

Alexander Gelbukh (Ed.)

LNCS 3878

Computational Linguistics and Intelligent Text Processing

**7th International Conference, CICLing 2006
Mexico City, Mexico, February 2006
Proceedings**



 Springer

Commenced Publication in 1973

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison

Lancaster University, UK

Takeo Kanade

Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler

University of Surrey, Guildford, UK

Jon M. Kleinberg

Cornell University, Ithaca, NY, USA

Friedemann Mattern

ETH Zurich, Switzerland

John C. Mitchell

Stanford University, CA, USA

Moni Naor

Weizmann Institute of Science, Rehovot, Israel

Oscar Nierstrasz

University of Bern, Switzerland

C. Pandu Rangan

Indian Institute of Technology, Madras, India

Bernhard Steffen

University of Dortmund, Germany

Madhu Sudan

Massachusetts Institute of Technology, MA, USA

Demetri Terzopoulos

New York University, NY, USA

Doug Tygar

University of California, Berkeley, CA, USA

Moshe Y. Vardi

Rice University, Houston, TX, USA

Gerhard Weikum

Max-Planck Institute of Computer Science, Saarbruecken, Germany

Alexander Gelbukh (Ed.)

Computational Linguistics and Intelligent Text Processing

7th International Conference, CICLing 2006
Mexico City, Mexico, February 19-25, 2006
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: 2006920389

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

LNCS Sublibrary: SL 1 – Theoretical Computer Science and General Issues

ISSN 0302-9743
ISBN-10 3-540-32205-1 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-32205-4 Springer Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

Springer is a part of Springer Science+Business Media

springer.com

© Springer-Verlag Berlin Heidelberg 2006
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11671299 06/3142 5 4 3 2 1 0

Preface

CICLing 2006 (www.CICLing.org) was the 7th Annual Conference on Intelligent Text Processing and Computational Linguistics. The CICLing conferences are intended to provide a wide-scope forum for discussion of the internal art and craft of natural language processing research and the best practices in its applications.

This volume contains the papers included in the main conference program (full papers) and selected papers from the poster session (short papers). Other poster session papers were included in a special issue of the journal *Research on Computing Science*; see information on this issue on the website. The previous CICLing conferences since 2001 were also published in Springer's *Lecture Notes in Computer Science* (LNCS) series, vol. 2004, 2276, 2588, 2945, and 3406.

The number of submissions to CICLing 2006 was higher than that of the previous conferences: 141 full papers and 35 short papers by 480 authors from 37 countries were submitted for evaluation, see Tables 1 and 2. Each submission was reviewed by at least two independent Program Committee members. This book contains revised versions of 43 full papers (presented orally at the conference) and 16 short papers (presented as posters) by 177 authors from 24 countries selected for inclusion in the conference program. The acceptance rate was 30.4% for full papers and 45.7% for short papers.

The book is structured into two parts subdivided into 14 sections representative of the main tasks and applications of natural language processing:

Computational Linguistics Research

- Lexical Resources
- Corpus-Based Knowledge Acquisition
- Morphology and Part-of-Speech Tagging
- Syntax and Parsing
- Word Sense Disambiguation and Anaphora Resolution
- Semantics
- Text Generation
- Natural Language Interfaces and Speech Processing

Intelligent Text Processing Applications

- Information Retrieval
- Question Answering
- Text Summarization
- Information Extraction and Text Mining
- Text Classification
- Authoring Tools and Spelling Correction

The volume features invited papers by Eduard Hovy of the Information Sciences Institute, University of Southern California, Nancy Ide of Vassar College, and Rada Mihalcea of the University of North Texas, who presented excellent

Table 1. Statistics of submissions and accepted papers by country or region

Country or region	Authors		Papers ¹		Country or region	Authors		Papers ¹	
	Subm	Accp	Subm	Accp		Subm	Accp	Subm	Accp
Algeria	2	–	1	–	Korea, South	67	17	29	7
Argentina	1	1	0.5	0.5	Lebanon	1	–	1	–
Austria	6	–	1	–	Mexico	51	24	17.65	7.23
Belgium	2	1	1.2	0.2	Netherlands	1	1	0.5	0.5
Brazil	10	10	3.33	3.33	Norway	2	–	1	–
Canada	10	5	5	2	Portugal	6	6	1.5	1.5
Chile	3	–	0.65	–	Romania	2	–	1.5	–
China	68	19	22	5.45	Russia	5	1	2.25	0.25
Costa Rica	1	–	0.5	–	Singapore	1	1	0.25	0.25
Cuba	1	1	0.25	0.25	Spain	49	22	14.1	6
Czech Republic	9	2	5	1	Sweden	2	–	2	–
France	12	1	5.7	1	Taiwan	12	3	4	1
Germany	2	1	0.53	0.33	Tunisia	3	3	1	1
Hong Kong	25	12	9.8	4.8	Turkey	3	–	2	–
India	21	2	10	1	UAE	2	–	1	–
Ireland	4	–	1	–	UK	3	2	0.8	0.6
Israel	3	–	1	–	USA	29	22	11.27	8
Italy	3	2	1	0.5	Uruguay	1	–	0.5	–
Japan	57	18	15.25	5.5	<i>Total:</i>	<i>480</i>	<i>177</i>	<i>176</i>	<i>59</i>

¹ Counted by authors; e.g. for a paper by 3 authors: 2 from Mexico and 1 from USA, we added $\frac{2}{3}$ to Mexico and $\frac{1}{3}$ to USA.

keynote lectures at the conference. Publication of extended full-text invited papers in the proceedings is a distinctive feature of CICLing conferences. What is more, in addition to presentation of their invited papers, the keynote speakers organized separate vivid informal discussions and encouraging tutorials—which is also a distinctive feature of this conference series.

The following papers received the Best Paper Awards and the Best Student Paper Award, correspondingly:

- 1st Place: Shallow Case Role Annotation Using Two-Stage Feature-Enhanced String Matching, by Samuel Chan
- 2nd Place: Finite State Grammar Transduction from Distributed Collected Knowledge, by Rakesh Gupta and Ken Hennacy
- 3rd Place: Automatic Acquisition of Question Reformulations for Question Answering, by Jamileh Yousefi and Leila Kosseim
- Student: Clustering Abstracts of Scientific Texts Using the Transition Point Technique, by David Pinto, Héctor Jiménez-Salazar and Paolo Rosso

The Best Student Paper was selected out of papers with the first author being a full-time student. The authors of the awarded papers were given extended time for their presentations. In addition, the Best Presentation Award and Best Poster Award winners were selected by a ballot among the participants of the conference.

Table 2. Statistics of submissions and accepted papers by topic²

Topic	Submitted	Accepted	
Theories and formalisms	9	2	22%
Lexical resources	29	13	44%
Statistical methods and machine learning	35	13	37%
Corpus linguistics	21	11	52%
Morphology	11	5	45%
Syntax (linguistic aspects)	12	4	33%
Parsing (technical aspects)	13	5	38%
Ambiguity resolution	16	9	56%
Word sense disambiguation	16	7	43%
Anaphora resolution	3	1	33%
Semantics	29	9	31%
Knowledge representation	26	4	15%
Text generation	4	4	100%
Machine translation	7	1	14%
Discourse and dialogue	8	4	50%
Natural language interfaces	10	3	30%
Speech recognition	8	3	37%
Speech synthesis	2	1	50%
Information retrieval	42	11	26%
Information extraction	25	4	16%
Text mining	26	6	23%
Summarization	6	3	50%
Text categorization	21	4	19%
Text clustering	12	5	41%
Spell checking	2	1	50%
Other: computational linguistics art and craft	12	2	16%
Other: text processing applications	38	13	34%

² According to the topics indicated by the authors. A paper may be assigned to more than one topic.

Besides its high scientific level, one of the success factors of CICLing conferences is their excellent cultural program. CICLing 2006 was held in Mexico, a wonderful country rich in culture, history, and nature. The participants of the conference had a chance to see the solemn 2000-year-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.

I want to thank everyone involved in the organization of this conference. Firstly, the authors of the papers constituting this book: it is the excellence of their research work that gives value to the book and sense to the work of all other people involved. I thank the Program Committee members for their hard and very professional work on reviewing so many submissions in a short time.

Very special thanks go to Manuel Vilares and his group, John Tait and his group, Nicolas Nikolov, Rada Mihalcea, Ted Pedersen, and Oana Postolache for their invaluable support in the reviewing process. The Best Paper Award selection working group included Alexander Gelbukh, Eduard Hovy, Rada Mihalcea, Ted Pedersen, and Yorick Wiks.

The entire submission, reviewing, and selection process, as well as putting together the proceedings, was supported for free by the EasyChair system (www.EasyChair.org); I express my gratitude to its author Andrei Voronkov for his constant support and help. I also express my most cordial thanks to the members of the local Organizing Committee for their considerable contribution to making this conference become a reality, and to our sponsoring organization—the Center for Computing Research (CIC) of the National Polytechnic Institute (IPN), Mexico—for hosting the conference. Last but not least, I deeply appreciate Springer staff’s patience and help in editing this volume—it is always a great pleasure to work with them.

December 2005

Alexander Gelbukh

Organization

CICLing 2006 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.

Program Chair

Alexander Gelbukh

Program Committee

Eneko Agirre	Rada Mihalcea
Christian Boitet	Ruslan Mitkov
Igor Bolshakov	Masaki Murata
Nicoletta Calzolari	Vivi Nastase
John Carroll	Olga Nevzorova
Dan Cristea	Nicolas Nicolov
Barbara Di Eugenio	Sergei Nirenburg
Gregory Grefenstette	Constantin Orasan
Linda van Guilder	Manuel Palomar
Cătălina Hallett	Ted Pedersen
Yasunari Harada	Viktor Pekar
Eduard Hovy	Stelios Piperidis
Nancy Ide	James Pustejovsky
Diana Inkpen	Fuji Ren
Frederick Jelinek	Fabio Rinaldi
Aravind Krishna Joshi	Horacio Rodriguez
Martin Kay	Vasile Rus
Alma Kharrat	Ivan Sag
Adam Kilgarriff	Franco Salveti
Richard Kittredge	Serge Sharoff
Kevin Knight	Grigori Sidorov
Alexander Koller	Thamar Solorio
Grzegorz Kondrak	Carlo Strapparava
Sandra Kuebler	Maosong Sun
Ken Litkowski	John Tait
Hugo Liu	Benjamin Ka-yin T'sou
Aurelio López López	Felisa Verdejo
Bernardo Magnini	Karin Verspoor
Daniel Marcu	Manuel Vilares Ferro
Carlos Martín-Vide	Yorick Wilks
Igor Mel'čuk	

Additional Referees

Mohamed Abdel Fattah
Mustafa Abusalah
Farooq Ahmad
Iñaki Alegria
Muath Alzghool
Bogdan Babych
Verginica Barbu Mititelu
Fco. Mario Barcala Rodríguez
Francesca Bertagna
Dimitar Blagoev
Hiram Calvo Castro
Anna Clark
Daoud Clarke
Andras Csomai
Victor Manuel Darriba Bilbao
Jeremy Ellman
Davide Fossati
Oana Frunza
Irbis Gallegos
Jorge Graña
Samer Hassan
David Hope
Scott K. Imig
Aminul Islam
Shih-Wen Ke
Stephan Kepser

Rob Koeling
Alberto Lavelli
Fennie Liang
Christian Loza
Xin Lu
Fernando Magán Muñoz
Raquel Martínez
Jaime Mendez
Dragos Stefan Munteanu
Crystal Nakatsu
Apostol Natsev
Matteo Negri
Michael Oakes
Octavian Popescu
Oana Postolache
Christoph Reichenbach
Francisco Ribadas Peña
German Rigau
Tarek Sherif
Radu Soricut
Chris Stokoe
Rajen Subba
Hristo Tanev
Martin Thomas
Jesus Vilares Ferro
Zhuli Xie

Organizing Committee

Hiram Calvo Castro
Hugo Coyote Estrada
Ignacio García Araoz
Alexander Gelbukh
Martín Haro Martínez

Oralia del Carmen Pérez Orozco
Marisol Pineda Pérez
Jorge Sosa Sánchez
Javier Tejada Cárcamo
Sulema Torres Ramos

Website and Contact

The website of CICLing conferences is www.CICLing.org. It contains information on past CICLing events and satellite workshops, abstracts of all published papers, photos from all CICLing events, and video recordings of some keynote talks, as well as information on forthcoming CICLing events. Contact: gelbukh@cicling.org, gelbukh@gelbukh.com; more contact options can be found on the website.

Table of Contents

Computational Linguistics Research

Lexical Resources

Invited paper:

- Integrating Semantic Frames from Multiple Sources
Namhee Kwon, Eduard Hovy 1

Invited paper:

- Making Senses: Bootstrapping Sense-Tagged Lists of
Semantically-Related Words
Nancy Ide 13

- Enriching Wordnets with New Relations and with Event and Argument
Structures
Raquel Amaro, Rui Pedro Chaves, Palmira Marrafa, Sara Mendes ... 28

- Experiments in Cross-Language Morphological Annotation Transfer
Anna Feldman, Jirka Hana, Chris Brew 41

- Sentence Segmentation Model to Improve Tree Annotation Tool
So-Young Park, Dongha Shin, Ui-Sung Song 51

- Markov Cluster Shortest Path Founded Upon the Alibi-Breaking
Algorithm
Jaeyoung Jung, Maki Miyake, Hiroyuki Akama 55

Corpus-Based Knowledge Acquisition

- Unsupervised Learning of Verb Argument Structures
Thiago Alexandre Salgueiro Pardo, Daniel Marcu,
Maria das Graças Volpe Nunes 59

- A Methodology for Extracting Ontological Knowledge from Spanish
Documents
Rafael Valencia-García, Dagoberto Castellanos-Nieves,
Jesualdo Tomás Fernández-Breis, Pedro José Vivancos-Vicente 71

- Automatically Determining Allowable Combinations of a Class of
Flexible Multiword Expressions
Afsaneh Fazly, Ryan North, Suzanne Stevenson 81

Web-Based Measurements of Intra-collocational Cohesion in Oxford Collocations Dictionary
Igor A. Bolshakov, Sofía Natalia Galicia-Haro 93

Probabilistic Neural Network Based English-Arabic Sentence Alignment
Mohamed Abdel Fattah, Fuji Ren, Shingo Kuroiwa 97

Morphology and Part-of-Speech Tagging

Towards the Automatic Lemmatization of 16th Century Mexican Spanish: A Stemming Scheme for the CHEM
Alfonso Medina-Urrea 101

Word Frequency Approximation for Chinese Without Using Manually-Annotated Corpus
Maosong Sun, Zhengcao Zhang, Benjamin Ka-Yin T'sou, Huaming Lu 105

Abbreviation Recognition with MaxEnt Model
Chunyu Kit, Xiaoyue Liu, Jonathan J. Webster 117

An Efficient Multi-agent System Combining POS-Taggers for Arabic Texts
Chiraz Ben Othmane Zribi, Aroua Torjmen, Mohamed Ben Ahmed ... 121

Syntax and Parsing

A Comparative Evaluation of a New Unsupervised Sentence Boundary Detection Approach on Documents in English and Portuguese
Jan Strunk, Carlos N. Silla Jr., Celso A.A. Kaestner 132

A General and Multi-lingual Phrase Chunking Model Based on Masking Method
Yu-Chieh Wu, Chia-Hui Chang, Yue-Shi Lee 144

UCSG Shallow Parser
Guntur Bharadwaja Kumar, Kavi Narayana Murthy 156

Evaluating the Performance of the Survey Parser with the NIST Scheme
Alex Chengyu Fang 168

Sequences of Part of Speech Tags vs. Sequences of Phrase Labels: How Do They Help in Parsing?
Gabriel Infante-Lopez, Maarten de Rijke 180

Word Sense Disambiguation and Anaphora Resolution

Verb Sense Disambiguation Using Support Vector Machines: Impact of WordNet-Extracted Features <i>Davide Buscaldi, Paolo Rosso, Ferran Pla, Encarna Segarra, Emilio Sanchis Arnal</i>	192
Preposition Senses: Generalized Disambiguation Model <i>Chutima Boonthum, Shunichi Toida, Irwin Levinstein</i>	196
An Unsupervised Language Independent Method of Name Discrimination Using Second Order Co-occurrence Features <i>Ted Pedersen, Anagha Kulkarni, Roxana Angheluta, Zornitsa Kozareva, Thamar Solorio</i>	208
Extracting Key Phrases to Disambiguate Personal Names on the Web <i>Danushka Bollegala, Yutaka Matsuo, Mitsuru Ishizuka</i>	223
Chinese Noun Phrase Metaphor Recognition with Maximum Entropy Approach <i>Zhimin Wang, Houfeng Wang, Huiming Duan, Shuang Han, Shiwen Yu</i>	235
Zero Anaphora Resolution in Chinese Discourse <i>Yuzhen Cui, Qinan Hu, Haihua Pan, Jianhua Hu</i>	245
Semantics	
<i>Invited paper:</i> Random Walks on Text Structures <i>Rada Mihalcea</i>	249
<i>Best paper award (first place):</i> Shallow Case Role Annotation Using Two-Stage Feature-Enhanced String Matching <i>Samuel W.K. Chan</i>	263
SPARTE, a Test Suite for Recognising Textual Entailment in Spanish <i>Anselmo Peñas, Álvaro Rodrigo, Felisa Verdejo</i>	275
Analysis of a Textual Entailer <i>Vasile Rus, Philip M. McCarthy, Arthur C. Graesser</i>	287

Text Generation

Referring Via Document Parts <i>Ivandr� Paraboni, Kees van Deemter</i>	299
Generation of Natural Language Explanations of Rules in an Expert System <i>Mar�a de los �ngeles Alonso-Lavernia, Argelio V�ctor De-la-Cruz-Rivera, Grigori Sidorov</i>	311
Generating a Set of Rules to Determine Honorific Expression Using Decision Tree Learning <i>Kanako Komiya, Yasuhiro Tajima, Nobuo Inui, Yoshiyuki Kotani</i>	315

Natural Language Interfaces and Speech Processing

NLP (Natural Language Processing) for NLP (Natural Language Programming) <i>Rada Mihalcea, Hugo Liu, Henry Lieberman</i>	319
Balancing Transactions in Practical Dialogues <i>Luis Albreto Pineda Cort�s, Hayde Castellanos, Sergio Rafael Coria Olguin, Varinia Estrada, Fernanda L�pez, Isabel L�pez, Ivan Meza, Iv�n Moreno, Patricia P�rez, Carlos Rodr�guez</i>	331
<i>Best paper award (second place):</i> Finite State Grammar Transduction from Distributed Collected Knowledge <i>Rakesh Gupta, Ken Hennacy</i>	343
Predicting Dialogue Acts from Prosodic Information <i>Sergio Rafael Coria Olguin, Luis Albreto Pineda Cort�s</i>	355
Disambiguation Based on Wordnet for Transliteration of Arabic Numerals for Korean TTS <i>Youngim Jung, Aesun Yoon, Hyuk-Chul Kwon</i>	366

Intelligent Text Processing Applications

Information Retrieval

<i>MFCRank</i> : A Web Ranking Algorithm Based on Correlation of Multiple Features <i>Yunming Ye, Yan Li, Xiaofei Xu, Joshua Huang, Xiaojun Chen</i>	378
---	-----

On Text Ranking for Information Retrieval Based on Degree of Preference <i>Bo-Yeong Kang, Dae-Won Kim</i>	389
Lexical Normalization and Relationship Alternatives for a Term Dependence Model in Information Retrieval <i>Marco Gonzalez, Vera Lúcia Strube de Lima, José Valdeni de Lima</i> . .	394
Web Search Model for Dynamic and Fuzzy Directory Search <i>Bumghi Choi, Ju-Hong Lee, Sun Park, Tae-Su Park</i>	406
Information Retrieval from Spoken Documents <i>Michal Fapšo, Pavel Smrž, Petr Schwarz, Igor Szöke, Milan Schwarz, Jan Černocký, Martin Karafiát, Lukáš Burget</i>	410
Automatic Image Annotation Based on WordNet and Hierarchical Ensembles <i>Wei Li, Maosong Sun</i>	417
Creating a Testbed for the Evaluation of Automatically Generated Back-of-the-Book Indexes <i>Andras Csomai, Rada F. Mihalcea</i>	429

Question Answering

<i>Best paper award (third place):</i> Automatic Acquisition of Semantic-Based Question Reformulations for Question Answering <i>Jamileh Yousefi, Leila Kosseim</i>	441
Using N-gram Models to Combine Query Translations in Cross-Language Question Answering <i>Rita M. Aceves-Pérez, Luis Villaseñor-Pineda, Manuel Montes-y-Gómez</i>	453
A Question Answering System on Special Domain and the Implementation of Speech Interface <i>Haiqing Hu, Fuji Ren, Shingo Kuroiwa, Shuwu Zhang</i>	458

Text Summarization

Multi-document Summarization Based on BE-Vector Clustering <i>Dexi Liu, Yanxiang He, Donghong Ji, Hua Yang</i>	470
---	-----

Deriving Event Relevance from the Ontology Constructed with Formal
 Concept Analysis
Wei Xu, Wenjie Li, Mingli Wu, Wei Li, Chunfa Yuan 480

A Sentence Compression Module for Machine-Assisted Subtitling
Nadjet Bouayad-Agha, Angel Gil, Oriol Valentin, Victor Pascual 490

Information Extraction and Text Mining

Application of Semi-supervised Learning to Evaluative Expression
 Classification
Yasuhiro Suzuki, Hiroya Takamura, Manabu Okumura 502

A New Algorithm for Fast Discovery of Maximal Sequential Patterns
 in a Document Collection
*René Arnulfo García-Hernández, José Francisco Martínez-Trinidad,
 Jesús Ariel Carrasco-Ochoa* 514

A Machine Learning Based Approach for Separating Head from Body
 in Web-Tables
Sung-Won Jung, Hyuk-Chul Kwon 524

Text Classification

Best student paper award:
 Clustering Abstracts of Scientific Texts Using the Transition Point
 Technique
David Pinto, Héctor Jiménez-Salazar, Paolo Rosso 536

Sense Cluster Based Categorization and Clustering of Abstracts
*Davide Buscaldi, Paolo Rosso, Mikhail Alexandrov,
 Alfons Juan Ciscar* 547

Analysing Part-of-Speech for Portuguese Text Classification
Teresa Gonçalves, Cassiana Silva, Paulo Quaresma, Renata Vieira . . . 551

Improving kNN Text Categorization by Removing Outliers from
 Training Set
Kwangcheol Shin, Ajith Abraham, Sang Yong Han 563

Authoring Tools and Spelling Correction

Writing for Language-Impaired Readers
Aurélien Max 567

Document Copy Detection System Based on Plagiarism Patterns <i>NamOh Kang, SangYong Han</i>	571
Regional vs. Global Robust Spelling Correction <i>Manuel Vilares Ferro, Juan Otero Pombo,</i> <i>Víctor Manuel Darriba Bilbao</i>	575
Author Index	587

Integrating Semantic Frames from Multiple Sources

Namhee Kwon and Eduard Hovy

Information Sciences Institute, University of Southern California,
Marina del Rey, CA 90292, USA
{`nkwon`, `hovy`}@isi.edu

Abstract. Semantic resources of predicate-argument structure have high potential to enable increased quality in language understanding. Several alternative frame collections exist, but they cover different sets of predicates and use different role sets. We integrate semantic frame information given a predicate verb using three available collections: FrameNet, PropBank, and the LCS database. For each word sense in WordNet, we automatically assign the corresponding FrameNet frame and align frame roles between FrameNet and PropBank frames and between FrameNet and LCS frames, and verify the results manually. The results are available as part of ISI's Omega ontology.

1 Introduction

With more accurate semantic analysis, systems should obtain higher performance in many applications such as machine translation, question answering, and summarization. Thanks to the release of annotated corpora with semantic argument structures and manually constructed lexical-semantic information such as FrameNet [1], PropBank [10], LCS database [3], and VerbNet [11], many models inducing semantic frames have been developed ([7], [6], [13], [17], [18]).

Such data collections cover different sets of predicates. Unfortunately, no collection covers all (or most) of the (English) predicates, and the roles and other definitional aspects of the collections differ. Due to these differences, most approaches to semantic analysis using these available resources (semantic role tagging) are specific to only one of these resources and their results are not comparable and usable over other resources.

We believe that we can build a broader and consistent semantic resource by integrating all semantic frame information from these disparate collections. The value of the integrated resource is apparent at many levels: first, as a theoretical device to highlight differences and generate further refinements in lexical semantic theory; second, as a practical resource that can be used by semantic analysis and other applications; third, as a testbed for an automatic aligning method between different resources that can also be applied to more general integration of lexical information. As more such annotated collections are created in the future, the value of (semi)automatic alignment/integration processes will increase.

In this paper, we provide a method to combine individual semantic resources—FrameNet, PropBank, and LCS¹—and investigate similarities and differences among these collections. We align frames, including semantic roles, for each lexical sense defined in WordNet 2.0 [4]. We currently work with verbs only, since PropBank and LCS define only verb predicates.

Shi and Mihalcea have attempted to combine FrameNet frames and VerbNet verb classes using WordNet semantic networks in [15] and generate a rule-based semantic parser in [14]. Their rule-based parser is a good example of utilizing an integrated resource. Different from their work, we also combine PropBank argument structures and provide automatic method to assign FrameNet frame to each WordNet sense with a careful evaluation/correction process.

The rest of the paper is organized as follows: Section 2 describes frame resources in detail, Section 3 explains the automatic process for frame and role alignment, Section 4 provides the system evaluation and results, Section 5 discusses a process of expanding in the ontology, and finally, Section 6 concludes.

2 Lexical Resources

We gather frame information from three different collections (FrameNet, PropBank, and LCS), and produce an integrated frame alignment, at the word sense level represented in the WordNet. Each resource has different coverage of lexicons, and provides different frame definitions and different generalization levels in role naming.

FrameNet. FrameNet [1] defines semantic frames which are semantic representations of situations involving various participants, properties, and other conceptual roles. For each frame, corresponding predicating words are associated, and each frame consists of a specified subset of relations drawn from a set of role names such as *Communicator*, *Speaker*, and *Traveler*. FrameNet release 1.1 (Jan. 2004) defines 487 distinct frames with 696 distinct roles, and 6,743 predicates (2,300 verbs, 3,103 nouns, 1,264 adjectives, etc.) are associated with the frames.²

PropBank. The Proposition Bank project adds semantic information to the Penn Treebank [10]. It defines a predicate-argument structure on a per-predicate basis, not at the frame level as the FrameNet. The core arguments of each predicate are simply numbered while keeping general role names such as “temporal” or “locative”. The numbered arguments (*Arg0*, *Arg1*, ..., *Arg5*) are specific to a predicate verb, in other words, *Arg1* of one verb and *Arg1* of another verb do not necessarily denote the same role. PropBank (Feb. 2004) covers 3,323 predicate verbs and 4,659 framesets.

¹ Both semantic structures of LCS (Lexical Conceptual Structure) database and VerbNet are based on Levin’s verb classes [12]. In this work, we integrated LCS (rather than VerbNet), since it has more subdivided classes.

² Data release 1.2 became available (Jul. 2005) after we had started this research.

LCS Database. The Lexical Conceptual Structure [3] represents abstract language-independent conceptual information including semantic structures. The semantic structure of a verb is based on Levin’s English verb classes [12] which assumes that syntactic verb-argument frames represent inherent semantics. The semantic structure uses 15 thematic roles such as *Agent*, *Theme*, and *Predicate*. The database contains 4,432 verbs in 492 classes.

WordNet. WordNet [4] is a lexical network covering most of English words (11,306 verbs, 114,648 nouns, 21,436 adjectives, and 4,669 adverbs) including semantic relations between lexical units. Several terms in this paper are from WordNet, and the following is a short description excerpted:

Sense: a meaning of a word in WordNet.

Synset: a synonym set; a set of words that are interchangeable in some context.

Hypernym: the generic term used to designate a whole class of specific instances. Y is a hypernym of X if X is a (kind of) Y.

Hyponym: The specific term used to designate a member of a class. X is a hyponym of Y if X is a (kind of) Y.

3 Integrating Process

Since a single lexical item can have more than one frame depending on its word sense, we align frames for the item’s WordNet sense. Given a WordNet sense, we assign the corresponding frames from each source to show the alignment between frames. The PropBank and LCS frames have been previously assigned to each sense by hand.³ Here, we associate word senses with FrameNet frames, and align frame roles between frames for one sense. We split the process into two steps: one is to assign a FrameNet frame to each sense, and the other is to map roles between FrameNet and PropBank frames, and between FrameNet and LCS frames.

3.1 Frame Alignment

FrameNet defines frames containing a frame name and a set of conceptual roles, and also defines corresponding predicate lexemes for each frame. Given a predicate, we search all senses in WordNet 2.0 and assign frames to a sense if there is enough evidence connecting the frame and the sense. The two steps of the process are:

Step I: Given a verb lexical unit from FrameNet, we search all WordNet senses and compute scores for each mapping between a sense and a frame. When the score is higher than a threshold⁴, we assign the frame to the sense.

³ Joint with PropBank project, we employed people to manually assign PropBank and LCS frames to each word sense.

⁴ All thresholds used in frame and role alignment are empirically determined by testing on 100 verbs.

Table 1. The features used for computing the degree of association between a frame and a synset

Input: synset c , frame f	
Output: scores of relation between c and f	
1	The frequency of the pair of (c, f) over all synsets associated with the frame f .
2	The frequency of the pair of (c, f) over all frames associated with the synset c .
3	The ratio of the lexemes associated to f in c to all lexemes in the synset c .
4	The ratio of the number of matching words between sense description and a FrameNet lexunit definition to the number of words in a FrameNet lexunit definition.
5	The number of matching words between a sense description and a FrameNet lexunit definition.
6	The number of matching stem words between a FrameNet frame definition and a sense description.
7	The frequency of the pair of (<i>immediate hypernym of c, f</i>) over all senses whose immediate hyponyms are associated with the frame f .
8	The frequency of the pair of (<i>immediate hypernym of c, f</i>) over all frames associated with the immediate hypernym of c .
9	The number of matching role names between FrameNet and LCS over the number of FrameNet frame element roles.
10	The number of matching role names between FrameNet and LCS over the number of LCS theta roles.
11	The number of matching words between a FrameNet frame name and a PropBank structure description.
12	The frequency of the pair of (<i>Lexicographer file name of c, f</i>) over the Lexicographer file names of all senses associated with the frame f . Verb synsets in WordNet are organized into 15 lexicographer files based on syntactic category and logical groupings such as <i>verb.body</i> , <i>verb.change</i> , and <i>verb.cognition</i> .
13	The frequency of the pair of (<i>Lexicographer file name of c, f</i>) over all frames associated with the Lexicographer file name of c .

We build scores mainly from the definition of frames and word senses, and also from the distribution of frame occurrences over senses. We also utilize the WordNet relations between senses, and pre-assigned PropBank and LCS frames. Since all verbs in a synset share a definition and WordNet relations, the scores are computed between a frame and a synset. The detailed description is in Table 1.

Step II: After assigning initial frame-sense mappings, some lexemes have no senses connected to a given frame, even though the lexeme is associated with the frame in FrameNet. However, if a lexeme is associated with the frame, it means at least one of the senses of the given lexeme has to be connected to the frame. Based on this assumption, we assign the frame to the sense having the highest score if none of the senses has a frame after Step I.

We associated 2,679 senses with FrameNet frames from 8,269 senses corresponding to the lexemes defined in FrameNet.

3.2 Role Alignment

After assigning frames to each word sense, we align roles among all FrameNet, Propbank, and LCS frames associated with a given sense. We find the mapping between FrameNet and LCS frames, and between FrameNet and PropBank frames. Given a FrameNet frame role⁵, we map a proper role from corresponding LCS and PropBank frames, since the FrameNet frame has a set of possible roles rather than an exhaustive set of roles for a sentence. For example, the FrameNet frame “Performer and roles” has the core roles: *Audience*, *Medium*, *Performance*, *role*, *Performer*, *Performer1*, *Performer2*, *score*, *script*, and *type*, while PropBank defines two roles, *Arg0(Actor)* and *Arg1(role)*, in the “act.01” and “play.02” structure. When there are two performers in a sentence, they are called *Performer1* and *Performer2* respectively in FrameNet, but all match *Arg0 (Actor)* in PropBank. In other words, different roles from a FrameNet frame can have the same corresponding roles from a PropBank or LCS frame.

We analyze FrameNet definitions and compare the example sentence patterns with the patterns of LCS frames and PropBank argument structures.

FrameNet vs. LCS: FrameNet uses 392 distinct role names for frames of verb predicate, specific to a frame. To map these roles to the more general theta roles in LCS, we use hierarchical frame information in FrameNet as a feature.⁶ We traverse up the role inheritance relations, and check if a role name matches the LCS role name. For example, *Deformer* in the “Reshaping” frame is inherited from *Agent* in the “Damaging” frame, so we assign points to the mapping from *Deformer* to *Agent* in the LCS frame. However, only 18% of the frame roles (including ancestor’s name) use the same name as in LCS.

To cover many frame roles, we use example sentences from the FrameNet frame definition. Figure 1 explains the mapping procedure with an example. From a sentence annotated with semantic roles, we delete parts not annotated with core roles, and substitute role names for real constituents. When a frame element starts with a preposition, we leave a preposition in front of the role name. With these simplified patterns, we compare LCS theta roles expressed in the same way. We count the frequency of matching roles between FrameNet and LCS.

With these features, we compute the sum of scores for a pair of a FrameNet role and a LCS role, and finally align the roles if the score is greater than a threshold.

FrameNet vs. PropBank: As in FrameNet vs. LCS, we check role names and role occurrence patterns in a sentence. PropBank uses numbered argument

⁵ FrameNet frames have three types of roles: core, peripheral, and extra-thematic. For the role alignment, we only consider core roles that are required in a given frame.

⁶ FrameNet defines inter-frame relations (Subframe, Inheritance, SeeAlso, Causative of and inchoative of, and Using) including super-sub relations between frame elements within related frames.

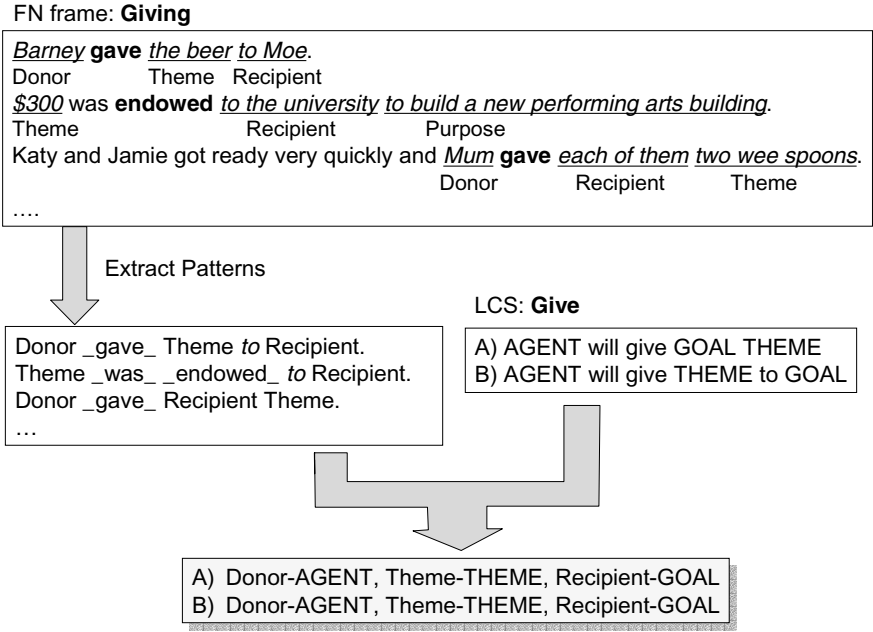


Fig. 1. Role mapping between FrameNet and LCS using annotated sentences

Table 2. The generalized patterns of role description of PropBank

-er
-ed
thing entity [being] -ed
thing entity -ing
-ed thing entity
-ing thing entity
-ing from to into...(preposition)
-ed from to into...(preposition)

names from *Arg0* to *Arg5*, and it includes short descriptions for the roles. We check the role description if it contains the FrameNet role name. In addition, we extract and compare role occurrence patterns in a sentence using example sentences from FrameNet and PropBank. The process is same as in FrameNet vs. LCS.

Since we already have the mappings between FrameNet frames and LCS frames, we use these mappings. The PropBank definition partially covers the mapping to VerbNet [11] verb classes, and 27% of total roles have corresponding VerbNet theta roles. Since VerbNet semantic structure is also based on Levin’s verb classes [12], the role definition is similar to roles in LCS. For arguments having no corresponding theta roles in PropBank, we generalize an argument

Table 3. The number of aligned roles

Automatically Aligned	FrameNet vs.	
	LCS	PropBank
Senses having frame mappings	1,724	1,714
Senses having role mappings	1,412	1,350
Sense-Frame pairs	3,897	1,601
Role pairs	9,635	4,514

description to the patterns in Table 2. For instance, *Arg0* in “abandon.03” is described as “entity abandoning something” and this is generalized to “thing|entity-ing”. We compute the probability of theta roles given one of these patterns.

From this probability given a generalized role description and from the example sentences, we compute the score of the relationship between the FrameNet role name and PropBank argument, and apply a threshold again to decide. Table 3 shows the number of aligned roles after this process. There are 1,724 senses having FrameNet and LCS frame. 1,412 of them have role mapping information, and there are 9,635 role pairs between frames.

4 Evaluation

Evaluation is performed by checking the system output manually. Not only to evaluate the result but also to use the integrated data for a future resource, we checked and corrected all the mapping output. Given the system output, one annotator first checked the frame alignment for each WordNet sense, and another checked the role mapping between FrameNet and LCS and between FrameNet and PropBank. In order to focus on the differentiation between frames given a verb, our alignment interface was designed to link a frame and a sense given a verb, and to align roles between aligned frames. We hired two annotators for parallel checking. They checked 2,300 verbs defined in FrameNet.

4.1 Agreement Between Annotators

In FrameNet, a predicate is often connected to multiple frames corresponding to its context, but it does not exactly match the WordNet sense, which results in disagreements between annotators.

Table 4 shows the agreement between two annotators for frame alignment and role alignment respectively. Agreement on frame alignment means if they provide the same answer (yes or no) for the association between the given frame and sense. For the frames that both annotators associate with a given sense, we check whether they also agree on each role mapping.

We compute the Kappa value for each frame alignment and role alignment. The Kappa statistic [16] is an agreement measure between judges’ assessment considering chance agreement. An assessment with $K \geq 0.8$ is generally viewed as reliable, and $0.67 < K < 0.8$ allows tentative conclusions. In our experiment, the Kappa value is 0.71 for frame alignment and 0.75 for role alignment.

Table 4. Agreement between annotators

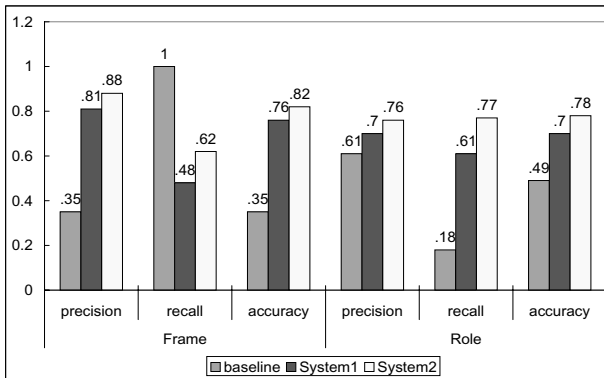
Annotation	Frame	Role
Agreement	0.85	0.82
Kappa	0.71	0.75

Based on the assumption that all verbs in a synset share the same frame information, we checked for inconsistencies within a synset and fixed them manually. Finally we asked a third person to check the mismatch between annotators, and we used the answers where two of the three agreed.

4.2 Automatic Alignment Accuracy

Figure 2 shows the performance of the automatic process when we assume the hand-corrected data is correct. *Precision* is a measure representing how many senses are actually associated with the automatically assigned frames, *Recall* measures how many sense-frame relations are detected by the process, and *Accuracy* is a measure how many of the considered sense-frame pairs are correctly answered, including saying “no relation”. The system shows 76% accuracy for frame alignment and 70% accuracy for role mapping. The precision is pretty high but the recall is not high enough because we don’t have enough information for some frames. Especially when the frame or verb is not common, we do not find adequate generalized information, or when the role definition is short and includes no words that provide clues. In other words, when there is enough evidence, we can get good matches, but if not, we cannot say anything.

The baseline is to assign all frames to all senses of the given verb when the verb is associated with the frame in FrameNet. Since we are considering only verbs defined in FrameNet, the baseline recall is 1. As a role mapping baseline, we match a role with a role having the same name or description and match *Agent*

**Fig. 2.** Automatic process evaluation

and *Arg0*. High accuracy results in high-speed hand-correction since annotators do not need to change a lot. To help annotators, we update the output of the automatic process (input of hand-checking process) using already corrected results. After annotating 1,000 senses, we update the remaining 7,295 senses when their synonyms already have frames that both annotators have agreed on. In Figure 2, System1 represents the original automatic process and System2 represents the modified system. It shows much progress for System2 (76% to 82% in frame alignment, and 70% to 78% in role mapping) and it implies more improvement as we add more resources to modify remaining annotations. As we obtain more aligned data, we will speed up the next manual annotation.

