one if the corresponding threshold rule including SCI values for both candidate and its malapropism actuates:

$$(\mathrm{SCI}(V_m, W_m) = NEG) \textbf{ and } (\mathrm{SCI}(V,W) > Q) \textbf{ or}$$
$$(\mathrm{SCI}(V_m, W_m) > NEG) \textbf{ and } (\mathrm{SCI}(V,W) > \mathrm{SCI}(V_m, W_m)),$$

where Q, $NEG < Q < 0$, is a constant chose experimentally.

The procedure **Filter** operates with whole groups of secondary candidates, ranking them by SCI values. The elite candidates are all with positive SCI (let be n of them), whereas from the negative valued one more is admitted, if n=1, and the two, if n=0.

6 Experiment with Yandex Statistics and Its Discussion

Similarly to other searchers, Yandex allows two options for a query:

- The strict option (in quotation marks): occurrences are searched strictly for the embedded word forms in their preset order;
- The loose option (without quotation marks): occurrences are searched of arbitrary inflected forms of embedded lexemes coming in any order at any distance.

The loose option raises the number of collocative co-occurrences too high; in their majority they are not the searched collocations. The strict option is not faultless either, since does not permit to search distant collocatives. We bypassed this obstacle deleting alien words between collocatives in a few samples of the experimental set.

From statistics that Yandex gives for each multiword query we take the page number of collocative co-occurrences (see the right column of Fig. 2) and numbers of pages for each collocative (in Fig. 2, the data for the collocatives containing in the malapropos line are omitted). Through the web session with 745 accesses we obtained significant statistics. The malapropisms have 65 zero co-occurrences, whereas primary candidates have 492 zeros. So the raw statistics is already a powerful tool for both detection and elimination of absurd corrections.

To obtain all negative SCI values for true malapropisms, we take P = 200. The constant G = –7.5 is adjusted so that all candidates with non-zero occurrences have SCI values greater then this threshold. The choice of the constant NEG = –9.999 is rather arbitrary; it suffices that all non-zero events have greater SCI.

Though all eight quasi-malapropisms were excluded while selecting the constant P, our algorithm detects seven of them. It is clear that if a collocation does exist in language and in the searcher, it may be inserted in the DB. However SCI of quasi-malapropisms is still too low for them to be acknowledged as collocations by our algorithm. This is a curious case when the noisy web exhibits higher precision of standalone malapropism detection than CL. As much as seven of these quasi-malapropisms contradict their extra-collocation context, so their detection is not in vain for the user.

The function **SemAdmissible** leaves 152 secondary candidates of 645 primary ones (the decrease 4.24), while the procedure **Filter** reduces them to 132 elite candidates (the total decrease 4.89). Among the elite candidates for the 99 malapropisms signaled, as many as 98 are true correction options, and only two of them are not first-ranked (see the samples with SCI values and decision qualifications in Fig. 3). The lists of the elite candidates contain 1 to 4 entries. Only 11 elite candidates are not

collocations but apparent the web noise. As few as 5 malapropisms have collocations among primary candidates not entering to the elite; only three malapropisms have elite candidates that try to change the correct word in the malapropisms.

```
1)2L 2.2 жизнь на Марксе            274, жизнь:34871341, Марксе:9021
  2L!! жизнь на Марсе               49288, Марсе:440004
2)1L 1.1 (совершать) полые акты     0, полые:37385, акты:2357875
  1L   голые акты                   1, голые:2729404
  1L!! подлые акты                  12, подлые:63984
  1L!  полные акты                  4, полные:1264157
  1L   пошлые акты                  0, пошлые:209498
  2L   полые акры                   0, акры:8065
  2L   полые пакты                  0, пакты:8676
  2L   полые такты                  0, такты:22226
  2L   полые факты                  0, факты:3729898
3)1L 1.1 истерический центр         0, истерический:46860, центр:33808389
  1M   истеричный центр             0, истеричный:14736
  1L!! исторический центр           46199, исторический:2029436
  1L   стерический центр            0, стерический:461
  2L   истерический цент            0, цент:174231
4)1L 2.1 (без) призраков жизни      42, призраков:293225, жизни:43588136
  1L!! признаков жизни              60581, признаков:1092302
5)2L 3.1 добиваться суждения (вора)0, добиваться:568690, суждения:387344
  2L!! добиваться осуждения         107, осуждения:163625
  2L!  добиваться сужения           18, сужения:56756
  1L   добираться суждения          1, добираться:233950
  1L   добываться суждения          0, добываться:7472
  1L   доживаться суждения          0, доживаться:28
  1L   доливаться суждения          0, доливаться:179
  1L   дошиваться суждения          0, дошиваться:3
6)1L!1.1 (смыло) сетевым потоком    36, сетевым:314290, потоком:531060
  1L!! селевым потоком              1346, селевым:1799
  1M   сеточным потоком             0, сеточным:971
  1M   сетчатым потоком             0, сетчатым:9198
  2L   сетевым пороком              0, пороком:57215
  2M   сетевым притоком             0, притоком:40353
  2L   сетевым протоком             0, протоком:3927
  2M!  сетевым током                33, током:447288
```