4.3 Results

By the integrating process, 4,240 senses are associated with FrameNet frames. In Table 5, we show the total number of senses associated with frames. 1,757 senses

Table 5. WordNet senses associated with frames

Associated Frames	Number of Senses
FrameNet	559
FrameNet & LCS	674
FrameNet & PropBank	1,250
FrameNet & PropBank & LCS	1,757

“assemble” in WordNet

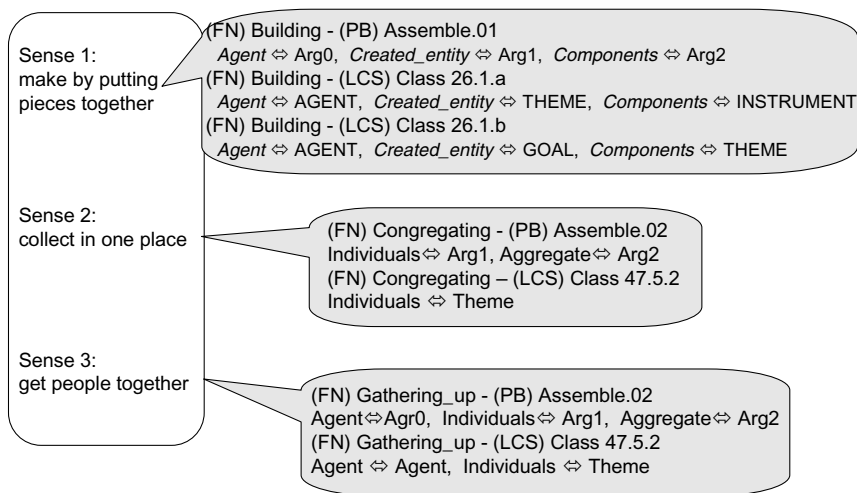


Fig. 3. The system output with an example. Given a verb “assemble”, our systems finds FrameNet frames (Gathering_up, Congregating, or Building) corresponding to each sense, and aligns roles between frames.

are associated frames from FrameNet, PropBank, and LCS, and 559 senses have only FrameNet frames. Figure 3 shows the system output with an example.

5 Expanding Frame Alignments

The verbs defined in FrameNet have been associated with the corresponding frames for each sense, but many senses still have no frame alignment. We attempt to expand the aligned frame information to other verb senses (not defined in FrameNet) using relations in WordNet.

LCS defines verb classes sharing semantics although it does not always match semantics (frames) in FrameNet [2]. We check the WordNet sense hierarchy within a verb class, and induce the FrameNet frame for the verbs defined in LCS. First, we check the immediate hyponyms of the word sense, and if most (90%) of the senses associated with a frame are connected to the same frame, and then we assign the frame to the sense (bottom-up approach). Second, given a sense, we find the frames associated with the immediate hypernym of the sense, and if the hyponyms of the hypernym do not have much conflict (if there is less than 10% hyponyms mapped to other frames), we assign the frames to the sense (top-down approach).

We randomly chose 170 senses from the output of bottom-up and top-down approach, and computed the precision to see how many frames were correctly assigned. Table 6 shows the precision for each step.

When verbs are not in FrameNet, it probably means that the appropriate FrameNet frames are not defined yet. However, we see the output provides an approximate for the real (undefined) frames with reasonable precision. Since we propagate the frames traversing the relations between senses within LCS verb classes, our system does not assign totally different frames but often mistakes fine-grained frame differences. For example, our system confuses *Attempt_suasion*, *Suasion*, and *Talking_into*. All three frames are about the situation that “the speaker expresses through the language his wish to get the addressee to act”. *Attempt_suasion* “does not imply that the addressee forms an intention to act, and let alone acts”, *Suasion* is for the situation “as a result, the addressee forms an intention to do so, but no implication that the addressee actually acts upon the intention”, and *Talking_into* implies “the addressee forms an intention to act and does so”. Even though the detailed meanings differ,

Table 6. The result of expanding alignment. 552 senses are newly associated with frames from 3,207 of senses defined in LCS (not in FrameNet).

Step	Frame-assigned Senses	Precision
I. Bottom-up	66	73%
II. top-down	486	67%
Overall Output	552	63%

they all have semantic roles “Speaker”, “Addressee”, and “Content”, and this disagreement may be tolerable for some applications.

6 Conclusion

We have described the process of aligning frames to each WordNet sense. For each verb sense, we associated FrameNet frame and aligned frame slots (semantic roles) between FrameNet and PropBank and between FrameNet and LCS. We aligned about 4,200 senses in FrameNet frame, and expanded the result to around 550 senses not defined in FrameNet and defined in LCS.

The alignment is performed for each word sense, and we could find the consistencies within a FrameNet frame. PropBank argument structure is defined per predicate and the same argument number does not represent the same role. The role mapping between FrameNet and PropBank within a FrameNet frame was consistent by 93%. For example, given a frame “Adorning”, *Location* maps to *Arg2* and *Theme* maps to *Arg1* in the PropBank structure “encircle.01”, but *Location* maps to *Arg1* and *Theme* maps to *Arg2* in “festoon.01”.

As a practical resource for the application, this resource can be used in diverse ways. By integrating and expanding the frame information, semantic role labeling can utilize this data directly. In addition, the frames assigned to each word sense can be understood as coarse-grained sense disambiguation information and it can be applied to computing similarity measures between senses in WordNet. We leave the application using this resource for the future work.

The method we showed in this paper is not restricted to frame alignment, and could be applied to other applications investigating similarities and differences between heterogeneous resources. We are encouraged by the good agreement levels exhibited by our algorithm and plan to explore theoretical verifications further.

References

1. Baker, C.F., Fillmore, C.J., Lowe, J.B.: The Berkeley FrameNet project. Proceedings of COLING-ACL, Montreal, Canada (1998)
2. Baker, C.F., Ruppenhofer, J.: FrameNet’s Frames vs. Levin’s Verb Classes. Proceedings of 28th Annual Meeting of the Berkeley Linguistics Society, (2002) 27–38
3. Dorr, B.J.: Large-Scale Dictionary Construction for Foreign Language Tutoring and Interlingual Translation. Machine Translation 12(4). (1997) 271–322
4. Fellbaum, C.: Wordnet: An Electronic Lexical Database. The MIT press (1998)
5. Fillmore, C.J.: Frame Semantics and the Nature of Language. Annals of the New York Academy of Science Conference on the Origin and Development of Language and Speech, 280 (1976) 20–32
6. Fleischman, M.B., Kwon, N., Hovy, E.: Maximum Entropy Models for FrameNet Classification. Proceedings of EMNLP-03, Sapporo, Japan. (2003)
7. Gildea D., Jurafsky D.: Automatic Labeling of Semantic Roles. Computational Linguistics, 28(3) (2002) 245–288

8. Green, R., Dorr, B.J., Resnik, P.: Inducing Frame Semantic Verb Classes from WordNet and LDOCE. Proceedings of ACL-04, Barcelona, Spain. (2004)
9. Green, R., Pearl, L., Dorr, B.J., Resnik, P.: Mapping Lexical Entries in a Verbs Database to WordNet Senses. Proceedings of ACL-01, Toulouse, France. (2001)
10. Kingsbury, P., Palmer, M.: Adding Semantic Annotation to the Penn Treebank. Proceedings of HLT-2002, San Diego, CA. (2002)
11. Kipper, K., Dang, H.T., Palmer, M.: Class-Based Construction of a Verb Lexicon. Proceedings of AAAI-2000, Austin, TX, USA. (2000)
12. Levin, B.: English Verb Classes and Alternations: A Preliminary Investigation. University of Chicago press (1993)
13. Pradhan, S., Ward, W., Hacioglu, K., Martin, J.H., Jurafsky, D.: Shallow Semantic Parsing Using Support Vector Machines. Proceedings of HLT/NAACL-04, Boston, MA. (2004)
14. Shi, L., Mihalcea, R.: An Algorithm for Open Text Semantic Parsing. Proceedings of the workshop on Robust Methods in Analysis of Natural Language Data, Geneva, Switzerland. (2004)
15. Shi, L., Mihalcea, R.: Putting Pieces Together: Combining FrameNet, VerbNet and WordNet for Robust Semantic Parsing. Proceedings of the 6th international Conference on Intelligent Text Processing and Computational Linguistics, Mexico city, Mexico. (2005)
16. Siegel, S., Castellan Jr., N.J.: Nonparametric Statistics for the Behavioural Sciences. McGraw-Hill (1988)
17. Surdeanu, M., Harabagiu, S., Williams, J., Aarseth, P.: Using Predicate-Argument Structure for Information Extraction. Proceedings of ACL-03, Sapporo, Japan. (2003)
18. Swier, R.S., Stevenson, S.: Unsupervised Semantic Role Labelling. Proceedings of EMNLP-04, Barcelona, Spain. (2004)

Making Senses: Bootstrapping Sense-Tagged Lists of Semantically-Related Words

Nancy Ide

Department of Computer Science, Vassar College, Poughkeepsie, New York, USA
ide@cs.vassar.edu

Abstract. The work described in this paper was originally motivated by the need to map verbs associated with FrameNet 2.0 frames to appropriate WordNet 2.0 senses. As the work evolved, it became apparent that the developed method was applicable for a number of other tasks, including assignment of WordNet senses to word lists used in attitude and opinion analysis, and collapsing WordNet senses into coarser-grained groupings. We describe the method for mapping FrameNet lexical units to WordNet senses and demonstrate its applicability to these additional tasks. We conclude with a general discussion of the viability of using this method with automatically sense-tagged data.

1 Introduction

Lists of semantically-related words and phrases are heavily used in many automatic language processing tasks. A common use of such lists in recent work is in attitude or opinion analysis, where words indicative of a given semantic orientation—often, “positive” or negative” polarity—are detected to classify documents such as movie and product reviews as more or less favorable ([1], [2], [3]). Approaches include simple term counting [4] as well as training machine learning algorithms to classify documents. In machine learning approaches, semantically-related words and phrases are often used as a part of the feature set (e.g., [2], [3], [5]). NLP tasks such as event recognition also typically rely on lists of semantically-related verbs coupled with frames or patterns that are used to identify participants, etc. (e.g., [6] [7]).

Largely due to the recent upsurge in work on attitude and opinion analysis, numerous lists of semantically-related words have been made available within the language processing community. The lists are compiled using a variety of means, including extraction from existing resources such as lexicons, thesauri, and pre-compiled content category lists such as the General Inquirer [8]; automated extraction [2] [3]; and manual production; and often include hundreds or even thousands of words.

Whatever the source, available lists of semantically-related words do not identify the sense of the included items, despite the fact that many of the words are highly polysemous.¹ As a result, work relying on such lists identifies word occurrences that may not represent the phenomenon in question. Sense-tagged lists of words could

¹ The General Inquirer includes sense tags using a sense inventory developed by the project; however, only words appearing in more than one sense in the same list are tagged.

significantly increase the accuracy of pattern-recognition and learning algorithms, if the data is also sense-tagged. For the moment, we put aside the issue of (accurately) sense-tagging corpora, and return to it in Section 8.

The work described in this paper was originally motivated by the need to map verbs associated with FrameNet 2.0 frames to appropriate WordNet 2.0 senses. As the work evolved, it became apparent that the developed method was applicable for a number of other tasks, including assignment of WordNet senses to word lists used in attitude and opinion analysis and collapsing WordNet senses into coarser-grained groupings. In the sections that follow, we describe our method and demonstrate its applicability to these additional tasks. We conclude with a general discussion of the viability of using our sense-tagged lists with automatically sense-disambiguated data.

2 Background

The work reported here was undertaken in the context of the FDR/Pearl Harbor Project², which is enhancing a range of image, sound, video and textual data drawn from the *Franklin D. Roosevelt Library and Digital Archives*. The project is undertaking the encoding, annotation, and multi-modal linkage of a portion of the collection, and development of a web-based interface that enables exploitation of sophisticated data mining techniques. The project focuses on a collection of 1,446 internal administration documents concerned with US-Japanese relations between 1931 and 1941, including memoranda of conversations, letters, diplomatic correspondence, intelligence reports, and economic reports. The corpus has been annotated for a wide range of entities and linguistic phenomena, and all words have been automatically tagged³ with WordNet2.0 senses. To support retrieval, an ontology including persons, locations, roles, organizations, and events and other entities specific to our data (ships, treaties, etc.) has been created, by extending and refining SUMO and MILO categories such as *government and military organizations* and *people related to organizations*. All annotation and ontology development in the project has been accomplished using the GATE (General Architecture for Text Engineering) system [9] developed at the University of Sheffield.

Historical research on Japanese-American relations in the ten years prior to the bombing of Pearl Harbor focuses on the nature of the relationship between representatives of the two countries. In particular, historians and political scientists are interested in the interplay of the dialogue between the two countries and how it conveys attitudes such as power and control vs. submission, hostility vs. friendliness and openness, cooperation vs. non-cooperation, etc., not only at a given time, but as these attitudes varied during interactions over the ten-year pre-war period. The FDR Project is therefore concerned with identifying evidence of such attitudes in the wording of documents in the corpus, and attributing this information to the appropriate person or entity. Because a large portion of the documents in the collection consists of so-called “memoranda of conversations”, many consist of near-transcriptions of meetings

² Supported by U.S. National Science Foundation grant ITR-0218997.

³ WordNet::SenseRelate (all words) [11] was used to provide WordNet sense annotations.

between Japanese and US officials.⁴ We have therefore focused on identifying *communication events*, down to the level of the utterance (e.g., “X asked that...”) and apply attitude-recognition procedures to each utterance attributed to a given speaker. Historians may thus request a synopsis of, for example, the attitudes conveyed by a Japanese official in conversations with the US Secretary of State over the ten-year period, and consider their development and change.

3 The Word List Problem

Annotation of the FDR document collection as described in Section 2 required automatic identification of semantically-related words signifying events and attitudes, followed by the application of pattern-recognition rules to extract contextual information (including role-fillers in the case of event recognition, polarity influencers [10] for attitude analysis, etc.). To detect lexical items indicative of a given attitude, we need to compile lists of words, in particular for specific affective categories such as “hostility”, “cooperation”, “power/control”, etc. For events, we require lists of verbs associated with a given event type, in particular, different categories of communication verbs (e.g., questioning, persuasion, reporting, etc.).

Rather than starting from scratch, we gathered information from available resources such as the General Inquirer, FrameNet[12], VerbNet [13], WordNet, and Levin’s verb list [14], and various additional lists compiled by individual researchers⁵, with the goal of merging as much of the information provided in these resources as possible. Existing resources from which word lists can be extracted come in several forms:

1. Flat word lists including no additional information. Some sources provide lists for relatively specific categories, such as “hostility”, “military”, etc, as, for example, one finds in the General Inquirer; others—especially lists that are becoming increasingly available within the NLP community—provide lists of words deemed to denote a positive or negative attitude. Typically, words in such lists are unlemmatized and may contain several inflectional variants of the same lexical item.
2. Word lists including a measure of relevance/relatedness, such as lists of positive/negative words that provide an associated measure of degree, or lists providing measures of semantic similarity (e.g., [15]).
3. Computational lexicons such as WordNet, FrameNet, and VerbNet. Depending on their intended use, computational lexicons contain additional syntactic and/or semantic information, such as definitions and examples and verb argument structure, and therefore, extracting a simple list of related words typically demands some degree of processing. Both WordNet and FrameNet additionally support their lexicons with hierarchical ontologies that provide several levels of increasingly general semantic classes associated with each word.

⁴ Note that the memoranda represent a complex communication event, in which, for example, Secretary Welles reports to FDR what the Japanese Ambassador said and how Secretary Hull replied. We make no judgment concerning the degree to which reports of, say, the Japanese Ambassador’s wording may have been colored by the reporter; our job is to simply provide the information to the historian and allow him to draw his or her own conclusions.

⁵ Our thanks go to Diana Inkpen, David Nadeau, and Maite Taboada for providing their lists of positive and negative words, and to Janyce Wiebe for her lists of subjective elements.

Merging information from these various resources is not a trivial matter, especially when the semantic categories involved go beyond broad categories such as “positive” and “negative”. First, semantic categories in different resources are determined using different criteria, and as a result, a straightforward mapping of categories is often impossible. This is particularly true for FrameNet and WordNet, whose categories and ontologies are, for practical purposes, virtually disjoint; furthermore, FrameNet’s ontology is shallow, often including only two or three levels, whereas WordNet’s is substantially deeper. VerbNet assigns FrameNet frames and WordNet senses to some of its entries, thus making a merge simpler; and the association of a given lexical item with a WordNet sense enables a mapping of words assigned to a given FrameNet frame to a WordNet sense. However, VerbNet’s coverage is relatively scant, especially for FrameNet categories, and it therefore provides very little information to link the three resources.

Extracting information from available resources can be a time-consuming task, and information from the different sources is various and occasionally conflicting. Furthermore, the results do not provide comprehensive lexical coverage, and they are in some cases inaccurate. Therefore, rather than attempting to merge existing resources, we developed a method to generate accurate, extensive sense-tagged word lists for lexical units associated with FrameNet frames, and generalized the method to produce sense-tagged lists for additional semantic categories.

3.1 Bootstrapping from FrameNet

FrameNet is creating a database of semantic frames, in which case roles dictated by the semantics of the lexical units (LUs) associated with the frame are specified. FrameNet provides a shallow inheritance hierarchy of frames corresponding to semantic categories; for example, the frame *complaining* inherits from *statement*, which inherits from the general category *communication*.⁶ Each frame is associated with a set of “frame-evoking” LUs consisting of word lemmas. Different senses of a lexical unit are defined on the basis of its association with different frames.

We use the set of LUs associated with each frame as defined by FrameNet as a starting point to develop more comprehensive word lists representing semantic categories. To be maximally useful for our application in the FDR project, this demanded several enhancements:

1. Extension of the lists of lexical units to provide more comprehensive coverage of words representative of a given category. As FrameNet is in the process of development, the number of lexical units associated with a given frame varies considerably, and coverage is incomplete.
2. Sense-tagging lexical units in order to eliminate “false hits” in our analysis. Many of the lexical items associated with the various FrameNet frames are highly

⁶ FrameNet also specifies a “using” relation among frames in cases where a particular frame makes reference in a general way to the structure of a more abstract frame; for example, the *judgment_communication* frame uses the *judgment* and *statement* frames, although it does not directly inherit from them. For the purpose of constructing a general hierarchy of semantic categories, we treat the “using” relation as inheritance. For a fuller explanation of FrameNet’s architecture and rationale, see [12].

polysemous, and identifying un-sense-tagged occurrences leads to considerable noise in our analysis. Because our corpus has been annotated with WordNet senses, it is desirable to associate each lexical unit in a given frame with a WordNet sense or senses.

3. Refinement of the FrameNet categories. FrameNet associates lexical units with a given frame on the basis of frame semantics, which often leads to word lists containing, for example, words with both positive and negative connotations that correspond to the same general semantic category. For example, the lexical units for the frame *judgment_communication* include not only “acclaim”, “commend”, and “praise” but also “condemn” and “denounce”. In addition, the possibility for more subtle semantic distinctions is apparent in many of the lists; in the same *judgment_communication* frame, we can isolate distinguishable semantic groupings of lexical items, such as “deride”, “mock”, and “ridicule”; “belittle”, “disparage”, “denigrate”; “blast” and “slam”; etc.

We developed a procedure to address issues (1) and (2) and applied a clustering algorithm to the results in order to accomplish (3), as described in the following sections.

3.2 Mapping FrameNet Lexical Units to WordNet Senses

Attempts to automatically or semi-automatically derive lists of semantically-related words and phrases has a long history in NLP, starting with a series of projects during the 1990’s (e.g., [15] [16] [17]) using similarity of distributional patterns in large corpora and clustering techniques. However, the use of distributional patterns in corpora has one major drawback: words may follow similar patterns not only because of semantic similarities, but also syntactic or pragmatic ones. As a result, many of the lists compiled using this strategy contain words that are not necessarily related semantically; for example, “achieve”, “frighten”, “invite”, and “penalize” are among the top-rated words in Lin’s publicly-available similarity list for the word “encourage”. For our purposes, where *semantic* similarity is the focus, corpus evidence is therefore not an ideal source.

WordNet::Similarity. WordNet::Similarity(WNS) [18] is a freely available package that includes six measures of similarity and three measures of relatedness that use information in WordNet, including links and path lengths for the various WordNet relations (synonymy, hyperonymy, etc.) and overlap among glosses and examples, shortest WordNet path length, information content, depth in the WordNet is-a hierarchy, and semantic density, to determine the degree to which two words are alike. The various measures are described and compared in [18]. Given a pair of words, WordNet::Similarity returns their most similar WordNet senses together with a numeric value reflecting their degree of similarity. The measure of similarity can also be determined for a pair of WordNet sense-tagged words, one sense-tagged word and one untagged word, etc., or for all senses of the input pair.

We use WNS to determine the “most similar” WordNet senses for each of the LUs associated with a particular FrameNet frame. To do this, we create a set of pairs $P_F = LU_F \times LU_F$, where LU_F is the set of lexical units associated with FrameNet frame, and feed P_F to WNS. The result set R_F includes the most similar senses for each pair of words and a measure of their similarity. The hypothesis is that since the words in the

pair sets have been associated with a specific FrameNet frame, they should be mutually disambiguating, and the most appropriate WordNet sense for the frame can be determined.

Preliminary experimentation with all nine of the similarity and relatedness measures provided in WNS confirmed that the *lesk* measure provided the most accurate results, which corresponds to the determination based on similar experiments reported in [REF]. The *lesk* measure [19] assigns a relatedness score by scoring overlaps between glosses of two senses and senses of other words that are directly linked to them in WordNet, according to a user-chosen set of relation “pairs” that specify which WordNet relations determine the score (for example, overlaps between synsets of the two words, overlaps in the gloss of the first word and example text of the second, etc.), any of which may be optionally weighted. WNS provides a default relation set for use with the *lesk* measure that determines the relatedness score based on overlaps among all possible relation pairs, a total of 88 in all.

We devised a reduced relation set that includes the following relation pairs: example - example, gloss - gloss, hypernym - hypernym, hypernym - hyponym, hyponym - hypernym, hyponym - hyponym, synset - example, synset - gloss, and synset - synset. Greatest weight (0.7) was given to synset overlaps, and additional weight (0.5) was given to overlaps in example texts, glosses, and synset overlaps with examples and glosses. The rationale for this choice was to focus on synonymy (same concept) and *is-a* relations (more/less general expression of the same concept). We also determined that gloss and example overlaps, as well as synset overlaps with glosses and overlaps, are highly reliable indicators of relatedness, often capturing commonalities that are not otherwise direct or explicit (e.g., the synset for *urge#v#3* includes “inspire”, which appears in the gloss for *encourage#v#2*, “inspire with confidence”).

Computing Sense Lists. We determine sense-tagged lists for LUs associated with FrameNet categories using WNS’s *lesk* procedure. A *sure sense* ss_{w_i} is identified for each word $w_i \in LU_F$ when any one of the following holds:

1. w_i has more than one sense and $freq(s_{w_i}) = 1$
2. w_i has only one sense and $simscore(s_{w_i}) > .2$
3. $freq(s_{w_i}) > T_{freq}$, and $tsim(s_{w_i}) > T_{sim}$

where T_{freq} and T_{sim} are user-defined threshold values in range 0-1 for the frequency and total similarity values, respectively.

The frequency score $freq(s_{w_i})$ is defined as

$$freq(s_{w_i}) = \frac{\sum_{s_{w_i} \in R_F} s_{w_i}}{pc_F} . \quad (1)$$

where pc_F is the number of pairs (w_b, w_j) in P_F for some w_i —i.e., $size(LU_F) - 1$. When $freq(s_{w_i}) = 1$, s_{w_i} has been returned as the most similar sense for every pair $(w_b, w_j) \in P_F$.

$Tsim(s_{w_i})$ is the sum of *lesk score* (*ls*) values returned by WNS reflecting the degree of relationship between sense s_{w_i} and all other senses s_{w_j} :

$$Tsim(s_{w_i}) = \sum_{\substack{j=1 \\ j \neq i}}^{pc_F} ls(s_{w_i}, s_{w_j}) . \quad (2)$$

$simscore(s_{w_i})$ is defined as:

$$simscore(s_{w_i}) = \frac{e^{simscore(s_{w_i})} - e^{-simscore(s_{w_i})}}{e^{simscore(s_{w_i})} + e^{-simscore(s_{w_i})}} . \quad (3)$$

This scales $simscore$ to a value between 0 and 1, and eliminates the impact of the size of the LU sets.

Condition 2 above handles the rare instance when the appropriate *sense* of an LU does not appear in WordNet; e.g., “grill”, which is an LU in the category *questioning*, appears in a single sense in WordNet (“cook over a grill”). However, because $simscore(\text{grill}\#\text{v}\#1)$ is only .04, it does not exceed the threshold, and therefore this sense is not added to the set.

We use the following algorithm to create a list of sursesens for LUs associated with a FrameNet category:

ALGORITHM A:

1. Compute SS_{LU_F} , the set of sursesens ss_{w_i} for lexical units in LU_F ⁷, using Algorithm A
2. Generate a new set of pairs P' from $SS_{LU_F} \times LU_F$
3. Compute $SS_{P'}$.

Note that some LUs in LU_F may not be assigned a sursesense. At the same time, more than one sense for a given word may qualify as a sursesense. Step 1 identifies highly-related senses from the original un-tagged list of LUs; since some words are not assigned a sense at this point, in Step 2 relatedness is computed using the set of *sense-tagged* words identified in Step 1 coupled with every un-tagged word in the original set. This strategy both provides better information for computing relatedness for the as-yet unassigned words, and may identify additional senses for words that were tagged in Step 1.

Manual evaluation determined that sursesense sets compiled using this method are highly accurate, but that in a few cases, “noise” was introduced into the set in the form of inappropriate sense assignments. This occurred in situations where, for example, two or more words in an LU share a second meaning, which was then introduced into the sursesense set. For example, the LUs for the *reasoning* frame include “demonstrate” and “show”, which share not only the appropriate sense of proving or establishing something, but also the sense of exhibiting to an audience. Therefore, to maximize the accuracy of our results, we modified the algorithm to include additional information derived from other sources, including WordNet::SenseRelate::WordToSet⁸ (SR) and a “master” sense list extracted from VerbNet (VN) and FnWnVerbMap 1.0⁹ (VM) [20].

⁷ In the course of computing the similarity measures, LUs that do not appear in WordNet are eliminated.

⁸ <http://www.d.umn.edu/~tpederse/senserelate.html>

⁹ Available at <http://lit.csci.unt.edu/~rada/downloads/FnWnVerbMap/FnWnVerbMap1.0>⁹ (VM) [20].

SR determines the relatedness of a given word w to a set of words. The results list all WordNet senses of w together with a relatedness score, sorted from highest to lowest score. We fed SR lists of word-set pairs for each LU_F consisting of (1) each $w_i \in LU_F$, coupled with (2) the set of all $w_j \in LU_F, i \neq j$. SR uses WNS to compute the relatedness scores and provides the same choice of similarity measures; we have used the similarity measure and relation set as described above for WNS. We derive two additional “suresense” sets from SR’s output:

1. SR_{top} , the sense of each LU determined to be most similar to the remaining words in a given LU_F ; and
2. SR_{cutoff} , the senses of each LU with a relatedness score above a pre-determined cutoff value.

Note that because SR computes a single, overall score for each sense based on its relatedness to all other LUs in a given frame, the results from WNS described above and results from SR provide somewhat different results; the correlation of results computed using WNS above and each of the two sets computed from SR is .8. We can characterize results in SR_{top} as highly precise but with low recall; whereas SR_{cutoff} and the SS sets computed using WNS have slightly lower precision but better recall.

To address this problem, we created another suresense set by combining the WordNet senses assigned to words in a given LU_F that is also tagged in VN and/or VM into a single set V . VN includes slightly over 4000 words, each of which is manually assigned a WordNet sense or senses; FrameNet frames are assigned to only a fraction of entries. VM provides a semi-automatically-assigned WordNet sense or senses for every verb lexical unit in FrameNet 2.0. Originally, we hoped to use VM as a gold standard against which to evaluate our results, but we discovered that the assigned senses in VM are often incomplete; that is, many senses that are viable alternatives are not included. Also, the identified senses are occasionally incorrect. More importantly, comparison of WordNet sense assignments for words that are tagged in both VN and VM show an agreement level of only 27%, which is no doubt a results of the well known problem with WordNet sense assignments, wherein distinctions are generally regarded as too fine-grained for most NLP applications and problematic for humans to distinguish. Collapsing WordNet senses to produce a sense list more appropriate for NLP applications has been proposed ([21]; see also [22]); in fact, because our method identifies multiple senses for each LU, it potentially identifies at least some senses of a given LU that can be collapsed.

Using information extracted from the various resources, the final set of senses for a FrameNet frame F is determined as follows:

ALGORITHM B:

1. Compute SS_{LU_F} , the set of suresenses ss_{w_i} for lexical units in LU_F
2. Set $SS'_{LU_F} = SS_{LU_F} \cap SR_{topF} \cap SR_{cutoffF} \cap V_F$
3. Generate a new set of pairs P' from $SS'_{LU_F} \times LU_F$
4. Compute $SS'_{P'}$

3.3 Evaluation

To test the accuracy of our method, we computed sense-tagged lists of LUs for 28 FrameNet2.0 categories that are classified as sub-types of *communication*. The number of LUs in the frames ranges from two to 58; the average number of LUs per frame is 12. In this experiment, T_{freq} was set to .3 and T_{sim} was set to .9. The algorithm identified at least one sense for 95% of the LUs and assigned an average of 1.35 senses to each.

Manual evaluation of sense-tagging is a notoriously problematic task, and even among human annotators there is typically no more than 80% agreement on the WordNet sense to be assigned to a given word in context. Our task here is somewhat simplified, for several reasons:

1. Sense assignments are not evaluated for words in context, but rather in terms of the word's association with a FrameNet category and in relation to the set of LUs associated with that category.
2. Multiple senses can be assigned to a given LU; there is no attempt to identify a unique sense assignment.
3. The task consists of validating the assignments produced by the algorithm, rather than assigning a sense or senses to LUs and comparing the results to the automatically-produced set.

Two undergraduate Cognitive Science majors with a background in linguistics performed manual validation of the sense assignments produced by our algorithm. Both verified that 100% of the senses assigned by the algorithm were appropriate for the FrameNet category with which they are associated. We note that given our method of determining the sense assignments, it is possible that some appropriate senses are not included, especially additional senses for words for which at least one sense has been identified. We address this issue in section 7. However, for our purposes it is preferable to maximize precision at the expense of recall, since the resulting sursesense sets are used to augment the lists, as described in the following section.

4 Augmenting the Lists

The highly accurate sursesense sets produced using the algorithm described in the previous section provide the base from which to generate additional sense-tagged words in order to augment the FrameNet LU sets. To do this, we apply the following algorithm:

ALGORITHM C:

1. Add synsets for all $s_{w_i} \in SS_{P'_F}$ to $SS_{P'_F}$
2. Generate $HYPE_F$, the set of hypernyms for all $s_{w_i} \in SS_{P'_F}$
3. Generate a new set of pairs P_{HYPE} from $SS_{P'_F} \times HYPE_F$
4. Compute SS_{HYPE} from P_{HYPE}
5. Generate $HYP0_F$, the set of hyponyms for all $s_{w_i} \in SS_{P'_F}$
6. Generate a new set of pairs P_{HYP0} from $SS_{P'_F} \times HYP0_F$
7. Compute SS_{HYP0} from P_{HYP0}
8. Generate a new set $U_F = SS_{P'_F} \cup SS_{HYPE} \cup SS_{HYP0}$
9. Add synsets for all $s_{w_i} \in U_F$ to U_F

Hyponym and hypernym sets occasionally include words that are less related to the category than desirable. For example, the set of hyponyms for sense 3 of “permit”, which is included in the category *grant_permission*, includes sense 4 of “pay” (“bear a cost or penalty in recompense for some action”). Verifying the hypernym set against the previously-generated set of sursesens for the category eliminates this and other less related words, including “take_lying_down” and “stand_for”. Hypernym sets often include general concepts, such as sense 2 of move (“cause to move, both in a concrete and in an abstract sense”), which is in the hypernym set for the category *attempt_suasion*; verification against the sursesense set also eliminates very general senses, as they are typically related only weakly to those sursesens for which they are not the hypernym.

A modified version of algorithm A is used to verify hypernym and hyponym sets, in which frequency scores--which tend to be near or equal to 1 in every case--are ignored; in these cases, relatedness is determined solely on the basis of *simscore*.

Algorithm C could be repeated one or several times to further augment the lists, although we have not tested this option: iterative addition of hypernyms and hyponyms could introduce increasing noise, and accuracy of the sets may degrade.

Lists of un-sense-tagged words from other sources can also be run against the sursesense sets to augment the sursesense sets. For example, we have run the list of verbs appearing in the FDR corpus (with the exception of “do”, “be”, “have”, and modal verbs) against the sursesense sets for the FrameNet communication categories, in order to ensure full coverage of our lexicon. Here, because the vast majority of the words in the list are unrelated to the sursesense set, we increased the threshold for eliminating words with one sense given in Algorithm A, step 2, to .5.

Similarity lists for each set of LUs associated with a FrameNet communication categories were also extracted from Lin’s data and run against the sursesense sets in order to extract additional word senses appropriate for the categories. The results were judged to be about 90% accurate overall, somewhat less than the accuracy rate for the sursesense sets, presumably because the words in Lin’s lists had already been selected for similarity to the target word by using contextual information. The failures typically involve words that have no sense that is relatively synonymous to the target word or with opposite polarity (e.g., (e.g., “engage” and “frighten” in relation to “encourage”). We are currently experimenting with Lin’s lists in order to improve accuracy, before adding the results to the FrameNet sursesense sets.

Our sense-tagged lists of words for each of the FrameNet communication categories is available at <http://www.cs.vassar.edu/~ide/FnWnSenseLists>. Both the original sursesense lists, including only the FrameNet LUs, and the augmented lists including synsets, hypernyms, and hyponyms, are available on the website.

5 Refining Categories

The LUs associated with FrameNet frames often fall into semantic sub-categories that are not isolated in FrameNet. The similarity measures produced by WNS can be exploited to produce a similarity matrix, which in turn can be used to identify semantic sub-groups of LUs for a given frame via clustering.

We applied a clustering algorithm using *weighted arithmetic average*, which assigns a weight to the distance between samples S_1 (in A) and S_2 (in B) of $(1/2)^G$, where G is the sum of the nesting levels (number of enclosing clusters) of S_1 and S_2 , which reduces the influence of groups of similar samples on the clustering process.

Table 1 shows the clustering results for the *judgment_communication* suresenses, obtained by “pruning” the cluster tree at edges with a weighted distance $> .85$ according to the algorithm. Each column contains word senses (WordNet2.0 sense number appended) included in one of the pruned sub-clusters.¹⁰ The results identify intuitively sensible semantic groupings, and correctly isolate the positive and negative senses. Further pruning within a sub-cluster could yield even finer semantic distinctions; for example, the “acclaim” sub-cluster includes two sub-clusters: *acclaim1*, *extol1*, *laud1*, and *commend4*; and *commend1*, *praise1*, and *cite2*.

Table 1. Clustering results for *judgment_communication*

ACCLAIM	DENIGRATE	BELITTLE	CONDEMN	CHARGE	ACCUSE	RIDICULE
acclaim1 extol1 laud1 commend4 commend1 praise1 cite2	denigrate1 deprecate2 execrate2	belittle2 disparage1 reprehend1 censure1 denounce1 remonstrate3 blame2 castigate1	condemn1 decry1 excoriate1 deprecate1	accuse2 charge2 recriminate1	accuse1 denigrate2	deride1 ridicule1 gibe2 scoff1 mock1 scoff2 remonstrate2

6 Generating Word Lists for Attitude Analysis

The procedure outlined in sections 3 and 4 can be applied to generate sense-tagged word lists for use in tasks such as attitude analysis. Here, the user provides an initial list of “seed” words to replace the FrameNet lists of LUs. An obvious source of seed words is the categorized word lists in the General Inquirer; however, the GI lists are extensive, some including over 1000 words, and often the semantic range of items in a given category is quite broad. In addition, the lists contain words in various parts of speech as well inflectional variants, the latter of which are not usable to retrieve information from WordNet.

To test the viability of creating sense-tagged lists of words for attitude analysis, we created lists of seed words by intersecting lemmas from a 150,000 word sub-corpus of the FDR data with the GI word lists for the categories “hostile”, “power/cooperation”, “submit”, “weak”, and “strong”. A seed suresense list is created using Algorithm B, replacing *LU* with the list of seed words, and using only SS , SR_{top} , and SR_{cutoff} in step 2. In step 3, the seed suresense list is run against the remaining words in the original list. Note that in processing the FrameNet categories, only verb senses were considered for inclusion, whereas here, senses of a given word as a noun, verb, adjective, or adverb are considered if they exist. Following the application of Algorithm B, the resulting suresense sets were split into subsets according to part of speech, and each subset was individually augmented by applying Algorithm C.

¹⁰ Senses *damn1*, *harangue1*, and *criticize1*, each of which appears in a cluster by itself, are not included in the table.

The resulting lists, averaging about 80 senses in length, were judged to be 98% accurate by the student validators. Our sense-tagged lists for GI categories are available at <http://www.cs.vassar.edu/~ide/GIsenselists/>.

Lists of “positive” and “negative” words are commonly used in opinion analysis, and several extensive lists are in circulation within the research community. The General Inquirer also provides lists of words with positive and negative connotations. Such lists include words from a broad range of semantic categories, since the only criteria for inclusion is their mutual polarity. For this reason it was not clear that the suresense procedure would be as effective in identifying relevant word senses. However, experimentation has so far shown that the results are better than anticipated. Inappropriate suresenses typically involve words whose inclusion in the list is questionable—for example, words like “colony” and “desire” in the GI’s list of negatives—although the procedure often fails to identify a suresense in such cases. We continue to experiment with producing suresense sets from polarity word lists; in particular, we are experimenting with threshold values as well as breaking the suresense sets into sub-sets based on clustering. Results will be posted on the website as they become available.

7 WordNet Sense Ambiguation

For the purposes of NLP, many WordNet senses can be viewed as identical—indeed, many of the problems with manual sense-tagging and word sense disambiguation that use WordNet senses arise from the at-times imperceptible shades of meaning that WordNet distinguishes. In an attempt to determine WordNet senses for the same word that can be, for practical purposes, collapsed, we applied WNS to determine the similarity among all *senses* of a given word. In this experiment, the full set of relations provided in WNS was used, rather than the reduced set applied in the experiments reported above.

Similarity scores between senses of the same word computed by WNS proved to be extremely low, which is not surprising given that the criterion for distinguishing senses in WordNet is membership in different synsets, one of the main criteria by which similarity is measured by WNS. Clustering based on the similarity matrix for the scores, however, indicated that the method holds some promise for collapsing WordNet senses: for example, the topology of the cluster tree for the verb “press” is given in Figure 1. The tree topology reflects sense grouping that are intuitively obvious, especially the close association of senses 10 and 7, 4 and 5, and 3 and 8; even more striking is the division between senses concerned with physical pressure and senses in which the use of “press” is abstract or metaphorical (excepting sense 11).¹¹ The clusters can be separated on the basis of varying distance cutoffs to create sense grouping at different levels of granularity.

We are currently experimenting with clustering WNS similarity measures to “ambiguate” WordNet senses, together with the incorporation of multiple suresenses for the same word identified by the algorithm. Based on the results so far, the method shows considerable promise for creating sense lists that are more usable for NLP.

¹¹ Note that the “press” example was randomly chosen, and is typical of the results we have seen so far in our experiments.

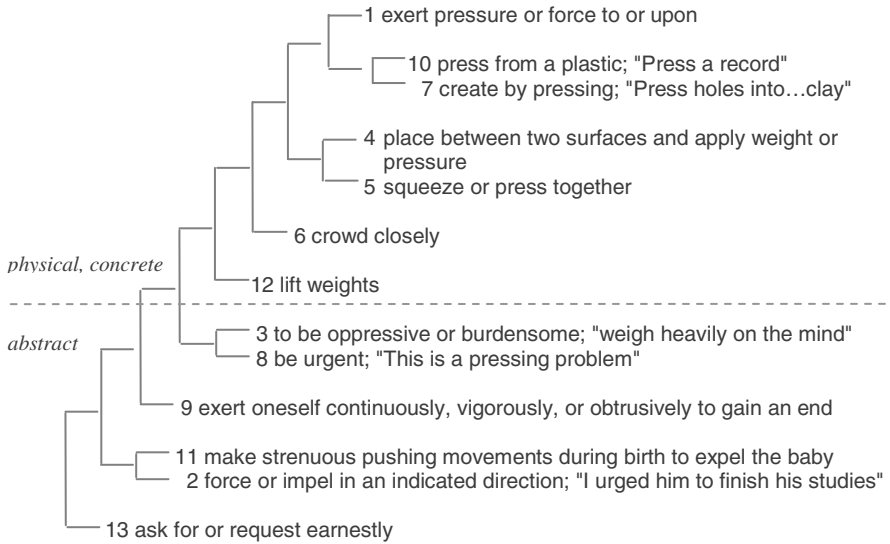


Fig. 1. Cluster tree topology for WordNet senses of “press”

8 Summary

The methods outlined in this paper demonstrate that similarity measures and clustering are effective methods for creating sense-tagged word lists for semantic analysis. Sense-tagged word lists are a valuable resource in their own right, but to be used in applications such as attitude an opinion analysis and event recognition, the corpus under analysis must be sense-tagged as well. This would seem to be a drawback to using sense-tagged word lists to accomplish these and other corpus-analytic tasks, since automatic disambiguation algorithms currently achieve, at best, about 80% accuracy, and the cost of hand-tagging or hand-validating sense-tags in even a modestly-sized corpus is prohibitive. However, automatic sense-tagging may soon cease to be the insurmountable problem it has traditionally been thought to be, if a “common” sense inventory is agreed upon within the community, and if the accuracy of entirely automatic sense disambiguation software is improved. We believe that both of these obstacles can be addressed, at least to a workable degree, with a single solution: adopt a set of senses derived from WordNet, in which senses that can be regarded as identical (at least, for the purposes of NLP) are collapsed.

We contend that a substantial portion of the “errors” generated using current disambiguation systems would be eliminated if WordNet senses were grouped into coarser-grained semantic categories. This is not a novel idea; the word sense disambiguation (WSD) community has long been aware that WordNet senses pose significant problems for the field because of their granularity, both for evaluating WSD systems and achieving agreement among human annotators. At the same time, the community is aware that homograph recognition—which can be automatically achieved with high rates of accuracy—is not enough for NLP. What has been recognized less frequently is that for most NLP applications, something not far from homo-

graph-level distinction *is* adequate, and that we have some good clues concerning what those distinctions are and how to identify them from a variety of sources, including cross-lingual information, psycholinguistic experiments, and, possibly, clustering “similar” WordNet senses as described in section 7, above (see [22] for a fuller discussion of this point). If the community can turn its attention to creating a usable sense inventory for NLP, then there is a future for automatic WSD.