Fig. 2. Some malapropisms and primary candidates with Yandex statistics

```
  1)2L 2.2 жизнь на Марксе          -0.023 DETECTED
      2L!! жизнь на Марсе            7.355 1ST CANDIDATE
  2)1L 1.1 (совершать) полые акты   -9.999 DETECTED
      1L!! подлые акты              -0.001 1ST CANDIDATE
      1L!  полные акты              -2.591 2ND CANDIDATE
  3)1L 1.1 истерический центр       -9.999 DETECTED
      1L!! исторический центр        6.526 1ST CANDIDATE
  4)1L 2.1 (без) призраков жизни    -3.751 DETECTED
      1L!! признаков жизни           2.866 1ST CANDIDATE
  5)2L 3.1 добиваться суждения (вора) -9.999 DETECTED
      2L!! добиваться осуждения     -2.524 1ST CANDIDATE
      2L!  добиваться сужения       -3.777 2ND CANDIDATE
  6)1L!1.1 (смыло) сетевым потоком  -1.736 IN CL BUT DETECTED
      1L!! селевым потоком           4.467 1ST CANDIDATE
```

Fig. 3. Some malapropisms and elite candidates supplied with SCI values

In Fig. 4 the distributions of SCI values for malapropisms and their true corrections are given. The values are rounded to the nearest integers. We can see that the zero is the upper bound for malapropisms (except the not signaled one), while $G = -7.5$ is the

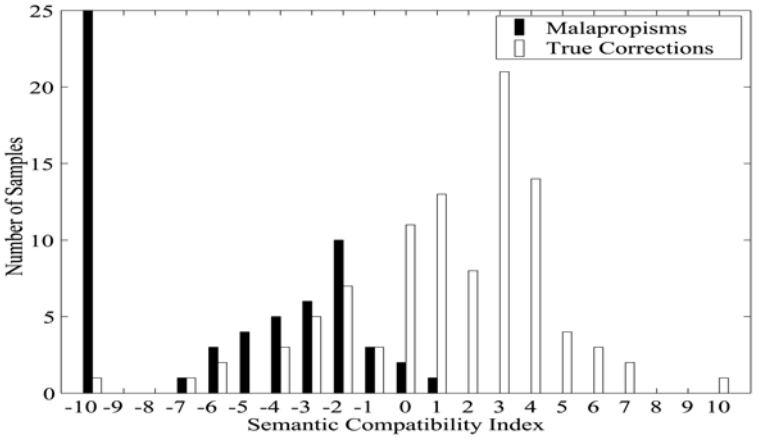


Fig. 4. Distribution of SCI for malapropisms and their true corrections

lower bound for the true corrections. The malapropisms have a bimodal distribution: the peak at –10 corresponds to "pure" errors, while the distributed peak at –2 corresponds to errors masked by the web noise. The corrections are distributed around +3. The difference in SCI values for malapropisms and their corrections is quite evident.

7 Conclusions and Future Work

A method is proposed for detection and correction of malapropisms based on calculation and comparison—for syntactically linked content word pairs—of a heuristically introduced numeric Semantic Compatibility Index. For its calculation, the web statistics on the content words and their pairs is used. The experiment on a set of a hundred malapropisms and their 645 correction candidates showed that as many as 99 malapropisms are detected and for 98 of them the true correction candidates are among the most high-ranked in the short lists of elite candidate delivered to the user.

It seems topical to affirm (or shake) the result of our study changing: (1) the experimental set (e.g., including some distant pairs); (2) the Web searcher (e.g. accessing to Google); (3) the time of the experiment (ours was done at the beginning of November, 2004); (4) the criterion (e.g., taking Mutual Information instead of SCI). Of course, it is quite topical to take another language, primarily, English.

A very serious flaw of the web is its tardiness. For example, to revise a sentence of, say, 15 word forms with 10 content words among them, 8 syntactical links between the latter and one malapropos pair, it is necessary to make 8+6=14 accesses to the web through heavy burdened networks, which take for our method about 14 sec. Additionally, the total number of accesses a day is limited for the well-known searchers.

We may propose two ways out from this situation. The first is to demand selling language-specific sectors of popular web searcher's storage—to use them as local corpora. Indeed, the progress in size of hard disc memories goes before the web growth. The second way is development of multistage systems including a collocation

DB of reasonable size, a large text corpus, and the web-oriented part, so that the system accesses the web in exceptional cases suggested by the collocation DB supplied with some inference ability.