In summary, we have within our means ways to significantly improve accuracy rates for WSD systems in the not-too-distant future. If this is done, it will in turn open the door to the use of systems such as WordNet::SenseRelate to perform accurate automatic WSD, and to the exploitation of these results in tasks such as attitude analysis and event recognition.

References

1. Turney, P.: Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews. Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (2002) 417-424
2. Pang, B., Lee, L., Vaithyanathan, S.: Thumbs up? Sentiment Classification using Machine Learning Techniques. Proceedings of Empirical Methods for Natural Language Processing (2002) 79-86
3. Pang, B., Lee, L., Vaithyanathan, S.: A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts. Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (2004) 271-278
4. Turney, P., Littman, M.: Unsupervised Learning of Semantic Orientation from a Hundred-Billion-Word Corpus, National Research Council, Institute for Information Technology, Technical Report ERB-1094 (NRC #44929) (2002)
5. Wiebe, J., Riloff, E.: Creating Subjective and Objective Sentence Classifiers from Unannotated Texts. In Gelbukh, A. (ed.): Computational Linguistics and Intelligent Text Processing, 6th International Conference Proceedings. Lecture Notes in Computer Science, Vol. 3406. Springer, Berlin Heidelberg New York (2005) 486-497
6. Vargas-Vera, M., Celjuska, D.: Event Recognition on News Stories and Semi-Automatic Population of an Ontology. In Shadbolt, N., O'Hara, K. (eds.): Selected Advanced Knowledge Technology Papers (2004) 159-163
7. Alphonse, E., Aubin, S., Bessières, P., Bisson, G., Hamon, T., Lagarrigue, S., Nazarenko, A., Manine, A.-P., Nédellec, C., Vetah, M., Poibeau, T., Weissenbacher, D.: Event-Based Information Extraction for the Biomedical Domain: The Caderige Project. In Collier, N., Ruch, P., Nazarenko, A. (eds.): Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications. (2004)
8. General Inquirer: <http://www.wjh.harvard.edu/~inquirer/> (2000)
9. Cunningham, H.: GATE, a General Architecture for Text Engineering. Computers and the Humanities, Vol. 36 (2002) 223-254
10. Polanya, L., Zaenen, A.: Contextual valence shifters. Proceedings of the AAAI Symposium on Exploring Attitude and Affect in Text: Theories and Applications (AAAI technical report SS-04-07) (2004) 106-111
11. <http://www.d.umn.edu/~tpederse/senserelate.html>
12. Ruppenhofer, J., Ellsworth, M., Petruck, M., Johnson, C.: FrameNet: Theory and Practice. On-line publication at <http://framenet.icsi.berkeley.edu/> (2005)

13. Kipper, K., Dang, H.T., Palmer, M.: Class-based construction of a verb lexicon. *Proceedings of Seventeenth National Conference on Artificial Intelligence* (2000) 691-696
14. Levin, B.: *English Verb Classes and Alternation: A Preliminary Investigation*. University of Chicago Press (1993)
15. Lin, D.: Automatic Retrieval and Clustering of Similar Words. *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics* (1998) 768-773
16. Pereira, F., Tishby, N., Lee, L.: Distributional Clustering of English Words. *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics* (1993) 183-190
17. Lee, L.: Measures of distributional similarity. *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics* (1999) 25-332
18. Pedersen, T., Patwardhan, S., Michelizzi, J.: WordNet::Similarity - Measuring the Relatedness of Concepts. *Proceedings of the Nineteenth National Conference on Artificial Intelligence* (2004) 1024-1025
19. Banerjee, S., Pedersen, T.: Extended gloss overlaps as a measure of semantic relatedness. *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence* (2003) 805-810
20. Shi, L., Mihalcea, R.: Putting Pieces Together: Combining FrameNet, VerbNet and WordNet for Robust Semantic Parsing. *Computational Linguistics and Intelligent Text Processing, 6th International Conference Proceedings. Lecture Notes in Computer Science, Vol. 3406*. Springer, Berlin Heidelberg New York (2005) 100-111
21. Palmer, M., Ng, H.T., Dang, H. Evaluation. In Agirre, E., Edmonds, P. (eds.): *Word Sense Disambiguation: Algorithms and Applications*. Springer, Berlin Heidelberg New York (forthcoming)
22. Ide, N., Wilks, Y.: Making Sense About Sense. In Agirre, E., Edmonds, P. (eds.): *Word Sense Disambiguation: Algorithms and Applications*. Springer, Berlin Heidelberg New York (forthcoming)

Enriching Wordnets with New Relations and with Event and Argument Structures*

Raquel Amaro^{1,**}, Rui Pedro Chaves¹, Palmira Marrafa^{1,2},
and Sara Mendes^{1,***}

¹ CLG – Group for the Computation of Lexical and Grammatical Knowledge,
Center of Linguistics, University of Lisbon, Portugal

² Department of Linguistics of the Faculty of Arts, University of Lisbon, Portugal
{ramaro, rui.chaves, palmira.marrafa, sara.mendes}@clul.ul.pt

Abstract. This paper argues that wordnets, being concept-based computational lexica, should include information on event and argument structures. This general approach is relevant both for allowing computational grammars to cope with a number of different lexical semantics phenomena, as well as for enabling inference applications to obtain finer-grained results. We also propose new relations in order to adequately model non explicit information and cross-part-of-speech relations.

1 Introduction

Wordnets are electronic databases developed along with the same general lines of the so-called Princeton WordNet, an electronic database of English [1, 2] containing nouns, verbs, adjectives, and adverbs. This database is structured as a network of relations between *synsets* (a set of roughly synonymous word forms). Several other wordnets have since been developed for many other languages and the number of relations adopted by the system has been enlarged (see for instance EuroWordNet [3]). In this paper we will show how wordnets can be integrated with a finer-grained lexical description framework in order to deal with various complex lexical semantics phenomena in a general and systematic way. Such an extension can be used both for deep lexical semantics analysis in computational grammars, and for a finer-grained linguistic knowledge-base in inference and question answering systems.

In Section 2 we will discuss the hyponymy/hypernymy relation. Following [4] we propose augmenting wordnet synset nodes with rich lexical-semantics descriptions which allow to explicitly capture the semantic inheritance patterns between hyponyms and hypernyms. We discuss some technical issues concerning this approach and provide a more general alternative view of semantic compatibility. Section 3 is dedicated to the verbal lexicon, focusing on argument

* The research reported in this paper has been partially supported by Instituto Camões, Fundação Calouste Gulbenkian, and Fundação para a Ciência e Tecnologia through the WordNet.PT, UniNet.PT, and COMPGRAM projects.

** Supported by FCT grant SFRH/BD/13875/2003.

*** Supported by FCT grant SFRH/BD/8524/2002.

structure. We will show that a decompositional approach to troponymy enables us to establish more precisely what is inherited through the hierarchy, accounting for different argument structures. We will show that co-troponyms' incompatibility is accounted for at the argument structure level, and that it is possible to state the exact prepositional complements selected by a verb at the lexical level, through the use of the available lexical structures in the lexicon. Section 4 is also dedicated to the verbal lexicon, but it focuses on event structure. We argue that telicity is not only a compositional property of syntactic structures but also a lexical property to be encoded in the lexicon. The analysis focuses on the behavior of complex telic predicates, in particular those which are deficient with regard to their lexical-conceptual structure. In order to represent appropriately such predicates in wordnets we propose a new relation, which has strong empirical motivation. In Section 5 we show that, despite the importance of the information that can be extracted from the hierarchical organization of lexical items, extending wordnets to all the main POS involves a revision of certain commonly used relations and the specification of several cross-part-of-speech relations. We focus on the specific case of adjective encoding and we present some strategies in order to mirror definitional features in the network, so that adjective classes emerge from the relations expressed in the database. In Section 6 we present some concluding remarks.

2 The Semantics of Hyponymy

Hyponymy is a relation which concerns not only world-knowledge, but also linguistic knowledge. Evidence for this comes from anaphoric constructions where the hypernym can be used to refer back to a more specific referent previously introduced (see [1]):

- (1) He owned a rifle, but the gun had not been fired.

The fact that the relation is hierarchical in nature does not allow hypernyms and hyponyms to be contrasted, as noted in [5]:

- (2) a. #A rifle is safer than a gun.
 b. #He owned a rifle, but not a gun.
 c. #He owned both a rifle and a gun.

Nominal hyponyms thus inherit all the information associated with the hypernym, and in turn further introduce specific semantic properties. Take for instance *school* and *bank*, two of the hyponyms of *institution*. The former is an institution which is dedicated to the teaching of students, and the latter is an institution dedicated to managing monetary funds. Although hyponymy is the main structuring relation in wordnets, there are other relations available such as meronymy and antonymy, but neither these relations nor the extended set of relations adopted in EuroWordNet are sufficiently expressive to adequately capture complex lexical semantic information.

Our goal is thus to enrich wordnets with a lexical semantics framework which allows to better describe the nature of lexical meaning as well as the specific semantic contribution made by a hyponym in relation to its hypernym. With this goal in mind, we have adopted the Generative Lexicon framework (henceforth GL, see [6]). In the approach we adopt, a synset is associated to a complex lexical description that encodes several kinds of semantic information, in particular, the specific semantic contribution of the synset as well as the meaning which is inherited from the hypernym. Such perspective has been put forth by [4], inspired by the distinction between *formal* and *telic* hypernymy in [7]. By enriching synsets with QUALIA descriptions, one can 'define' (to a reasonable extent) in what sense does one synset function as the hyponym of another. Take for instance the words *sword* and *rifle*. While man-made physical objects, both are hyponyms of *artifact*. In GL terms, *artifact* specifies properties about composition (via the FORMAL quale) which are inherited and further elaborated in *sword* and *rifle*. However, both synsets are also hyponyms of *weapon*, in the sense that these are entities devised for violent attack. In GL terms, the meaning of *weapon* is largely *agentive* and *telic*. Although the prototype of *weapon* is an object one can wield, some references are quite different: organic material can be a weapon (“*anthrax and other bio-chemical weapons*”), software (“*a computer virus is a weapon used to attack other computers (...)*”), violent coercion (“*terrorism is the political weapon of choice for some factions*”), and so on. It is not the case that ideas or molecules are weapons, but it is the case that there are in principle no incompatible properties between these concepts. [4] proposes to compute this referential possibility as a consequence of the QUALIA information associated to each synset, via an operation that integrates QUALIA features monotonically (*feature unification of QUALIA roles*). More specifically, two QUALIA structures are said to be *compatible* if the values of the pair-wise QUALIA roles are not inconsistent. Consider the multi-inheritance hypernymy structure of quales illustrated below:

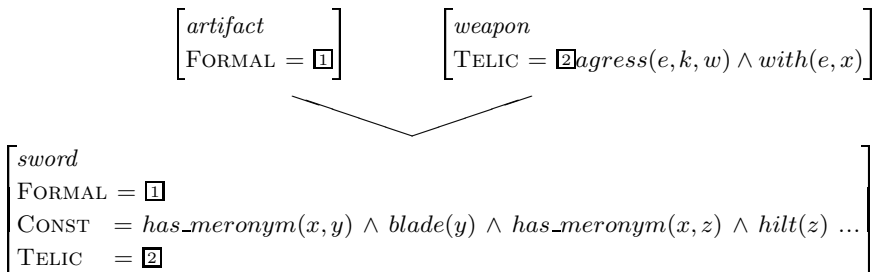


Fig. 1. QUALIA Inheritance and Hypernymy

Since these particular hypernyms are relatively underspecified, the relevant inherited information only concerns the FORMAL and the TELIC roles. Note that all the information present in the hypernym must be inherited, but that the

hyponym needs not be confined to it and should be able to add further information to any given quale. For instance, the TELIC role of *sword* coincides with the TELIC of *weapon* in the example above, but it may be the case that a hyponym introduces specific information in addition to the inherited properties.

This approach also allows to distinguish *compatible* from *incompatible* co-hyponyms: synsets X and Y that share the same hypernym (inheriting the same information) but where X introduces specific properties which may or not be consistent with the ones introduced by Y. For instance, *feline* and *canine* are incompatible co-hyponyms because the constitutive quale of *mammal* is extended with mutually inconsistent information about the animal’s morphology (cf. [8]). Compatible co-hyponymy obtains whenever QUALIA properties are orthogonally extended. E.g. some of the hyponyms of *dog* are compatible: *police dog* (extending the TELIC role), and any co-hyponym extending the constitutive role, such as *german shepperd*. Another example is *lap dog* (extending the TELIC role) and any co-hyponym that does not extend dog along the same dimensions, such as *poodle* (extending the constitutive role). Virtually every hyponym of *person* or *profession* is compatible with the remaining hyponyms (*man*, *sibling*, *teacher*, *witness*, *biologist*, *musician*, *lawyer*, etc). The alternative to the general approach in [12] seems to be to exhaustively mark all the pairs of compatible co-hyponyms.

However, the method proposed for determining if two given QUALIA are consistent – subsumption – is too restrictive. It correctly obtains that *canine* constitutive properties are incompatible with *feline* constitutive properties, but a noun may receive more than one kind of TELIC value, for instance. Nouns like *professor* and *biologist* have different TELIC properties, and yet are not incompatible. In our view, QUALIA should simply be conjoined rather than unified. It is world knowledge that imposes the relevant constraints: nothing needs to be said about entities having more than one function (i.e. TELIC role), but different *simultaneous* physical properties along the same dimension (“short fur” and “thick fur” are orthogonal and thus not inconsistent properties, while “thick fur” and “thin fur” are inconsistent) should be prohibited. This can be achieved by background world knowledge rules. We note also that such rules may even be suppressed in hypothetical contexts (“*If square circles existed (...)*”), children stories, or in metaphorical uses.

3 Argument Structure and the Semantics of Movement Verbs

Verbal concepts are related through a hyponymy relation that refers a special subtyping relation: troponymy. Troponymy establishes a relation between verbal concepts concerning types of manner (see [9, p. 79]):

(3) *to V1 is to V2 in some particular manner*

The types of manner denoted in the verbal concepts that determine hypernym/troponym relations can be of different kinds, accounting – as for nouns –

for the occurring sets of compatible co-troponym verbs, as explored in [10, 11] in a decompositional approach to troponymy within the set of verbs of movement.

- (4) a. He *came walking*.
 b. He *exited* the house *limping*.
 c. *He *walked flying*.
 d. *He *exited* the house *entering*.

In the GL framework, the argument structure is the representation level for logical arguments. It is thus natural that the semantic content of a verb meaning is reflected on the number and/or the type of arguments selected. Consider for instance the verbs *to move* (change the location of), hypernym of *to put* (move into a given location), and *to box* (put into a box), troponym of *to put*:

- (5) a. $\{box\} @-> \{put\} @-> \{move\}$
- b.
$$\left[\begin{array}{l} move \\ ARG-STR = \left[\begin{array}{l} ARG_1 = x: entity \\ ARG_2 = y: entity \end{array} \right] \end{array} \right]$$
- c.
$$\left[\begin{array}{l} put \\ ARG-STR = \left[\begin{array}{l} ARG_1 = x: entity \\ ARG_2 = y: entity \\ ARG_3 = z: goal \end{array} \right] \end{array} \right]$$
- d.
$$\left[\begin{array}{l} box \\ ARG-STR = \left[\begin{array}{l} ARG_1 = x: entity \\ ARG_2 = y: entity \\ S-ARG_1 = z: box \end{array} \right] \end{array} \right]$$

The meaning specificity of *to put* – denoting a specified GOAL – is reflected in an increase of the list of true arguments that is inherited from the hypernym *to move*. The expression of the final location (GOAL), ARG₃ – introduced by a preposition –, becomes obligatory in the case of the verb *to put*. Again, this argument structure is inherited by the immediate troponym, *to box*, that expresses a specific GOAL location, a box, through lexical shadowing, changing the predicate type of argument from true argument (ARG₃) to shadow argument (S-ARG₁). This level of representation allows us to establish more precisely what is inherited through the hierarchy and how, making it possible to account for different argument structures within a troponymy tree. Moreover, the account for compatibility issues among co-troponyms makes use of the argument structure of verbs. As stated in Section 2, two compatible nominal co-hyponyms are synsets that share the same hypernym and whose specific properties – information added to any given quale – are consistent with the properties of each other. However, this operation cannot be directly applied to the QUALIA structure of verbs. Verbal QUALIA structure is fulfilled with semantic predicates that establish the relations between the arguments of a verb. The QUALIA are also used

to reflect the internal structure of the events. For instance, the agentive and formal QUALIA are typically used, respectively, to represent the causal chain and the final state for accomplishment and achievement type events. This way, the compatibility among co-troponyms is checked at argument structure level.

Considering that it is in the argument structure that the logical arguments of a predicate are listed, reflecting the added information responsible for the meaning specificities between hypernyms and troponyms, we account for co-troponym compatibility indirectly, checking for arguments' incompatibilities, recurring also to the inheritance structure in the wordnet. Following the proposal in Section 2:

- (6) Two co-troponym verbs are incompatible *iff* the non-inherited arguments in their argument structure refer to incompatible co-hyponyms, i.e. if the QUALIA values of these arguments refer to opposite *simultaneous* properties along the same dimension. (see Section 2).

This indirect checking enables us to predict that the co-troponym verbs to *exit* (move out of) and to *enter* (move into) are incompatible, see (4d), since the QUALIA structure of out and in – which are co-hyponyms – are not consistent.

$$(7) \text{ a. } \left[\begin{array}{l} \textit{exit} \\ \text{ARG-STR} = \left[\begin{array}{l} \text{ARG}_1 = x: \textit{entity} \\ \text{ARG}_2 = y: \textit{location} \\ \text{S-ARG}_1 = z: \textit{out} \end{array} \right] \end{array} \right] \quad \text{b. } \left[\begin{array}{l} \textit{enter} \\ \text{ARG-STR} = \left[\begin{array}{l} \text{ARG}_1 = x: \textit{entity} \\ \text{ARG}_2 = y: \textit{location} \\ \text{S-ARG}_1 = z: \textit{in} \end{array} \right] \end{array} \right]$$

Conversely, co-troponym verbs *exit* (move out) and *limp* (move using a leg deficiently) (in (4b)) are compatible since there are no co-hyponym arguments in their structure. The integration of an argument structure can also provide means for some syntactic mapping. In the GL, it is assumed that the type of arguments and their listing order (from less oblique to more oblique) account for the syntactic mapping of the arguments [6, pp. 62–67]. However, the representation of the arguments in these terms does not state which type of oblique argument – typically expressed by prepositional phrases – is selected by a given verb. For instance, the argument structure of the verb *to exit* in (7a) does not reveal that the ARG₂ is a prepositional phrase, nor which preposition heads this particular phrase.

It is our proposal that the use of the lexical structures available in the lexicon should make possible to state at the lexical level the exact prepositional complement selected by the verb. This proposal assumes the integration of prepositions in the lexicon, following [12] that states that the semantic contribution of prepositional phrases is consistent across uses, regardless of their status as complements or adjuncts. Prepositional lexical entries allow to account for the semantic contribution of the prepositional phrase in sentences such as “*He pulled the box from here*”, as well as when the semantic content of the preposition is part of the semantic content of the verb itself, as in the case of the verb *to exit*.

4 Event Structure and the Semantics of Telic Verbs

The semantics of telic verbs involves a change of state of their theme argument. In other words, the sub-event that closes the whole event is an atomic event (i.e. a state) that affects the theme and is different from its initial state, as briefly represented below.

$$(8) \left[\begin{array}{l} \text{EVENT-STR} = [{}^{\sigma}_e e_1: process <_{\infty} e_2: state] \\ \text{ARG-STR} = \left[\begin{array}{l} \text{ARG}_1 = x: \textit{entity} \\ \text{ARG}_2 = y: \textit{entity} \end{array} \right] \\ \text{QUALIA} = \left[\begin{array}{l} \text{AGENTIVE} = \textit{act}(e_1, x, y) \\ \text{FORMAL} = \textit{result}(e_2, y) \end{array} \right] \end{array} \right]$$

It becomes apparent from (8) that the event denoted by the verbs at stake has a typical transition type geometry, with an initial head sub-event (e_1 : *process*) and a definite endpoint sub-event (e_2 : *state*) which corresponds to the final state of the argument that undergoes the result of the event. In most cases e_2 is shadowed or externalized by means of a subtyping operation.

- (9) a. John washed his shirt.
 b. John washed his shirt white/*washed.

- (10) a. John painted his house.
 b. John painted his house yellow/*painted.

Sentence (9b) entails that John's shirt is white as a result of washing. Similarly, sentence (10b) entails that John's house became yellow as a result of painting. Following [13] and previous work, we assume that the constituent that expresses the result of the event denoted by the verb integrates the predicate. In other terms, the verb plus the resultative constitute a complex predicate, as extensively argued in [14]. However this is not an uncontroversial issue. As a matter of fact, despite the general assumption that resultative constructions are telic constructions (i.e. they describe events with a definite endpoint), there is a major controversy on whether or not the telic aspect of such constructions is an inherent feature of the meaning of the corresponding verbs. The compositional hypothesis, defended by [15], has been argued for in more recent works (see, for instance [16]) on the basis of contrasts like the following:

- (11) a. John painted his house in one year / *for one year.
 b. John painted houses *in one year / for one year.

At a first glance, these examples suggest that (11a) is telic and (11b) is atelic and, consequently, that telicity depends on the nature of the internal argument. Hence, telicity would be a compositional feature of VP and not a lexical feature of V. However, the relevant opposition seems to be transition *vs* process (in the sense of [17]) and not telic *vs* atelic aspect.

As defended in [14], though the global event in (11a) is a process, its main sub-events are not atomic events, but transitions. Let us compare the structure of the global event of (11a) and (11b), represented by (12a) and (12b), respectively (T: Transition; P: Process; e: atomic event):

- (12) a. $[_T [_T e_1 \dots e_n] e_m] : e_m > e_n$
 b. $[_P [_{T_1} [_P e_1^t \dots e_n] e_{m1}] \dots [_{T_t} [_P e_1^t \dots e_k] e_{m2}] \dots] : e_{m1} > e_n, e_{m2} > e_k$

Similarly to e_m , in (12a), e_{m1} and e_{m2} , in (12b), are telic states. This suggests that, although telicity is a compositional feature regarding the whole sentence, it is also an intrinsic feature of the verb. By default, verbs like *paint* are associated to the following Lexical-Conceptual Structure (LCS' in [17]):

- (13) $[_T [_P \text{act}(x, y) \text{ and } \sim Q(y)], [_e Q(y)]] : Q: \text{atomic event}$

Instantiating the variables with the data of the first example above, we obtain:

- (14) $[[\text{act}(\text{john}, \text{his_house}) \text{ and } \sim \text{painted_yellow}(\text{his_house})],$
 $\qquad\qquad\qquad [\text{painted_yellow}(\text{his_house})]]$

The absence of the resultative (*yellow*) does not have any impact on the LCS:

- (15) $[[\text{act}(\text{john}, \text{his_house}) \text{ and } \sim \text{painted}(\text{his_house})], \quad [\text{painted}(\text{his_house})]]$

However, in the case of verbs like *to make*, discussed below, it seems impossible to assign a value to Q independently of the resultative. Consider the sentence given below in (16). The LCS associated with it seems to be (17a) and not (17b):

- (16) He made Mary happy.
 (17) a. $[[\text{act}(\text{he}, \text{Mary}) \text{ and } \sim \text{happy}(\text{Mary})], [\text{happy}(\text{Mary})]]$
 b. $[[\text{act}(\text{he}, \text{Mary}) \text{ and } \sim \text{made_happy}(\text{Mary})], [\text{made_happy}(\text{Mary})]]$

Therefore, Q is instantiated just with the resultative. The absence of the resultative induces ungrammaticallity, as expected:

- (18) *He made Mary.

Along the same lines of [14] and [13], verbs like *to make* are defended here to be LCS deficient, in the following sense (informal definition):

- (19) $\forall v((v \text{ a verb}, \exists \varepsilon, \varepsilon \text{ the LCS of } v, \exists \pi, \pi \text{ the set of content properties of } \varepsilon, \pi = \emptyset) \Rightarrow \text{LCS_deficitary}(v))$

Since $\pi = \emptyset$, the LCS cannot bear an appropriate interpretation. A syntactic structure that projects an anomalous LCS is, then, expected to be ruled out, since it does not satisfy the commonly accepted requirement of full interpretation. In this case, the resultative fills the gap of the LCS of the verb. Therefore, these facts render evident that the representation of the predicates at issue has to include information regarding the telic expression. Obviously, it would not be

adequate to overtly include in the synset all the expressions that can integrate the predicate, among other reasons, because they seem to constitute an open set. Rather, we claim that we can capture the telicity of these verbs by the inclusion of a new relation in the set of the internal relations of wordnets: the telic sub-event relation, which has two inverse counterparts, as exemplified below.

- (20) $\{make\}$ has_telic_sub-event $\{state\}$
 $\{state\}$ is_telic_sub-event_of $\{make\}$

Relating *make* to *state* by means of this relation, we capture the telic properties of the verb and let underspecified the specific nature of the final state. This way, we also account for the weakness of the verb selection restrictions. As expected, we can also use this relation to encode telicity in the case of the troponyms of the class of verbs discussed so far. Let us examine an example:

- (21) a. He saddened Mary.
 b. He made Mary sad.
 c. *He saddened Mary sad.

Verbs like *sadden* incorporate the telic state. This fact justifies that *sadden* can be paraphrased by *make sad* ((21a) is semantically equivalent to (21b)) and cannot co-occur with *sad* (cf. (21c)). In these cases, we use the telic sub-event relation to relate the verb to the expression corresponding to the incorporated telic information:

- (22) $\{sadden\}$ has_telic_sub-event $\{sad\}$
 $\{sad\}$ is_telic_sub-event_of $\{sadden\}$

It should be noticed that the existing sub-event relation in the EuroWordNet framework is different from the relation proposed here. It only stands for lexical entailment involving temporal proper inclusion. Therefore, it does not account for the geometry of the event. On the contrary, the telic sub-event relation regards the atomic sub-event that is the ending point of the global event.

As shown, the telic sub-event relation allows straightforwardly the encoding of lexical telicity in wordnets, in accordance with the empirical evidence.

5 Encoding Cross-Part-of-Speech Relations

In section 2, we focused on *hyponymy* since it is the main structuring relation in wordnets. Even if we claim here that more detailed semantic information should be introduced in computational lexica (cf. Sections 2 and 3), it is undeniable that important structural information can be extracted from the hierarchical organization of lexical items, namely of nouns and verbs. However, extending wordnets to all the main POS involves a revision of certain commonly used relations and the specification of several cross-part-of-speech relations. In this section we will focus on adjectives.

As pointed out by [1, 18], the semantic organization of adjectives is unlike that of nouns and verbs, as this POS does not generally show a hierarchical organization. Thus, encoding adjectives in wordnets calls for the specification of a number of cross-part-of-speech semantic relations. In the following subsections we will present some strategies in order to mirror adjectives main features in wordnets, namely definitional ones. This way, it is possible to make adjective classes emerge from the relations expressed in the network.

5.1 Adjectives in Wordnets

In Princeton WordNet, descriptive and relational adjectives are distinguished by both being encoded in separate files and by the relations holding between synsets. Descriptive adjectives are organized in clusters of synsets, each cluster being associated by semantic similarity to a focal adjective linked with a contrasting cluster via an *antonymy* relation. Relational adjectives, on the other hand, do not have antonyms and cannot be organized in opposite clusters. Thus, relational adjectives are linked to the nouns they relate to.

[19] discusses this organization of adjectives in GermaNet. It abandons the cluster structuring of adjectives in favor of an uniform treatment of all POS in taxonomic chains. The distinct treatment of relational and descriptive adjectives is also abandoned in GermaNet, as the distinction between these two classes is considered to be 'not at all clear'. Here, along with [18] and [1], we will claim that, even if the distinction between relational and descriptive adjectives is not always clear-cut, it is however a relevant one, as these adjectives differ in terms of their intrinsic meaning, as well as with regard to their syntactic and semantic behavior. To put it somewhat simplistically, descriptive adjectives ascribe a value of an attribute to a noun. We introduce a new relation, the *characterizes with regard to/can be characterized by*,¹ linking each descriptive adjective to the attribute it modifies. Thus, instead of linking adjectives amongst themselves by a similarity relation, all adjectives modifying the same attribute are linked to the noun that lexicalizes this attribute. This way we obtain the cluster effect, argued in [18, 1] to be the basis of the organization of adjectives, without having to encode it directly in the network (see [20]).

As shown by word association tests, *antonymy* is also a basic relation in the organization of descriptive adjectives. Nonetheless, this relation does not correspond to conceptual opposition: *antonymy* holds between word forms and not word meanings. We argue that conceptual opposition does not have to be explicitly encoded either, since it is possible to make it emerge from the combination of *synonymy* and *antonymy* relations as in [20]. This way, we are able to define adjective clusters without the *indirect antonymy* relation used in Princeton WordNet, as we manage to obtain the cluster effect via the *antonymy* and the *characterizes with regard to / can be characterized by* relations. In fact, our strat-

¹ This semantic relation is very close to the *is a value of/attributes* relation used in Princeton WordNet. In WordNet.PT we changed its label in order to make it more straightforward to the common user.

egy is more intuitive and descriptively adequate, since many attributes are not bipolar, but can take many values along a continuum.

Concerning relational adjectives, and unlike what is done in other wordnets, we claim that these should be encoded in the same file as descriptive adjectives, avoiding having to decide beforehand whether an adjective is relational or descriptive, for instance. Rather, membership to these classes emerges from the relations expressed in the database. Being, like descriptive adjectives, property ascribing adjectives, relational adjectives usually entail more complex and diversified relations between the set of properties they introduce and the modified noun, often pointing to a domain exterior to it, the denotation of another noun. We introduce the *is related to* relation to encode this.

Thus, the *characterizes with regard to / can be characterized by* and the *antonymy* relation for descriptive adjectives, and the *is related to* relation for relational adjectives, allow us to encode the basic characteristics of these adjectives in the database, on the one hand, while making it possible to derive membership to these classes from the relations expressed in the database, on the other hand.

5.2 Additional Relations

Ideally, the distinctive syntactic and semantic properties of lexical items would be encoded in lexical models such as wordnets. The SIMPLE project, for instance, addresses the semantics of adjectives (see [21]), identifying a set of features claimed to be relevant for classifying and describing their behavior. Adjectives are organized in terms of semantic fields, but these authors note that, even though similarities exist, the classes proposed in SIMPLE are not homogeneous, as adjectives belonging to the same semantic class often differ from each other in various ways.

We introduce a new relation to encode salient characteristics of nouns expressed by adjectival expressions: *is a characteristic of / has as a characteristic*. Despite the fact that we can object the status of this relation is not clear, concerning the lexical knowledge, it regards crucial information for many wordnet-based applications, namely those using inference systems, allowing for richer and clearer synsets.

Also, it may allow deducing semantic domains from the database: if synsets are encoded in this fine-grained way, it may be possible to identify the typical semantic domains of application of adjectives. The research on the classes and semantic domains emerging from the relations expressed in the database is still ongoing. Future work should include a comparative study between the classes extracted from the database and classes defined by several authors.

6 Conclusion

We have motivated the introduction of information on event and argument structures in wordnets, showing how this general approach is relevant both for allowing computational grammars to cope with a number of different lexical semantics

phenomena, as well as for enabling inference applications to obtain finer-grained results. We have also proposed some new relations in order to adequately model non explicit information. Focusing on the specific case of adjective encoding in wordnets, new cross-part-of-speech relations are also introduced in order to mirror definitional features of this POS in the network and to allow for the deduction of adjective classes from the information encoded in the database.

Future work will focus on methods for the specification of the information on QUALIA, event and argument structures, ideally through new relations, in the WordNet model.

References

1. Miller, K.J.: Modifiers in wordnet. In Fellbaum, C., ed.: *WordNet: an electronic lexical database*. Cambridge, MA: The MIT Press (1998) 47–68
2. Fellbaum, C.: A semantic network of english: the mother of all WordNets. In Vossen, P., ed.: *EuroWordNet: a Multilingual Database with Lexical Semantic Networks*. Dordrecht, Kluwer Academic Publishers (1998) 137–148
3. Vossen, P.: Introduction to EuroWordNet. *Computers and the Humanities* (32) (1998) 73–89 Reprinted In: *EuroWordNet – A Multilingual Database with Lexical Semantic Networks*, Kluwer Academic Publishers (1998), 73–89.
4. Mendes, S., Chaves, R.P.: Enriching wordnet with qualia information. In: *Proceedings of the Workshop on WordNets and Other Lexical Resources, NAACL 2001 Conference, Pittsburgh (2001)* 108–112
5. Bever, T.G., Rosenbaum, P.S.: Some lexical structures and their empirical validity. In Jacobs, R.A., Rosenbaum, P.S., eds.: *Readings in English Transformational Grammar*. Waltham, Mass., Ginn (1970)
6. Pustejovsky, J.: *The Generative Lexicon*. MA: The MIT Press (1995)
7. Pustejovsky, J.: *The Generative Lexicon*. *Computational Linguistics* **17**(4) (1991) 409–441
8. Cruse, A.D.: *Lexical Semantics*. Cambridge University Press, Cambridge (1986)
9. Fellbaum, C.: A semantic network of english verbs. In Fellbaum, C., ed.: *WordNet: an electronic lexical database*. MA: The MIT Press (1998) 69–104
10. Amaro, R.: Semantic incorporation in a portuguese WordNet of verbs of movement: on aktionsart shifting. In: *Proceedings of the Third International Workshop on Generative Approaches to the Lexicon, École de Traduction et d’Interpretation, University of Genève (2005)* 1–9
11. Amaro, R.: Wordnet as a base lexicon model for the computation of verbal predicates. In: *Proceedings of the GWA 2006, Global WordNet Association Conference, Jeju Island, Korea (2006)* to appear.
12. Verspoor, C.M.: *Contextually-Dependent Lexical Semantics*. Phd dissertation, University of Edinburgh (1997)
13. Marrafa, P.: The representation of complex telic predicates in WordNets: the case of lexical-conceptual structure deficitary verbs. In nosa, J.C., Gelbukh, A., Tovar, E., eds.: *Research on Computing Science. Volume 12.* (2005) 109–116
14. Marrafa, P.: *Predicação Secundária e Predicados Complexos: Modelização e Análise*. Phd. dissertation, University of Lisbon (1993)
15. Verkuyl, H.: *On the Compositional Nature of Aspects*. D. Reidel (972)
16. Schmitt, C.: *Aspect of the Syntax of Noun Phrases*. Phd. dissertation, University of Maryland (1996)

17. Pustejovsky, J.: The syntax of event structure. *Cognition* **41** (1991) 47–81
18. Fellbaum, C., Gross, D., Miller, K.: Adjectives in WordNet. In Miller et al., ed.: *Five papers on WordNet*, Technical Report, Cognitive Science Laboratory, Princeton University (1993) 26–39
19. Hamp, B., Feldweg, H.: GermaNet – a lexical semantic net for german. In: *Proceedings of ACL workshop on Automatic Information Extraction and Building of Lexical Semantic Resources for NLP Applications*, Madrid (1997)
20. Mendes, S.: Adjectives in WordNet.PT. In: *Proceedings of the GWA 2006, Global WordNet Association Conference*, Jeju Island, Korea (2006) to appear.
21. Peters, I., Peters, W.: The treatment of adjectives in simple: theoretical observations. In: *Proceedings of LREC 2000*. (2000)

Experiments in Cross-Language Morphological Annotation Transfer

Anna Feldman, Jirka Hana, and Chris Brew

Ohio State University,
Department of Linguistics,
Columbus, OH 43210-1298, USA

Abstract. Annotated corpora are valuable resources for NLP which are often costly to create. We introduce a method for transferring annotation from a morphologically annotated corpus of a source language to a target language. Our approach assumes only that an unannotated text corpus exists for the target language and a simple textbook which describes the basic morphological properties of that language is available. Our paper describes experiments with Polish, Czech, and Russian. However, the method is not tied in any way to these languages. In all the experiments we use the TnT tagger ([3]), a second-order Markov model. Our approach assumes that the information acquired about one language can be used for processing a related language. We have found out that even breathtakingly naive things (such as approximating the Russian transitions by Czech and/or Polish and approximating the Russian emissions by (manually/automatically derived) Czech cognates) can lead to a significant improvement of the tagger's performance.

1 Introduction

Genetically related languages possess a number of properties in common. For example, Czech and Russian are similar in many areas, including lexicon, morphology, and syntax (they have so-called free word-order). This paper explores the resemblances between Czech, Russian, and Polish, as well as exploits linguistic knowledge about these languages for automatic morpho-syntactic annotation without using parallel corpora or bilingual lexicons. Our experiments use these three languages; however, a broader goal of this work is to explore the general possibility of porting linguistic knowledge acquired in one language to another. This portability issue is especially relevant for minority languages with few resources.

Cross-language information transfer is not new; however, most of the existing work relies on parallel corpora (e.g. [7, 11, 12]) which are difficult to find, especially for lesser studied languages, including many Slavic languages. In our work, we explore a new avenue — We use a resource-rich language (e.g. Czech/Polish) to process a resource-poor genetically related language (e.g. Russian) without using a bilingual lexicon or a parallel corpus.

We tag Russian by combining information from a resource-light morphological analyzer ([5]) and information derived from Czech and Polish.

In the following we report both the overall performance of the model as well as its performance limited to nouns. We deliberately choose nouns, because:

1. As the most open class, nouns are extremely difficult to cover with manually created resources. The set of named entities and proper names is virtually infinite.
2. Nouns is the most challenging category. In the majority of Slavic languages, noun inflection is less systematic than, say, inflection of adjectives or verbs. Moreover, the morphemes are highly homonymous.
3. For practical reasons, we have to limit the scope of our work.

We report tagging accuracy on both the tag as a whole and five categories corresponding to five sub-parts of the complete tag (see Table 1) – part of speech (12 possible values, incl. N/A), detailed part of speech (32, e.g. infinitive or ordinal numeral), gender (5), number (4), and case (8). Note that the number of possible values for detailed part of speech is comparable to the size of Penn Treebank tagset with 36 non-punctuation tags ([8]).

Table 1. Overview and comparison of the tagsets

No.	Description	Abbr.	No. of values		
			Cz	Ru	Po
1	POS	P	12	12	12
2	SubPOS – detailed POS	S	75	32	20
3	Gender	g	11	5	5
4	Number	n	6	4	4
5	Case	c	9	8	9
6	Possessor’s Gender	G	5	4	2
7	Possessor’s Number	N	3	3	2
8	Person	p	5	5	5
9	Tense	t	5	5	5
10	Degree of comparison	d	4	4	4
11	Negation	a	3	3	3
12	Voice	v	3	3	3
13	Unused		1	1	1
14	Unused		1	1	1
15	Variant, Style	V	10	2	1

2 Tag System

We have adopted the Czech tag system ([4]) for Russian and Polish. Every tag is represented as a string of 15 symbols each corresponding to one morphological category ([6]).

The tagset used for Czech (4290+ tags) is larger than the tagset we use for Russian (about 900 tags). There is a good theoretical reason for this choice – Russian morphological categories usually have fewer values (e.g. 6 cases in

Russian vs. 7 in Czech; Czech often has formal and colloquial variants of the same morpheme); but there is also an immediate practical reason – the Czech tag system is very elaborate and specifically devised to serve multiple needs, while our tagset is designed to capture only the core of Russian morphology, as we need it for our primary purpose of demonstrating portability and feasibility of our technique. The Polish corpus contains 600 tags. This is due to the fact that the original Polish corpus is tagged with a different tagset, which had to be translated into our system and the correspondences are not always isomorphic (see the discussion in section 5.2).

3 Corpora

The experiments below are based on several corpora. One is the first 630K tokens of the morphologically annotated Prague Dependency Treebank ([2]). The other is 630K tokens of the IPI PAN Polish corpus ([9]), translated into our tag system (see 5.2 for the tag translation details).

For development purposes, we selected and morphologically annotated (by hand) a small portion from the Russian translation of Orwell’s *1984*. This corpus contains 1858 types (856 types). In the following sections we discuss our experiments and report the results.¹

4 Morphological Analysis

Our morphological analyzer is a knowledge and labor light system, which takes the middle road between completely unsupervised systems on the one hand, and systems with extensive manually-created resources on the other. Our position is that for the majority of languages and applications neither of these extreme approaches is warranted. The knowledge-free approach lacks precision and the knowledge-intensive approach is usually too costly.

The analyzer is an open and modular system. It allows us to combine modules with different levels of manual input – from a module using a small manually provided lexicon, through a module using a large lexicon automatically acquired from a raw corpus, to a guesser using a list of paradigms, as the only resource provided manually. The general strategy is to run modules that make fewer errors and less overgenerate before modules that make more errors and overgenerate more. This, for example, means that modules with manually created resources are used before modules with resources automatically acquired.

5 Tagging

Our approach assumes that information acquired about a language can be used for processing a related language, in our case information acquired about Czech or Polish can be used to tag Russian.

¹ Note that we do not report the results for tag position 13 and 14, since these positions are unused; and therefore, are always trivially correct.

In ([6]), we describe an n-gram Russian tagger, where transition probabilities were approximated by Czech transition probabilities and emission probabilities were approximated by uniformly distributed output of the morphological analyzer.²

In this section, we report on some of the experiments testing both limits and possible enhancements to this basic approach. All the results are summarized in Table 2 (all tokens) and Table 3 (nouns only). In all experiments (except the lower bound), we use the TnT tagger ([3]), which is a second-order Markov model.

5.1 Bounds

Our main practical goal is to develop a portable system for morphological tagging. From the theoretical point of view, we want to understand and isolate general properties of languages that seem to make a difference in the cross-language transfer approach. The experiments discussed in the following two sections simulate two ideal situations: 1) when the word order of a source language is identical to that of the target language (the word order upperbound); 2) when the lexicon of a source language is identical to the target lexicon (the emission upperbound). The upperbounds are given in columns 1 and 2 of Tables 2 and 3.