References

1. Bolshakov, I.A. Getting One's First Million…Collocations. In: A. Gelbukh (Ed.). *Computational Linguistics and Intelligent Text Processing*. Proc. CICLing-2004. Lecture Notes in Computer Science, N 2945, Springer, 2004, p. 229–242.
2. Bolshakov, I.A., A. Gelbukh. On Detection of Malapropisms by Multistage Collocation Testing. In: A. Düsterhöft, B. Talheim (Eds.) Proc. 8th Intern. Conference on Applications of Natural Language to Information Systems NLDB´2003, June 2003, Burg, Germany, GI-Edition**,** LNI, V. P-29, Bonn, 2003, p. 28–41.
3. Bolshakov, I.A., A. Gelbukh. Paronyms for Accelerated Correction of Semantic Errors. *International Journal on Information Theories & Applications.* V. 10, 2003, p. 198-204.
4. Gelbukh, A., I.A. Bolshakov. On Correction of Semantic Errors in Natural Language Texts with a Dictionary of Literal Paronyms. In: J. Favela et al. (Eds.) *Advances in Web Intelligence*. Proc. Intern. Atlantic Web Intelligence Conf. AWIC 2004, Cancun, Mexico, May 2004. LNAI 3034, Springer, 2004, p. 105-114.
5. Gelbukh, A., G. Sidorov, L. Chanona-Hernández. Compilation of a Spanish representative corpus. In: A. Gelbukh (Ed.). *Computational Linguistics and Intelligent Text Processing*. Proc. CICLing-2002. Lecture Notes in Computer Science, N 2276, Springer, 2002, p. 285–288.
6. Keller, F., M. Lapata. Using the Web to Obtain Frequencies for Unseen Bigram. *Computational linguistics*, V. 29, No. 3, 2003, p. 459-484.
7. Kilgarriff, A., G. Grefenstette. Introduction to the Special Issue on the Web as Corpus. *Computational linguistics*, V. 29, No. 3, 2003, p. 333-347.
8. *The New Encyclopædia Britannica*. Micropædia Vol. 7. Encyclopædia Britannica, Inc., 1998.
9. Hirst, G., D. St-Onge. Lexical Chains as Representation of Context for Detection and Corrections of Malapropisms. In: C. Fellbaum (ed.) *WordNet: An Electronic Lexical Database*. MIT Press, 1998, p. 305-332.
10. Hirst, G., A. Budanitsky. Correcting Real-Word Spelling Errors by Restoring Lexical Cohesion. *Natural Language Engineering*, 2004 (to appear).
11. Manning, Ch. D., H. Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, 1999.
12. Mel'čuk, I. *Dependency Syntax: Theory and Practice*. SUNY Press, NY, 1988.
13. Pedersen, T. A decision tree of bigrams is an accurate predictor of word senses. Proc. 2nd Annual Meeting of NAC ACL, Pittsburgh, PA, 2001, p. 79-86.

A Paragraph Boundary Detection System

Dmitriy Genzel

Department of Computer Science, Box 1910,
Brown University, Providence, RI 02912, USA
dg@cs.brown.edu

Abstract. We propose and motivate a novel task: paragraph segmentation. We discuss and compare this task with text segmentation and discourse parsing. We present a system that performs the task with high accuracy. A variety of features is proposed and examined in detail. The best models turn out to include lexical, coherence, and structural features.

1 Introduction

In this paper we will introduce the problem of paragraph boundary detection (or paragraph segmentation). Given a collection of semantically coherent texts, such as news articles or book chapters, with sentence boundaries marked, we wish to mark the paragraph boundaries in each text.

The system that solves this problem has several applications. The most interesting one is to be used as a part of a grammar checker in a word processing system like Microsoft Word [1]. The system would suggest places where a paragraph could be split in two, or even suggest a paragraph break as soon as the user typed in an appropriate sentence.

Furthermore, it could be used to restore paragraph breaks after OCR processing in the cases when the OCR module is unable to accomplish this based on spatial information. This happen fairly often when the paragraph starts at the top of the page.

It is also the case that some text summarization systems prefer extracting paragraphs from original text [2], and thus can use this system when the paragraph boundaries are not specified in the text.

This problem has received very little attention, although it is similar to several well-known tasks, such as text segmentation and discourse parsing. The problem is interesting in its own right as well as through its potential applications. By building such a system, one can obtain information about the regularity of paragraph boundary location. After all, the paragraph boundary placement is somewhat arbitrary and depends on the author's style and taste. Furthermore, by examining the most relevant features of the model, one can learn which elements of the text are sensitive to paragraph boundary placement. This knowledge could then be used to better predict these features based on the paragraph boundary information, which has been rarely used in models, even when available.

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 816–826, 2005.

2 The Problem

We wish to build a system that can segment a text into paragraphs. We will frame this problem as one involving the discrimination between paragraph-starting and non-paragraph-starting sentences. We do this because we have reason to believe that there are structural and lexical differences between these two kinds of sentences [3], and also because there is an approximately equal number of sentences of each kind. We classify each sentence independent of the decisions made for preceding sentences.

The problem appears to be very hard, since even humans would presumably have low agreement on the location of the paragraph boundaries. This ought to be particularly the case when the texts are not homogeneous, e.g., when the authors are different.

3 Previous and Related Work

One earlier approach to this problem taken by [4] relied on the view of paragraphs as lexically cohesive units. In this work we use a variety of features, including measures of lexical cohesion, but also purely syntactic and other measures.

We have mentioned above that this task is similar to that of text segmentation and discourse parsing. It is worth noting how it differs from either of these tasks.