Table 2. Tagging the Russian Development Corpus: All experiments, all categories

All POS	1	2	3	4	5	6	7	8
Tagger	Max		Even emissions				Cognates	
Accuracy	trans	emis	Cz t	Po t	Interlg	Po&Cz	Manual	Auto
Tags	81.2	95.6	78.6	74.9	79.7	79.1	81.0	80.4
POS	94.5	99.7	92.7	92.0	92.3	91.8	92.8	92.1
SubPOS	87.4	99.7	90.9	90.6	90.0	90.4	91.1	90.5
Gender	93.7	99.7	91.1	90.7	92.1	91.9	92.6	92.2
Number	95.8	99.7	94.0	93.8	94.7	94.6	94.8	94.7
Case	91.7	95.6	87.6	82.6	86.7	86.2	88.3	88.3

Upperbound – Word Order. First, we decided to test how close the Czech and Russian word order is. If they were, it would mean we can train language models relying on word-order, e.g. n-grams, on one language and use it for another.

To measure the upper-bound of the performance of such a model, i.e. the perfect match between the word order in Czech and Russian, we trained the transitions on a small corpus of Russian, and ran [5]’s morphological analyzer to obtain evenly distributed emissions. The results obtained are summarized in column 1 (82.6% accuracy for the nouns). What this means is that the remaining 17.4% deficits are **not** due to word order divergence.

² Since Russian and Czech do not use the same words we cannot use the Czech emissions (at least not directly).

Table 3. Tagging the Russian Development Corpus: All experiments, nouns

Nouns	1	2	3	4	5	6	7	8
Tagger	Max		Even emissions				Cognates	
Accuracy	trans	emis	Cz t	Po t	Interlg	Po&Cz	Manual	Auto
Tags	82.6	94.8	65.8	57.0	66.1	66.9	71.3	68.9
POS	97.2	99.7	94.5	93.1	93.1	93.9	94.8	94.2
SubPOS	97.2	99.7	94.5	93.1	93.1	93.9	94.8	94.2
Gender	89.0	98.9	83.5	84.6	84.6	84.3	87.3	85.1
Number	92.0	99.7	90.1	88.4	89.3	89.8	91.2	90.6
Case	87.9	95.6	76.9	65.8	73.8	76.0	79.1	78.2

Upperbound – Lexical Similarities. In the next step, we test how useful the source language lexicon is for the tagging of the target language (here, Russian). We use Czech transitions and Russian emissions, obtained by training TnT on our development corpus. This is the upper-bound performance corresponding to the situation where the source language and the target Russian words behave the same way, all occur in the training data, and we have their perfect translations. The results are in column 2 (94.8% accuracy for nouns). It is clear that the knowledge about Czech-Russian lexical correspondences would definitely help to improve the tagger’s performance.

5.2 Approximating Transitions

Below we discuss a number of experiments exploring possibilities of transferring transition probabilities necessary for tagging Russian from a related language.

Approximating by Czech or Polish. In section 5.1, we discuss the word order upperbound. This is an approximation to the performance of the model that would be obtained if there were a perfect correspondence in the word order of Czech and Russian. We wish to know if this result which is obtained by using information about the transitions in the Russian test data, information that we do not have in any realistic situation, can be approximated using Czech.

We train the transitions on 630K Czech tokens, and use the morphological analyzer to create evenly distributed emissions for Russian. The results are given in column 3 in Tables 2 and 3. Such a method approaches the upperbound on transitions.

We also ran an identical experiment with Polish, using the IPI PAN corpus ([9]). This corpus is morphosyntactically annotated, but the structure of its morpho-syntactic tags is different from the tagset we used for Czech and Russian. The repertoire of grammatical categories used in the IPI PAN corpus is different from the Czech tagset. For example, some Polish pronouns are tagged as adjectives, since they have adjectival inflections, whereas the Czech system makes more fine-grained distinctions. Traditional grammatical categories which are represented only partially in the IPI PAN tagset include tense, mood and voice. In addition, since we intentionally did not use a native speaker’s expertise

for checking the translations (keeping the project resource light), in addition to many differences in the tagset conventions, the translations are not 100% reliable. More importantly, there are obvious linguistic differences between Polish, Czech and Russian. Animacy agreement for adjectives and nouns is obligatory in Polish, whereas in Czech it is manifested only partially and does not exist in Russian at all. There are two types of obligatory copula in Polish (*byc, to*), only one in Czech and none in Russian (for present tense). So, we did not expect a pure Polish model to perform better than the Czech when tagging Russian text.

The results of the experiments are given in Table 2 and Table 3, column 4, for all categories and nouns, respectively. The performance of the Polish model is not as good as of the Czech.

Slavic Interlingua. We discuss one possible solution in detail in [6]. We train the tagger on individual components of the full tag (thus in addition, reducing data sparsity) and then combine them by simple voting. The relative reduction of error rate is 3.3%.

Another possible solution is to create a training corpus which will look more like Russian. Simple “russifications” of Czech lead to 10.5% reduction in relative error rate ([6]).

Every person who knows a Slavic language is able to translate this text, even though it does not belong to any living language: *Korchagin oxvatil glavu rukami i gluboko sa zamyslil. Pred ochami mu prebezhel cely jeho zhiivot, od detinstva i do poslednix dni. Dobre li on prozhil svoje dvadeset i chetyri let, ili je zle prozhil?*³ The purpose of this example is to show that it is possible to construct texts that are intelligible to all Slavic speakers.

With a similar idea in mind and with the goal of using minimal resources and minimal knowledge that will not require native speakers’ expertise, we decided to create a pseudo-Slavic language which would fuse elements of Czech and Russian and have more Russian-like properties without relying on sophisticated linguistic knowledge. The simple way of doing it is diluting the Czech training data with another resource-rich language which has more Russian-like properties, complementary to Czech. One such language is Polish. We concatenate the Czech 630K tokens with the Polish 630K tokens to create a new training corpus. Polish has some properties that Czech does not. We expect that if we train a tagger on the combination of the two texts, the overall tagging result will improve. The reasons are that negation in Polish is expressed by the particle, whereas in Czech it is expressed by prefixation. Russian is somewhere in the middle – it has cases where negation is a particle, but there is also a class of words, e.g. certain verbs or adverbs, that negate by prefixation. Polish has obligatory genitive of negation. Czech does not have this phenomenon. Russian

³ This text is a translation of an excerpt from the book *How the Steel Was Tempered* by Nikolai Ostrovsky. Greg Kondrak constructed the translation on the basis of Old-Church-Slavonic, and by consulting translations into the following Slavic languages: Serbo-Croatian, Slovenian, Bulgarian Russian, Ukrainian, Belarusian, High and Low Sorbian, Czech, Slovak, and Polish (from *Introduction to the phonological history of the Slavic languages* by Terence Carlton).

genitive of negation is only partial. With certain noun phrases it is optional, with certain noun phrases it is obligatory. Possessive sentences in Polish are more like Russian (omitting "have", dative constructions) rather than in Czech. The performance of the Russian tagger trained on the Slavic interlingua is given in Table 2, column 5, for all parts-of-speech, and in Table 3, column 5, for the nouns. Our expectations have been met. the interlingua model improves the performance of the Czech model by 1.1% and the pure Polish model by 4.8%, which is a significant improvement. On nouns, The tagging result of the interlingua model is better than the Polish by 9.1%.

Combining Two Language Models. Another possibility is to train the transitions separately on Czech and on Polish and then combine the resulting models into one, taking into account the typological facts about these languages. Based on our linguistic knowledge, we assumed that Polish gender and number for nouns and verbs are more reliable than Czech. The results, given in Tables 2, column 6 for all 12 categories, are better than the models with transition probabilities from individual languages, but not as good as results from the interlingua model. However, in the case of nouns, the situation is reversed. The hybrid model performs better than the interlingua one. The reason, we think, is that the gender and number category is the most relevant for nouns, and our linguistic intuition was correct. We believe that a more sophisticated combination of models would create better results.

5.3 Approximating Emissions – Czech-Russian Cognates

As we said above, since Russian and Czech do not use the same words we cannot use the Czech emissions directly. Instead, the models above approximated Russian emissions by uniformly distributing output of a morphological analyzer. This is a very crude approximation. In this section we explore a different possibility. Although it is true that forms and distributions of Czech and Russian words are not the same, they are also not completely unrelated.

The corresponding Czech and Russian words can be cognates, i.e. historically they descend from the same ancestor root or they are mere translations. We assume that (1) translation/cognate pairs will have similar morphological and distributional properties;⁴ (2) cognate words are similar in form.

Manually Selected Cognates. To test the first assumption, we created by hand a list of 202 the most frequent noun Russian-Czech pairs that occurred in our development corpus, which constitutes 60% of all noun tokens in our development corpus. This is clearly not very resource-light, but we do it to

⁴ This is obviously an approximation, since certain cognate words in Czech and Russian, even though have similar meanings and morphological properties, do not have the same distributional behavior. For example, the word *zivot* means 'belly' in Russian, while *život* means 'life' in Czech; or *krasnij* means 'red' in Russian, while *krasný* means 'nice' in Czech. Yet, in the former case both words are masculine nouns, and in the latter case both are adjectives.

find out if cognates have any potential. We limit ourselves to nouns due to the reasons outlined above. We used these manual translations for transferring the information about the distribution of Czech words into Russian. In order to do that we normalize and project the tag-frequencies of Czech word into its Russian translation in the case their tags match. The rest of the tags offered by the Russian morphological analyzer for that particular word are redistributed evenly again. For example if *cognate_{czech}* appears with *tag₁* 30 times in the Czech corpus, with *tag₂* 100 times and with *tag₃* 50 times, after the normalization, the distribution is *tag₁ 17, tag₂ 56, tag₃ 27*. If the corresponding Russian word is analyzed by the morphological analyzer as either *tag₁*, or *tag₂*, *tag₄*, *tag₅*, then the new distribution for *ruword* is *tag₁ 17, tag₂ 27, tag₄ 28, tag₅ 28*. With this naive procedure, the relative reduction in error rate is 16.1% on nouns, and 11.2% overall – see columns 7, for the detailed information.

Discussion. In our development corpus there are 363 noun tokens, 290 noun types. We are using 202 cognate/translation pairs (types) (= 273 tokens), which means if all these pairs did the expected job, the overall tagging performance on nouns would be (at least) 75.2% (i.e. we would improve the performance on nouns (which is 65.8% without the cognates) by (at least) 9.4%, but in fact we improve only by 5.5%. One of the problems that we have noticed by analyzing the errors is that about half of the Czech manual cognates are not actually found in the 25% most frequent Czech words, which means that we might have been too restrictive by limiting ourselves to the most frequently used words. Nevertheless, even such a naive approach suggests that it is worthwhile to explore this avenue.

Automatic Cognates. In reality, we have no Czech-Russian translations and we do not work with a parallel corpus. In the absence of this knowledge, we automatically identify cognates, using the (normalized by word length) edit distance algorithm. We assume that in a development of a language, vowel changes are more common and less regular than changes of consonants. So, rather than treating all string edits as equal, the operations on vowels have lower costs than on consonants. Yarowsky et al. (2000) use a synchronic version of these assumptions for inflection. This does not require language-intense resources and is general enough to apply to any language we want to work with. In addition, to obtain a more sensitive measure, costs are refined based on phonetic-orthographic regularities, e.g. replacing an ‘h’ with ‘g’ (as in the Czech ‘kniha’ (‘book’) and Russian ‘kniga’ (‘book’) is less costly than replacing ‘m’ with, say ‘sh’. However, we do not want to do a detailed contrastive morpho-phonological analysis, since we want our system to be portable to other languages. So, some facts from a simple grammar reference book should be enough.

Once we identify Czech-Russian cognate pairs automatically, we use the same approach as in the case of manual translations described above. In the case, several cognate candidate pairs have the same edit cost, one of them is selected randomly. Column 8 summarizes the performance of the tagger that uses 149 automatically derived cognates. The cognates we are able to extract by this

method help a little in some cases, but the pattern is not as clear as we would like. It looks as if the cognate detector needs further work.

Clearly, the upper-bound on emissions is unreachable, since not for all words in the Russian data there are corresponding Czech words, since even true Russian-Czech cognate pairs might not correspond in their morpho-syntactic behavior. For instance, the words “tema”, borrowed from Greek, exists both in Russian and Czech, but in Russian it is feminine, while in Czech it is neuter; moreover, there are definitely false cognates in the two languages, which might mislead the transfer from Czech into Russian (e.g. *matka*: ‘uterus’ (Russian) vs. ‘mother’ (Czech)). Finally, our cognate detector is not 100% precise.

6 Discussion and Ongoing Work

This work aims to explore the portability of linguistic knowledge from one language to another. The upper-bounds on TnT transitions and emissions suggest that given we utilize the linguistic knowledge about Czech, Polish and Russian effectively, we can obtain a rather good performance of the tagger. What we showed about Czech, Polish, and Russian surprised us. The model that is trained on a mixture of the two languages, Czech and Polish, outperforms models which were trained on these languages individually. We realize that this is due to the fact that Polish and Czech have complementary Russian-like properties and the mixture of the two creates more Russian-like training data. The fact that the hybrid model outperforms the interlingua model on nouns, where the combination was done using our linguistic intuition about the gender and number assignment in Czech and Polish, is a strong motivation for exploring and exploiting further the linguistic knowledge about the source and the target languages for more accurate tagging.

Our results suggest that the transfer is possible. The system we have developed uses comparable corpora, as opposed to parallel corpora, which makes it very suitable for languages where parallel corpora is not easy to find.

In our ongoing work we are developing an algorithm which will detect cognate stems and generate word forms using the Czech/Russian morphologies. The identification of cognate stems should give more reliable cognate classes, but the next challenge is to map the generated Russian forms into Czech.

Finally, we are extending our work to other languages. We are currently running experiments with Portuguese and Spanish.

Even though the overall performance of our system is not yet comparable to the tagging standard, say, for English, the accuracy of the tagger on the SubPOS position, which is comparable to the Penn Tree bank tagset (32 values) is close to 93%. For many applications this information is useful on its own.

Acknowledgements

We thank Adam Przepiórkowski and Lukasz Debowski for letting us use a fragment of the IPI PAN corpus, as well as for the help with the tagset analysis.

References

1. Agirre E., Atutxa A., Gojenola K., Sarasola K. (2004) Exploring Portability of syntactic information from English to Basque. In Proceedings of LREC 2004, Lisbon, Portugal.
2. Bémová A., Hajič J., Hladká B., Panevová J. (1999). Morphological and Syntactic Tagging of the Prague Dependency Treebank. In Proceedings of ATALA Workshop, Paris, France. pp. 21–29.
3. Brants T. (2000) TnT — A Statistical Part-of-Speech Tagger. Proceedings of ANLP-NAACL. pp. 224–231.
4. Hajic J. (2000) Morphological Tagging: Data vs. Dictionaries. In Proceedings of ANLP-NAACL Conference, pp. 94-101, Seattle, WA, USA.
5. Hana J. (2005) Knowledge and labor light morphological analysis of Czech and Russian. Ms. Linguistic Department. The Ohio State University,
6. Hana J., Feldman A., Brew C. (2004) A Resource-light Approach to Russian Morphology: Tagging Russian using Czech resources. In Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp.222–229
7. Hwa R., Resnik P., Weinberg A., Cabezas C., Kolak O. (2004) Bootstrapping Parsers via Syntactic Projection across Parallel Texts. *Natural Language Engineering* 1 (1):1-15.
8. Marcus M., Santorine B., Marcinkiewicz M.A. (1993). Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics* 19(2), 313-330.
9. Przepiórkowski A., (2004). The IPI PAN Corpus: Preliminary version. IPI PAN, Warszawa.
10. Yarowsky D., Wicentowski R. (2000). Minimally Supervised Morphological Analysis by Multimodal Alignment. In Proceedings of the 38th Meeting of the Association for Computational Linguistics. pp. 208-216.
11. Yarowsky D., Ngai G. (2001) Inducing Multilingual POS Taggers and NP Brackets via Robust Projection Across Aligned Corpora. In Proceedings of NAACL-2001. pp. 200-207.
12. Yarowsky D., Ngai G., Wicentowski R. (2001) Inducing Multilingual Text Analysis Tools via Robust Projection across Aligned Corpora. In Proceedings of HLT 2001, First International Conference on Human Language Technology Research.

Sentence Segmentation Model to Improve Tree Annotation Tool

So-Young Park¹, Dongha Shin¹, and Ui-Sung Song²

¹ College of Computer Software & Media Technology, SangMyung University,
7 Hongji-dong, Jongno-gu, SEOUL, 110-743, Korea
ssoya@smu.ac.kr, dshin@smu.ac.kr

² Dept. of Computer Science & Engineering, Korea University,
5-ka 1, Anam-dong, Seongbuk-ku, SEOUL, 136-701, Korea
ussong@disys.korea.ac.kr

Abstract. In this paper, we propose a sentence segmentation model for a semi-automatic tree annotation tool using a parsing model. For the purpose of improving both parsing performance and parsing complexity without any modification of the parsing model, the tree annotation tool performs two-phase parsing for the intra-structure of each segment and the inter-structure of the segments after segmenting a sentence. Experimental results show that it can reduce manual effort about 28.3% by the proposed sentence segmentation model because an annotator's intervention related to cancellation and reconstruction remarkably decrease.

1 Introduction

A treebank is a corpus annotated with syntactic information. In order to reduce manual effort for building a treebank by decreasing the frequency of the human annotators' intervention, several approaches have tried to assign an unambiguous partial syntactic structure to a segment of each sentence. The approaches [1, 2] utilize the reliable heuristic rules written by the grammarians. However, it is too difficult to modify the heuristic rules, and to change the features used for constructing the heuristic rules [3]. On the other hand, the approaches [3, 4] use the rules which are automatically extracted from an already built treebank. Nevertheless, they place a limit on the manual effort reduction and the annotating efficiency improvement because the extracted rules are less credible than the heuristics.

In this paper, we propose a tree annotation tool using an automatic full parsing model for the purpose of shifting the responsibility of extracting the reliable syntactic rules to the parsing model. In order to improve both parsing performance and parsing complexity without any modification of the parsing model, it utilizes a sentence segmentation model so that it performs two-phase parsing for the intra-structure of each segment and the inter-structure of the segments after segmenting a sentence. Next, section 2 will describe the proposed sentence segmentation model for the tree annotation tool, and section 3 shows the experimental results. Finally, we conclude this paper in section 4.

2 Segmentation Model to Improve Tree Annotation Tool

As shown in Fig. 1, the tree annotation tool is composed of *sentence segmentation* segmenting a long sentence, *tree annotation for intra-structure* annotating the intra-structure for each segment, and *tree annotation for inter-structure* annotating the inter-structure for all segments. Fundamentally, each component automatically generates candidates such as segments or partial syntactic structures based on the sentence segmentation model or the parsing model, and then the annotator cancels the incorrect constituents of the candidates, and reconstructs the correct constituents.

$$\operatorname{argmax}_{t_{0n}} P(w_{1n}, t_{0n}) = \operatorname{argmax}_{t_{0n}} P(w_{1n}) \times P(t_{0n}|w_{1n}) \tag{1}$$

$$= \operatorname{argmax}_{t_{0n}} P(t_0|w_{1n}) \times \prod_{i=1}^{n-1} P(t_i|w_{1n}, t_{i-1}) \times P(t_n|w_{1n}, t_{n-1}) \tag{2}$$

$$\approx \operatorname{argmax}_{t_{0n}} \prod_{i=1}^{n-1} P(t_i|w_{i-1}, w_i, w_{i+1}, w_{i+2}) \tag{3}$$

The proposed sentence segmentation model estimates the probabilities of generating the segment-tagged sentence by using the equation (3). The equation (1) is rewritten as the equation (2) based on the chain rule and the fact that the probability of generating the given sentence is a constant. The equation (2) is replaced by the equation (3) according to the independence assumption and the fact that t_0 and t_n are always “(” and “)”. In the equations, w_i indicates the i -th part-of-speech tagged word in a sentence, and t_i indicates the i -th segment tag between two words w_i and w_{i+1} where t_i is either null or “(”.

(김 씨는) (어머니가 강가에 나와) (아들을 애타게 찾았다는) (사실을) (최근에야 전해 들었다)
 $w_3 \quad w_4 \quad t_4 \quad w_5 \quad w_6$

For example, the first segment tag t_1 is “(” while the second segment tag t_2 is null in the above sentence. The probability $P(t_4|w_3, w_4, w_5, w_6)$ is estimated for the fourth segment tag. Its context includes the lexical items and their part-of-speech tags of two previous words w_3, w_4 and two next words w_5, w_6 .

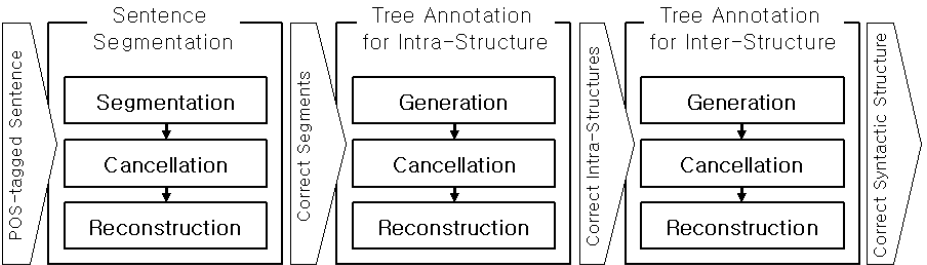


Fig. 1. Tree Annotation Tool

3 Experiments

In order to examine the effect of the proposed segmentation model, the tree annotation tool integrated with a parsing model [5] (F-measure 86.83%) is evaluated on 3,108 Korean sentences which never have been used for training the sentence segmentation model and the parsing model in a Korean treebank [1]. *#Cancellations* indicates the number of incorrect constituents cancelled by the annotator. *#Reconstructions* indicates the number of constituents reconstructed by the annotator. *#Interventions* includes *#Cancellations* and *#Reconstructions*. In this experiment, we compare the annotator’s intervention of the four methods: the fully manual tree annotation tool (*only human*), the tree annotation tool using the parsing model (*no segmentation*), the tree annotation tool using the parsing model with the manual segmentation (*manual segmentation*), and the tree annotation tool using the parsing model with the proposed segmentation model (*proposed*).

The left part of Figure 2 shows the performance of the proposed sentence segmentation model. In this figure, precision indicates the ratio of correct candidate segment tags “()” from the candidate segment tags generated by the proposed model while recall indicates the ratio of correct candidate segment tags from the all correct segment tags “()” excluding the null tags. On the other hand, “1:pos” indicates $P(t_i|w_i^*, w_{i+1}^*)$ while “2:pos,lex” indicates $P(t_i|w_{i-1}, w_i, w_{i+1}, w_{i+2})$ where w_i indicates the i-th part-of-speech tagged word and w_i^* indicates its part-of-speech tag. Roughly, the precisions are high while the recalls are low because of data sparseness problem. Although the more information makes precision higher, it does not cause *#Interventions* lower on account of too low recall. Nevertheless, the figure indicates that *#Interventions* of every segmenter is lower than *#Interventions* of *manual segmentation* which is 1,840.

As shown in the right part of Figure 2, *only human* does not need any manual effort related to segmentation and cancellation but it requires too expensive reconstruction cost. *No segmentation* does not require manual effort related to segmentation. This figure shows that the parsing model can reduce manual effort by roughly 50% although the parsing model generates some incorrect constituents. Furthermore, the sentence segmentation model also reduces the annotator’s intervention by about 28.3% as compared with *No segmentation*.

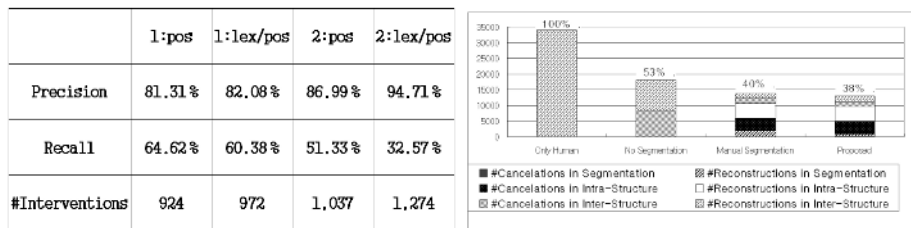


Fig. 2. Experimental Results

4 Conclusion

In this paper, we propose a sentence segmentation model in order to improve a semi-automatic tree annotation tool using a parsing model. The proposed tree annotation tool has the following characteristics. First, it can reduce manual effort to build a treebank. Experimental results show that it can improve approximately 62.0% and 28.3% as compared with the tree annotation tool without a parsing model and the tree annotation tool without a sentence segmentation model. Second, it can prevent the initial syntactic errors of a word or a phrase from propagating to the whole syntactic structure without any modification of the parsing model because the annotator can correct errors in the sentence segmentation step and the tree-annotation for the intra-structure step. Third, it can shift the responsibility of extracting the reliable syntactic rules to the parsing model. For future works, we will try to develop an better sentence segmentation model to minimize manual effort.

Acknowledgements

This work was supported by Ministry of Education and Human Resources Development through Embedded Software Open Education Resource Center(ESC) at Sangmyung University.

References

1. Choi, Ki-Sun. 2001. "KAIST Language Resources ver. 2001." *The Result of Core Software Project from Ministry of Science and Technology*, <http://kibs.kaist.ac.kr>. (written in Korean)
2. Mitchell, P. Marcus, B. Santorini, and M. A. Marcinkiewicz. 1993. Building a Large Annotated Corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313-330.
3. Joon-Ho Lim, So-Young Park, Yong-Jae Kwak, Hae-Chang Rim. 2004. A Semi-automatic Tree Annotating Workbench for Building a Korean Treebank. *Lecture Note in Computer Science*, 2945:253-257.
4. Yong-Jae Kwak, Young-Sook Hwang, Hoo-Jung Chung, So-Young Park, Hae-Chang Rim. 2001. FIDELITY: A Framework for Context-Sensitive Grammar Development. *In Proceedings of International Conference on Computer Processing of Oriental Languages*, 305-308.
5. So-Young Park, Yong-Jae Kwak, Joon-Ho Lim, Hae-Chang Rim, and Soo-Hong Kim. 2004. Partially Lexicalized Parsing Model Utilizing Rich Features. *In Proceedings of the 8th International Conference on Spoken Language Processing*, 3:2201-2204.

Markov Cluster Shortest Path Founded Upon the Alibi-Breaking Algorithm

Jaeyoung Jung, Maki Miyake, and Hiroyuki Akama

Tokyo Institute of Technology, Department of Human System Science,
2-12-1 O-okayama, Meguro-ku, Tokyo, 152-8552 Japan
{catherina, mmiyake, akama}@dp.hum.titech.ac.jp

Abstract. In this paper, we propose a new variant of the breadth-first shortest path search called Markov Cluster Shortest Path (MCSP). This is applied to the associative semantic network to show us the flow of association between two very different concepts, by providing the shortest path of them. MCSP is obtained from the virtual adjacency matrix of the hard clusters taken as vertices after MCL process. Since each hard cluster grouped by concepts as a result of MCL has no overlap with others, we propose a method called Alibi-breaking algorithm, which calculates the adjacency matrix of them in a way of collecting their past overlapping information by tracing back to the on-going MCL loops. The comparison is made between MCSP and the ordinary shortest paths to know the difference in quality.

1 Introduction

In the leading network science, the graph structure and scale problem has risen as a renewed matter of concern. The same thing is true of the corpus or cognitive linguistics that allows us to see the world of language as a large-scale graph of words. If a word is associated in a certain sense to the other, it is told that they are connected with each other and all the words taken in this way as nodes (vertices) are linked together by a set of edges corresponding here with the lexical association. In this structure, the shortest path between two random words or concepts represents their distance in semantic networks. Steyvers et al. (2003) showed that large-scale word association data possess a small-world structure characterized by the combination of highly clustered neighborhoods and a short average path length. According to them, the average shortest path (SP) length between any two words was 3.03 in the Undirected Associative Network of Nelson et al, 4.26 in their Directed Associative Network, 5.43 in Roget's thesaurus and 10.61 in WordNet.

It also held true in Ishizaki Associative Concepts Dictionary of Japanese Words (in abbreviation, ACD), which offered us lexical association data for graph manipulation. Its average shortest path (SP) length was 3.442 in the 43 word pairs randomly chosen from it. Despite such low values, however, it took a relatively long time (according to our experiment mentioned below, more than 1 minute on average) by the usual searching method that automatically traces the shortest routes based on the word node connectivity in semantic networks. This kind of word-to-word distance measure not only takes time, but might restrict a way to present any other possible semantic

structures of networks. Accordingly to make the best use of the small-world feature of the semantic network, we propose a new shortest path detection algorithm by using a customized strategy of graph clustering: *Recurrent Markov Cluster Algorithm* (RMCL).

2 Markov Cluster Process

The original MCL algorithm proposed by Van Dongen (2000) can be formulized as the alternation of two steps--expansion and inflation-- to reach the convergence of a stochastic matrix through which a whole graph is subdivided into the clusters without any overlaps one another.

In this work, we used GridMathematica to write the original MCL program and applied it to ACD. The dictionary, made as a result of the free association by ten participants, is composed of 33,018 words and 240,093 pairs of them. But to make a significant and well-arranged semantic network, we selected 9,373 critical words from it by removing the rarest words. And then MCL process was applied to a $9,373 \times 9,373$ adjacency matrix which was calculated based on 187,113 pairs of the critical words. Finally it made a nearly-idempotent stochastic matrix at the 16th cluster stage to allow us to gain 1,408 hard clusters corresponding with as many concepts sustained by a series of similar words.

3 Recurrent MCL and Markov Cluster Shortest Path

The final concept clusters generated by MCL have no common word node. Since they don't expose their adjacent relations at the final stage of convergence, we need to find out their connections to search for the shortest path based on them. For this, we thought out a way to restore the *virtual* connections of them by tracing back to the previous cluster stages before the convergence. In this procedure, each concept cluster is newly considered as a vertex, or meta-vertex including word nodes (later, each of them is identified by the representative word node with the largest degree value). This back-tracing search collects the *evidence* that the final clusters *have had* any common word nodes somewhere in the previous cluster stages, and then based on it reconnects the final clusters. This procedure may allow us to call it "alibi-breaking algorithm", in a sense that it takes up evidence of the past "*implication*" to show the connections of the final clusters.

The alibi-breaking algorithm is shown below, which is the core part of our Recurrent Markov Cluster Algorithm (RMCL). ClusterStagesList means a set of the clustering results still in progress of MCL loops, except for the ClusterStage_{*k*} that represents the final converged clusters. First OverlappingNodes(ClusterStage_{*i*}) looks for all the multiply-attributed nodes (abbreviated as *oln*(*p*)) in each of the on-going ClusterStage_{*i*}. And then in OverlappingClusters(*oln*(*p*)), the set of *olc*(*p*), the union of all the soft clusters including *oln*(*p*) at ClusterStage_{*i*} is generated. For each *oln*(*p*), all the past co-occurring nodes in *olc*(*p*) (we call them "conodes") are enumerated, and by searching for the clusters containing conodes(*p*) at ClusterStage_{*k*} we newly settle adjacency relationships between the final clusters.

Formula of Alibi-breaking Algorithm

```

ClusterStagesList
= {ClusterStage1, ClusterStage2, ..., ClusterStagek};
OverlappingNodes(ClusterStagei)
= {oln(1), oln(2), ..., oln(p), ..., oln(m)};
OverlappingClusters(oln(p))
= olc(p) =  $\bigcup_j (\text{ClusterStage}_i(j) \supset \text{oln}(p))$ ;
For each oln(p) {conodes(p) = olc(p)  $\cap$   $\neg$  {oln(p)}
= {con(1), con(2), ..., con(q), ..., con(n)};
MakeAdjacency(ClusterStagek(j)  $\supset$  conodes(p)); end.

```

The breadth-first routing is a method to trace the shortest path in a traversal search. Namely, from a starting node all the adjacent nodes to it are searched by constructing spanning trees from connected graphs. This way is adopted here since we are interested in representing the coordination of a series of *paradigms* (sets of similar words) instead of the straight-forward chain of every single word. Markov Cluster Shortest Path (MCSP) uses the breadth-first traversal strategy, yet unlike the ordinary shortest paths, it is applied to the adjacency matrix of the concept cluster nodes, not of word nodes.

4 Results

Using RMCL and the ordinary search, we computed the shortest paths in the 43 word pairs that were randomly chosen from ACD as a sample set at the size of 1.0e-6 of population. The results were distinguished into three types. a) Type 1 of Markov Cluster shortest path (MCSP1), which is the breadth-first shortest path detected in the graph of 1,408 clusters obtained by MCL process. Its results are given under the form of cluster-to-cluster flows. b) Type 2 of Markov Cluster shortest path (MCSP2), of which computation is the same as in MCSP1. But in MCSP2, the specific paths between words are searched from the result clusters of MCSP1. c) Ordinary breadth-first shortest path (SP), which is searched in the graph of 9,373 words. The core function of breadth-first research was identical through all the three types.

Consequently we could see a tendency that SP would permit us to grasp the relatively precise denotative semantic relationships between any two words, whereas MCSP would show us the large sphere of meaning that is joined together by a free association using the extensive connotative uses of the words. As for the quantitative data, there was a highly significant difference in average time for calculation (on Windows XP, 2.01GHz, by Mathematica5.0). (5.071 sec, 2.342 sec and 84.487 sec on average in a), b) and c) respectively, $F_{(2,126)}=16.066$, $p<.001$). In this respect MCSP1 and MCSP2 turned out to be much more effective than SP when implemented in practical systems.

The average lengths of paths were a) 1.767, b) 17.277 and c) 3.442, which means that even if the results of MCSP2 are by nature inevitably approximate and redundant, these characteristics might have under some circumstances positive effects due to the sufficient average length of shortest paths, by permitting us to complement the shortage of information resulted from the "small world" structures of semantic networks.

Furthermore according to the subjective evaluation of three types (three participants evaluated the results of each type into five scales, in the three aspects of *natural to understand*, *appropriate in size* and *inspirational to write a story*), the results of SP were more favorable for natural precision than inspirational effects especially when it comes to the word pairs with high similarity (subjectively selected by a specialist of linguistics and language education), whereas such difference was not shown in the results of MCSP2 (Figure 1).

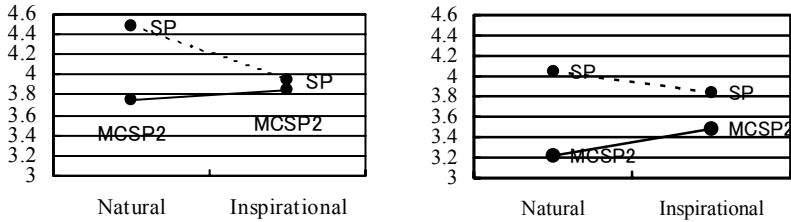


Fig. 1. Evaluation of the shortest paths of word association data by MCSP2 and SP (5-scale in two aspects here). The left graph shows the average score of the word pairs with high similarity and the right one for the all pairs.

5 Conclusion and Future Works

The concept clusters generated by the MCL process and the Markov Cluster shortest paths based on the alibi-breaking algorithm allowed us to see the detailed small-world structures in the semantic network. They can be useful information to language learners in the aspect of providing extensive, associative and connotative relations between words. We will develop by using this technique a web-based composition support system and would like to evaluate it from the viewpoints of educational technology and cognitive science.

References

1. Van Dongen, S.: Graph Clustering by Flow Simulation. PhD thesis, University of Utrecht (2000) <http://www.library.uu.nl/digiarchief/dip/diss/1895620/inhoud.htm>
2. Steyvers, M., Tenenbaum, J.: The Large Scale Structure of Semantic Networks: Statistical Analyses and a Model of Semantic Growth, *Cognitive Science*, 29(1) (2005) 41-78
3. Okamoto, J., & Ishizaki, S.: Associative Concept Dictionary and its Comparison Electronic Concept Dictionaries (2001) <http://afnlp.org/pacling2001/pdf/okamoto.pdf>

Unsupervised Learning of Verb Argument Structures

Thiago Alexandre Salgueiro Pardo¹, Daniel Marcu², and
Maria das Graças Volpe Nunes¹

¹ Núcleo Interinstitucional de Lingüística Computacional (NILC),
CP 668 – ICMC-USP, 13.560-970 São Carlos, SP, Brasil
<http://www.nilc.icmc.usp.br>

² Information Sciences Institute (ISI),
4676 Admiralty Way, Suite 1001, Marina del Rey, CA 90292
<http://www.isi.edu>
tasparado@gmail.com, marcu@isi.edu, gracan@icmc.usp.br

Abstract. We present a statistical generative model for unsupervised learning of verb argument structures. The model was used to automatically induce the argument structures for the 1,500 most frequent verbs of English. In an evaluation carried out for a representative sample of verbs, more than 90% of the induced argument structures were judged correct by human subjects. The induced structures also overlap significantly with those in PropBank, exhibiting some correct patterns of usage that are not present in this manually developed semantic resource.

1 Introduction

Inspired by the impact that the availability of Penn Treebank (Marcus et al., 1993; Marcus, 1994) had on syntactic parsing, several efforts have recently focused on the creation of semantically annotated resources. The annotation of verb arguments, their roles, and preferential linguistic behaviors represents a significant fraction of these efforts. The annotations that we are focusing on here pertain to the argument structures of a verb. In particular, we look for the words/concepts that constitute the arguments required by the verbs when these are used in real sentences.

The determination of verb argument structures has been shown to be a hard task for several reasons. Little agreement exists with respect to (a) how many canonical usages a verb has, (b) which arguments are really required by a verb and (c) in what order they may be realized in sentences. For instance, examples (1)-(3) show some patterns of usage for the verb *bought*.

- (1) He had bought them gifts.
- (2) He bought it 40 years ago.
- (3) About 8 million home water heaters are bought each year.

Intuitively, one can induce from these examples that the object/thing that is bought (“gifts” in sentence (1), “it” in sentence (2), and “about 8 million home water heaters” in sentence (3)) is more likely to be a required argument for the verb than the time when the buying event occurred, since the thing bought is specified in all the cases and time is not. The examples also show the variation in the order in which the arguments are realized: in (1) and (2), the thing bought is stated after the verb; in (3), it is

stated before the verb. Ideally, all the possibilities should be acknowledged in the semantic specification of verbs.

There is also little agreement with respect to how the arguments should be labeled. Figures 1, 2, and 3 show the information associated with the verb “buy” in FrameNet (Baker et al., 1998), VerbNet (Kipper et al., 2000), and PropBank (Kingsbury and Palmer, 2002), respectively. These are large scale projects that aim at developing semantic information repositories for verbs, mainly. FrameNet shows the pattern in which a verb occurs and provides representative examples; the resource also organizes the verbs into a hierarchy that implicitly encodes how verb structures can be inherited from ancestors. VerbNet shows the thematic roles the verb asks for, their semantic features, and possible subcategorization frames; VerbNet also provides examples for each categorization frame. PropBank makes explicit the argument roles of a verb, the possible subcategorization frames, and provides examples for each one. PropBank also distinguishes between obligatory verb arguments and optional ones, i.e., adjuncts. The adjunct in Figure 3, for example, is the ArgM-MNR argument (i.e., argument of manner). By inspecting Figures 1-3, it is not difficult to see that little agreement exist with respect to the ontological status of argument labels. What is ARG1 after all? *Goods*? A *Theme*? Or the *Thing Bought*? What is the most appropriate level of abstraction for argument labels?

<p>Typical pattern: BUYER buys GOODS from SELLER for MONEY</p> <p>Example: Abby bought a car from Robin for \$5,000.</p>
--

<p>Thematic Roles: Agent[+animate OR organization], Asset[-location -region], Beneficiary[+animate OR +organization], Source[+concrete], Theme[]</p> <p>Frames: Basic Transitive: "Carmen bought a dress" (Agent, Theme)</p>

Fig. 1. FrameNet annotation for the verb *buy*

Fig. 2. VerbNet annotation for the verb *buy*

<p>Roles: Arg0:buyer Arg1:thing bought Arg2:seller Arg3:price paid Arg4:benefactive</p> <p>Examples: Intransitive: Consumers who buy at this level are more educated than they were. Arg0: Consumers REL: buy ArgM-MNR: at this level</p>
--

Fig. 3. PropBank annotation for the verb *buy*

Given the difficulty of the task, it is not surprising that FrameNet, VerbNet, and PropBank have been manually built. However, some research efforts have targeted the problem of automatic (Brent, 1991; Resnik, 1992; Grishman and Sterling, 1992; Manning, 1993; Framis, 1994; Briscoe and Carroll, 1997; Rooth et al., 1999; McCarthy, 2000; Sarkar and Zeman, 2000; Merlo and Stevenson, 2001; Sarkar and Tripasai, 2002; Gildea, 2002) and semi-automatic (Korhonen, 2002; Green et al., 2004; Gomez, 2004) verb argument structures induction (including the related task of verb subcategorization frames learning). In general, these approaches rely on syntactic information and/or subcategorization dictionaries for identifying the arguments of a verb in a sentence, and/or assume as known the structure types in terms of number and order of arguments a verb can assume. The main goal in these approaches is to identify the lexemes that are most likely to fill a given verb argument slot. Some researchers (Grishman and Sterling, 1994; Framis, 1994; Lapata, 1999; Gomez, 2004) try to go beyond these lexemes and generalize the structures that are learned, by computing the similarity between the words occurring across similar structure instances or by using lexical resources such as WordNet and Levin (1993)'s verb classes. Most of these approaches implement a filtering step, in which inadequate learned structures are discarded on frequency-based grounds.

In this paper, we propose an alternative approach to the problem of determining verb argument structures. We present an unsupervised method for learning the argument structures, modeled over the noisy-channel framework, with the following characteristics:

- It does not assume that the number and order of arguments are known in advance. The argument structures are completely learned from naturally occurring texts.
- The argument structures that we learn are grounded in both lexemes and abstractions (named entities), with the most appropriate abstraction level being automatically determined.
- It ranks competing structures according to their probability.
- It makes use of simple tools, such as part of speech and named entity taggers, that are both widely available and easy to port across languages and domains.

In the rest of the paper, we first describe our statistical model and the algorithms we used to train it (Section 2). We introduce the training data (Section 3) and present a human-based evaluation for a representative sample of verb argument structures that we learn automatically (Section 4). We end with a discussion of the strengths and weaknesses of our model and future work (Section 5).