3.1 Text Segmentation

Text segmentation is very similar to our task in the problem setting and one can imagine retraining and running existing text segmentation systems on it (see, for example, [5]). There are, however, a few differences.

First, a very large part of the text segmentation system's accuracy depends on indicator words – words that tend to trigger beginning of a new segment. For instance, [6] report that one of the best features for their news corpus was the word *C.*, as in *This is C. N. N.* which often precedes a new CNN segment. Even in the less extreme cases, for Wall Street Journal articles they report words such as *Corporation* and *Incorporated* which usually occur only in the first sentence of an article, as the company is being introduced in full. Obviously, this feature would not be as useful for paragraph boundaries.

Second, another very important feature is lexical coherence. It is to be expected that the vocabulary would change drastically at the text boundary. Paragraph boundaries, however, rarely indicate a completely new topic, and we expect this feature to be less important.

Third, the text segmentation task looks for rare events, since average text length in (for example) Wall Street Journal is about 20 sentences, whereas average paragraph length is only slightly above two. Paragraph segmentation is, therefore, as much about identifying paragraph-starting as about non-paragraph-starting sentences.

3.2 Discourse Parsing

Discourse parsing and similar methods (see [7] for an example of recent work) indirectly create a segmentation of the text into discourse segments. These segments may span multiple paragraphs or single ones, or sometimes parts of paragraphs. Often they cross paragraph boundaries altogether. This is not surprising, since a paragraph might be used by the author for a variety of purposes that do not always correspond to the discourse elements. It has also been pointed out by [8] that the rhetorical structure might not even correspond to continuous chunks of text. In short, discourse parsing is a similar, but different problem. It focuses on the deeper structure of the text, which is more or less deterministic, whereas paragraph structure at least partially depend on the stylistic considerations.

4 The Model and the Algorithm

We need a model that classifies objects into two classes based on a variety of features which can depend on one another. We chose a sparse voted perceptron model, similar to that of [9], because we are primarily interested in the features, rather than a specific model. The basic perceptron algorithm [10] works as follows:

Let $X = \{x_i\}$ be the training set of sentences. Let $F = \{f_j(x) : X \to \mathbb{R}\}$ be the set of feature functions, mapping each sentence into a point in $\mathbb{R}^n$.
Define a set $W = w_j$ of weights, one per each feature function, with all weights initially 0.
Let $s(x) : X \to \mathbb{R} = \sum_j (w_j * f_j(x))$ be the scoring function.
Let $y(x) : X \to \{0, 1\}$ be 1 for paragraph-starting sentences, and 0 for non-paragraph-starting ones, as specified in the training data. For every sentence x, it is judged to be paragraph-starting if $s(x) > 0$, and non-paragraph-starting otherwise. The algorithm is as follows:

```
Repeat until done:
  for i=1..|X|
    if s(x_i) > 0
       z=1
    else
       z=0
  if z ≠ y(x_i)
    for j=1..|W|
       w_j=w_j + (y(x_i) − z) * f_j(x_i)
```

In other words, whenever a mistake in prediction is made, the algorithm increments the weights for those features which would have helped to make the correct predictions, and decrements those responsible for the incorrect prediction.

We continue the outer loop until our performance on the development corpus stabilizes. The sparse voted version of the algorithm periodically saves the set W of feature weights, obtaining several perceptrons. During the testing phase each

of the perceptrons votes and the result supported by the majority is reported. For this particular model, we skip first 5 rounds of training (since it is less reliable) and then save 10 perceptrons per round, equally spaced.

5 The Feature Set

Let us now discuss the feature set used for these experiments. We initially propose a great variety of features, with the intent to learn which will be useful for this task. Our feature selection scheme which is discussed in the next section.

The particular set of features used was selected for several major reasons. Some features were already known to be successful in text segmentation tasks. Several syntactic features were introduced to evaluate the utility of the parsing information. Other features, added based on the results of [3], suggest that the sentence structure varies for paragraph-starting and non-paragraph-starting sentences. Finally, some features were introduced to check the usefulness of discourse information.

We introduce all the features (in alphabetical order) with name, description, motivation, and the number of features (count) of this type in the two corpora considered here: Wall Street Journal and War and Peace (see the next section for details).

Centering Type

Description: For each of the centering types, whether the subject of the current sentence is in the corresponding relationship with the previous sentence. See [11] for details.
Motivation: Centering is related to local coherence which is relevant at paragraph boundary
Count: 5, 5 (current sentence has no subject, previous subject became current subject, previous object became current subject, other word from previous sentence became subject, new word became subject)

Cosine Features (5 Feature Types)

Description: Measure lexical similarity between the current sentence and the previous ones (5 different measures)
Motivation: Lexical similarity was useful in text segmentation.
Count: 5, 5 (between sentences: previous - current, previous 5 - current, previous - current and next 4, 5 previous - current and next 4, previous - current (measured for upper case words only))

First Word

Description: Same as lexical feature (see below), but only for the first word in the sentence
Motivation: First sentences tend to start differently
Count: 5157, 1525