2 Our Approach

We couch our learning problem in a probabilistic noisy-channel framework. This framework has been widely used in statistical natural language processing¹. In this framework, one concocts a generative story that explains how data of interest comes into existence. For instance, Knight and Marcu (2002)'s generative story shows how

¹ For a more detailed discussion about the noisy-channel model in natural language processing tasks and its characteristics, see Marcu and Popescu (2005).

short sentences can be mapped into long sentences; Brown et al. (1990, 1993) show how sentences in English are probabilistically mapped into French sentences; Soricut and Brill (2004) show how answers can be mapped into questions. In our model (see Figure 4), the generative story explains how natural language sentences (S) are produced by generating first an abstract argument structure (A) and then mapping this structure into strings. Our generative story goes like this:

1. (a) The head (verb) of the argument structure is first chosen with probability $P(v)$. (b) The number of arguments the verb takes is chosen with probability $\text{narg}(\text{no_arg} \mid v)$. (c) Each argument is generated with probability $\text{arg}(\text{argument} \mid v)$. Each argument can be either an abstraction/concept (named entity in our case) or a word/lexeme.
2. Once the verb argument structure is generated, a probabilistic parameter $\phi(N \mid v)$ decides the number of extra words/concepts that are going to be eventually produced in the sentence.
3. Each extra word/concept is stochastically generated according to the distribution $\text{ew}(\text{word})$.
4. If the generative process produces concepts c (named entities), these are translated into words, with probability $t(\text{word} \mid c)$.

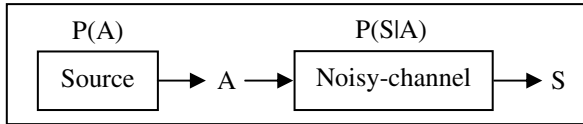


Fig. 4. A noisy-channel model for learning verb argument structures

For instance, the sentence *Santa has bought them gifts* can be generated by the following process.

1. (a) The head verb *bought* is first chosen with probability $P(\textit{bought})$. (b) The verb is associated with 3 arguments with probability $\text{narg}(3 \mid \textit{bought})$, which are *PERSON1*, *gifts*, and *PERSON2* (c) with probabilities $\text{arg}(\textit{PERSON1} \mid \textit{bought})$, $\text{arg}(\textit{gifts} \mid \textit{bought})$ and $\text{arg}(\textit{PERSON2} \mid \textit{bought})$, respectively. At the end of this sequence, we have available the following verb argument structure: $\textit{bought}(\textit{PERSON1}, \textit{gifts}, \textit{PERSON2})$.
2. One extra word is added with probability $\phi(1 \mid \textit{bought})$
3. which turns out to be the word *has* with probability $\text{ew}(\textit{has})$.
4. The named entities are translated into words: *PERSON1* into *Santa* and *PERSON2* into *them*, with probabilities $t(\textit{Santa} \mid \textit{PERSON1})$ and $t(\textit{them} \mid \textit{PERSON2})$.

In order to make the training of our model tractable, we make some simplifying assumptions. That is, we assume that the subsequence corresponding to steps 1.a-c happens in one shot: an entire event is generated stochastically with probability $\text{event}(\text{verb}(\text{arg}_1, \dots, \text{arg}_n))$. Since named entity taggers work at levels of accuracy above 90%, we also assume that it is not necessary to translate concepts into words as part of the generative process – we can pre-tag the sentences used for training with

named entity tags and learn argument structures that include such entities directly. From a generative story perspective, this means that we no longer need Step 4 to model the translation of entities into words. Mathematically, these choices simplify our model tremendously. According to the resulting model, the probability of a sentence S is thus given by the following formula:

$$P(S) = \prod_A P(S, A) = \prod_A P(A) \times \prod_A P(S|A) = \prod_A \text{event}(A) \times \prod_A \text{phi}(N | \text{verb}) \times \prod_{i=1}^N \text{ew}(w_i)$$

where A is a possible argument structure, N is the number of extra words/concepts that are generated, and w_i is the i^{th} extra word being generated. In this view, the probability of the sentence $P(\text{Santa/PERSON1 has bought them/PERSON2 gifts})$ is $\text{event}(\text{bought}(\text{PERSON1}, \text{gifts}, \text{PERSON2})) \times \text{phi}(1 | \text{bought}) \times \text{ew}(\text{has})$.

We use the Expectation-Maximization (EM) algorithm (Dempster et al., 1977) to estimate the parameters of the model (which are uniformly initialized). To restrict the search space and make the training feasible, we assume that a verb can have at most 3 arguments and that arguments can be only open class words (verbs, adjectives, adverbs and nouns – including pronouns). In order to impose these restrictions, we pre-tag the data with a part of speech tagger (Ratnaparki, 1996). Now, we are capable of doing full EM training on our data, as the number of hidden alignments/argument structures that we have to consider for every sentence is reasonable. For example, Figure 5 shows all possible hidden argument structures for the sentence *He has bought them gifts*. The arrows leave from the verb and point to the arguments. The words not pointed to by any arrow are the extra words. For simplicity, the named entity and part of speech tags are not shown.

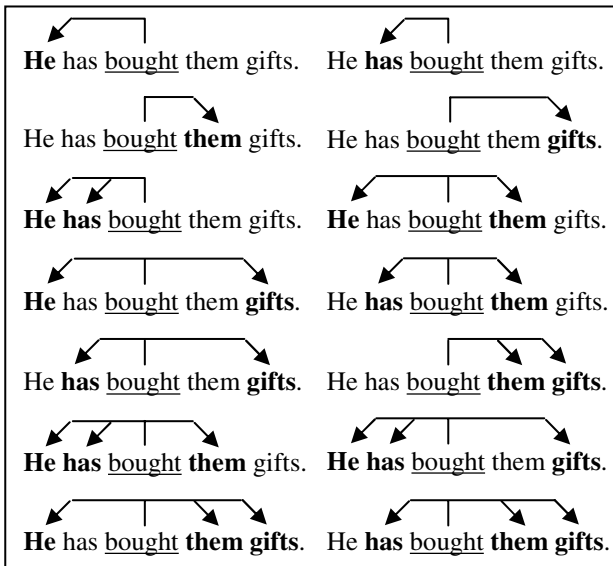


Fig. 5. Possible argument structures in the sentence *He has bought them gifts*

Because we use EM, low probabilities are naturally assigned to uncommon or, hopefully, inadequate argument structures. Therefore, it is not necessary to filter our results in an ad-hoc manner.

3 Data Preparation

From TREC'2002 data collection (Voorhees and Buckland, 2002), we have selected the 1,500 most frequent verbs for training our model. We extracted from TREC'2002 corpus all sentences containing occurrences of these verbs. Since our model is not ready to properly cope with very long sentences (especially those that contain complex verb sentential complements), we filtered out the sentences longer than 10 words. We tagged every sentence using the BBN Identifinder named entity tagger (Bikel et al., 1999) and Ratnaparki's (1996) part of speech tagger. On average, we ended up with nearly 1,400 sentences per verb and a total of 14 million words in the collected corpus.

The use of a named entity tagger is not necessary for our model to work; its use, however, enable the model to learn more general argument structures. If entities are not used, the structures we learn are completely lexicalized; if a named entity tagger is used, we expect to learn both lexicalized and generalized verb argument structures. As expected, named entities overcome some of the data sparseness problems and yield argument structures that are more likely than the fully lexicalized ones. It is worth noting that the most appropriate level of abstraction for arguments (lexemes vs. named entities) is learned automatically by the EM algorithm.

Using WordNet concepts for representing the abstraction level is also possible, like many works do. We chose named entities because of the following advantages: the set of entities is more intuitive and small, making the learning process more effective; during tagging, the correct sense of the word is determined.

Other arrangements we did to our data include: all numbers were replaced by the general entity *number*; excepting *it*, *they* and *them*, all personal pronouns were replaced by the entity *person*; *it*, *they* and *them* were considered to be both *person* and the generic entity *thing* (that can be anything but *person*), since they can refer to anything.

Figure 6 shows a sample of our learning data, with entities in bold.

about/IN **money**/NN home/NN water/NN heaters/NNS are/VBP bought/VBN
each/DT year/NN

organization/NNP bought/VBD **organization**/NN from/IN **organization**/NN
last/JJ year/NN

thing/PRP bought/VBD the/DT outstanding/JJ shares/NNS on/IN **date**/NNP
the/DT cafeteria/NN bought/VBD extra/JJ plates/NNS

Fig. 6. Data sample

It is easy to note that some sentences are completely lexicalized, without entities (e.g., the last sentence), while others have several entities. In the first sentence, one can also note an error introduced by BBN *IdentiFinder*: *8 million* was misclassified as **money**. Such errors should be naturally discarded by EM as valid arguments, since they are not frequent in our corpus.

4 Evaluation and Analysis

To assess the correctness of the verb argument structures we learned automatically, we carried out two experiments with a randomly selected sample of 20 verbs, assuring that it includes low (i.e., rare), medium and high-frequency (i.e., common) verbs in our corpus. The first row in Table 1 shows the selected verbs: “hook”, “spin” and “yell” are examples of low-frequency verbs; “raise” and “spend” are examples of medium-frequency verbs; “buy”, “die” and “help” are examples of high-frequency verbs.

We compare our results to the results obtained with a baseline algorithm. This algorithm uses a frequency-based method to produce the argument structures: it computes all possible structures that the sentences in our corpus can have, in the same way we show in the example in Figure 5, and ranks the produced argument structures according to their frequencies. Like in our model, the part-of-speech tags and entities are also taken into consideration, i.e., the baseline algorithm is informed about which words can be arguments and is able to learn generalized structures. As will be noted, this baseline algorithm turns out to be very strong.

For the experiments we carried out, for each verb, we kept only the argument structures learned with probabilities above a threshold of 10^{-3} in order to make the evaluation feasible (for some verbs, our model learns hundreds of possible argument structures). Having this, for each verb, we took the same number of structures produced by the baseline algorithm, selecting the most probable ones. This way, we guarantee that the evaluation is fair.

In the first experiment, we wanted to verify how many correct/plausible argument structures were learned by our model in relation to all structures learned. This is a precision measure. We presented the argument structures to three judges (computational linguists) and asked them to independently judge their correctness/plausibility. Each argument structure could be classified as “correct”, “wrong” or “can’t tell” by each judge: it should be classified as “correct” if the judge could come up with a sentence from the structure; “wrong” in the case it is not possible to come up with a sentence; and “can’t tell” when it is not possible to know for sure.

In the second experiment, in order to verify the correspondence of the learned structures to the ones predicted by humans for the verbs, we compared our structures to the ones in PropBank. We computed how many structures in PropBank were learned by our model, observing the number of arguments and their types in each structure and the overall frame. This is, basically, a recall measure. It is important to note that precision was not evaluated in relation to PropBank structures because PropBank is not complete and, as will be discussed here, our model learns argument structures not predicted by this repository.

The same evaluation was carried out for the structures learned by the baseline algorithm.

The 2nd and 3rd columns in Table 1 show the number of sentences used for training our model for each verb and the number of argument structures considered in the experiments. The 4th column in Table 1 shows **Precision** (the average for the three judges) and **Recall** for each verb. In relation to precision, the annotation agreement between judges was high: the kappa statistic (Carletta, 1996) was 0.69. A kappa figure between 0.6 and 0.8 indicates high agreement. In average, our model achieved 93.7% precision and 73.5% recall, showing good results for low, medium and high-frequency verbs. The 5th column in the table shows the corresponding results for the structures produced by the baseline algorithm: on average, it achieved 81.4% precision and 59.2% recall. The baseline showed to be a strong one, but our model outperformed it. We suspect the good performance of the baseline is explained by the methodology we used to do data collection.

Table 1. Performance of verb argument structure induction algorithm

Verbs	Sentences	Structures	P & R (%)	
			<i>Our model</i>	<i>Baseline</i>
<i>abandon</i>	171	3	100, 50.0	100, 0
<i>aspire</i>	25	3	100, 100	100, 100
<i>avoid</i>	482	6	100, 100	100, 100
<i>buy</i>	2326	44	85.5, 70.0	75.6, 70.0
<i>cause</i>	1301	29	93.0, 100	63.1, 100
<i>collapse</i>	153	4	91.6, 75.0	66.6, 50.0
<i>die</i>	4334	70	85.2, 100	57.5, 100
<i>earn</i>	971	43	88.3, 75.0	76.7, 50.0
<i>expect</i>	2597	64	84.3, 100	70.8, 100
<i>fix</i>	270	18	86.9, 40.0	75.8, 20.0
<i>hate</i>	594	27	91.3, 100	71.5, 100
<i>help</i>	3706	54	89.4, 100	76.4, 100
<i>hook</i>	46	2	100, 33.3	100, 0
<i>issue</i>	955	10	100, 75.0	73.3, 50.0
<i>offer</i>	3071	41	95.8, 20.0	77.9, 20.0
<i>paint</i>	253	5	93.3, 33.3	100, 16.6
<i>raise</i>	1422	63	93.0, 83.3	66.6, 50.0
<i>spend</i>	1560	22	96.9, 100	77.2, 25.0
<i>spin</i>	111	4	100, 66.6	100, 33.3
<i>yell</i>	110	5	100, 50.0	100, 100
Avg.	1223	26	93.7, 73.5	81.4, 59.2

We computed the same results for the 10 and 20 most probable structures for each verb in order to verify how the consideration of more low-probability structures interfere in the performance of our model. Table 2 shows the results obtained. As expected, one can note that, as more argument structures we consider, precision decreases and recall increases.

Table 2. Performance of the algorithm for top-10, top-20 and all argument structures

Structures	Our model		Baseline	
	P (%)	R (%)	P (%)	R (%)
Top-10	95.2	63.1	86.6	47.5
Top-20	93.8	65.6	84.8	52.4
All	93.7	73.5	81.4	59.2

We investigated what led to both recall and precision problems. The recall problems, i.e., the inability of the algorithm to induce certain PropBank structures, is explained by the following reasons:

- Some of the argument types found in PropBank structures were not part of our training corpus, and, therefore, they were not learned.
- A few PropBank structures had more than three arguments; our model predicts at most 3 arguments. For instance, PropBank lists the sentence *John killed Mary with a pipe in the conservatory*, for which the words *John*, *Mary*, *pipe* and *conservatory* are arguments, while our model predicts structures with two entities of type *person* as arguments and a third argument being the instrument or the location of the event, but not both together.

There were two main reasons that explained our precision problems; they pertained to improper handling of adverbs and phrasal verbs. Most of times adverbs are adjuncts, instead of arguments, and, therefore, should not be included in argument structures. However, in some cases, the adverbs are too frequent, co-occurring a lot with some verbs, and look essential to the sentence meaning, like in *He asked rhetorically* and *He asked incredulously*. Corroborating this, PropBank includes adverbs in some argument structures. Phrasal verbs are also a nuisance to our model. For instance, from the sentence *He gave up*, the model learns that either *up* is a possible argument for *gave* or that *gave* asks for 1 argument only, ignoring the particle *up* completely.

For exemplifying the learned structures, Figure 7 shows the top 10 structures learned for the verb *buy* with their associated probabilities.

1	buy(organization,organization)	1.20e-01
2	buy(person,number)	8.44e-02
3	buy(person,thing)	7.10e-02
4	buy(organization,thing)	5.63e-02
5	buy(person,organization)	4.28e-02
6	buy(organization,person)	3.51e-02
7	buy(person,house)	1.54e-02
8	buy(person,thing,anyway)	1.54e-02
9	buy(money,money)	1.40e-02
10	buy(organization,organization,date)	8.63e-03

Fig. 7. Argument structures for the verb *buy*

It is worth noting the following:

- the 5th and 6th structures are very similar (in the former, a person buys an organization; in the latter, an organization is bought by a person);
- the 7th structure has a lexicalized item (*house*);
- in the 8th structure, there is an error caused by the inclusion of an adverb (*anyway*) in the structure (because it co-occurred enough times with the verb *buy* in the corpus to be learned by the EM algorithm);
- in the 9th structure, there is an error caused by the phrasal verb *buy down* (like in *dollar bought down the yen*) (because the system is not able to identify this as a phrasal verb and ignore it).

For several verbs, our model was able to learn senses and behaviors not listed in PropBank. For instance, for the verb *raise*, our model learned structures for the ‘growing’ sense of the verb (like in *Peter was raised in a big city*), which is not annotated in PropBank. Our model could also learn many possible behaviors (not listed in PropBank) for the verb *die*, for instance: (a) In *date*, *person* died; (b) *Person* died in *date*; (c) *Person* died in *date* in *location*; (d) *Person* died in *location* in *date*.

5 Conclusion

The experiments reported in this paper make explicit the strengths and weaknesses of our approach. On the positive side, our model is able to yield high accuracy verb argument structures with no annotation effort, using relatively simple language tools. Our model learns both abstract argument structures, which are grounded in named-entity types, and specific structures, which are grounded in the lexicon. Not only does our method find most of the verb argument structures that are already annotated in the PropBank, but it is also able to suggest structures that are not part of this resource.

On the negative side, our model is still not robust enough to properly handle phrasal verbs, adverbs and complex verb sentential complements. Like FrameNet and VerbNet, our model does not explicitly differentiate between obligatory verb arguments from adjuncts. According to our interests, in the way the model works, we also do not distinguish between active and passive constructions in the learned argument structures. However, this is a simple adaptation that could be done by simply distinguishing the sentences types.

A natural extension of this work consists in complementing the argument structures with more information, e.g., thematic roles and syntactic realization of the arguments. However, these extensions require the use of more sophisticated tools (like syntactic and semantic parsers) to identify the arguments roles and syntactic realizations, making the proposed model less language independent.

The usefulness of a repository of verb argument structures is unquestionable. It is easy to imagine how the learned structures can be used in a variety of natural language generation applications (summarization and machine translation, for example) to assess whether the generated outputs are consistent with a set of pre-learned structures. For instance, if such a system generates text that subsumes an inconsistent structure, that text is probably semantically ill-formed.

This paper presented a first investigation on the statistical modeling of argument structures learning in the noisy-channel framework. All the detected limitations and the improvement possibilities to the model should be investigated in future work, as well as the use of the learned argument structures in natural language applications. Other models with different generative stories should also be tested.

References

- Baker, C.F.; Fillmore, C.J.; Lowe, J.B. (1998). The Berkeley FrameNet project. In the *Proceedings of COLING/ACL*, pp. 86-90, Montreal.
- Bikel, D.M.; Schwartz, R.; Weischedel, R.M. (1999). An Algorithm that Learns What's in a Name. *Machine Learning* (Special Issue on NLP).
- Brent, M.R. (1991). Automatic acquisition of subcategorization frames from untagged text. In the *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics*, pp. 209-214, Berkeley, CA.
- Briscoe, T. and Carroll, J. (1997). Automatic extraction of subcategorization from corpora. In the *Proceedings of the 5th ANLP Conference*, pp. 356-363, Washington, D.C.
- Brown, P.; Cocke, J.; Della Pietra, S.; Della Pietra, V.; Jelinek, F.; Lafferty, J.; Mercer, R.; Roossin, P. (1990). A statistical approach to machine translation. *Computational Linguistics*, Vol. 16, N. 2, pp. 79-85.
- Brown, P.; Della Pietra, S.; Della Pietra, V.; Mercer, R. (1993). The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, Vol. 19, N. 2, pp. 263-311.
- Carletta, J. (1996). Assessing Agreement on Classification Tasks: The Kappa Statistic. *Computational Linguistics*, Vol. 22, N. 2, pp. 249-254.
- Dempster, A.P.; Laird N.M.; Rubin, D.B. (1977). Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society*, Ser B, Vol. 39, pp. 1-38.
- Framis, F.R. (1994). An experiment on learning appropriate selection restrictions from a parsed corpus. In the *Proceedings of the International Conference on Computational Linguistics*, Kyoto, Japan.
- Gildea, D. (2002). Probabilistic Models of Verb-Argument Structure. In the *Proceedings of the 17th International Conference on Computational Linguistics*.
- Gomez, F. (2004). Building Verb Predicates: A Computational View. In the *Proceedings of the 42nd Meeting of the Association for Computational Linguistics*, pp. 359-366, Barcelona, Spain.
- Green, R.; Dorr, B.J.; Resnik, P. (2004). Inducing Frame Semantic Verb Classes from WordNet and LDOCE. In the *Proceedings of the 42nd Meeting of the Association for Computational Linguistics*, pp. 375-382, Barcelona, Spain.
- Grishman, R. and Sterling, J. (1992). Acquisition of selectional patterns. In the *Proceedings of the International Conference on Computational Linguistics*, pp. 658-664, Nantes, France.
- Grishman, R. and Sterling, J. (1994). Generalizing Automatically Generated Selectional Patterns. In the *Proceedings of the 15th International Conference on Computational Linguistics*, Kyoto, Japan.
- Kingsbury, P. and Palmer, M. (2002). From Treebank to PropBank. In the *Proceedings of the 3rd International Conference on Language Resources and Evaluation*, Las Palmas.
- Kipper, K.; Dang, H.T.; Palmer, M. (2000). Class-based Construction of a Verb Lexicon. In the *Proceedings of AAAI 17th National Conference on Artificial Intelligence*. Austin, Texas.
- Knight, K. and Marcu, D. (2002). Summarization beyond sentence extraction: A Probabilistic Approach to Sentence Compression. *Artificial Intelligence*, Vol. 139, N. 1.
- Korhonen, A. (2002). Semantically Motivated Subcategorization Acquisition. In the *Proceedings of the Workshop of the ACL Special Interest Group on the Lexicon*, pp. 51-58.

- Lapata, M. (1999). Acquiring lexical generalizations from corpora: A case study for diathesis alternations. In the *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pp. 394-404.
- Levin, B. (1993). *Towards a lexical organization of English verbs*. Chicago University Press, Chicago.
- Manning, C. (1993). Automatic acquisition of a large subcategorization dictionary from corpora. In the *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, pp. 235-242, Columbus, Ohio.
- Marcu, D. and Popescu, A.M. (2005). Towards Developing Probabilistic Generative Models for Reasoning with Natural Language Representations. In the *Proceedings of the 6th International Conference on Computational Linguistics and Text Processing* (Lecture Notes in Computer Science 2406 Springer 2005, ISBN 3-540-24523-5). Mexico City, Mexico.
- Marcus, M.; Santorini, B.; Marcinkiewicz, M.A. (1993). Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, Vol. 19, N. 2, pp. 313-330.
- Marcus, M. (1994). The Penn Treebank: A revised corpus design for extracting predicate-argument structure. In the *Proceedings of the ARPA Human Language Technology Workshop*, Princeton, NJ.
- McCarthy, D. (2000). Using semantic preferences to identify verbal participation in role switching alternations. In the *Proceedings of the 1st NAACL*, pp. 256-263, Seattle, Washington.
- Merlo, P. and Stevenson, S. (2001). Automatic Verb Classification Based on Statistical Distributions of Argument Structure. *Computational Linguistics*, Vol. 27, N. 3.
- Ratnaparki, A. (1996). A Maximum Entropy Part-Of-Speech Tagger. In the *Proceedings of the Empirical Methods in Natural Language Processing Conference*, University of Pennsylvania.
- Resnik, P. (1992). Wordnet and distributional analysis: a class-based approach to lexical discovery. In the *Proceedings of AAAI Workshop on Statistical Methods in NLP*.
- Rooth, M.; Stefan, R.; Prescher, D.; Carroll, G.; Beil, F. (1999). Inducing a semantically annotated lexicon via EM-based clustering. In the *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pp. 104-111, College Park, Maryland.
- Sarkar, A. and Zeman, D. (2000). Automatic extraction of subcategorization frames for Czech. In the *Proceedings of the 18th International Conference on Computational Linguistics*.
- Sarkar, A. and Tripasai, W. (2002). Learning Verb Argument Structures from Minimally Annotated Corpora. In the *Proceedings of the 19th International Conference on Computational Linguistics*.
- Soricut, R. and Brill, E. (2004). Automatic Question Answering: Beyond the Factoid. In the *Proceedings of the Human Language Technology and North American Association for Computational Linguistics Conference*.
- Voorhees, E.M. and Buckland, L.P. (eds.) (2002). *NIST Special Publication 500-251: The Eleventh Text REtrieval Conference* (TREC 2002). Department of Commerce, National Institute of Standards and Technology.

A Methodology for Extracting Ontological Knowledge from Spanish Documents

Rafael Valencia-García¹, Dagoberto Castellanos-Nieves²,
Jesualdo Tomás Fernández-Breis¹, and Pedro José Vivancos-Vicente¹

¹ Departamento de Informática y Sistemas. Facultad de Informática.,
Universidad de Murcia 30071 Espinardo (Murcia). España

Tel: +34 968364613, Fax: +34 968364151
{valencia, jfernand, pedroviv}@um.es

² Facultad de Informática y Matemática,
Universidad de Holguín, Holguín, Cuba
dcn.ncd@gmail.com

Abstract. This paper presents a semi-automatic approach for extracting knowledge from natural language texts in Spanish. The knowledge is acquired and learned through the combination of NLP techniques for analyzing text fragments, the ontological technology for representing knowledge and MCRDR, a case-based reasoning methodology. This approach has been applied in the ontology domain and the results of this application are discussed in this work.

1 Introduction

Spanish is the official language of a significant amount of countries, and it has millions of speakers world-wide. Hence, there is a huge amount of information and knowledge in Spanish documents. So, extracting knowledge from such texts would be beneficial and of great help for the Spanish speaking community. The recognition of natural language has been traditionally viewed as a linguistic issue and based on grammars. However, grammars have different drawbacks, such as the fact that they are unable of managing ambiguity, imprecision, variability, etc. In order to overcome the drawbacks of grammar approaches, we have developed a methodology for acquiring knowledge from texts in an incremental way based on knowledge engineering and natural language processing techniques. In this paper, we describe such methodology and how it is capable of extracting knowledge from pieces of Spanish free texts. The combination of knowledge engineering technologies with natural language processing techniques provides us the goodnesses of both areas. As far as knowledge engineering technologies are concerned, two have been included in the methodology, namely, ontologies and MCRDR. Let us introduce now both technologies and the reason why they are used in the methodology.

1.1 Ontologies

An ontology is viewed in this work as a formal specification of a domain knowledge conceptualization [10]. In this sense, ontologies provide a formal, structured knowledge representation, having the advantage of being reusable and shareable. In our methodology, ontologies are used to represent the knowledge extracted from texts, as

it is done in [12], so that, ontologies are obtained as a result of knowledge extraction processes. In addition to this, we represent ontologies by means of *multiple hierarchical restricted domains* (MHRD) in a similar manner to that employed by other authors. In particular, we term MHRD to a set of concepts holding that: (1) they are defined through a set of attributes; (2) there can be taxonomic relations among the concepts, so that attribute (multiple) inheritance is permitted; (3) there can be temporal relationships among the concepts; (4) there can be mereological relationships among the concepts; (5) other type of generic relations among concepts, which are considered the most common relations in problems [4] are initially also included in our model, such as equivalency, dependency, topology, causality, functionality, similarity, conditionality, purpose, etc; (6) there are "structural" axioms, that is, axioms drawn from the proper structure of the ontology. Beyond this definition, the methodology allows for the definition of new types of relations.

1.2 Multiple Classification Ripple Down Rules (MCRDR)

MCRDR [5] is an incremental methodology to build knowledge-based systems. It is based on the construction of a knowledge rules tree. Each rule has a series of conditions to satisfy that are used to infer a series of conclusions and to evaluate the successor rules in the tree. When an MCRDR tree makes a wrong classification, then it can learn from this error by specifying the reasons of the error. So, the intervention of the expert is required at this stage in order to tune the MCRDR module until the accuracy of the system is considered good enough. This is the semi-automatic component of the approach. The MCRDR module classifies automatically, but in case of wrong conclusions, the expert component must correct such decisions. In MCRDR, an n-ary tree is used; i.e., a rule may have multiple refinement rules. A conclusion is provided by the last rule satisfied in a pathway. All children of a satisfied parent rule are evaluated, allowing for multiple conclusions. The conclusion of the parent rule is only given if none of the children are satisfied. In our methodology, MCRDR is combined with the grammatical category of the words to infer knowledge entities (i.e., concepts, attributes and values) from pieces of text. So, the nodes of our MCRDR tree contain knowledge entities such as concepts, attributes, values and relations, as well as the conditions to conclude such entities.

Once described the knowledge technologies used, we proceed to describe how the paper is organized. In Section 2, the knowledge extraction process is described. Then, the different stages of the knowledge extraction subprocesses are explained in Section 3. The application of the methodology is illustrated through the example presented in Section 4. A validation in the breast cancer domain is shown in Section 5. Finally, some conclusions are put forward in Section 6.

2 Overview of the Knowledge Extraction Framework

The framework used in this methodology is comprised of three main modules, namely, concept knowledge base, relation knowledge base and MCRDR sub-system. At the beginning of the extraction process, they are likely to be empty unless previous knowledge extraction processes had been carried out. The goal of the framework is to extract all the possible knowledge from pieces of free texts. This process will be supported by an expert (at least) while the performance of the framework is not satisfactory. So far, provided that the three components are empty, the framework will

be unable to find any knowledge in the input text and the expert has to introduce the knowledge manually. The text is processed on a sentence-by-sentence basis. When the expert manually extracts the knowledge, (s)he has to identify the type of relation associated to the main verb (if any) of the current sentence and all the other knowledge entities of the current textual fragment. (S)he will also tell the system in which *linguistic expressions* they appear. An association between the linguistic expression and the knowledge it represents is stored; in particular, the expressions associated with concepts are stored in the conceptual knowledge base, and the expressions associated with relationships are stored in the verbal knowledge base) in order to be used for new knowledge findings thereafter. Thus, the MCRDR sub-system is used for extracting the knowledge entities participating in the relationship under question. The MCRDR sub-system rules are based on the type of relation that represents the main verb in the current sentence, the grammatical categories of the knowledge entities, which appear in that sentence, and the position of these knowledge entities with respect to the verb in the sentence. The framework obtains the type of relation and the knowledge entities (concepts, attributes, values and relationships) in the current sentence, and the expert will just have to confirm these results. If the real conclusion is different from the conclusion inferred by the system, then the expert has to introduce the correct conclusion. After that, the new case is added into the MCRDR sub-system which will have the rules updated. This process is repeated for each sentence of the text and the framework will incrementally learn and it will extract the knowledge from the text in a more accurate way as it is fitted.

3 Knowledge Extraction Subprocesses

After describing the framework, more attention should be paid to the proper knowledge extraction processes (see Figure 1) included in this methodology. For each sentence in the text, three sequential phases, called POS-Tagging, concept search and inference, are applied. Then, the expert can correct the knowledge inferred through these phases or validate it. This process is done for all the sentences of the text.

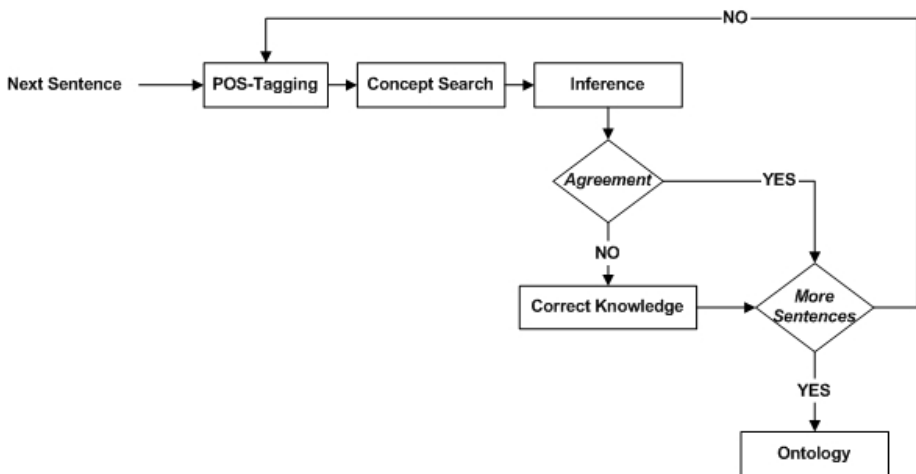


Fig. 1. The knowledge acquisition process

3.1 POS-Tagging

The main objective of this process is to obtain the grammatical category of each word in the current sentence. For this purpose, the eTiKeT@ POS-tagger [8] is used. This is the first subprocess to be carried out.

For example, the morphological analysis for the sentence “El cáncer de piel es una enfermedad producida por el desarrollo de células cancerosas en las capas exteriores de la piel.” obtains the following grammatical categories:

El [det] cáncer [noun] de [prep] piel [noun] es [verb (auxiliary)] una [det] enfermedad [noun] producida [Verb (lexical)] por [prep] el [det] desarrollo [noun] de [prep] células [noun] cancerosas [adj] en [prep] las [det] capas [noun] exteriores [adj] de [prep] la [det] piel [noun].

3.2 Concept Search

Through this process, linguistic expressions representing concepts are identified. The associations between linguistic expressions and concepts are stored in the conceptual knowledge base. This process is quite simple and, as a result, all the expressions of the fragment which are already in the conceptual knowledge base are obtained.

In this approach, it is assumed that there exist some semantically meaningless words. These words usually have the following grammar categories: Preposition, Conjunction, Interjection, Particle, Pronoun and Determiner. So, only concepts associated to Nouns, Adjectives and Adverbs will be searched.

With all, the system works as follows. For each noun, adjective or adverb, it looks for *similar* words in the expressions existing in the concept knowledge base. Then, for each of these expressions, if it is considered to be *acceptable*, it is added to the list of expressions with its knowledge associated. Obviously, there might be cases where no good options are found. In that case, the user may define new knowledge associated with the expression or just ignore it.

The *similar* function identifies the expressions of the knowledge base that are “similar” to the current word of the fragment. In its simplest case, it would be an “equal” function. Nevertheless, this function cannot deal with compound expressions by itself, so a function for checking whether the current word is a substring of another word. In this case, a word in the current fragment is “similar” to an expression in the knowledge base if the expression starts with the current word.

The main drawback of this function is that unrelated concepts might be identified as similar. This is the case of “prop” and “property”. Hence, the similar function is currently being improved to deal with families of words (i.e., types associated to a single lemma/lexeme) and other linguistic phenomena. For instance, if the expression “causes” already exists in the knowledge base and the current fragment contains the word “caused”, it would be desirable to realize of the fact that both words actually allude to the same verb (lemma). This issue is being partially implemented using part-of-speech taggers and lemmatizers.

The *acceptable* function is an extension of the “*similar*” one. Since the “*similar*” function can be very permissive, the “acceptable” function is introduced in order to determine whether the current word and a similar expression are not just “similar by chance”. An example of the “isPrefix” function is: if the current word is the noun

“cáncer”, any expression starting with “cáncer”, as “cáncer de mama”, “cancer de piel”, or “cáncer de próstata” will be (candidates to be) considered similar.

Therefore, this function limits the number of acceptable options amongst the similar ones. This function has been designed with strong requirements: an existing expression in the database is acceptable if it actually appears in the current fragment.

Current words in a text fragment are always single constituents. However, database expressions can contain more than one word (multiple-word expressions). If a word is acceptable, then the current fragment will contain all the words of the database expression. That is, the current word needs to be enlarged to cover all the words of the database expression, creating a new object that contains all the words.

As an example, let us suppose that the concept knowledge base contains the linguistic expression-concept associations showed in Table 1.

Table 1. Concept Knowledge Base

Linguistic expressions	Concept Associated
Cáncer	Cáncer
Cáncer de mama	Cáncer de mama
Cáncer de piel	Cáncer de piel
Enfermedad	Enfermedad

Let us suppose the sentence “El cáncer de piel es una enfermedad producida por el desarrollo de células cancerosas en las capas exteriores de la piel.”.

Let us also suppose that the tagger has previously labelled all the words in the sentence with their grammatical category: “El [det] cáncer [noun] de [prep] piel [noun] es [verb (auxiliary)] una [det] enfermedad [noun] producida [Verb (lexical)] por [prep] el [det] desarrollo [noun] de [prep] células [noun] cancerosas [adj] en [prep] las [det] capas [noun] exteriores [adj] de [prep] la [det] piel [noun]”.

Only concepts associated to Nouns, Adjectives, and Adverbs are searched. The first word in the sentence “cáncer” has been labelled as a noun, so that we look for similar linguistic expressions into the concept knowledge base. The result of similar words is {cáncer, cáncer de mama, cáncer de piel}. Then, the acceptable function obtains that “Cáncer de piel” is a concept.

The next word to be processed is “enfermedad”, because all the previous words are labelled as neither a noun, neither an adjective, neither an adverb.

Similar: {enfermedad }

Acceptable: {enfermedad }

The methodology would proceed analogously for the rest of words in the sentence.

3.3 Inference

In natural language, relationships between concepts are usually associated to verbs [11]. Although the sub-process in which MCRDR acts is mainly concerned with obtaining relationships between concepts, it can also be used to get other knowledge categories like concepts, attributes, or values. This MCRDR component is formed by a knowledge-base that contains linguistic expressions representing generic conceptual relationships, and by an MCRDR subsystem that infers the participants in these relationships. We describe next the modus operandi of this process. The main verb of the current sentence

is identified. Then, there is a search for the type of semantic relationship associated to that verb. This search is performed on the relation knowledge base by using the *similar* and *acceptable* functions. Once the type of relation associated to the main verbal expression in the current sentence is found, the MCRDR sub-system is applied to extract knowledge by means of the grammatical category of the words, their position in the current sentence, and the type of relation associated to the verbal expression, if any.

A new case is then created, formed by the type of relation, which represents the verbal expression, and the categories of the other words in the sentence. This case would be processed by the MCRDR sub-system and would generate a conclusion.

Once a conclusion is obtained, the expert has to identify the knowledge entities and the relationship(s), if any, in the current sentence in order to check whether that conclusion is correct (i.e., if it is a wrong, or it is an incomplete ontology). Then, the system identifies the differences between the previously stored case and the new one. After that, the system updates the MCRDR sub-system rules, and introduces the expression and the type of relation associated to it into the knowledge base.

4 Example of the Extraction Process

We can illustrate how the methodology works through the following example. Both the MCRDR sub-system and the knowledge base, which contains verbs and the type of relation associated to them, are initially empty. Let us suppose the sentence “El cáncer de piel es una enfermedad producida por el desarrollo de células cancerosas en las capas exteriores de la piel.”. Its POS-tagging analysis has these results:

El [det] cáncer [noun] de [prep] piel [noun] es [verb (auxiliary)] una [det] enfermedad [noun] producida [Verb (lexical)] por [prep] el [det] desarrollo [noun] de [prep] células [noun] cancerosas [adj] en [prep] las [det] capas [noun] exteriores [adj] de [prep] la [det] piel [noun].

As both the relations knowledge base and the MCRDR sub-system are empty, no knowledge is inferred. So, the expert identifies that the expression “es una” is associated to an “IS-A” relationship between the concepts “Cáncer de piel” and “enfermedad”. Thus, the corresponding expression-type of relation association is stored into the knowledge base, and then the new case is introduced in the MCRDR sub-system, so the following rule is created:

If relation=“IS-A” and category (pos(verb)-1)=Noun and category (pos(verb)-2)=prep and category(pos(verb)-3)=Noun and category(pos(verb)+2)=Noun **then** {**Concept**(word(pos(verb)-3)+ word(pos(verb)-2)+ word(pos(verb)-1)), **Concept**(word(pos(verb) +1)), **Relation**(IS-A, word(pos(verb)-3)+ word(pos(verb)-2)+ word(pos(verb)-1),word(pos(verb)+1))}.

Here ‘pos(verb)’ is a function that returns the position of the ‘verbal’ expression in the current sentence. The function ‘category(i)’ returns the category of the word in the position i, the function word(i) returns the word in the position i. The operator ‘+’ concatenates two linguistic expressions.

The next sentence is “El [det] cáncer[noun] de[prep] próstata[noun] consiste [Verb(lexical)] en[prep] un[det] tumor[noun] maligno[adj] que[pro] se[Pro] desarrolla[Verb(lexical)] en[prep] la[adj] glándula[noun] prostática[adj]”.

Given this situation, expressions *similar* to the verb “consiste” are looked in the knowledge base up. So far, the knowledge base does not contain *similar* expressions to “consiste”, so no knowledge is inferred. Then, the expert identifies that the expression “consiste en un” is associated to an IS-A relation. After that, a rule is activated inferring that *Cáncer de próstata* IS-A *tumor*.

The next sentence to analyse is “Una [det] exposición[noun] excesiva[adj] al[det] sol[noun] puede[Verb(auxiliary)] producir[Verb(lexical)] cáncer[noun] de[prep] piel[noun], sin[adv] el[det] uso[noun] de[prep] cremas[noun] protectoras[adj] solares[adj]”

Expressions *similar* to the verb “puede” are then looked in the knowledge base up. Again, no knowledge is inferred and the expert associates the expression “puede producir” to a CAUSE relation, and extracts the concepts “sol” and “cáncer de piel”, and the relation “sol CAUSE cáncer de piel”.

As a result, new knowledge associations are stored and the following rule is created:

If relation=“CAUSE” and category (pos(verb)-1)=Noun and category(pos(verb)+1)=noun and category(pos(verb)+2)=prep and category(pos(verb)+3)=noun **then** { **Concept**(word(pos(verb)-1)), **Concept**(word(pos(verb)+1)+ word(pos(verb)+2)+ word(pos(verb)+3)), **Relation**(CAUSE, word(pos(verb)-1), word(pos(verb)+1)+ word(pos(verb)+2)+ word(pos(verb)+3)) }

Next sentence:

El[det] hábito[noun] del[prep] tabaco[noun] es[verb(auxiliary)] la[det] causa[noun] principal[adj] de[prep] cáncer[noun] de[prep] pulmón[noun] en[prep] el[det] 90%[num] de[prep] los[det] varones[noun] y[conj] en[prep] el[det] 70%[num] de[prep] las[det] mujeres[noun].

“Es una” is found *similar* to the verb “es” but it is not an *acceptable* expression so, the system infers nothing. The expert identifies then that the expression “es la causa principal” is associated to a CAUSE relation, so the rule 2 is activated obtaining that “tabaco CAUSE cancer de pulmón”.

In Figure 2, the ontological components extracted from this example is shown.

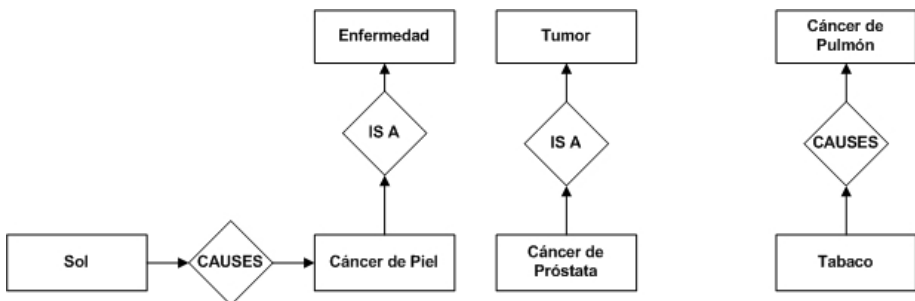


Fig. 2. Partial Ontology of the example

Through the example, the *conceptual and relational knowledge bases* have been modified, and new elements have been inserted into them. Tables 2 and 3 show these knowledge bases after the example.

Table 2. *Conceptual knowledge base after the example*

Linguistic Expressions	Associated Concept
Cáncer	Cáncer
Cáncer de mama	Cáncer de mama
Cáncer de Páncreas	Cáncer de Páncreas
Cáncer de Piel	Cáncer de Piel
Cáncer de Pulmón	Cáncer de Pulmón
Enfermedad	Enfermedad
Sol	Sol
Tabaco	Tabaco
Tumor	Tumor

Table 3. *Relational knowledge base after the example*

Linguistic Expressions	Associated Relation
consiste en un	<i>IS A</i>
es una	<i>IS A</i>
es la causa principal	<i>CAUSE</i>
Puede producir	<i>CAUSE</i>

5 Validation

The methodology described in this work has been validated across the breast cancer application domains. Four PhD students used the system with a corpus of 5683 words divided into four texts. The aim of this experiment was to analyze whether the tool was capable of learning and suggesting the correct knowledge to the experts. In the following table, the accuracy of the knowledge suggestions proposed by the system to the users is shown. The accuracy score is the result of the division between the amount of knowledge entities suggested by the system and that are accepted by the expert user, and the total amount of knowledge entities existing in the text. In order to complete the conditions in which the experiment was run, it should be noticed that the knowledge base was initially empty, that is, each student started with an empty knowledge base which was augmenting its content as the knowledge acquisition process evolved.

In Table 2, the partial and accumulated results for each text are displayed. The first four columns represent the partial accuracy obtained for each student; the remaining ones are the accumulated accuracy of each student. From the results contained in that table, the learning capability of the system can be affirmed, since the accuracy of the system increases along the number of fragments processed. In the experiment we obtain a maximum accumulated accuracy of 60,33% and for the first student in the four text we obtain a partial accuracy of 80,60%.

Table 4. Partial and Accumulated Accuracy of the experiment

Text	Partial Accuracy				Accumulated Accuracy			
	Student 1	Student 2	Student 3	Student 4	Student 1	Student 2	Student 3	Student 4
1	49,52	52,38	35,02	37,50	49,52	52,38	35,02	37,50
2	45,33	50,67	45,79	50,44	48,42	51,93	38,19	41,64
3	66,67	50,00	63,11	60,00	51,33	51,47	43,68	45,85
4	80,67	65,41	61,16	60,86	60,33	56,07	50,88	52,35

These evaluation results are promising. The performance of the systems improves through the sessions, so helping the experts to extract the knowledge contained in a text in a more efficient way.

6 Discussion and Conclusion

In this paper, a human-driven method for acquiring knowledge from texts has been presented. This approach is based on the use of the grammatical categories of words, a small verbal knowledge base and a small conceptual knowledge base for performing inferences. The framework is divided into three main processes: part-of-speech tagging, search for concepts and inference. The inference process, which is likely to be the most challenging one, allows storing only the linguistic expressions that represent type of relations in the knowledge base, different knowledge entities might be inferred. It is also remarkable that the methodology is capable of inferring many relationships and knowledge entities by only knowing their respective grammatical category as the MCRDR case and knowledge bases grow.

Ontology building from free texts is an important activity for the knowledge engineering community. One of the hottest research trends in this area is ontology learning from Web documents, which is considered an important activity to promote the Semantic Web [1;2;9]. Knowledge acquisition from texts is a process that has already been considered part of the ontology learning process [6]. The approach presented in this work is human-driven as most of the ontology construction approaches. Another feature of this approach is that it works with multiple semantic relations (not only taxonomy).

We are currently planning the use of Natural Language Ontologies as WordNet [3] to verify the correctness of the relations inferred by our system and to obtain the new relations as it is done in [7], which use Wordnet to interpret semantic terms and to identify mainly taxonomic and similarity relations.

A larger validation of the system is planned by applying the system to texts from different medical domains and by using statistical methods for analysing the results obtained. Moreover, we intend to extend the system to cover axioms. The main forecast problem concerning axioms is however that the number of participants in axioms is a priori unknown. However, the amount of axioms present in a text is not significant compared to the amount of other knowledge entities.

Acknowledgements

This work has been possible thanks to the Spanish Ministry for Science and Education through the projects TIC2002-03879, and TSI2004-06475-C02-02; the Seneca Foun-

dition through the Project 00670/PI/04; and FUNDESOCO through project FDS-2004-001-01.

References

1. H. Alani, S. Kim, D.E. Millard, M. J. Weal, W. Hall, P.H. Lewis, and N.R. Shadbolt, 'Automatic Ontology-Based Knowledge Extraction from Web Documents', *IEEE Intelligent Systems*, January/February 2003 14-21, (2003)
2. H. Davulcu, S. Vadrevu, S. Nagarajan, and I.V. Ramakrishnan, 'OntoMiner: Bootstrapping and Populating Ontologies from Domain-Specific Web Sites'. *IEEE Intelligent Systems*, September/October 2003, 24-33, (2003)
3. C. Fellbaum, *WordNet: An Electronic Lexical Database*. The MIT Press, (1998).
4. A. Gómez-Pérez, A. Moreno, J. Pazos and A. Sierra-Alonso, 'Knowledge maps: An essential technique for conceptualization'. *Data and Knowledge Engineering*, **33**, 169-190, (2000).
5. B. Kang, *Multiple Classification Ripple Down Rules*. PhD thesis, University of New South Wales.1996
6. A. Maedche, and S. Staab, 'Ontology Learning for the Semantic Web', *IEEE Intelligent Systems*, vol. **16**, no. 2, pp. 72 – 79, (2001)
7. R. Navigli, P. Velardi and A. Gangemi. 'Ontology Learning and Its Application to Automated Terminology Translation'. *IEEE Intelligent Systems*, January/February 2003, 22-31, (2003)
8. J. M. Ruiz-Sanchez, R. Valencia-García, J. T. Fernández-Breis, R. Martínez-Béjar, P. Compton. 'An approach for incremental knowledge acquisition from text'. *Expert Systems with Applications*. **25(2)**, 77-86, (2003)
9. M. Shamsfard and A. Barforoush 'Learning ontologies from natural language texts', *International Journal of Human-Computer Studies* **60(1)**: 17-63, (2004)
10. G. Van Heijst, A. T. Schreiber, & B. J. Wielinga, 'Using explicit ontologies in KBS development'. *International Journal of Human-Computer Studies*, **45**, 183-292, (1997).
11. R. Valencia-García, J.M. Ruiz-Sánchez, P.J. Vivancos-Vicente, J.T. Fernández-Breis, R. Martínez-Béjar 'An incremental approach for discovering medical knowledge from texts'. *Expert Systems with Applications* 26(3):291-299, (2004)
12. R. Valencia-García, D. Castellanos-Nieves, P. J. Vivancos Vicente, J. T. Fernández-Breis, R. Martínez-Béjar and F. García-Sánchez. 'An approach for Ontology Building from Text Supported by NLP techniques' Lecture Notes in Artificial Intelligence **3040** (10th Conference of Spanish Association for Artificial Intelligence, CAEPIA 2003, and 5th Conference on Technology Transfer, TTIA 2003, San Sebastián, Spain, November 2003) : 126-135 (2004)

Automatically Determining Allowable Combinations of a Class of Flexible Multiword Expressions

Afsaneh Fazly, Ryan North, and Suzanne Stevenson

Department of Computer Science, University of Toronto,
Toronto, ON M5S 3H5, Canada
{afsaneh, ryan, suzanne}@cs.toronto.edu

Abstract. We develop statistical measures for assessing the acceptability of a frequent class of multiword expressions. We also use the measures to estimate the degree of productivity of the expressions over semantically related nouns. We show that a linguistically-inspired measure outperforms a standard measure of collocation in its match with human judgments. The measure uses simple extraction techniques over non-marked-up web data.

1 Light Verb Constructions

Recent work in NLP has recognized the challenges posed by the rich variety of multiword expressions (MWEs) (e.g., Sag et al., 2002). One unsolved problem posed by MWEs is how they should be encoded in a computational lexicon. Many MWEs are syntactically flexible; for these it is inappropriate to treat the full expression as a single word. However, fully compositional techniques can lead to overgeneralization, because flexible MWEs are often *semi*-productive: new expressions can only be formed from limited combinations of semantically and syntactically similar component words. In order to achieve accurate lexical acquisition methods, we must determine computational mechanisms for capturing the allowable combinations of such MWEs.

Our focus here is on light verb constructions (LVCs); these are largely compositional and semi-productive MWEs having a high frequency of occurrence across many diverse languages (Karimi, 1997; Miyamoto, 2000; Butt, 2003). LVCs combine a member of a restricted set of light verbs, such as *give*, *take*, and *make* among others in English, with a wide range of complements of varying syntactic categories. We consider a common class of LVCs, in which the complement is a noun generally used with an indefinite article, as in (a–c) below:

- | | |
|--|--|
| a. Priya <i>took a walk</i> along the beach. | d. Priya <i>walked</i> along the beach. |
| b. Allene <i>gave a smile</i> when she saw us. | e. Allene <i>smiled</i> when she saw us. |
| c. Randy <i>made a joke</i> to his friends. | f. Randy <i>joked</i> to his friends. |

Moreover, the complement nouns in these expressions, such as *walk*, *smile*, and *joke* in (a–c), have a stem form identical to a verb. Because the light verb is “semantically bleached” to some degree (Butt, 2003), most of the meaning of these LVCs comes from the complement. The predicative nature of the complement is illustrated by the fact that the noun complements in (a–c) contribute the verbs of the corresponding paraphrases in (d–f).

These LVCs are of interest because they are very frequent, and moreover, their productivity appears to be patterned (Kearns, 2002; Wierzbicka, 1982). For example, one can *take a walk*, *take a stroll*, and *take a run*, but it is less natural to *?take a groan*, *?take a smile*, or *?take a wink*. These patterns of semi-productivity depend on both the semantics of the complement and on the light verb itself. For example, in contrast to *take*, we observe *?give a walk*, *?give a stroll*, *?give a run*, compared to *give a groan*, *give a smile*, *give a wink*. Thus, these constructions provide a rich testbed for exploring computational means for capturing the range of allowable combinations of semi-productive MWEs.

We approach this problem by developing and evaluating acceptability measures for LVCs, which draw on their linguistic properties. Furthermore, we investigate the hypothesis that the acceptability of a candidate LVC depends on the semantic class of its complement. Two semantic classifications for the potential complements are compared, to explore the impact of different semantic similarity criteria. Human acceptability judgments of each candidate expression are gathered, and used as the standard against which our statistical measures are evaluated.

Our results indicate a high level of compatibility between our computational measures of acceptability and human judgments. Moreover, we find that the measures can be used to quantify the productivity of a class of complements, i.e., the extent to which the semantic class forms acceptable expressions with a particular light verb. Automatically assessing both acceptability of individual expressions, and productivity across semantic classes, enables us to take a first step toward adequate representation of LVCs in a computational lexicon. Since such semi-productive behaviour arises frequently (e.g., in verb-particle constructions and other phrasal verbs), we believe our approach yields insights for the automatic extraction and representation of MWEs more generally.

2 Acceptability Measures

We present three statistical measures for a continuously valued assessment of LVC acceptability. These measures capture in differing ways the association between a light verb (LV) and a noun complement. Since the complement of such LVCs contributes event semantics to the expression (as illustrated in (a–f) above), the noun must be a predicative noun (PN)—i.e., a noun that has an argument structure. Because the PN is preceded by an indefinite determiner (*a* or *an*) in these expressions, we refer to the complement in our formulas below as *aPN*.¹

2.1 The PMI_{LVC} Measure

Following Stevenson et al. (2004), our first measure uses pointwise mutual information (PMI), a standard measure of collocation (Church et al., 1991), to assess the strength of association between a given LV and PN:²

¹ Since LVCs are somewhat flexible (*give it a try*, *take a nice walk*), we allow other intervening words between the LV and PN in some of our counts, as described in detail in §3.3.

² PMI has some limitations with very low frequency items, but since we use the web as our corpus (see §3), we do not expect counts of such low frequency.

$$\text{PMI}_{\text{LVC}}(LV; aPN) = \log \frac{nf(LV, aPN)}{f(LV)f(aPN)},$$

where n is the corpus size. Higher values of PMI_{LVC} reveal a greater degree of association between the LV and PN, which can be interpreted as an indication of LVC acceptability.

2.2 The Prob_{LVC} Measure

While common LVCs typically appear as good collocations, the PMI_{LVC} measure fails to incorporate other important properties of LVCs. Here, we propose a linguistically-motivated measure, Prob_{LVC} , which captures the likelihood that a given LV and PN form an acceptable LVC, i.e., $\text{Pr}(LV, PN, \text{LVC})$. The joint probability is factored as:

$$\text{Pr}(PN) \text{Pr}(\text{LVC}|PN) \text{Pr}(LV|PN, \text{LVC}).$$

The first factor, $\text{Pr}(PN)$, reflects the observation that higher frequency words are more likely to be used as complements in LVCs (Wierzbicka, 1982). We estimate this probability by $f(PN)/n$, where n is the number of words in the corpus.

The $\text{Pr}(\text{LVC}|PN)$ factor captures the general tendency of the PN to form LVCs with any light verb. This factor is estimated by the number of times we observe the prototypical LVC pattern “LV a PN” with this PN across possible LVs:

$$\text{Pr}(\text{LVC}|PN) \approx \frac{\sum_{i=1}^v f(LV_i, aPN)}{f(aPN)},$$

where v is the number of LVs in our study. Since we are only counting usages of the PN in the context of an indefinite determiner in the numerator, we normalize over counts of aPN . Note that simply counting “LV a PN” as an LVC is an overestimate, since we cannot determine which of such usages are indeed LVCs, as opposed to literal usages of the LV as in *give a present*. However, we expect that true predicative nominals will have a higher probability of usage in LVCs than other nouns, since the noun complement must contribute an argument structure to the LVC.

Finally, $\text{Pr}(LV|PN, \text{LVC})$ reflects the specific tendency of the PN to form an LVC with this particular light verb, LV. We similarly estimate this factor with counts of the LV and PN in the typical LVC pattern: $f(LV, aPN)/f(aPN)$.

Combining the estimation of the three factors results in the following formula:

$$\text{Prob}_{\text{LVC}}(LV, PN) \approx \frac{f(PN)}{n} \times \frac{\sum_{i=1}^v f(LV_i, aPN)}{f(aPN)} \times \frac{f(LV, aPN)}{f(aPN)},$$

where v is the number of LVs and n the corpus size.

2.3 The Freq_{LVC} Measure

We also propose here an additional measure, Freq_{LVC} , for which the primary goal is inexpensive extraction from noisy but plentiful data. Freq_{LVC} assesses the acceptability

of a candidate LVC simply according to its raw frequency in the corpus. But from that raw frequency, we subtract an estimate of the amount of noise affecting the candidate expression, in order to better approximate the frequency of “true” LVC usage.

Our estimate of noise is based on the intuition that the likelihood of seeing an LVC with m internal modifiers—intervening words between the LV and PN—approaches zero as m increases. For example, we expect to find *take a walk* more often than *take a long walk*, which in turn is more probable than *take a long relaxing walk*, etc. We assume there exists a threshold, t , at which the likelihood of producing an LVC involving $m \geq t$ words of internal modification is negligible. At this threshold, any results found must be noise—i.e., cooccurrences of the LV and PN that are unrelated to LVC usage.

Let $f_m(LV, aPN)$ be the frequency of the string “LV a *word* ^{m} PN”. As we increase the value of m from 0 to t , the number of actual LVC usages included in $f_m(LV, aPN)$ gradually decreases. Under the assumption that the amount of noisy results remains roughly constant, we can use $f_t(LV, aPN)$ as the estimate of noise for each count. Thus if $f_0(LV, aPN)$ is the count of “LV a PN”, including both actual LVCs and noise, then if we subtract from it the estimate of noise, $f_t(LV, aPN)$, we have an estimate of the actual LVC usage when $m = 0$.

The assumption that noise remains constant does not hold in practice (as actual LVC usage decreases, noise increases). However, we find that by taking an average of $f_t(LV, aPN)$ across a range of values of t , we achieve a useful estimate of noise. The resulting measure is defined as:

$$\text{Freq}_{\text{LVC}}(LV, PN) = f_0(LV, aPN) - \text{mean}_t f_t(LV, aPN).$$

In our experiments, t is in the range [6,10], empirically established through experiments on the development data.

3 Materials and Methods

3.1 Light Verbs Used

Linguists have identified a small set of verbs which, crosslinguistically, are commonly used as light verbs. We focus on three frequent light verbs in English, *take*, *give*, and *make*. *Take* and *give* have nearly opposite, but highly related, semantics, while *make* differs from both. Also, the line between light and literal uses of *make* is less clear. We expect then that *make* will show contrasting behaviour.

3.2 Experimental Expressions

Experimental expressions—i.e., potential LVCs using *take*, *give*, and *make*—are formed by combining the three light verbs with predicative nouns from (i) selected semantic verb classes of (Levin, 1993); or (ii) generated WordNet classes. In each case, some classes are used as development data, and some classes as test data. The following paragraphs explain the selection of Levin classes, and the process of generating corresponding classes using WordNet.

Table 1. Seed words selected according to acceptability trends identified for each Levin test class

Levin Class	Acceptability Trend			Seed Word
	<i>take</i>	<i>give</i>	<i>make</i>	
<i>Hit, Swat Verbs</i>	fair	good	fair	<i>knock</i>
<i>Peer Verbs</i>	fair	fair	poor	<i>check</i>
<i>Verbs of Sound Emission</i>	poor	good	fair	<i>ring</i>
<i>Verbs of Motion Using a Vehicle^a</i>	good	fair	poor	<i>sail</i>

^a The subset that are verbs that are not vehicle names.

Selection of Levin Classes. It may seem odd to use a verb classification for noun complements. However, recall that an important property of the type of LVCs we are considering is that the complement noun has an argument structure, and is identical in stem form to a verb. The verb classes of Levin (1993), defined by similarity of argument structure, therefore provide natural similarity sets to consider. As long as we only use verbs identical in form to a noun, we are assured that such complements are PNs.

Our three development and four test classes from Levin are taken from Stevenson et al. (2004). These classes reflect a range of productivity in combination with the three light verbs. For classes with more than 35 verbs (30 for development classes), a random subset of that size is selected for experimentation.

Generation of WordNet Classes. Although the use of Levin verb classes has linguistic motivation, it may be that semantic classes which also incorporate nominal similarity are more appropriate for this task (Newman, 1996). We therefore also use semantic classes generated from both the noun and verb hierarchies of WordNet 2.0 (Fellbaum, 1988). In determining these WordNet-derived classes, it is important that they are comparable to each of the Levin classes, so that we can relate performance of our measures across the corresponding classes from the two classifications. We achieve this by generating a WordNet set that is semantically similar to a representative word from a given Levin class.

Specifically, for each Levin class, we first determine the general pattern of acceptability of that class with the different light verbs. As described in §3.4 below, human ratings are put into buckets of ‘poor’, ‘fair’, and ‘good’. We then determine the predominant bucket for each class and light verb, and select a representative PN seed that best reflects the typical ratings across the three light verbs (see Table 1). For each seed, we examine both the noun and verb hypernym hierarchies of WordNet, and select all words which have a parent in common with the seed. We filter from this set those words which do not appear in both hierarchies, thereby excluding items which are not nouns identical in form to a verb.³ A random selection of 35 of the remaining words forms a WordNet class, which we refer to by “WN-” plus the seed word (e.g., WN-*knock*).

Our final experimental data consists of 195 PNs in the development set (90 from Levin classes and 105 from WordNet classes), and 238 PNs in the test set (98 from Levin classes and 140 from WordNet classes). These PNs are combined with each of

³ In contrast to the Levin expressions, we also filter rare PNs, whose frequency as a verb in the British National Corpus is less than 50.

the three light verbs to yield 585 development expressions, and 714 test expressions, all of the form “*give/take/make a PN*”.

3.3 Corpus and Data Extraction

LVCs of the type we consider are, as a class, very frequent. Interestingly, however, individual expressions may be highly acceptable but not very frequently attested in a traditional corpus. We therefore decided to use the web (the subsection indexed by Google) to estimate frequency counts required by the statistical measures. Each count is calculated via an exact-phrase search; counts including LVs are collapsed across separate searches using three tenses of the verb: base, present, and simple past. The number of hits is used as the frequency of the string searched for. The size of the corpus, n , is estimated at 5.6 billion, the number of hits returned in a search for “*the*”. Note that frequency counts for candidate expressions are likely underestimated, as a phrase may occur more than once in a single web page; we make the simplifying assumption that this affects all counts similarly.⁴ Such web-based frequency estimates have been successfully used in many NLP applications (Turney, 2001; Villavicencio, 2003), and have been shown to highly correlate with frequency counts from a balanced corpus (Keller and Lapata, 2003).

Most LVCs allow their noun component to be modified, as in *take a long walk*. To capture such uses, we use the “*” wildcard (as in “*take a * walk*”), which matches exactly one word. Moreover, many LVCs using the light verb *give* frequently appear in the dative form; some of these can only appear in this form. For example, one can *give NP a try*, but typically not *?give a try to NP*. To address this, we perform individual searches for each of a set of 56 common object pronouns intervening between the LV and PN components. The estimated frequency of an expression is the sum over its bare, adjectival, and dative forms. (The additional searches are not run for Freq_{LVC} , as it is designed to explore rating LVCs using little information.)

3.4 Human Acceptability Judgments

Two expert native speakers of English rated the acceptability of each experimental expression. The ratings range from 1 (unacceptable) to 4 (completely natural), by 0.5 increments. On Levin test expressions, the two sets of ratings yield kappa values of .72, .39, and .44, for *take*, *give*, and *make*, respectively, and .53 overall. (We use linearly weighted kappa, since our ratings are ordered.) Wide differences in ratings typically arose when one rater missed a possible meaning for an expression; these were corrected in the reconciliation process. Discussion of disagreements when rating Levin expressions led to more consistency in ratings of WordNet expressions, which yield (linearly weighted) kappa values of .79, .66, and .69, for *take*, *give*, and *make*, respectively, and .71 overall. Ratings were reconciled to within one point difference, and then averaged to form a single consensus rating. We also place the consensus ratings in buckets of ‘poor’ (range [1–2]), ‘fair’ (range [2–3]), and ‘good’ (range 3 and higher) for coarser-grained comparison.

⁴ This is clearly not the case for the estimate of the corpus size, since “*the*” likely occurs frequently within each page. However, in our formulas, this value appears as a constant.

4 Experimental Results

We evaluate our measures by comparing their acceptability scores with the consensus human ratings: Spearman rank correlation coefficient (r_s) is used to compare the rankings provided by the two (§4.1); linearly weighted observed agreement (p_o) is used to examine their agreement at the coarser level of the acceptability buckets (§4.2). The acceptability buckets are further used to determine the appropriateness of our measures for predicting the productivity of a class with respect to LVC formation (§4.3). We focus on the performance on unseen test data; trends are similar on development data.

4.1 Correlation Between Human Ratings and Statistical Measures

We perform separate correlation tests between the consensus human ratings and the three measures over each of the three LVs in combination with each of the four test classes within the two classifications (Levin and WordNet). In Figure 1, we show the results graphically, so that patterns are easier to see. Each rectangle represents a separate correlation calculation. Values of r_s which are not significant are shown as the lightest rectangles; significant values from .30 to over .70 (by deciles) are shown as increasingly darker rectangles.⁵ We discuss the results in terms of the measures, the light verbs, and the two classifications, in turn.

The Prob_{LVC} measure is the most consistent of the three measures, performing best overall and achieving good correlations in most cases. The PMI_{LVC} measure does surprisingly well, as a simple measure of collocation; it even performs comparably to Prob_{LVC} on the WordNet classes. Freq_{LVC} has reasonably good performance on the Levin

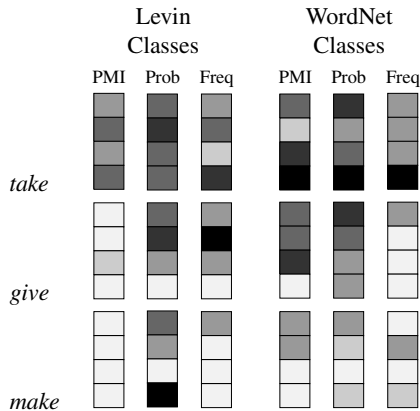


Fig. 1. Greyscale representation of the correlation coefficients (r_s) of each measure, across the three light verbs, for the four Levin and WordNet test classes

⁵ We used a significance cut-off of $p < .07$, since some tests achieved reasonably good correlations that were marginally significant at this level. Numerical r_s values are available in an unpublished TR at the authors' website.

classes, but relatively poor performance on WordNet classes. It is the most knowledge-poor of the measures, and also the most inconsistent performer, indicating that such simple methods are inappropriate for fine-grained acceptability measures in this task.

Examining the patterns in Figure 1 by light verb, we see that *take* achieves the best correlations on both Levin and WordNet expressions, followed by *give*, then *make*, which has particularly poor results. The poorer correlations with *give* and *make* may be partly due to the difficulty in rating them; note the lower interannotator agreement on expressions involving *give* and *make* (see §3.4).

Now looking at the patterns across the two semantic classifications, we note that the performance of Prob_{LVC} is overall comparable across the two, while PMI_{LVC} shows a marked improvement with WordNet, and Freq_{LVC} a marked decline. A closer look at the WordNet and Levin expressions reveals an interesting difference between the two: the average frequency of PNs in the WordNet classes is significantly higher than that of PNs in the corresponding Levin classes (26M vs. 8M, respectively). This observation provides evidence for the robustness of Prob_{LVC} , which appears to be less sensitive to frequency factors than PMI_{LVC} or Freq_{LVC} .

The effect of semantic classification on the measures also interacts with the specific light verb being used. We see that PMI_{LVC} is particularly inferior on Levin classes with *give* and *make*. It seems that expressions with *give* and *make* are less treatable as straightforward collocations, especially with lower frequency items.

4.2 Agreement Between Human Ratings and Statistical Measures

We now inspect the performance of our statistical measures when the coarser level of acceptability—‘poor’, ‘fair’, or ‘good’—is considered. For each measure, thresholds for dividing the continuous ratings into the discrete buckets are chosen such that the bucket sizes for development data match as closely as possible those of the human ratings. We then compare the measures on unseen test data to a chance baseline using the (linearly weighted) proportion of observed agreement with the “bucketized” human ratings.⁶ For most LV-class pairs, our chance baseline considers all items to be labelled ‘poor’, since that is the largest bucket size in the human ratings. The one exception is *take* with the Levin class of Verbs of Motion, in which the baseline assignment is ‘good’.

Table 2 presents the observed agreement scores (p_o) averaged across classes in each classification (Levin or WordNet); values of p_o above the baseline are in boldface. On Levin and WordNet expressions with *take* and *give*, both Prob_{LVC} and Freq_{LVC} mostly outperform the baseline, with Prob_{LVC} performing the best. On expressions involving *make*, however, none of the measures perform better than the baseline, reinforcing our initial hypothesis that *make* has differing properties from the other two light verbs. This coarser-grained level of acceptability shows a similar pattern across Levin and WordNet classes to that revealed by the correlation scores. Here again, PMI_{LVC} does better on WordNet classes, and Freq_{LVC} on Levin classes.

⁶ Because our ratings are skewed toward low values, slight changes in observed agreement cause large swings in kappa values (the “paradox” of low kappa scores with high observed agreement; Feinstein and Cicchetti, 1990). Since we are concerned with comparison to a baseline, observed agreement better reveals the patterns.

Table 2. Weighted observed agreement (p_o) between statistical measures and human judgments, over Levin and WordNet test expressions

Light Verb	Class Type	Chance Agreement	PMI _{LVC}	Prob _{LVC}	Freq _{LVC}
<i>take</i>	Levin	.78	.77	.85	.80
	WordNet	.81	.88	.86	.82
<i>give</i>	Levin	.80	.59	.77	.86
	WordNet	.75	.74	.80	.73
<i>make</i>	Levin	.87	.81	.82	.82
	WordNet	.85	.80	.74	.73

We look next at the productivity of these classes with the different light verbs. Because accurate assessment of class productivity depends on a measure having a reasonable level of agreement with the human ratings, we exclude the light verb *make* from the consideration of productivity.

4.3 Predicting Class Productivity

The ultimate goal of this study is to devise statistical measures that are good indicators of the semi-productivity of LVC formation for a semantic class of predicative nouns, given a particular light verb. One aspect of this is our proposed measures of the individual acceptability of a particular LV and PN combination as an LVC. We also want to assess the overall acceptability of a class of semantically related PNs, which indicates the productivity of the class with respect to the LV. Such class knowledge can be useful in extending our measures of acceptability to new or low frequency PNs. For example, if our measure predicts that the class of sound emission nouns, such as *groan* and *yell*, productively forms acceptable LVCs with *give*, the acceptability of an unseen LVC such as *give a moan* should be promoted.

The productivity level of a class is indicated by the proportion of PNs that form acceptable LVCs with the given LV. We consider an acceptable LVC to be one that is either ‘fair’ or ‘good’ according to human judgments. Thus, to investigate the appropriateness of each proposed measure as an indicator of class productivity, we compare (for each combination of LV and semantic class of PNs) that measure’s proportions of PNs in the ‘fair’ and ‘good’ buckets with those of the human judgments. The better the match between the two proportions, the better the measure at assessing class productivity.

Using the bucket thresholds described above, we determine the productivity level of each combination of LV (*take* and *give*) and semantic class. As an example, Table 3 presents the productivity of each WordNet test class for *take*, as determined by human judges and by each of the statistical measures. The variability across the classes according to the human judgments clearly shows that LVC acceptability is a class-based phenomenon.

We quantify the “goodness” of each measure for predicting productivity by calculating the divergence of its assessed productivity levels from those of the human judges, across all classes and light verbs. The divergence is measured as the sum of squared errors (SSE) between the two sets of numbers, averaged over all light verbs and classes.

Table 3. Proportion of acceptable expressions (those rated ‘fair’ or ‘good’) for *take* and each WordNet test class, as determined by human ratings and the statistical measures

Class	Human	PMI _{LVC}	Prob _{LVC}	Freq _{LVC}
WN- <i>knock</i>	.26	.40	.26	.40
WN- <i>check</i>	.14	.09	.26	.34
WN- <i>ring</i>	.09	.17	.23	.20
WN- <i>sail</i>	.46	.40	.37	.34

Table 4. Divergence of the productivity levels assessed by each of the statistical measures from those determined by human judges, averaged across Levin or WordNet classes

Class Type	SSE × 100		
	PMI _{LVC}	Prob _{LVC}	Freq _{LVC}
Levin	22.0	9.0	12.0
WordNet	5.7	3.5	8.1

Table 4 shows the average SSE values for each measure and each classification (Levin or WordNet). The lowest SSE (best match to human judgments) is shown in bold. For both classifications, Prob_{LVC} gives the closest predictions, i.e., the lowest SSEs. Notably, here we see overall better performance with WordNet than with Levin classes across all three measures.

4.4 Summary of Results

Our results indicate that Prob_{LVC}, the measure that incorporates more linguistic knowledge about LVCs, performs well at assessing acceptability at both the fine- and coarse-grained levels, according to the observed r_s and p_o values, respectively. Prob_{LVC} also accurately predicts the degree of productivity of a semantic class of complements with a light verb, according to the reported SSE values. PMI_{LVC} achieves reasonably good performance at both tasks when using WordNet classes, while Freq_{LVC} shows inconsistent performance across the tasks and the classifications.

In general, the classes generated from WordNet seem most useful in our tasks, especially when considering generalization of knowledge of possible LVC complements. Whether this is due to their higher item frequency noted above, or to the fact that our generation process draws on both nominal and verbal similarity, is an issue for future work.

5 Related Work

Compared to other types of MWEs, such as verb particle constructions, LVCs have not been studied computationally in great detail. Grefenstette and Teufel (1995) and Dras and Johnson (1996) examine the problem of choosing the best support verb (similar to an LV) for a given deverbal noun complement (similar to a PN). This is too restrictive for our purposes, since the same complement may form acceptable LVCs with different light verbs. Like us, Moirón (2004) links surface syntactic behaviour of LVCs

to their underlying semantics; however, her approach requires a great deal of manual analysis.

Sag et al. (2002) address the lexical encoding of LVCs more directly, but consider the selection of complements by an LV mainly idiosyncratic. Although they mention the use of selectional restrictions for LVs, they do not give an explicit means for determining the allowable combinations of semi-productive LVCs. Stevenson et al. (2004) particularly focus on the issue of semi-productivity of LVCs using Levin classes, but lack a clear proposal for extending their PMI-based acceptability measure to assess productivity. Here we propose a measure, Prob_{LVC} , that captures linguistic properties of LVCs relevant to their acceptability in a more appropriate manner, and explore its effectiveness across WordNet classes as well. We also show that Prob_{LVC} fits well with the human judgments on predicting the productivity level of both types of classes.

Our study of semantic classes is related to the idea of substitutability in other types of MWEs, i.e., substituting part of an MWE with a semantically similar word to determine the productivity of the expression (McCarthy et al., 2003; Lin, 1999; Villavicencio, 2003). However, the approach in this work differs not only in focusing on LVCs, but also in its goal of quantifying degree of acceptability of an expression in order to more precisely assess productivity. Moreover, a contribution of this paper is the investigation of different classifications and their impact on performance of our measures.

6 Conclusions

We have developed three statistical measures of the acceptability of light verb constructions, for use in automatically determining the allowable complements of a light verb. In comparisons against human judgments, we find that the Prob_{LVC} measure, which incorporates some linguistic insight within a probabilistic formulation, performs best and most consistently overall, for both fine- and coarse-grained assessment of acceptability. The results demonstrate that LVCs are best treated as more than simple collocations. Moreover, estimation of the Prob_{LVC} measure requires only simple extraction techniques over non-marked-up web data.

Our findings also show that Prob_{LVC} yields an accurate assessment of the productivity of a class of semantically related nouns as potential complements of a light verb. Due to the semi-productive nature of LVCs, such an assessment is crucial for generalizing the knowledge in a computational lexicon to previously unseen potential complements.

Given the crosslinguistic prominence of light verb constructions, our future work aims to extend these techniques to similar constructions in languages other than English. Moreover, while we have focused here on LVCs, we believe that similar techniques can be useful in dealing with other semi-productive MWEs, especially other types of phrasal verbs which are crosslinguistically frequent as well.

References

- Miriam Butt. 2003. The light verb jungle. <http://edvarda.hf.ntnu.no/ling/tross/Butt.pdf>.
- Kenneth W. Church, William Gale, Patrick Hanks, and Donald Hindle. 1991. Using statistics in lexical analysis. In U. Zernik, editor, *Lexical Acquisition: Exploiting On-line Resources to Build a Lexicon*, pages 115–164. Lawrence Erlbaum Associates.

- Mark Dras and Mike Johnson. 1996. Death and lightness: Using a demographic model to find support verbs. In *Proceedings of the 5th ICCSNLP*, pages 165–172.
- Alvan R. Feinstein and Domenic V. Cicchetti. 1990. High agreement but low Kappa: I. The problems of two paradoxes. *Journal of Clinical Epidemiology*, 43(6):543–549.
- Christiane Fellbaum, editor. 1988. *WordNet: An Electronic Lexical Database*. The MIT Press.
- Gregory Grefenstette and Simone Teufel. 1995. Corpus-based method for automatic identification of support verbs for nominalizations. In *Proceedings of the 7th EACL*, pages 98–103.
- Simin Karimi. 1997. Persian complex verbs: Idiomatic or compositional? *Lexicology*, 3(1):273–318.
- Kate Kearns. 2002. Light verbs in English. Manuscript.
- Frank Keller and Maria Lapata. 2003. Using the web to obtain frequencies for unseen bigrams. *Computational Linguistics*, 29(3):459–484.
- Beth Levin. 1993. *English Verb Classes and Alternations, A Preliminary Investigation*. University of Chicago Press.
- Dekang Lin. 1999. Automatic identification of non-compositional phrases. In *Proceedings of ACL-99*, pages 317–324.
- Diana McCarthy, Bill Keller, and John Carroll. 2003. Detecting a continuum of compositionality in phrasal verbs. In *Proceedings of the ACL-SIGLEX Workshop on Multiword Expressions*, pages 73–80.
- Tadao Miyamoto. 2000. *The Light Verb Construction in Japanese: the role of the verbal noun*. John Benjamins Publishing Company.
- M. Begoña Villada Moirón. 2004. Discarding noise in an automatically acquired lexicon of support verb constructions. In *Proceedings of the 4th LREC*.
- John Newman. 1996. *Give: A Cognitive Linguistic Study*. Mouton de Gruyter.
- Ivan A. Sag, Timothy Baldwin, Francis Bond, Ann A. Copestake, and Dan Flickinger. 2002. Multiword expressions: A pain in the neck for NLP. In *Proceedings of the 3rd CICLING*, pages 1–15.
- Suzanne Stevenson, Afsaneh Fazly, and Ryan North. 2004. Statistical measures of the semi-productivity of light verb constructions. In *Proceedings of the ACL'04 Workshop on Multiword Expressions*, pages 1–8.
- Peter D. Turney. 2001. Mining the web for synonyms: PMI-IR versus LSA on TOEFL. In *Proceedings of the 12th ECML*, pages 491–502.
- Aline Villavicencio. 2003. Verb-particle constructions in the World Wide Web. In *Proceedings of the ACL-SIGSEM Workshop on the Linguistic Dimensions of Prepositions and their use in Computational Linguistics Formalisms and Applications*.
- Anna Wierzbicka. 1982. Why can you Have a Drink when you can't *Have an Eat? In *The Semantics of Grammar*, pages 293–358. John Benhamins Publishing Co.

Web-Based Measurements of Intra-collocational Cohesion in Oxford Collocations Dictionary*

Igor A. Bolshakov¹ and Sofía Natalia Galicia-Haro²

¹Center for Computing Research (CIC),
National Polytechnic Institute (IPN), Mexico City, Mexico
igor@cic.ipn.mx

²Faculty of Sciences,
National Autonomous University of Mexico (UNAM),
Mexico City, Mexico
sngh@fciencias.unam.mx

Abstract. Cohesion between components of collocations is already acknowledged measurable by means of the Web, and cohesion measurements are used for some applications and extraction of new collocations. Taking a specific cohesion criterion *SCI*, we performed massive evaluations of collocate cohesion in Oxford Collocations Dictionary. For three groups of modificative collocations (adjective—noun, adverb—adjective, and adverb—verb) *SCI* distributions proved to be one-peaked and compact, with rather close mean values and standard deviations. Thus we suggest a reliable numeric criterion for extraction of collocations from the Web.

1 Introduction

Let us transitorily define collocations as syntactically linked and semantically compatible pairs of content words. They are rather specific for each language, so electronic collocation databases compiled beforehand are needed for many applications (text editing, foreign language learning, syntactic analysis, word sense disambiguation, detection & correction of errors etc.). Though the tools for automatic collocation extraction are being developed more than 15 years [7], large electronic collocation databases do not exist to the date for well-known languages.

The Web is acknowledged now as a huge corpus for automatic collocation extraction and this it is supposed possible with a numeric criterion of coherence between collocates [2, 6]. An application of corpus-oriented criteria to Web statistics theoretically is not grounded, since the Web counts occurrences and co-occurrences in pages, not words. Since a theory allowing to recalculate the numbers of relevant pages to the numbers of words occurred in them does not exist nowadays, we are free to use both formulas recommended for corpuses, re-conceptualizing page numbers as word numbers, and any analogous formulas operating by numbers of relevant pages.

In this paper we apply a criterion analogous to the well-known mutual information [3] for evaluation of inner coherence for 2100 modificative collocations arbitrary

* Work done under partial support of Mexican Government (CONACyT, SNI) and CGEPI-IPN, Mexico.

picked from Oxford Collocations Dictionary [5]. OCD is the only large collocation collection for English (ca. 150,000 units) in printed form. It was tested against British National Corpus. We plot distributions of values of a selected cohesion criterion for ‘modifier—modified’ collocations in order to see if these distributions are compact enough. If so, we could be sure that the method of British lexicographers is quantitatively grounded in other way, and our criterion suits for automatic extraction of collocations from the Web—for English and other languages. Additionally, we explore if qualitative adjectives—an important group of modifiers for nouns—form collocations more cohesive than adjectives in general.

2 Possible Criteria of Cohesion

The well-known criterion of cohesion of words P_1 and P_2 in a corpus sized S words is Mutual Information [2]:

$$MI(P_1, P_2) \equiv \log \frac{S \cdot N(P_1, P_2)}{N(P_1) \cdot N(P_2)},$$

where $N()$ is number of the entity in parentheses met through the corpus. If MI is greater than a threshold, the syntactically linked pair is considered collocation.

For Web evaluations, we can re-conceptualize $N()$ as numbers of relevant pages, and S as the page total under the search engine’s control. Since repeated evaluation of S is an additional tedious task, we can use instead the occurrence number N_{max} of pages with the word most frequent in the engine (in English it is *the*), thus obtaining Modified Mutual Information criterion (the constants k_1 and k_2 are commented later):

$$MMI(P_1, P_2) \equiv k_1 \log_2 \frac{k_2 \cdot N_{max} \cdot N(P_1, P_2)}{N(P_1) \cdot N(P_2)}.$$

Instead of MI and MMI , we prefer Stable Connection Index SCI that depends only on $N(P_1, P_2)$ and $N(P_1) \cdot N(P_2)$:

$$SCI(P_1, P_2) \equiv 16 + \log_2 \frac{N(P_1, P_2)}{\sqrt{N(P_1) \cdot N(P_2)}},$$

where the additive constant 16 and the logarithmic base 2 are chosen to allocate the majority of SCI values for collocations of various languages in the interval [0...16]. Similarly to MI and MMI , SCI is scalable, i.e. it conserves its value when all $N()$ change proportionally. This feature is very important, since Web statistics fluctuate synchronously and have ever-increasing trend.