Form-Function Tags

Description: Form function tags are used in Penn Treebank. See [12] for details.
Motivation: Paragraph-starting sentences are structurally marked. Moreover, this set includes tags like *subject* which might be particularly relevant
Count: 9, 9 (specific tags listed in [12])

Internal Node

Description: For each internal node in the parse tree, number of times it occurs in the sentence. Note: this includes "extended" node labels, with form/function tags, such as S-TMP-TPC (sentence - temporal - topicalized)
Motivation: Paragraph-starting sentences are structurally marked, according to [3]
Count: 255, 229

Internal Node Size

Description: For each internal node in the parse tree: its average size
Motivation: Paragraph-starting sentences are structurally marked
Count: 255, 229

Lexical (Two Feature Types)

Description: For each word in the vocabulary, number of times occurring in the current sentence (feature type 1) and in the preceding sentence (feature type 2)
Motivation: Indicator words help in text segmentation
Count: 49206*2, 19715*2

Local Tree

Description: For each rule used in the derivation (or, for each local subtree of size 1) of the parse tree, average number of times it is used in the sentence
Motivation: Paragraph-starting sentences are structurally marked
Count: 34307, 22554

Number of Uppercase

Description: Number of uppercase words in the sentence
Motivation: Large number may indicate an introduction of a new entity
Count: 1, 1

Part of Speech (Two Feature Types)

Description: For each part of speech, number of times occurring in the current sentence (feature type 1) and in the preceding sentence (feature type 2)
Motivation: If indicator words are too fine-grained, perhaps parts of speech are useful
Count: 45*2, 43*2

Sentence Length

Description: Length of the current sentence (in words)
Motivation: Sentence length is known to be (on average) the highest in the first sentences of the text
Count: 1, 1

Subject Head

Description: Same as lexical feature, but only for the head of the subject
Motivation: Perhaps certain subjects are more likely in paragraph-starting sentences.
Count: 7387, 2437

Subject Type (2 Feature Types)

Description: For each of the subject types, whether the subject of the current (previous) sentence is of this type?
Motivation: Perhaps indefinite subjects are more likely in paragraph-starting sentences?
Count: 7, 7 (no subject, subject is an empty node, indefinite, definite, pronoun, proper name, possessive)

Text Boundary

Description: For the first sentence in each text – 1, otherwise – 0
Motivation: First sentence in the text always starts a paragraph
Count: 1, 1

6 Data and Results

We do our experiments on two data sets: Penn Treebank [13] as a collection of articles; and *War and Peace* [14] as a collection of chapters, processed by a variety of tools [15, 12, 16] to recover parses, form/function tags and empty nodes.

We randomize the order of texts used, and then extract 70% as training, 20% as development, and 10% as testing. We do feature selection by starting with all feature types enabled, removing the most useless types one by one, until performance starts to drop.

The full feature set is highly redundant, and choice of one feature type over another may not be very significant for similar types, e.g. one of the cosine measures may be almost as good as another, but probably only one or two will be chosen.

6.1 Penn Treebank Results

Our best model (automatically learned as described above) includes the following features:

- Text boundary
- Part of speech (current sentence)
- Sentence length
- First word
- Subject types
- Internal nodes (current sentence)
- Cosine between the last one and next 5 sentences (including current one)

Its classification accuracy is 67% with the baseline (guess all sentences to be non-paragraph-starting) of 55%. Dropping each of the features makes the performance worse, and the decrease is statistically significant ($P<0.001$) for each feature, as determined by a paired Monte Carlo test [17].

This result confirms that the problem is very hard. Wall Street Journal articles are written by different people with different writing styles. Moreover, paragraph boundaries are often inconsistent with regard to quotations. It is also the case that WSJ articles are focused on one topic, and thus consistency-based measures tend to do poorly.

In table 1 you will find selection of the most useful features, with high usefulness measure defined as

$$u(f_j) = \frac{w_j \sum_i f_j(x_i)}{|X|}$$

Note: Using the weight itself tends to bring up obscure features which are rarely used. This measure takes into account how often the feature is triggered.

Negative score means the weight of the feature is negative, i.e. it selects for non-paragraph-starting sentences.

Table 1. Selected top features: Wall Street Journal

Type	Feature	Score
Int. nodes	S	-188
Int. nodes	NP-SBJ	141
Part of speech	.	-129
Sentence length		119
Text boundary		79
Int. nodes	VP	-66
Part of speech	NNP	62
Part of speech	NN	49
Cosine	prev - 5 cur	35
First word	The	-14
First word	“	-10
First word	It	-8

6.2 War and Peace Results

Our best model included the following features:

- Lexical (current and previous sentences)
- Sentence length
- Cosine (between the last and current, and between last 5 and current)
- First word
- Internal node size

Classification accuracy is 76%, with a baseline of 63%. Dropping each of the features makes the result worse, and the drop is statistically significant ($P<0.001$ for all types except internal node size, for which $P=0.01$), as shown by a paired Monte Carlo test.

It is somewhat easier to deal with this corpus, both because the baseline is higher and because the corpus itself is consistent. It is also worth noting that the features are generally similar between two models. For instance, internal node and internal node size tend to capture similar structural variations. Both models contain lexical, coherence, and structural features.

You will find selection of the most useful features in the table 2. Many of these are intuitively clear. For instance, an opening quote in the previous sentence makes it unlikely that the current sentence starts a paragraph (the previous sentence probably starts it). On the other hand, a closing quote in the previous sentence makes it more likely. The pronoun *he* is unlikely to be used in the first sentence, whereas *Pierre* (the main character in the book and the likely referent of *he*) is likely to appear there. A question or exclamation in the previous sentence is probably answered in the same paragraph. The word *said* usually occurs in the first sentence, and does not occur elsewhere. The size of SINV node is a proxy for whether SINV is present[1], and if it is present it is an indication of a start of a paragraph.

There are also several non-trivial features. For instance, apparently conjunctions rarely appear in the first sentence. Moreover, while the above features may seem obvious after the fact, it is in fact very hard to invent them on one's own.

6.3 Cross-Text Results

The potential utility of the proposed task lies in being able to train the model on one dataset and test on another. It is obvious that if we take a random pair of texts our system will perform poorly. The more interesting question is what will happen if the training and testing texts are generally of the same style. The proper and systematic way to do this would be to take a major corpus that records stylistic subdivisions and investigate all or most combinations of texts in the same style as training and testing. Unfortunately, we were unable to perform this task due to time and processing considerations, and also because most corpora contain large numbers of small texts, rather small numbers of large

[1] SINV indicates inverted sentence constructions, such as: *"Hello", said he.*

Table 2. Selected top features: *War and Peace*

Type	Feature	Score
Lexical (previous)	.	-345
Lexical (previous)	”	201
Lexical (previous)	“	-102
First word	“	92
Cosine	prev - cur	-68
Sentence length		65
Cosine	5 prev - cur	-46
Lexical (current)	the	36
Lexical (current)	.	-31
Lexical (current)	he	-31
Lexical (current)	and	-28
Lexical (previous)	and	26
Lexical (previous)	!	-25
Lexical (previous)	?	-25
Int. node size	SINV	20
Int. node size	S-TPC	20
Int. node size	NP	-17
Int. node size	VP	17
Lexical (current)	Pierre	15
Lexical (current)	said	14
Lexical (previous)	said	-13

Table 3. Cross-text Testing

Test text	Accy	Self accy	Base
Anna Karenina	77%	78%	62%
3 Musketeers	75%	78%	58%
Kim (Kipling)	82%	89%	69%
David Copperfield	64%	57%	62%
Essays (B. Russell)	75%	78%	81%

texts which we need for training. Instead we have randomly chosen several large texts (freely available from Project Gutenberg) and used the model described above to train on *War and Peace*[2] and test on each of the texts. We report the results in Table 3. Under **Accy** we report accuracy for cross-text performance, under **Self accy** performance for training and testing on parts of testing text, and under **Base** the baseline performance: always guess “yes” or always guess “no”, whichever is better. Note, that the texts used in testing are *much* smaller than the training text, and their **Self accy** is often low, due to lack of training data.

[2] Because it is the largest of our texts

The results show that at least for several texts that are somewhat similar to *War and Peace* we can train on a different text and lose only a little in performance. The crucial stylistic element is the presence of dialogues, with each person's speech starting a new paragraph. A collection of essays which does not have that property is not sufficiently similar to *War and Peace*, and the system does not do as well on it.

7 Discussion and Future Work

We built a system that can assign paragraph boundaries with accuracy significantly exceeding baseline. This accuracy, however, is not very high. Undoubtedly, this stems from the intrinsic complexity of the problem. While we have yet to measure how accurate humans are at this task, but we anticipate the accuracy to be within 10% of our best results. Anecdotal evidence (one person, one WSJ article) shows accuracy of 70%, which is only slightly above the system's performance.

It appears that the features that are useful for text segmentation tend to be useful for this task as well, but to a lesser extent. Indeed, our results indicate that the best lexical features tend to be punctuation, unlike in the case of the text segmentation. Of course, this result is not surprising, since it would be fairly strange for first sentences of paragraphs to differ lexically from the other sentences to any significant extent. Minor structural changes, however, do occur, since they correspond to stylistic changes made by the author. We have seen examples of them in the case of *War and Peace*. The coherence measures (cosine measures in our model) appear to be useful, which indicates that there indeed happen to be topic shifts around the paragraph boundaries.

In both datasets parsing information does indeed turn out to be useful, as we observed in several examples. The discourse-based features turned out to be only indirectly useful.

We also found that it is often possible to train on a similar text and achieve reasonable performance. While we have yet to do a comprehensive study to verify that the effect is pervasive and not an artifact of choosing a particular set of texts, our early results are very promising. It appears likely that a system can be built that can suggest potential paragraph breaks to a text editor or word processor user as he or she finishes typing a sentence. No training would be required from the user, except perhaps to identify the writing style used.