As it was shown in [1], SCI and MMI with specific k_1 and k_2 give very close results for a rather big experimental set. Considering SCI independence of the corpus size, we can steadily use it to evaluate collocate cohesion in other experimental sets too.

3 Cohesion Evaluations with Oxford Collocations Dictionary

We took arbitrarily ca. 2,100 OCD collocations of modificative type with nouns, adjectives and verbs as the modified collocates. The majority of modifiers for nouns are

adjectives (*unlawful act*, *wrong attitude*), but some collocations with attributively used preceding nouns were taken too (*garlic bread*, *business call*). Indeed, their semantic role is similar to relational adjectives. Such modifiers usually come in texts just before the modified noun, and both collocate are invariable, so we easily obtain statistics for the bigram by one Web query.

The modifiers for adjectives and past participles in their adjectival role are adverbs (*quite easy*, *elaborately carved*). Again, these collocates are invariable and adjacent.

Verbs are inflective, and modifying adverbs can stay just before them just after or be distant to the right. Hence, to exhaustively evaluate cohesion between a form of the verb *assist* and the adverb *actively*, we have to test the bigrams “*actively assist*” and “*assist actively*”, and the distant pair “*assist * actively*”, where * is the Web wild card operator omitting several words. Then we have to repeat queries with all forms of *assist* and take maximal *SCI* as cohesion measure for the whole paradigm.

The *SCI* distributions for collocations of adjective—noun type (1025 units), adverb—adjective type (581 units), and adverb—verb type (498 units) are given in the Figure 1. All three are compact and unimodal (one-peaked), their standard deviations *D* are equal, while their mean values *M* are rather close to each other.

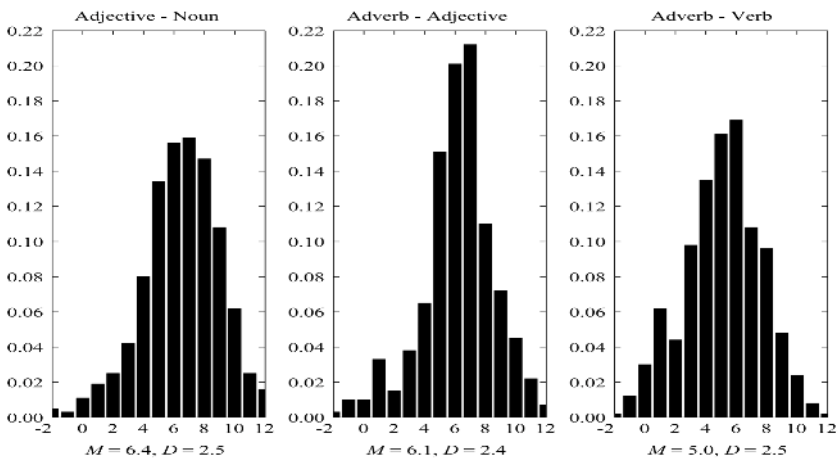


Fig. 1. *SCI* distributions for three types of collocations

The OCD collocations with negative *SCI* comprise only 0.9%, 1.3%, and 1.4% of the subgroups. So the compact distributions in the interval (0...12) allow taking *SCI* as a numeric substitute of lexicographers’ intuition: we may consider collocation any syntactically linked combination of content words that has positive *SCI* value.

4 Collocations with Qualitative Adjectives

Many adjectives in European languages are qualitative with very distinctive features:

- Corresponding adverbs are simply derived from them, e.g. *clear* → *clearly*;
- Their meaning is gradable and they have morphologic grades of comparison;

- Some of them are values of Lexical Functions by I. Mel' uk [4]. LF values express habitual senses but are to be selected depending on syntactically linked words (LF arguments). E.g., the sense 'intensive' for the noun *tee* is expressed by the adjective *strong*, not *mighty*, *powerful*, etc. So corresponding collocations are idiomatic, in comparison with free word combinations like *Indian tee*.

We have manually marked adjective—noun collocations with qualitative adjectives (they were 661 of 1025) and compare their *SCI* distribution with that of the whole set. The results ($M=6.2$, $D=2.2$ for qualitative adjectives *Vs.* $M=6.4$, $D=2.5$ for the whole set) show a negligible difference. So we cannot consider qualitative adjectives—and LF values among them—as more cohesive than other modifiers for nouns.

5 Conclusion

We used a specific measure for cohesion between components of collocations in the Web. A syntactically linked combination of content words is considered collocation if the proposed measure is positive. Our measure was tested against 2,100 collocations from Oxford Collocations Dictionary. Only each hundredth collocation of our experimental set did not pass the threshold test. Our criterion seems useful for extracting collocations from the Web. It may be used also in various applications of computational linguistics.

References

1. Bolshakov, I.A., E.I. Bolshakova. Measurements of Lexico-Syntactic Cohesion by means of Internet. *LNAI 3789*, Springer, 2005, p. 790-799.
2. Keller, F., M. Lapata. Using the Web to Obtain Frequencies for Unseen Bigram. *Computational linguistics*, V. 29, No. 3, 2003, p. 459-484.
3. Mitkov, R. (Ed.). *The Oxford Handbook of Computational Linguistics*. OU Press, 2003.
4. Mel' uk, I. Phrasemes in Language and Phraseology in Linguistics. In: M. Everaert *et al.* (Eds.) *Idioms: Structural and Psychological Perspectives*. Lawrence Erlbaum Associates Publ., Hillsdale, NJ / Hove, UK, 1995, p. 169-252.
5. *Oxford Collocations Dictionary for Students of English*. OU Press, 2003.
6. Seretan, V., L. Nerima, E. Wehrli. Using the Web as a Corpus for the Syntactic-Based Collocation Identification. Proc. Int. Conf. *Language Resources and Evaluation (LREC 2004)*, Lisbon, Portugal, 2004, p.1871-1874.
7. Smadja, F. Retrieving Collocations from text: Xtract. *Computational Linguistics*. Vol. 19, No. 1, 1990, p. 143-177.

Probabilistic Neural Network Based English-Arabic Sentence Alignment

Mohamed Abdel Fattah, Fuji Ren, and Shingo Kuroiwa

Faculty of Engineering, University of Tokushima,
2-1 Minamijosanjima, Tokushima, Japan 770-8506
{mohafi, ren, kuroiwa}@is.tokushima-u.ac.jp

Abstract. In this paper, we present a new approach to align sentences in bilingual parallel corpora based on a probabilistic neural network (P-NNT) classifier. A feature parameter vector is extracted from the text pair under consideration. This vector contains text features such as length, punctuation score, and cognate score values. A set of manually aligned training data was used to train the probabilistic neural network. Another set of data was used for testing. Using the probabilistic neural network approach, an error reduction of 27% was achieved over the length based approach when applied on English-Arabic parallel documents.

1 Introduction

Recently, much work has been reported on sentence alignment using different techniques [1]. Length-based approaches (length as a function of sentence characters [2] or sentence words [3]) were the most interesting. These approaches work quite well with clean input, such as the Canadian Hansards corpus, however they do not work well with noisy document pairs. Moreover, these approaches require that the paragraph boundaries be clearly marked, which is not the case for most of document pairs. Cognate approaches have also been proposed and have been combined with length-based approaches to improve alignment accuracy [4, 5]. They have used sentence cognates such as digits, alphanumerical symbols, punctuation, and alphabetical words. However both of Simard and Thomas did not take the text length between two successive cognates (Simard case) or punctuations (Thomas case) into account which increased the system confusion that leads to execution time increase and accuracy decrease (we have avoided this drawback in this work).

In this paper we present a non-traditional approach for the sentence alignment problem. In the sentence alignment problem, we may have 1-0 (One English sentence does not match any of the Arabic sentences), 0-1, 1-1, 1-2, 2-1, 2-2, 1-3 and 3-1. There may be more categories in bi-texts, however they are rare, hence we consider only the previous mentioned categories. As illustrated above, we have eight sentence alignment categories. Hence, we can consider sentence alignment as a classification problem. This classification problem may be solved by using a probabilistic neural network classifier.

2 English–Arabic Text Features

There are many features that can be extracted from any text pair. The most important feature is text length, since Gale [2] achieved good results using this feature.

The second text feature is punctuation symbols. We can classify punctuation matching into the following categories: A. 1-1 matching type, where one English punctuation mark matches one Arabic punctuation mark, similarly there are: B. 1-0 and C. 0-1 matching types.

The probability that a sequence of punctuation marks $AP_i = Ap_1Ap_2.....Ap_i$ in a text of the Arabic language translates to a sequence of punctuation marks $EP_j = Ep_1Ep_2.....Ep_j$ in a text in the English language is $P(AP_i, EP_j)$. The system searches for the punctuation alignment that maximizes the probability overall possible alignments given a pair of punctuation sequences corresponding to a pair of parallel sentences from the following formula:

$$\arg \max_{AL} P(AL | AP_i, EP_j) \quad (1)$$

Since “AL” is a punctuation alignment. Punctuation symbols for 1000 English – Arabic sentence pairs were manually aligned to calculate each punctuation mark pair probability. After specifying the punctuation alignment that maximizes the probability overall possible alignments given a pair of punctuation sequences, the system calculates the punctuation compatibility factor for the text pair under consideration as:

$$\gamma = \frac{c}{\max(m, n)}$$

Where γ = the punctuation compatibility factor, c = the number of

direct punctuation matches, n = the number of Arabic punctuation marks, m = the number of English punctuation marks. The punctuation compatibility factor is considered as the second text pair feature.

The third text pair feature is cognates. Many UN and scientific Arabic documents contain some English words and expressions. These words may be used as cognates. We define the cognate factor (cog) as the number of common items in the sentence pair. When a sentence pair has no cognate words, the cognate factor is 0.

3 The Proposed Sentence Alignment Model

The proposed sentence alignment model consists of two modes of operations:

- * Training mode where features are extracted from 7653 manually aligned English-Arabic sentence pairs and used to train the probabilistic neural network (P-NNT).
- * Testing mode where features are extracted from the testing data and aligned using the P-NNT. (Training and testing data are UN document pairs extracted from the Internet archive).

The input pattern X is propagated through the P-NNT in the following way: The probability distribution function (PDF) for a feature vector (X) to be of a certain category (class A for example as one of the 8 output categories) is given by:

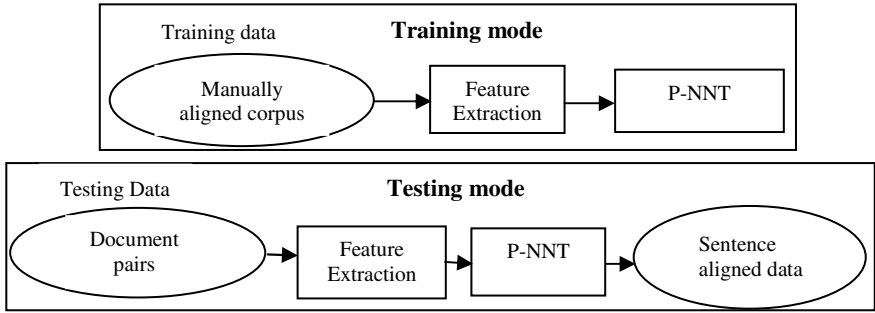


Fig. 1. The proposed sentence alignment model

$$f_a(X) = 1/(2\Pi)^{p/2} \sigma^p (1/n_a) \sum_{i=1}^{n_a} \exp(-(X - Y_{ai})^\tau (X - Y_{ai})/2\sigma^2) \quad (2)$$

Where: $f_a(X)$ = the value of the PDF for class A at point X , X = test vector to be classified, i = training vector number, p = the training vector size, n_a = number of training vectors in class A, $Y_{ai} = i^{th}$ training vector for class A, τ = transpose, σ = the standard deviation of the Gaussian curves used to construct the PDF.

Introducing a term to represent the relative number of trials in each category (n_a/n_{total}) simplifies the expression. Hence $(1/n_a)$ term is canceled out. Terms common to all classes such as $1/(2\Pi)^{p/2}$, σ^p , and n_{total} can also be eliminated. Hence the classifier can be expressed as follows: A feature parameter X to belong to a category (r); the following equation must be verified:

$$\sum_i \exp((X^\tau Y_{ri} - 1)/\sigma^2) \geq \sum_i \exp((X^\tau Y_{si} - 1)/\sigma^2) \quad (3)$$

Every neuron in layer one of the neural network receives the elements of vector X to be classified as input. The weight matrix scaling these inputs is formed by the elements of the training vectors divided by the constant (σ^2). The first layer has a bias of $-1/\sigma^2$. The inputs of layer one are summed, producing $(X^\tau Y_{ri} - 1)$. Then, this value is divided by σ^2 and the exponential transfer function is applied, so the outputs of layer one are $\exp((X^\tau Y_{ri} - 1)/\sigma^2)$ and $\exp((X^\tau Y_{szi} - 1)/\sigma^2)$ where (sz) represents the rest categories for $z=1, 2, 3, 4, 5, 6, 7$.

The second layer has eight neurons: each one is associated with a specific output mentioned before. The inputs from the first layer of each category are summed to produce the expressions $\sum_{i=1}^m \exp((X^\tau Y_{ri} - 1)/\sigma^2)$ and $\sum_{i=1}^m \exp((X^\tau Y_{szi} - 1)/\sigma^2)$ for all values of z . The output of each neuron represents the probability that the vector X belongs to its class. The neuron in layer 2 with the greatest output determines the classification of the vector.

4 Experimental Results

A dynamic programming framework was constructed to conduct experiments using the length based approach as a baseline experiment in order to compare the results with the proposed system. Table 1 illustrates the results using 1200 English–Arabic sentence pairs as a test set. Table 1 also illustrates the results when the proposed approach is applied on the 1200 English – Arabic sentence pairs. It is clear from the table that there is an improvement in accuracy using the proposed approach.

Table 1. The results using length based approach & the proposed approach

Category	Frequency	Length based approach	Probabilistic neural network
		% Error	% Error
1-1	1099	4.9%	3.6%
1-0, 0-1	2	100%	100%
1-2, 2-1	88	15.9%	10.2%
2-2	2	50%	50%
1-3, 3-1	9	66%	55.5%
Total	1200	6.4%	4.7 %

5 Conclusions

In this paper, we have investigated the use of a probabilistic neural network on sentence alignment. We have applied our new approach on a sample English – Arabic parallel corpus. Our approach outperforms the length base approach. The approach used feature extraction criteria which give researchers an opportunity to use many varieties of these features based on the used language pair and text type.

Acknowledgment. This research was partially supported by the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Scientific Research (B), 14380166 and 17300065, Exploratory Research 17656128 in 2005, International Communications Foundation (ICF).

References

1. Wang, X., Ren, F.,: Chinese-Japanese Clause Alignment. CICLing, LNCS 3406, (2005), 400-412.
2. Gale, W. A., Church, K. W.: A program for aligning sentences in bilingual corpora. Computational Linguistics, vol. 19, (1993) 75-102.
3. Brown, P., Lai, J., Mercer, R.: Aligning sentences in parallel corpora. In Proceedings of the 29th annual meeting of the association for computational linguistics, Berkeley, CA, USA, (1991).
4. Simard, M., Foster, G., Isabelle, P.: Using cognates to align sentences in bilingual corpora. Proceedings of TMI92, Montreal, Canada, (1992) 67-81.
5. Thomas, C., Kevin, C.: Aligning Parallel Bilingual Corpora Statistically with Punctuation Criteria. Computational Linguistics and Chinese Language Processing, vol. 10 (1), (2005), 95-122.

Towards the Automatic Lemmatization of 16th Century Mexican Spanish: A Stemming Scheme for the CHEM

Alfonso Medina-Urrea

GIL, Instituto de Ingeniería, UNAM,
Ciudad Universitaria, 04510 Coyoacán, DF, Mexico
`amedinau@ii.unam.mx`

Abstract. Two of the problems that should arise when developing a stemming scheme for diachronic corpora are: (1) morphological systems of natural languages may vary throughout time, and these changes are normally not documented sufficiently; and (2) they exhibit very diverse orthographic characteristics. In this short paper, a stemming strategy for a diachronic corpus of Mexican Spanish is briefly described, which partially faces up to these problems. Success rates of the method are contrasted to those of a Porter stemmer.

1 Introduction

Diachronic corpora for the Spanish language have become available for various kinds of research. Two widely known corpora are the RAE's *Corpus diacrónico del español*, CORDE (<http://www.rae.es/>), and Mark Davies' *Corpus del español* (<http://www.corpusdelespanol.org/>). Recently, a first version of the *Corpus histórico del español de México*, CHEM (<http://www.iling.unam.mx/chem/>), became available to the public for the study of the Spanish used in Mexico from the arrival of Europeans to the 19th century.

Many tools for the exploitation and analysis of corpora require a lemmatization process, which is often reduced to simple stemming or graphical word truncation to eliminate inflections. Simple techniques such as the Porter algorithm [1] are regularly applied to corpora of many languages, but they require knowledge of their morphology. Fortunately, in comparison with other languages, Spanish morphology has changed relatively little during the last five centuries. So, a Porter stemmer for today's Spanish could presumably be applied to those centuries in order to accomplish inflection removal. However, given that techniques exist which can be used for stemming without having to code morphological knowledge into the algorithm, it is worthwhile to compare them to the Porter method in order to appreciate what scheme would be better for the CHEM.

In this short paper, the stemming strategy devised for this corpus is described and contrasted with an implementation of the Porter stemmer.¹ The strategy

¹ Various implementations of the Porter algorithm for Spanish are available (based on <http://snowball.tartarus.org/>). In this experiment a version for contemporary Spanish developed at GIL-IINGEN-UNAM, was used.

proposed is based on automatic segmentation techniques previously tested for synchronic corpora of Spanish,² Czech, Chuj and Ralámuli.

2 Entropy and Economy Based Automatic Stemming

The stemmer basically examines each n -gram of each graphical word, estimating uncertainty at each point, by measuring Shannon's entropy, and determining economy relations, by counting corpus evidence of syntagmatic and paradigmatic relations of word fragments. The techniques to accomplish this were sufficiently exposed for Czech and Chuj corpora in [2, 3, 4]. Also, they were presented with more detail for Spanish in [5]. In essence, each word segmentation yields, according to corpus evidence, entropy and economy measurements of how likely a morphological border is bound to occur at that segmentation. The highest values are expected to occur at the borders between bases and affixes, so they are taken as criteria to determine the best morphological segmentation within the graphical word. The stemmer simply eliminates the assumed suffix sequence.

3 Stemming a 16th Century Sample of Mexican Spanish

The target corpus for the stemming experiment is constituted by 95 CHEM documents from the 16th century. These documents comprise around 257,385 graphical token words, which correspond approximately to 15,834 graphical word types. Capitalized words were assumed to be proper names. These and words of length less than four were not stemmed. Upon examination of the documents, it becomes obvious that the orthographic idiosyncrasies of the 16th century cause referents to have multiple graphical forms (*e.g. admynistración, admynystracjon, administraçjon, adminystracjón*, etc.). Thus, although some unwanted homophony for short items may be introduced, it is clear that the text should be normalized in order to conflate, into a lesser number of graphical word types, several orthographical forms sharing a referent (*e.g. merced, merçed* → *merced*; *cantava, cantaba* → *cantaba*; *yndio, jndio* → *indio*, etc.). Therefore a simple set of rules to modify some characters was introduced to enhance grapheme-phoneme correspondence. These rules³ appear in Table 1.

To be able to stem, the stemmer determines suffix sequences from the corpus estimating, as mentioned above, entropy and economy measurements. Using the corpus without the grapheme-phoneme correspondence rules, the method yielded 565 suffix strings. Then, given that stress marks distinguish verbal inflection morphemes (towards the end of words), the rules of Table 1 were applied to the corpus respecting last syllable stress marks (rendering 487 relevant suffix strings) and omitting all stress marks (rendering 470 suffix strings). The suffix

² An unpublished experiment indicates that these techniques perform, for contemporary Mexican Spanish, slightly better than the Porter method.

³ There are, of course, problems that arise when applying these rules, but there is no space here to discuss them.

Table 1. Character modifications (grapheme-phoneme correspondence)

rules	ph.	contexts
'h' → ϵ	-	all contexts.
'v' → 'b'	[b]	all contexts.
'ch' → 'ç'	[ç]	all contexts.
'rr' → 'r̄'	[r̄]	all contexts.
'ç', 'z', 'c' → 's' ^a	[s]	'çe', 'çi'; every 'z'; 'ce', 'ci'.
'c', 'qu' → 'k'	[k]	'ca', 'que', 'qui', 'co', 'cu'.
'g' → 'j'	[h]	'ge', 'gi'.
'gu' → 'g'	[ɣ]	'gue', 'gui'.
'y' → 'i'	[i]	end of syllable preceded by vowel ('ay', 'ey', 'oy', 'uy'); or word beginning, before consonant ('yn', 'yd', etc.).
'j' → 'i'	[i]	between consonants or consonant and vowel; or word beginning before consonant ('jn', 'jd', etc.).
'r' → 'r̄'	[r̄]	beginning of word; or preceded by syllable ending with 'n', 'l' or 's'.

^a By the 16th century, the Spanish stridents system was collapsing and most probably Castilian [θ] never made it to America; see [6].

sequences discovered by applying the rules and keeping last syllable stress marks were used for the stemming experiment, which, as mentioned above, consisted of finding the best segmentation of each word and then eliminating its right side.

4 Evaluation

For this evaluation, one of the 16th century documents was picked randomly. Then, both stemmers were applied to it and a specialist judged separately whether the segmentations were morphologically appropriate or not. Table 2 shows success rates for both stemmers applied to the selected document (12,424 token words). The CHEM column shows success percentages for the stemmer developed for this corpus. Since all errors occurred only once (*i.e.* in types of frequency one), the rate for types and the one for tokens is the same. The last two columns exhibit results for the Porter stemmer. As one would expect, rates improved somehow when the orthographic normalization rules devised for the CHEM (see Table 1) were applied before the Porter stemmer. Still, rates for the entropy-economy stemmer are much closer to 1.0.

Table 2. Success Rates for 16th Century

	CHEM	Porter	
		without rules	with rules
types	0.9932	0.9328	0.9597
tokens	0.9932	0.8926	0.9328

5 Closing Remarks and Future Work

It is interesting that a Porter stemmer for contemporary Spanish would get such good rates when applied to a document belonging to an earlier stage of Spanish. This corroborates the observation that the morphology of this language has changed relatively little in the last centuries. One might ask, how far back can a method developed for one stage of a language be applied to earlier stages of the same language? The answer is obviously language dependent. More innovative languages like French or English have gone through considerably more changes in lesser time, so very likely Porter stemming based on their current states would be less adequate than it appears to be for Spanish. Another question would be, since the Porter stemmer obtained a type success rate of 0.96, why not just use such method, instead of going through the overhead of calculating entropies and finding affix economical relations, especially when the latter is more expensive? At least for this experiment, such expensive method did show an improvement, reaching a success rate of 0.99. It is not clear whether that small improvement is due to the robustness of the entropy-economy stemmer or simply to morphological differences between two stages of Spanish. At any rate, it is more desirable to apply the best existing method than to develop a Porter stemmer specifically designed for the 16th century. There is, however, lots of room for improvement. The stemming scheme presented will be improved and applied to the 17th, 18th and 19th centuries of the CHEM. This would be a first step towards lemmatization of all the corpus documents, a step necessary for the development of future exploitation tools.

Acknowledgments

The work reported in this paper has been supported by a DGAPA UNAM grant for PAPIIT Project IN400905.

References

1. PORTER, M.F.: “An Algorithm for Suffix Stripping”. *Program* 14(3) (1980) 130–137
2. MEDINA-Urrea, A., HLAVÁČOVÁ, J.: “Automatic Recognition of Czech Derivational Prefixes”. In: *Proceedings of CICLing 2005*. Volume 3406 of *Lecture Notes in Computer Science*. Springer, Berlin/Heidelberg/New York (2005) 189–197
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. MEDINA-Urrea, A., ALVARADO García, M.: “Análisis cuantitativo y cualitativo de la derivación léxica en ralamuli”. In: *Primer Coloquio Leonardo Manrique*, Mexico, Conaculta-INAH (2004)
5. MEDINA-Urrea, A.: “Automatic Discovery of Affixes by Means of a Corpus: A Catalog of Spanish Affixes”. *Journal of Quantitative Linguistics* 7(2) (2000) 97–114
6. HARRIS, J.: “Historical Excursus: Reflexes of the Medieval Stridents”. In: *Spanish Phonology*. MIT Press, Cambridge, Mass. (1969) 189–206

Word Frequency Approximation for Chinese Without Using Manually-Annotated Corpus

Maosong Sun¹, Zhengcao Zhang¹, Benjamin Ka-Yin T'sou², and Huaming Lu³

¹ The State Key Laboratory of Intelligent Technology and Systems,
Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China
sms@tsinghua.edu.cn

² Language Information Sciences Research Center, City University of Hong Kong
rlbtsou@cityu.edu.hk

³ School of Business, Beijing Institute of Machinery, Beijing 100085, China
huaminglu@sohu.com

Abstract. Word frequencies play important roles in a variety of NLP-related applications. Word frequency estimation for Chinese is a big challenge due to characteristics of Chinese, in particular word-formation and word segmentation. This paper concerns the issue of word frequency estimation in the condition that we only have a Chinese wordlist and a raw Chinese corpus with arbitrarily large size, and do not perform any manual annotation to the corpus. Several realistic schemes for approximating word frequencies under the framework of STR (frequency of string of characters as an approximation of word frequency) and MM (Maximal matching) are presented. Large-scale experiments indicate that the proposed scheme, MinMaxMM, can significantly benefit the estimation of word frequencies, though its performance is still not very satisfactory in some cases.

1 Introduction

Word frequencies play important roles in a variety of NLP-related applications, for example, TF in information retrieval. The estimation of word frequencies for English is very easy – it can be done by running a simple program to count word occurrences in a (in fact, any arbitrarily huge) corpus. In case of Chinese where no explicit word boundaries like spaces exist between words in texts, the task becomes very complex.

In general, a fully correct word-segmented Chinese corpus is a prerequisite for calculating word frequencies (Liu 1973). However, we face two difficulties in this respect. The first one is such a ‘fully correct’ corpus, or, a corpus with ideal segmentation consistency, is extremely difficult to obtain due to a main characteristic of Chinese word-formation: the borders between morphemes, words, and phrases of Chinese are fuzzy in nature (Dai 1992; Chen 1994), though the definition for ‘word’ from the linguistic perspective seems very clear (Zhu 1982; Tang 1992). A large number of linguistic constituents could be regarded as words by some linguists whereas be regarded as phrases by others (even for a specific linguist, his feeling to some constituents may change in between from time to time), resulting in serious inconsistency in a manually word-segmented corpus. For example, a constituent,

‘猪肉’, may be thought of as either a compounding word, *pork*, or a phrase consisting of two single-character words ‘猪’(pig) and ‘肉’(meat), *meat of pig*. Thus, the word frequency of ‘猪肉’ could be pretty high if it is treated in the corpus in the former way, and could also be zero if treated in the latter way. The second difficulty is, manual annotation of a large-scale corpus (in our experience, to obtain a satisfactory frequency estimation for a medium-sized Chinese wordlist, a corpus with several hundred million characters, rather than several million characters, is needed) is labor-intensive and time-consuming, always failing to capture the rapid update of language usage well and in time, particularly in the context of the web and increasingly sophisticated and pervasive applications of Chinese NLP.

A more feasible strategy is therefore trying to estimate word frequencies without using a manually-annotated corpus. Basically, we have the following choices:

Alternative 1: Use a word segmenter to segment the corpus automatically, then obtain the approximated word frequencies from the resulting corpus. Theoretically, it would be best if we could use a very powerful word segmenter to obtain word frequencies as accurate as could be expected in terms of a machine-segmented corpus (Liu and Liang 1986). However, current state-of-the-art word segmenters are not satisfactory in performance, though tremendous efforts have been invested in it in the last two decades. In 1995, the ‘863’ High Technology Programme of China carried out an evaluation of Chinese word segmentation (Liu 1997). Several systems participated in the evaluation, and the highest accuracy for word segmentation is 89.4%. In 1998, a second round evaluation was done. The best accuracy for word segmentation is 87.4% (‘863’ High Technology Program of China et al. 1998). SIGHAN organized the First International Chinese Word Segmentation Bakeoff in 2003 (Sproat and Emerson 2003). The highest F-scores for word segmentation in the open test on four small-scale corpora (PK corpus with 34,955 word tokens, HK corpus with 17,194 word tokens, AS corpus with 11,985 word tokens, and CTB corpus with 39,922 word tokens) are 95.9%, 95.6%, 90.4% and 91.2% respectively. In the Second International Chinese Word Segmentation Bakeoff (Emerson 2005), the situation remains unchanged in nature, although the performance of word segmentation increases to some extent. As can be seen, for word segmentation, the most basic task in Chinese language processing, a satisfactory solution is still not quite within reach for researchers. The performance of segmentation can be further damaged if words whose frequencies are to be estimated are out-of-vocabulary words to the word segmenter. In addition, exploring more sophisticated word segmentation techniques may introduce more inconsistency in segmentation and thus generate more confusion in the estimation of word frequencies.

Alternative 2: Use ‘Maximal matching’ (MM), the most basic method for Chinese word segmentation, to segment the corpus automatically, then obtain the approximated word frequencies from the resulting corpus. Liu and Liang (1986) first used MM to handle large-scale texts. According to the direction of sentence scanning, MM can be further sub-categorized as forward MM (FMM, scanning the sentence from left to right) and backward MM (BMM, scanning the sentence from

right to left). Experimental results in Liang (1987) showed that MM is both effective (the error rate of MM ranges from once / 245 characters to once / 169 characters, provided that the wordlist is complete) and efficient (fast and easy to implement). Sun and T'sou (1995) distinguished four cases when FMM and BMM are both considered, and found that for 90.30% sentences, the outputs of FMM and BMM are identical and correct; for 9.24% sentences, the outputs of FMM and BMM are different, and only one is correct; for 0.41% sentences, the outputs of FMM and BMM are identical but incorrect; and for 0.05% sentences, the outputs of FMM and BMM are different, and both are incorrect. This observation provides a very strong evidence for supporting MM-based schemes to be reasonable estimations of word frequencies.

Another advantage of MM-based schemes is their high consistency in word segmentation.

Alternative 3: Use the frequency of a string of characters as an approximation of the word frequency of a constituent (Sun, Shen, and T'sou 1998). This statistic can be derived directly from any raw corpus. Obviously, its value is always larger than the value of word frequency for any word given a corpus. This scheme may over-estimate word frequencies seriously for some words (in particular for mono-syllabic words), but one of its good properties is it is free of any kind of word segmentation errors.

This paper concerns the issue of word frequency estimation in the condition that we only have a Chinese wordlist and a raw Chinese corpus. The size of the corpus can be arbitrarily large, for example, to the extreme, the collection of all texts available on the web. For the realistic consideration, we only account for alternative 2 and 3 at this stage. Consequently, two key questions come: are such approximations proper estimations of word frequencies? If not, can we find other, more reasonable and reliable methods of estimation, in which we still need not resort to any manual annotation? These will be the focus of the rest of the paper.

2 Data Set

In experiments throughout the paper, we use PDA9801-06, a manually word-segmented and part-of-speech-tagged corpus constructed from the People's Daily (January to June 1998 issues), which was developed by Institute of Computational Linguistics, Peking University, as a benchmark for word frequency estimation. PDA9801-06 contains a total of 10,930,237 Chinese characters. At the same time, a raw corpus, PDR9801-06, is derived from PDA9801-06 by ignoring all word boundary delimiters and part-of-speech information.

We collect all words from PDA9801-06, delete those with frequencies less than 10, and mix the surviving words with words from the CCWL which appear in PDA9801-06, forming a wordlist PCL with 58,571 entries. CCWL is a core wordlist of contemporary Chinese consisting of 92,843 entries and is expected to become a National Standard of China for text information processing (Sun, Wang et al. 2001). The distribution of PCL is shown in Table 1.

Table 1. Distribution of PCL

Word length	Subset of PCL	Number of words
1	PCL1	2,661
2	PCL2	36,872
3	PCL3	11,580
4	PCL4	7,002
5	PCL5	308
6	PCL6	77
7	PCL7	56
>=8	PCL8+	15
Total	PCL	58,571

3 Experimental Study on Word Frequency Approximation

3.1 The Approximation Schemes

As discussed in Section 1, we choose the following alternative schemes for consideration, because of their effectiveness, simplicity, and consistency:

- (1) STR (the frequency of string of characters)
- (2) FMM (Forward MM)
- (3) BMM (Backward MM)
- (4) Two weighted combinations of FMM and BMM, denoted FBMMComb and MinMaxMMComb respectively (See below for details)

A manually segmented corpus will serve as the golden standard, denoted GS (here, PDA9801-06).

In our experiments, for any word $w \in PCL$, we assign a series of frequencies with respect to the schemes:

- (1) $f_{GS}(w)$: frequency of word w in PDA9801-06 (golden standard).
- (2) $f_{STR}(w)$: frequency of string w in PDR9801-06.
- (3) $f_{FMM}(w)$: frequency of word w in PDR9801-06 segmented by FMM.
- (4) $f_{BMM}(w)$: frequency of word w in PDR9801-06 segmented by BMM.
- (5) $f_{FBMMComb}(w)$:

$$k \times f_{FMM}(w) + (1-k) \times f_{BMM}(w), k \in [0,1] \tag{1}$$

obviously,

$$f_{FBMMComb}(w) = f_{FMM}(w) \text{ when } k=1;$$

$$f_{FBMMComb}(w) = f_{BMM}(w) \text{ when } k=0.$$

$$(6) f_{MinMaxMMComb}(w):$$

$$k \times f_{Min}(w) + (1-k) \times f_{Max}(w), k \in [0,1] \tag{2}$$

where

$$f_{Min}(w) = \min \{ f_{FMM}(w), f_{BMM}(w) \}$$

$$f_{Max}(w) = \max \{ f_{FMM}(w), f_{BMM}(w) \}$$

3.2 Hypothesis Testing for the Schemes

For any scheme $SC \in \{STR, FMM, BMM, FBMMComb, MinMaxMMComb\}$, we can establish a hypothesis

$$H_0: SC=GS$$

$$H_1: SC \neq GS$$

The chi-square statistic is computed as a means of comparing SC with GS on PCL:

$$Q_{PCL}(SC, GS) = \sum_{w_i \in PCL} \frac{(f_{SC}(w_i) - f_{GS}(w_i))^2}{f_{GS}(w_i)} \tag{3}$$

The level of significance is arbitrarily set at $\alpha=0.05$, for which the critical value is supposed to be $\delta_{PCL}(SC, GS)$. If the chi-square statistic $Q_{PCL}(SC, GS)$ is smaller than $\delta_{PCL}(SC, GS)$, then we accept H_0 , signaling that SC is a good estimation of GS on PCL; otherwise, reject H_0 and accept H_1 , signaling that SC is a poor estimation of GS on PCL.

For the sake of comparisons hereafter, we use the average chi-square statistic:

$$AvgQ_{PCL}(SC, GS) = \frac{Q_{PCL}(SC, GS)}{|PCL|} \tag{4}$$

and the average critical value for the level of significance $\alpha=0.05$:

$$\frac{\delta_{PCL}(SC, GS)}{|PCL|} \tag{5}$$

3.3 Fixing the Factor k in FMM and BMM Combinations

We need to fix the factor k in schemes FBMMComb and MinMaxMMComb.

It is reasonable to assume that k is sensitive to word length. We compute

$$AvgQ_{PCLX}(SC, GS) = \frac{Q_{PCLX}(SC, GS)}{|PCLX|} \text{ for } k = 0, 0.1, 0.2, \dots, 0.9, 1, PCLX =$$

{PCL1, PCL2, ..., PCL7, PCL8+} and $SC = \{FBMMComb, MinMaxMMComb\}$ respectively, as shown in Fig. 1, 2, 3, 4, 5, 6 and 7 (the figure for PCL8+ is omitted because all associated AvgQ values are zero).

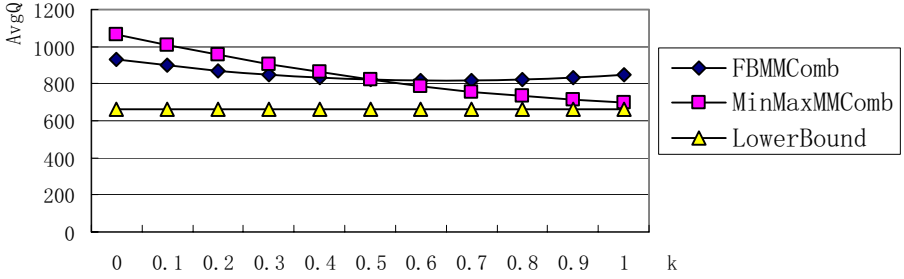


Fig. 1. AvgQs with k on PCL1

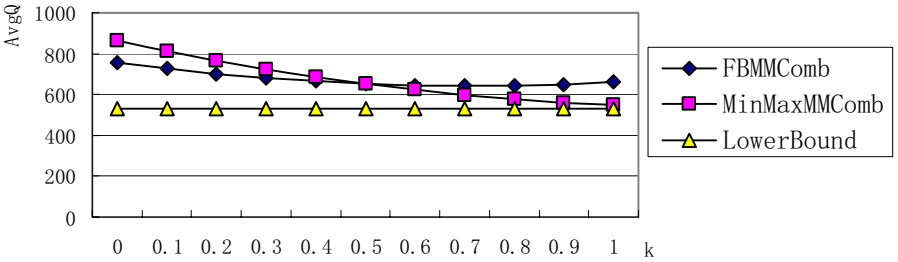


Fig. 2. AvgQs with k on PCL2

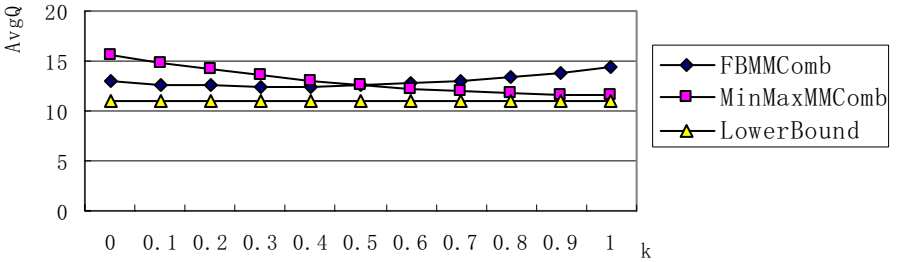


Fig. 3. AvgQs with k on PCL3

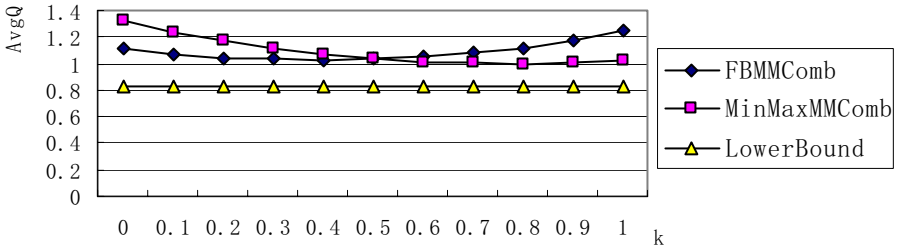


Fig. 4. AvgQs with k on PCL4

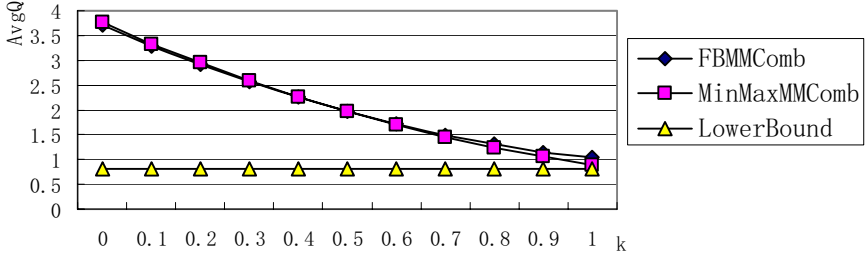


Fig. 5. AvgQs with k on PCL5

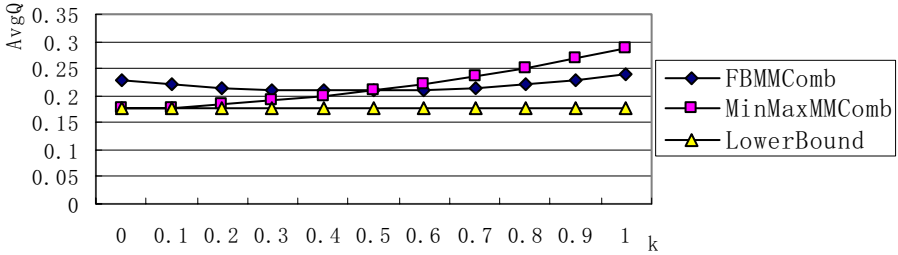


Fig. 6. AvgQs with k on PCL6

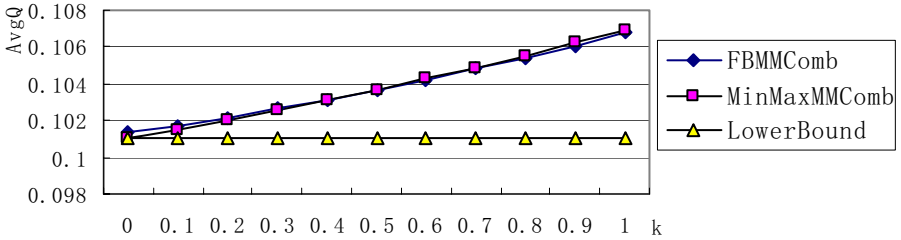


Fig. 7. AvgQs with k on PCL7

Note that in each of the figures:

(1) The leftmost and the rightmost points of the curve FBMMComb correspond to the average chi-square statistics of BMM and FMM;

(2) The curve LowerBound indicates the optimal solution expected when combining FMM and BMM in the context of GS and PCLX. For any w in PCLX, we define $f_{LowerBound}(w)$ to be $f_{FMM}(w)$ if $|f_{FMM}(w) - f_{GS}(w)| < |f_{BMM}(w) - f_{GS}(w)|$ otherwise $f_{BMM}(w)$. Summing $f_{LowerBound}(w)$ for all w and averaging the summation over PCLX, we get the value of LowerBound for PCLX:

$$AvgQ_{PCLX}(LowerBound, GS) = \frac{Q_{PCLX}(LowerBound, GS)}{|PCLX|} \quad (6)$$

The best k values with which the minimal AvgQs are achieved in cases PCL1 to PCL7 can be easily located in the figures, as demonstrated in Fig. 8.