Our future work will involve trying more features, as well as examining in detail why particular features help. We also plan to use a more complex algorithm to see if accuracy can be further improved. In particular, it is worth exploring whether an algorithm that is sequential in nature can capture certain other constraints, such as average paragraph length. We might use a Conditional Random Field to model the discourse state.

We also intend to verify the system's cross-text performance in a more systematic way. In particular, we want to make sure that for most writing styles we have a training corpus that provides good performance for this style.

References

1. Richardson, S.: Microsoft natural language understanding system and grammar checker. In: Proceedings of Fifth Conference on Applied Natural Language Processing: System Demos (ANLP–97). (1997)
2. Fukumoto, F., Suzuki, Y.: Detecting shifts in news stories for paragraph extraction. In: Proceedings of 19th International Conference on Computational Linguistics (COLING-02). (2002)
3. Genzel, D., Charniak, E.: Variation of entropy and parse trees of sentences as a function of the sentence number. In: Proceedings of EMNLP–03, Sapporo, Japan. (2003)
4. Bolshakov, I.A., Gelbukh, A.F.: Text segmentation into paragraphs based on local text cohesion. In: Lecture Notes in Artificial Intelligence #2166. Springer-Verlag (2001) 158–166
5. Hearst, M.: TextTiling: Segmenting text into multi-paragraph subtopic passages. Computational Linguistics **23** (1997)
6. Beeferman, D., Berger, A., Lafferty, J.: Text segmentation using exponential models. In: Proceedings of EMNLP–97. (1997)
7. Soricut, R., Marcu, D.: Sentence level discourse parsing using syntactic and lexical information. In: Proceedings of HLT/NAACL–03. (2003)
8. Bouayad-Agha, N., Power, R., Scott, D.: Can text structure be incompatible with rhetorical structure? In: Proceedings of the International Natural Language Generation Conference (INLG-2000). (2000)
9. Collins, M.: Ranking algorithms for named-entity extraction: Boosting and the voted perceptron. In: Proceedings of ACL–02. (2002)
10. Rosenblatt, F.: The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review **65** (1958) 386–408
11. Grosz, B., Joshi, A., Weinstein, S.: Centering: a framework for modelling the local coherence of discourse. Computational Linguistics **21** (1995) 203–226
12. Blaheta, D., Charniak, E.: Assigning function tags to parsed text. In: Proceedings of NAACL–00. (2000)
13. Marcus, M.P., Santorini, B., Marcinkiewicz, M.A.: Building a large annotated corpus of English: the Penn treebank. Computational Linguistics **19** (1993) 313–330
14. Tolstoy, L.: War and Peace. Available online, in 4 languages (Russian, English, Spanish, Italian): http://www.magister.msk.ru/library/tolstoy/wp/wp00.htm (1869)
15. Charniak, E.: A maximum-entropy-inspired parser. In: Proceedings of ACL–01, Toulouse. (2001)
16. Johnson, M.: A simple pattern-matching algorithm for recovering empty nodes and their antecedents. In: Proceedings of ACL–02. (2002)
17. Cohen, P.: Empirical methods for artificial intelligence. MIT Press, Cambridge, MA (1995)

Author Index

Amith, Jonathan D. 474
Araki, Kenji 620
Arranz, Victoria 250
Atserias, Jordi 250

Bae, Myung Jin 429, 437
Bandyopadhyay, Sivaji 649
Bao, Yubin 593, 735
Bellynck, Valérie 324
Berdichevsky, Alexander S. 388
Bhembe, Dumisizwe 704
Biemann, Chris 773
Bigert, Johnny 142
Bilac, Slaven 413
Boguslavsky, Igor M. 377, 388
Boitet, Christian 324, 357
Bolshakov, Igor A. 803
Bu, Ki-Dong 624
Buscaldi, Davide 263, 267

Calvo, Hiram 177
Cardeñosa, Jesús 377
Carthy, Joe 645
Castillo, Mauro 250
Chen, Huowang 214
Chen, Qing 167
Chen, Yaodong 214
Chen, Ying 548
Chen, Zushun 333
Choi, Ho-cheol 198
Chollet, Gérard 441
Chowdhury, Nirmalya 715
Christodoulakis, Dimitris 604
Cristea, Dan 632
Cruz, Carlos Méndez 653

Dalli, Angelo 723
Debusmann, Ralph 25
de llarraza, Amanza Díaz 793
Desai, Kirtan 112
Díaz, Isabel 560
Dinu, Anca 83, 785
Dinu, Liviu P. 83, 785
Doran, William 645
Douglas, Benjamin 548
Dunnion, John 645

Fuentes, Inmaculada 560

Galicia-Haro, Sofía N. 337
Gallardo, Carolina 337
Gao, Feng 452
Garrett, Edward John 463
Gelbukh, Alexander 177, 337, 628
Genzel, Dmitriy 816
Giuliano, Claudio 498
Gliozzo, Alfio Massimiliano 242, 498
Gojenola, Koldo 793
Graehl, Jonathan 1
Graña, Jorge 120
Gross, Zuriel 657
Guthrie, Louise 723

Hacioglu, Kadri 548
HaCohen-Kerner, Yaakov 657
Han, Sang-yong 198, 628
Hilario, Melanie 522
Hlaváčová, Jaroslava 189
Hroza, Jiří 608

Iomdin, Leonid L. 388
Iraola, Luis 377
Isahara, Hitoshi 293

James, Vinosh Babu 789
Ji, Donghong 155, 238, 572
Jiménez-Salazar, Héctor 719
Jo, Wangrae 437
Johansson, Christer 694
Johnsen, Lars G. 694
Jordan, Pamela W. 704

Kanamaru, Toshiyuki 293
Kang, NamO 628
Kenwright, John 324
Kilgarriff, Adam 177

Kim, Hae-Jung 624
Kim, Jee-Hyub 522
Kim, Jong-Bok 60
Kim, Jongkuk 429, 437
Kim, Junghyun 624
Knight, Kevin 1
Knutsson, Ola 142
Koster, Cornelis H.A. 48
Kreydlin, Leonid G. 388
Kühnlein, Peter 222
Kulkarni, Anagha 226

Lazursky, Alexander V. 388
Le, Viet Bac 433
Lee, Ki Young 429
Lee, Sang-Jo 624
Li, Xiaoguang 593, 735
Lin, Shouxun 132
Liu, Shaoming 584
Liu, Qun 132
López-López, Aurelio 612, 762
Lu, Ru-Zhan 452
Luo, Shengfen 202

Mahmud, Rohana 116
Makagonov, Pavel 746
Marcu, Daniel 88
Marge, Matthew 341
Marrafa, Palmira 37
Masa, Asaf 657
Maxwell, Mike 474
Medina Urrea, Alfonso 189, 653
Méndez Cruz, Carlos 653
Meza, Ivan V. 73
Mihalcea, Rada 100
Mityushin, Leonid G. 388
Montes-y-Gómez, Manuel 246, 263, 267, 433, 539, 612
Moreno, Lidia 560
Moyotl-Hernández, Edgar 719
Murata, Masaki 293

Nahm, Un Yong 535
Nastase, Vivi 312
Newman, Eamonn 645
Niu, Zheng-Yu 238
Nunes, Maria das Graças Volpe 352

Orăsan, Constantin 670
Oronoz, Maite 793
Otero, Juan 120

Pala, Karel 305
Pancardo-Rodríguez, Aarón 246, 267
Pappuswamy, Umarani 704
Paşca, Marius 280
Pastor, Oscar 560
Pedersen, Ted 226
Pelizzoni, Jorge Marques 352
Pérez-Coutiño, Manuel 433, 612
Pineda, Luis A. 73
Pineda, Luis Villaseñor 267, 539, 612
Pistol, Ionuț 632
Popescu, Ana-Maria 88
Postolache, Oana 25, 632
Purandare, Amruta 226

Ramsay, Allan 116
Ranieri, Marcello 242
Ren, Fuji 400
Riloff, Ellen 486
Rinaldi, Raffaella 498
Rosso, Paolo 246, 263, 267
Ruiz Figueroa, Alejandro 746
Rus, Vasile 112

Saha, Diganta 715
Sahlgren, Magnus 142
Sankaran, Baskaran 789
Sarkar, Kamal 649
Sassen, Claudia 222
Schneider, Karl-Michael 682
Sedláček, Radek 305
Shi, Lei 100
Shi, Wuguang 769
Siefkes, Christian 510
Singhai, Mohit 341
Sizov, Victor G. 388
Sjöbergh, Jonas 142
Skowron, Marcin 620
Solorio, Thamar 612, 762
Stamou, Sofia 604
Stent, Amanda 341
Stokes, Nicola 645
Strapparava, Carlo 242

Su, Jian 155, 218, 750
Sun, Maosong 202, 584
Szpakowicz, Stan 312

Tan, Chew-Lim 238
Tan, Yongmei 167
Tanaka, Hozumi 413
Téllez-Valero, Alberto 539
Teresniak, Sven 773
Tomokiyo, Mutsuko 441
Traat, Maarika 25
T'sou, Benjamin K. 202

VanLehn, Kurt 704
Vilares, Manuel 120
Villaseñor-Pineda, Luis 246, 267, 433, 539, 612

Wang, Daling 593, 735
Wang, Hongtao 584
Wang, Houfeng 769
Wang, Ruichao 645
Wang, Ting 214
Wang, Xiaojie 400
Wiebe, Janyce 486
Wilks, Yorick 723
Wu, Wei-Lin 452

Xia, Yunqing 723
Xiong, Deyi 132

Yang, Jaehyung 60
Yang, Lingpeng 155, 218, 238, 572, 750
Yang, Xiaofeng 218
Yao, Tianshun 167
Yoon, Kyuchul 425
Yu, Ge 593, 735
Yuan, Yan 452
Yu, Nie 572

Zhou, Guodong 155, 572, 750
Zhou, Qiang 333
Zhu, Jingbo 167
Žižka, Jan 608