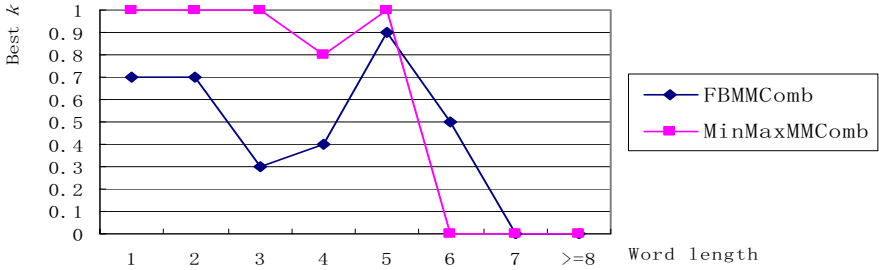


Fig. 8. Determination of the best k

Note an interesting phenomenon in Fig. 8. The curve FBMMComb vibrates quite strongly; while the curve MinMaxMMComb is relatively simple. The best k value for word length 4 using MinMaxMMComb (=0.8) might be regarded as an ‘accidental’ deviation in the region of word length from 1 to 5 (where the best k values are 1), because the AvgQ values for $k = 0.8$ and $k = 1$ of MinMaxMMComb are very close (0.997618 vs. 1.019498). We thus move the best k value from 0.8 to 1 for word length 4, resulting in an elegant pattern for MinMaxMMComb: for words with length less than 6, the best k is 1 otherwise it is 0.

FBMMComb and MinMaxMMComb using the best k values over all word lengths are considered to be two solutions for combining FMM and BMM. We name these two solutions FBMM and MinMaxMM respectively, as given in Table 2.

Table 2. The description of schemes FBMM and MinMaxMM

Word length	FBMM		MinMaxMM	
	Best k	Formula	Best k	formula
1	0.7	$0.7 * f_{FMM}(w) + 0.3 * f_{BMM}(w)$	1	$f_{Min}(w)$
2	0.7	$0.7 * f_{FMM}(w) + 0.3 * f_{BMM}(w)$	1	$f_{Min}(w)$
3	0.3	$0.3 * f_{FMM}(w) + 0.7 * f_{BMM}(w)$	1	$f_{Min}(w)$
4	0.4	$0.4 * f_{FMM}(w) + 0.6 * f_{BMM}(w)$	1	$f_{Min}(w)$
5	0.9	$0.9 * f_{FMM}(w) + 0.1 * f_{BMM}(w)$	1	$f_{Min}(w)$
6	0.5	$0.5 * f_{FMM}(w) + 0.5 * f_{BMM}(w)$	0	$f_{Max}(w)$
7	0	$f_{BMM}(w)$	0	$f_{Max}(w)$
>=8	0	$f_{BMM}(w)$	0	$f_{Max}(w)$

3.4 Evaluation of the Schemes

We now compare all schemes STR, FMM, BMM, FBMM and MinMaxMM.

AvgQ is a mixture of contributions from all words under consideration. The result of comparison on PCL is illustrated in Fig. 9, and the results on all PCLX are in

Fig. 10, 11, 12 and 13. Table 3 gives detailed information on AvgQ values for all schemes in all word length cases, supplemented with the average critical value at $\alpha=0.05$ as well as the status of H_0 for every case.

As can be seen in Fig. 9, MinMaxMM outperforms all the other schemes on PCL; STR is the worst, showing a striking contrast to the others. The difference between MinMaxMM and FMM, BMM, FBMM is quite significant (please refer to the last row in Table 3), though the presence of a very high STR peak in the figure makes this less readily apparent.

Refer to Fig. 10, 11, 12 and 13: MinMaxMM significantly outperforms all the other schemes on PCL1, PCL2, and PCL3, and slightly outperforms all the others on PCL4, PCL5 and PCL6; STR outperforms all the others on PCL7; and, all the schemes perform equally well on PCL8+ with AvgQ = 0.

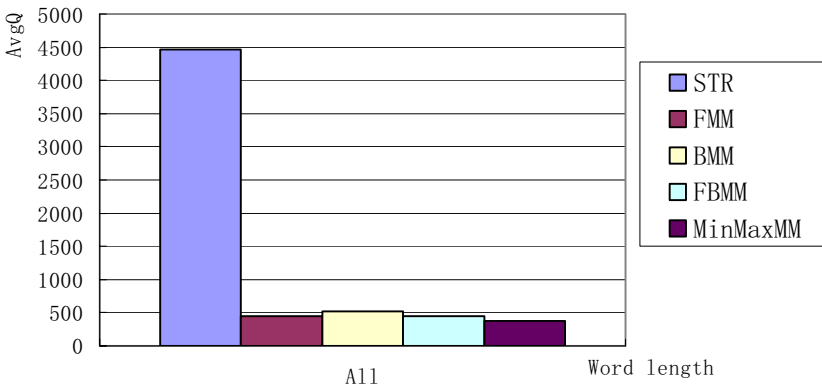


Fig. 9. Comparison of AvgQs of all schemes on PCL

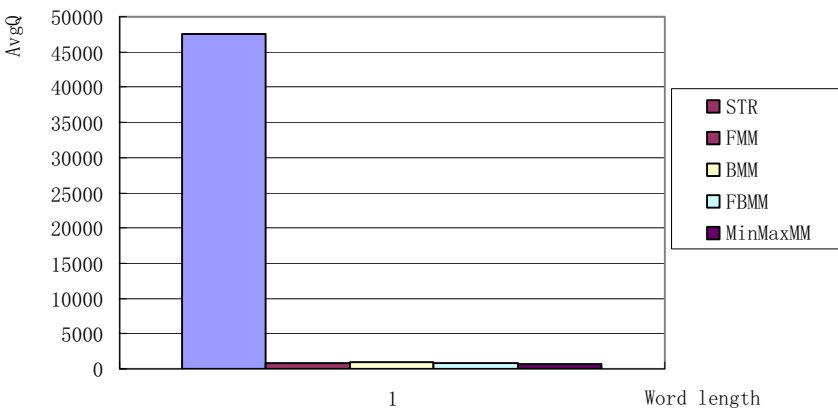


Fig. 10. Comparison of AvgQs of all schemes on PCL1

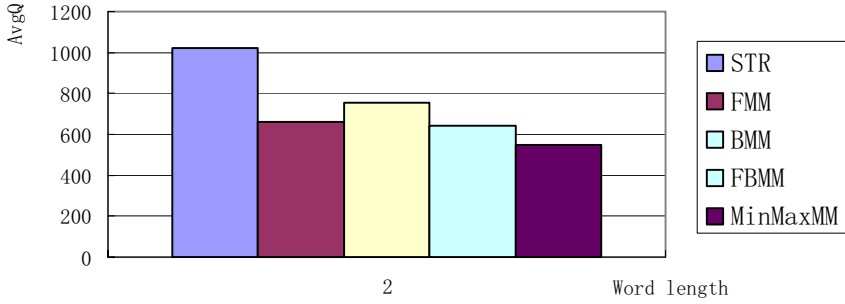


Fig. 11. Comparison of AvgQs of all schemes on PCL2

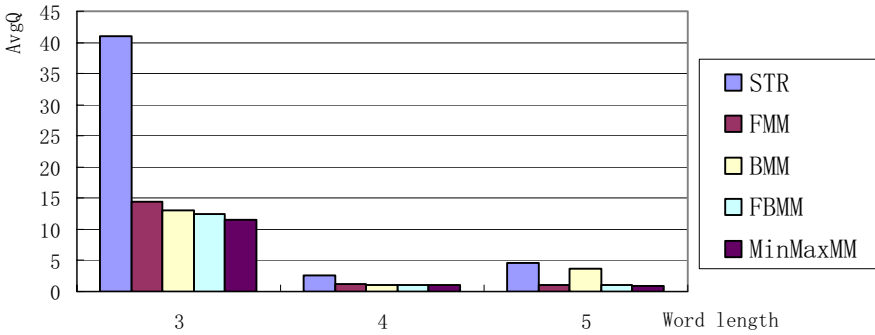


Fig. 12. Comparison of AvgQs of all schemes on PCL3, PCL4 and PCL5

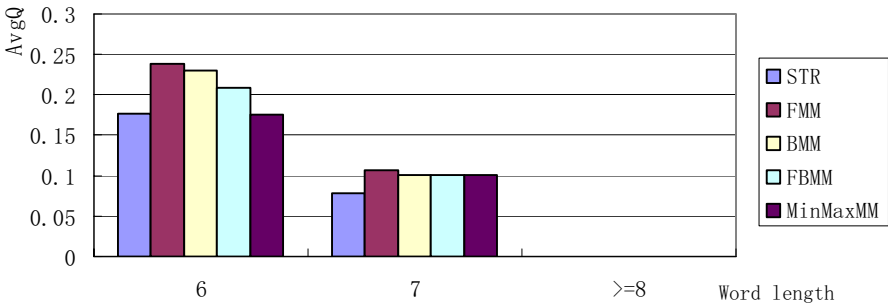


Fig. 13. Comparison of AvgQs of all schemes on PCL6, PCL7 and PCL8+

Consider Table 3 with its results of hypothesis testing (at the 0.05 level of significance): all the schemes accept H_0 on PCL6, PCL7 and PCL8+; FMM and MinMaxMM accept H_0 on PCL5; FBMM and MinMaxMM accept H_0 on PCL4; and all the schemes reject H_0 in PCL1, PCL2 and PCL3. In general, no scheme can accept H_0 on PCL.

Table 3. AvgQs of all schemes in all word length cases, supplemented with the average critical value at $\alpha=0.05$, and status of H_0 (‘√’ for ‘accept’, ‘×’ for ‘reject’) for every case (the underlined values are the minimums for their rows)

Scheme		The avg. critical value at $\alpha=0.05$	STR	FMM	BMM	FBMM	MinMax MM
Word length							
1	AvgQ	1.0455	47580.54	848.125	931.8895	815.7786	<u>698.8394</u>
	H_0	N.A.	×	×	×	×	×
2	AvgQ	1.0121	1021.537	660.6742	756.9079	642.0452	<u>547.4175</u>
	H_0	N.A.	×	×	×	×	×
3	AvgQ	1.0217	40.9966	14.40495	13.04044	12.46911	<u>11.57837</u>
	H_0	N.A.	×	×	×	×	×
4	AvgQ	1.028	2.633609	1.251279	1.109525	1.027401	<u>1.019498</u>
	H_0	N.A.	×	×	×	√	√
5	AvgQ	1.1361	4.545972	1.040558	3.706553	1.138404	<u>0.898344</u>
	H_0	N.A.	×	√	×	×	√
6	AvgQ	1.279	0.177263	0.238617	0.229614	0.209025	<u>0.175087</u>
	H_0	N.A.	√	√	√	√	√
7	AvgQ	1.3298	<u>0.078343</u>	0.106774	0.101338	0.101338	0.101019
	H_0	N.A.	√	√	√	√	√
≥	AvgQ	1.6664	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>
	H_0	N.A.	√	√	√	√	√
All	AvgQ	1.0096	4458.887	452.4088	518.8171	440.8097	<u>376.3869</u>
	H_0	N.A.	×	×	×	×	×

From the above observations and analyses, we can draw the conclusion that the proposed scheme, MinMaxMM, can serve as a better approximation to word frequency than other schemes when we only have a wordlist and a raw Chinese corpus.

4 Conclusion

Experimental results in the paper indicate that MinMaxMM, compared to other alternatives schemes under the framework of STR and MM, can benefit the estimation of word frequencies, in the condition that only a wordlist and a raw Chinese corpus are provided, and no any manual annotation on the corpus is carried out. But the performance of MinMaxMM is still not very satisfactory in some cases, as indicated by ‘×’ in the last column of Table 3. How to obtain a more accurate word frequency estimation for Chinese is still a big challenge, – the road ahead for this is not easy and perhaps quite long.

Acknowledgements. The research is supported by the National Natural Science Foundation of China under grant number 60321002, and the Tsinghua-ALVIS Project co-sponsored by the National Natural Science Foundation of China under grant number 60520130299 and EU FP6.

References

- [1] '863' High Tech Program of China et al.: When Computers Can Have Ability to listen, to speak, and to read? -- The Results of the Fifth Evaluation of Chinese Character Recognition, Speech Recognition, Speech Synthesis and Natural Language Processing. Computer World. E9, June 22, (1998)
- [2] Chen G.L.: On Chinese Morphology. Xuelin Publisher, Shanghai, (1994)
- [3] Dai X.L.: Chinese Morphology and its Interface with the Syntax. Ph.D Dissertation, Ohio State University, USA, (1992)
- [4] Emerson T.: The Second International Chinese Word Segmentation Bakeoff. Proceedings of the Third SIHAN Workshop on Chinese Language Processing. Jeju, Korea, (2005)
- [5] Liang N.Y.: CDWS: A Word Segmentation System for Written Chinese Texts. Journal of Chinese Information Processing. Vol. 1, No. 2, 44-52, (1987)
- [6] Liu E.S.: Frequency Dictionary of Chinese Words. Mouton & Co N.V. Publishers, (1973)
- [7] Liu K.Y.: Study on the Evaluation Technique for Word Segmentation of Contemporary Chinese. Applied Linguistics (Beijing). No. 1, 101-106, (1997)
- [8] Liu Y., Liang N.Y.: Counting Word Frequencies of Contemporary Chinese -- An Engineering of Chinese Processing. Journal of Chinese Information Processing. Vol. 0, No. 1, 17-25, (1986)
- [9] Sproat R., Emerson T.: The First International Chinese Word Segmentation Bakeoff. Proceedings of the Second SIHAN Workshop on Chinese Language Processing. Sapporo, Japan, 133-143, (2003)
- [10] Sun M.S., Shen D.Y., T'sou B.K.Y.: Chinese Word Segmentation without Using Lexicon and Hand-crafted Training Data, Proceedings of 36th ACL & 17th COLING, 1265-1271, Montreal, Canada, (1998)
- [11] Sun M.S., T'sou B.K.Y.: Ambiguity Resolution in Chinese Word Segmentation. Proceedings of the 10th Pacific Asia Conference on Language, Information & Computation. Hong Kong, 121-126, (1995)
- [12] Sun M.S., Wang H.J. et al.: Wordlist of Contemporary Chinese for Information Processing. Applied Linguistics (Beijing), No. 4, (2001), 84-89
- [13] Tang T.C.: Chinese Morphology and Syntax: Vol. 3. Taiwan Student Publisher, Taipei, (1992)
- [14] Zhu D.X.: Lectures on Grammar. The Commercial Press, Beijing, (1982)

Abbreviation Recognition with MaxEnt Model

Chunyu Kit, Xiaoyue Liu, and Jonathan J. Webster

Department of Chinese, Translation and Linguistics,
City University of Hong Kong, 83 Tat Chee Ave., Kowloon, Hong Kong
{ctckit, xyliu0, ctjjw}@cityu.edu.hk

Abstract. Abbreviated words carry critical information in the literature of many special domains. This paper reports our research in recognizing dotted abbreviations with MaxEnt model. The key points in our work include: (1) allowing the model to optimize with as many features as possible to capture the text characteristics of context words, and (2) utilizing simple lexical information such as sentence-initial words and candidate word length for performance enhancement. Experimental results show that this approach achieves impressive performance on the WSJ corpus.

1 Introduction

The literature in many special domains, e.g., biomedical, has been growing rapidly in recent years with a large number of abbreviations carrying critical information, e.g., proper names and terminology. There is an increasing interest in practical techniques for identifying abbreviations from plain texts.

There are several typical forms of abbreviation, including acronyms, blending, and dotted strings. Previous research [2, 7] illustrated significant success in identifying and pairing short form terms, referred to as abbreviations, most of which are acronyms, and their original long forms, referred to as definitions, e.g., <HIV, Human Immunodeficiency Virus>. This paper is intended to report our recent work to apply the Maximum Entropy (MaxEnt) model to identify abbreviations in another form, i.e., dotted strings, e.g., “abbr.” for “abbreviation”, “Jan.” for “January”. Popular abbreviations of this kind such as “Mr.”, “Dr.”, “Prof.”, “Corp.” and “Ltd.” are available from an ordinary dictionary. There is no point to invent any complicated techniques for recognizing them. The availability of such a sample set, however, gives us great convenience to evaluate the performance of a learning model on recognizing abbreviations with a particular surface form. The significance of this approach lies in the plausibility that similar methodology can be applied to abbreviations with some other common surface characteristics, e.g., in parentheses.

Aiming at this objective, we intend to allow the MaxEnt model to optimize with as many features as possible to capture the text form characteristics of context words and with some special features to utilize simple lexical information such as candidate word length and sentence-initial words that can be derived from the training data straightforwardly. Section 2 below presents feature selection for MaxEnt model training, and Sect. 3 the experiments for evaluation. Section 4 concludes the paper in terms of experimental results.

2 Feature Selection

MaxEnt models have been popular since [6, 1, 3] to cope with various NLP tasks that can be formulated as a classification problem. A MaxEnt model is trained with available data to achieve a probability distribution as unbiased as possible with a set of parameters λ_i for feature f_i , as in the following exponential form:

$$p(a|f) = \frac{1}{Z(f)} \exp\left(\sum_i \lambda_i \delta(f_i, a)\right), \quad (1)$$

where the feature function $\delta_i(f_i, a) = v_i \in \{1, 0\}$ for the feature f_i contributing to the answer a as in the data entry: $f = [f_1 = v_1, f_2 = v_2, \dots, f_n = v_n] \rightarrow a$, and the normalization factor $Z(f) = \sum_a \exp(\sum_i \lambda_i \delta(f_i, a))$.

By taking advantage of the *OpenNLP MAXENT* package available from <http://maxent.sourceforge.net/>, acknowledged here with gratitude, our research focuses on examining the effectiveness of various features in utilizing contextual information to identify dotted abbreviations. The contextual information comes from a few tokens next to the candidate word in question that carries a dot. A dot is highly ambiguous in English. Our task here is to determine whether a dot within the following general format of context indicates an abbreviation.

[preceding-words prefix.suffix following-words] $\rightarrow a \in \{\text{true}, \text{false}\}$

In order to train a MaxEnt model with such contextual information, all tokens in a context window as above have to be converted to a set of features representing the string characteristics of each token. To achieve a model as unbiased as possible, it is also critical to allow as many features as possible so as to let the training process determine the significance of each feature. The feature set developed in our experiments uniformly for each token in the context window is presented in Table 1. It is intended to cover as many observable text form characteristics as possible.

Table 1. Features for a context word

Feature	Description	Example
Hiphenated	If containing a dash	non-U.S.
AllCap	If of only capital letters (and dots)	D.C.
InitCap	If starting with a capital letter	Mr.
ContainDot	If containing a dot	B.A.T
EndDigit	If ending with a digit	45. <u>6</u>
InitDigit	If starting with a digit	<u>78</u> .99
IsNum	If of only numerical letters (and dots)	1.27
EndPunct	If ending with a punctuation	today."
InitPunct	If starting with a punctuation	"Dr.
EndDot	If ending with a dot	slightly.
EndComma	If ending with a comma	Inc.,
IsEword	If an English word	billions.

3 Experiments

Corpus. The corpus used for our experiments is the WSJ corpus from PTB-II, a refined version of PTB [5], of 1.2 million words in 53K sentences, involving 76,518 dotted words in total. This set of dotted words is divided into two by the ratio 2:1 for training and testing.

Effectiveness of Context Window. To examine how context words affect the learning performance, we carry out a number of experiments with context windows of various size. The experimental results are presented as the baseline performance in Table 2, where 1 and 0 surrounding a dot “.” in a context window indicate whether or not a context token in the correspondent position is in use. These results show that (1) the features from the dotted words themselves enable an accuracy beyond 91% and (2) a wider context window gives better performance in general.

Effectiveness of Sentence-Initial Words (SIWs) and Word length (WL). In addition to the features in Table 1, two more features, namely `IsSentInit` and `WL`, are introduced to utilize more specific lexical information for performance enhancement. `IsSentInit` indicates whether the context word, especially the one immediately following a candidate, is among the set of SIWs in use. Our previous work on period disambiguation with MaxEnt model [4] shows that SIWs are indicative that their preceding dotted words are not abbreviations. In total 4,190 sentence-initial words are collected from the training corpus. Our experiments show that using the top 200 ones in terms of frequency is most effective. Also, abbreviations are short in general, most of which do not exceed five in length, including the dot. With regard to this observation, `WL` is introduced as a special feature for a candidate word w as follows: `WL = true`, if $|w| \leq 5$; `false`, otherwise.

Table 2. Recognition accuracy (%) and performance enhancement by SIWs and WL

Context	001.100	001.110	011.100	011.110	011.111	111.110	111.111
Baseline	91.2954	92.2362	90.9317	93.6237	94.1732	95.1101	95.0508
+SIWs	96.3909	96.7387	96.3830	96.7150	96.6597	96.7980	96.7941
Increment	+5.0955	+4.5025	+5.4513	+3.0913	+2.4865	+1.6879	+1.7433
+SIWs+WL	98.1618	98.2093	98.2488	98.2765	98.3279	98.2804	98.2725
Increment	+6.8664	+5.9731	+7.3171	+4.6528	+4.1547	+3.1703	+3.2217

Table 3. Performance on a few popular abbreviations

Abbreviations	“Oct.”	“Nov.”	“Mr.”	“No.”	“U.S.”	“Corp.”
Total	158	89	1574	26	704	378
Correct	158	89	1546	25	695	333

The performance enhancement by SIWs, via `IsSentInit`, and by both SIWs and WL is presented in Table 2, showing a significant enhancement in both cases. Interestingly, the enhancement tends to be more significant for smaller context windows. A comparable accuracy is observed across various window sizes with the two features: around 96.5% for +SIWs and around 98.25% for +SIWs+WL. The performance on a few popular abbreviations is illustrated in Table 3.

4 Conclusions

Presented in the above sections is our investigation into how context window, feature space and lexical information affect the performance of the MaxEnt model in recognizing dotted abbreviations. A number of experiments on PTB-II WSJ corpus suggest that (1) the candidate words themselves provide the most useful information for a decision, achieving an accuracy near 91.3%, and (2) extending the feature space to utilize simple lexical information such as sentence-initial words and candidate word length leads to a significant enhancement, giving the best accuracy 98.33%.

Acknowledgements

The work described in this paper was supported by the Research Grants Council of HKSAR, China, through the CERG grant 9040861 (CityU 1318/03H), and we wish to thank Alex Fang and Haihua Pan for their help.

References

1. Berger, A., Della Pietra, S., and Della Pietra, V.: A maximum entropy approach to natural language processing. *Computational Linguistics*. (1996) 22(1):39–71
2. Chang, J.T., Schütze, H., and Altman, R.B.: Creating an online dictionary of abbreviations from MEDLINE. *The Journal of the American Medical Informatics Association*. (2002) 9(6):612–620
3. Della Pietra, S., Della Pietra, V., and Lafferty, J.: Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. (1997) 19(4):380–393
4. Kit, C. and Liu, X.: Period disambiguation with MaxEnt model. In *Natural Language Processing - IJCNLP-05*, LNAI 3651. Springer. (2005) 223–232
5. Marcus, M.P., Santorini, B., and Marcinkiewicz, M.A.: Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*. (1993) 19(2):313–330
6. Rosenfeld, R.: *Adaptive Statistical Language Modeling: A Maximum Entropy approach*. PhD thesis CMU-CS-94. (1994)
7. Schwartz, A.S. and Hearst, M.A.: A simple algorithm for identifying abbreviation definitions in biomedical text. In *Proceedings of the Pacific Symposium on Biocomputing (PSB 2003)*, Kauai. (2003) 451–462

An Efficient Multi-agent System Combining POS-Taggers for Arabic Texts

Chiraz Ben Othmane Zribi, Aroua Torjmen, and Mohamed Ben Ahmed

RIADI laboratory, National School of Computer Sciences, 2010,
University of La Manouba, Tunisia
{Chiraz.benothmane, Aroua.torjmen,
Mohamed.benahmed}@riadi.rnu.tn

Abstract. In this paper, we address the problem of Part-Of-Speech tagging of Arabic texts with vowel marks. After the description of the specificities of Arabic language and the induced difficulties on the task of POS-tagging, we propose an approach combining several methods. One of these methods, based on sentences patterns, is original and very attractive. We present, afterward, the multi-agent architecture that we adopted for the conception and the realization of our POS-tagging system. The multi-agent architecture is justified by the need for collaboration, parallelism and competition between the different agents. Finally, we expose the implementation and the evaluation of the system implemented.

1 Introduction

The process of Part-Of-Speech tagging was widely automated for English and French and for many others European languages giving a rate of accuracy ranging from 95 % to 98 %. We find on the Web, many tagged corpora as well as programs of POS-tagging for these languages. The methods used by these POS-taggers are various, namely stochastic approaches such as the Hidden Markov Model [1], the decision trees [2], the maximum entropy model [3], rules-based approaches inspired in their majority of the transformation rules-based POS-tagging [4], hybrid approaches [5] (statistics and rules-based), or combined ones [6] and [7].

Unfortunately, the situation is different for Arabic as there are neither POS-taggers nor tagged corpora available. Nevertheless, some Arabic POS-taggers [8], [9] and [10] started to appear with an accuracy going from 85% to 90% on average for texts with vowel marks and by about 65% for texts without vowel marks.

This gap noted for Arabic language is especially due to, its particular characteristics, which, involve firstly, a rate of grammatical ambiguity relatively more significant than for other languages, and secondly, make impossible the application of existing POS-taggers without any change. Thus, obtaining improving accuracy remains a challenge to reach for Arabic language.

Accordingly, we propose a POS-tagging system for Arabic texts. Due to the complexity of the problem, and in order to decrease grammatical ambiguity, we have restricted the scope of our investigation: we only treat texts with vowels marks.

The remainder of this paper is organized as follows: First, we present the Arabic language characteristics making the task of POS-tagging more difficult. We then present the general principle of our combined approach. Next, we show the general architecture of our multi-agent system and present a detailed description of the work of each agent. Finally, we present the method we used to evaluate the efficiency of our system and the results obtained.

2 Difficulties of Arabic Languages

In Arabic, the problem of POS-tagging is much more complicated than in other languages. Indeed, Arabic has numerous writing constraints such as vowels, agglutination and grammatical ambiguity, which can lead to ambiguities.

2.1 Vowel Marks

The vowel marks in words are vocalic signs that facilitate the reading and the comprehension of texts written in Arabic. Without vowels, the reader has to see the context to find the good vowels of the textual form, because Arabic words are vocalically ambiguous. This vocalic confusion involves naturally much more grammatical ambiguity.

Table 1. Example of vocalic ambiguity

كُتِبَ		
كُتِبَ	<i>Kattib</i>	Make write
كُتِبَ	<i>Kuttiba</i>	Has been made write
كُتِبَ	<i>Kutiba</i>	Has been written
كُتِبَ	<i>Kataba</i>	Wrote
...

2.2 Agglutination

Arabic is an agglutinative language. Textual forms are made of the agglutination of prefixes (articles, prepositions, conjunctions) and suffixes (linked pronouns) to the stems (inflected forms). In general, to obtain the different decompositions of a textual form, a morphological analyzer is needed. The ambiguities of decomposing textual forms induce a significant ambiguity of tagging. When the text is without vowel marks, the decomposing ambiguity increases.

Table 2. Example of decomposing ambiguity

أَكْبَرُ		
+ أكبر +	<i>Akabara</i>	Did it grow?
+ أكبر +	<i>Akbar</i>	Higher
+ أكبر +	<i>AkaBir</i>	Like benevolence

2.3 Grammatical Ambiguity

Arabic words are grammatically ambiguous. The statistics carried out in definition by [11] confirm this ambiguity. The author noted the importance of the rate of

grammatical ambiguity for the lexical forms with vowel marks, which is equal to 2.8 on average. This rate increases by the absence of the vowels to reach 5.6 possible tags per lexical form. Because of the agglutination of affixes to lexical forms, the rate of grammatical ambiguity is more significant for textual forms. According to the counting carried out by [8] on texts with vowel marks this rate is equal to 5.6 on average, and could reach an average of 8.7 for texts without vowel marks.

3 Suggested Approach

To achieve our POS-tagging system, we opted for combining methods (probabilistic and rules-based) in a multi-agent architecture.

3.1 Combined Method

We combine different methods trying to benefit from advantages for each method used and to improve our system's accuracy. This implies the construction of a number of POS-taggers where each operates according to the principle of the method that it represents. Each POS-tagger proposes one tag for the treated word and by voting the best one is assigned as the final tag to the target word.

3.2 A Multi-agent Architecture

The following arguments can justify the choice of this architecture, in addition to its originality:

- Combination of several methods: we combine several methods to realize our POS-tagging system.
- Competition and parallel work of agents: the POS-taggers agents treat the same sentence, which is extracted from the text to be tagged concurrently.
- Communication and cooperation between agents: The agents' system can communicate and cooperate for example to solve unknown words.

4 Part-of-Speech Tagger

We considered the following hypotheses to accomplish our POS-tagging system:

- We chose a supervised training mode to construct linguistic and probabilistic training data, from a pre-treated corpus (morphologically analyzed and manually tagged).
- We considered a sentence as a sequence of words limited by punctuations.
- The input of our system is the set of part of speech tag proposed by the morphological analyzer for each textual.

4.1 Tag Sets

In this work, we manipulate two main tag sets. The first one involves simple tags, also called micro tags. These tags are assigned to lexical units. We count 223 tags for the

inflected forms and 65 for the affixes. The second tag set is devoted to textual forms and is constructed by the licit combination of the simple tags {*prefix's tag + simple form's tag + suffix's tag*}.

We consider two other tag sets as well. Firstly, we use 22 macro tags, which are less detailed than micro tags. Secondly, we use a tag set representing the three principle part-of-speech tags: **S**ubstantive, **V**erb and **P**article (SVP). We use these tag sets to a simple matching between their tags and the micro tags. This is, in order, to make a comparison between the results given by taggers and to adapt the results of our system to various applications requirements.

4.2 System Agents

Some agents participate to accomplish the global objective of our POS-system that consists in assigning appropriate tags to each textual form of a given text. We cite:

- Sentences' extracting agent;
- Tagger agents,
- Unknown words solver agent;
- Voting agent.

The following figure illustrates the general architecture of this system.

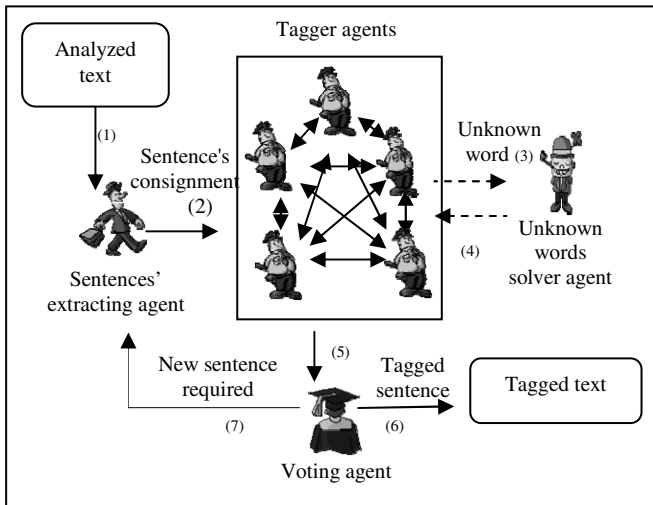


Fig. 1. General architecture of the POS-tagging system

Sentences' Extracting Agent. This agent is responsible of the extraction of the sentences from the text to tag. Each word in a sentence has a set of tags proposed by the morphological analyzer¹ developed by [11]. When it loads a sentence, the

¹ The morphological analyzer gives for each word all possible partitions in prefix, stem and suffix and for each partition, a set of all potential tags.

sentences' extracting agent activates all tagger agents to start the tagging of this sentence.

Tagger Agents. Given a sentence, five POS-tagger agents will work in parallel, each applying its own method, aiming to find for each word of the sentence the suitable tag among the tags proposed by the morphological analyzer.

Unigram Tagger Agent. For each word of a sentence received, the unigram tagger:

1. recuperates tags proposed by the morphological analyzer;
2. accedes to a lexicon which is containing the different words of the training corpus and their tags with their occurrence's frequencies;
3. seeks the target word in this lexicon;
4. chooses the most frequent tag for this word.

Bigram Tagger Agent. This tagger uses the binary succession probabilities recovered from the training corpus and saved in a binary succession matrix. We calculate the binary succession probability as follows:

$$p(t_i \setminus t_{i-1}) = \frac{\text{number of occurrences of the succession } (t_{i-1}, t_i)}{\text{number of occurrences of } t_{i-1}} . \quad (1)$$

The bigram tagger follows these steps to tag a word, it:

1. recovers the tags proposed by the morphological analyzer;
2. recovers the tag of the word preceding the target word;
3. accedes to the matrix of binary succession probabilities;
4. chooses the tag belonging to the set of tags proposed by the morphological analyzer having the higher binary transition probability considering the tag of the previous word;
5. assigns the tag that it found to the word to tag.

Trigram Tagger Agent. This trigram tagger agent works similarly to the precedent one, but it takes into account ternary succession probabilities recovered from the training corpus and saved in a ternary succession matrix. We determine the ternary succession probability as follows:

$$p(t_i \setminus t_{i-2}, t_{i-1}) = \frac{\text{number of occurrences of the succession } (t_{i-2}, t_{i-1}, t_i)}{\text{number of occurrences of the succession } (t_{i-2}, t_{i-1})} . \quad (2)$$

Here, the principle of tagging each word in a given sentence consists in:

1. recovering the tags proposed by the morphological analyzer;
2. recovering the two previous tags in relation with the target word;
3. acceding to the matrix of ternary transition probabilities;
4. choosing the grammatical tag, which belongs to the tags proposed by the morphological analyzer and has the higher ternary transition probability considering the two tags of the two previous words;
5. assigning the tag found to this word.

Hidden Markov Model Tagger Agent. This tagger agent operates according to Hidden Markov Model's principle.

Given a sequence of n words $W = w_1 \dots w_n$, this tagger tries to find the tag sequence $T = t_1 \dots t_n$, that maximizes the conditional probability $p(T|W)$.

We note:

$$\text{Max } T = \arg \text{Max}_T p(T|W)$$

By some assumptions² :

$$\text{Max } T = \arg \text{Max}_T \prod_{i=1}^n p(w_i \setminus t_i) \times p(t_i \setminus t_{i-1}) . \quad (3)$$

Where:

$p(w_i \setminus t_i)$ is the emission probability that is calculated with the following formula :

$$p(w_i \setminus t_i) = \frac{\text{number of occurrences of } w_i \text{ tagged with } t_i}{\text{number of occurrences of } t_i} . \quad (4)$$

and $p(t_i \setminus t_{i-1})$ is the transition probability that is determined as follows:

$$p(t_i \setminus t_{i-1}) = \frac{\text{number of occurrences of succession } (t_{i-1}, t_i)}{\text{number of occurrences of } t_{i-1}} . \quad (5)$$

Where:

$p(t_1 \setminus t_0) = p(t_1)$ called initial probability.

When it receives a sentence, including for each word all the tags proposed by the morphological analyzer, this tagger agent applies the VITERBI algorithm [12]. The latter takes all the needed frequencies from the training corpus and tries to find the tag sequence that has the maximum likelihood.

Agent based on Sentences Patterns. The sentences patterns-based method presented here is new and has not been approached before. We define a sentence pattern as a model of sentence made of a succession of tags.

Example:

The sentence: "The child eats a cake" can matches with the following pattern: "Definite-Article + Noun + Verb + Indefinite-Article + Noun".

In the practice, the possession of all sentences patterns for a language is difficult. That is why this tagger manipulates the longest successions of tags of adjustable size. The sentence pattern considers the positional character of tags in the sentence (1st tag, 2nd tag...).

² Independency assumption and Markov assumption $k=1$ (using binary successions).

The principle of this tagger consists in:

1. considering the first word of the sentence and extracting the tags that have been assigned by the morphological analyzer;
2. acceding to the set of sentences patterns and seeking the patterns that start with one of the tags proposed by the morphological analyzer;
3. treating the second word. Among models found in patterns, the second tag correlates to one of tags proposed by the morphological analyzer for this word.
4. this process is repeated until the words of the sentence are tagged completely considering the position of words while the matching between the tags proposed by the morphological analyzer for the treated word and the tags of patterns are proposed. Thus, the number of the candidates patterns decreases when the tagger goes forward in the treatment of the sentence.

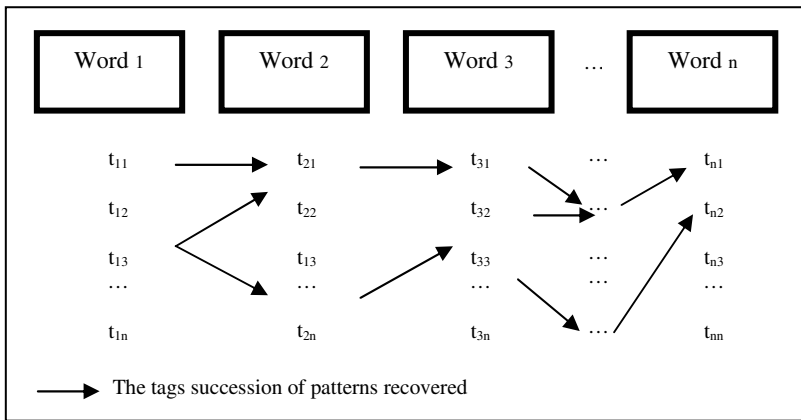


Fig. 2. Example of the progress of sentences patterns exploration

If for a given word no pattern is founded, this tagger agent examines all the training patterns to extract the longest succession tags matching to the tags proposed by the morphological analyzer. When the extraction is made, the tagger joins the segments patterns to the segments previously retained, to form new patterns that are going to serve to the research of the tags of the following words. When all words of the sentence have been treated, and if several candidate patterns were kept as result of the tagging, the tagger chooses the pattern having the highest weight that is calculated from the sum of the initial probabilities of its words' tags. If several patterns have the same weight then it keeps the one that is most frequent in the training corpus.

Voting Agent. After achieving their works, the tagger agents activate the voting agent to decide which tag to assign for a word. We have three cases:

1. If all taggers elect the same tag then this tag is affected to the target word;
2. If the majority and not the totality of taggers agree about a given tag, this tag is assigned to the treated word;

3. If all taggers are in a total disagreement, the voting agent uses heuristics to decide and to choose one and only one tag to assign to this word.

These heuristics are:

- The reliance degree in progress: Voting agent considers the tag of the tagger having the higher reliance degree. For each tagger an indicator is provided and is incremented each time the voting process considers its tag in the vote.
- The reliance degree in historic: In case two or several taggers have the same highest reliance degree, the voting agent sees the historic of every tagger in competition and chooses the one which previously achieved the best tagging accuracy.

Unknown Words Resolution Agent. We have two cases of unknown word:

- If the morphological parser does not propose tags for the treated word:
 - The tagger agent asks the assistance help of the other tagger agents. If one of them solves this problem, it sends to him the found tag.
 - If no tagger could help it, the tagger agent calls the unknown word resolution agent.
- If the morphological parser proposes a set of tags for the target word and the tagger does not find the suitable tag, because of lack of training data :
 - If one of the tagger agents solves this problem before the tagger agent asking help, it sends to him the found tag.
 - Otherwise, the unknown word resolution agent is required:
 - If the unknown word resolution agent proposes a tag that exists among the set of tags proposed by the morphological analyzer, this tag will be considered as the final tag to assign to the word to tag;
 - If the unknown word resolution agent proposes a tag which is not among the set of tags proposed by the morphological analyzer, then the tagger accedes to the training data and recovers the most frequent tag among the set of tags proposed by the morphological parser to assign it to this word.

To guess a tag, the unknown words resolution agent works according to this principle:

1. uses the schemes of verbs and personal names as well as the lexical rules to determine the nature of the treated word (noun, personal name, verb...);
2. takes the corresponding tag from a training list containing schemes and their relative tags, if the word to tag has a scheme of a verb or personal name;

Examples:

Verb: If a word has the scheme *افعلتوا* it can be tagged *فعل أمر*.

Personal name: If a word has the scheme *فعلن* it can be tagged *اسم علم مرفوع*.

3. applies lexical rules if the word is assumed to be a noun.

Example: noun starting with *ال* and ending with *ضمّة* is likelier to have the tag *اسم معرف مرفوع*.

5 Experimentations and Results

Our experiment was carried out in two stages: one stage of training during which, a textual corpus containing about **6000** textual forms was manually annotated and probabilities were collected. The second stage is the testing, which consists in using these probabilities to tag a testing text. We tested two different environments. In the first one, we took the testing text from training corpus. In the second, we chose the testing text out of the training corpus.

For the system evaluation, we used the accuracy rate that is calculated as follows:

$$\text{Tagging accuracy} = \frac{\text{number of correctly tagged words}}{\text{total number of tagged words}} . \quad (6)$$

5.1 Ad Hoc Environment

As shown in the table 3, the probabilistic taggers are not very efficient (maximum accuracy of **93.73%** for simple tags and **95.78%** for composed tags) since they require a big training data. In general, the accuracy for all taggers increases (except for the Trigram tagger), when we manipulate the composed tags. The accuracy of the global system increases as well, and it is due to the diversity of mistakes that taggers provoke. We observe also that the use of macro tags increases significantly the accuracy of the taggers. This is due to the nature of mistakes caused by taggers that confuses tags belonging to the same class of tags.

Table 3. Tagging accuracy in the ad hoc environment

Taggers	Simple tags (%)	Macro tags (%)	Substantives (%)	Verbs (%)	Particles (%)	Composed tags (%)
Unigram	92.17	94.68	97.33	90.43	99.45	94.42
Bigram	90.47	95.88	93.44	76.33	98.26	92.35
Trigram	93.73	96.84	94.37	87.5	98.85	90.99
HMM	91.52	94.23	96.01	89.88	100	95.78
Sentences Patterns	95.33	97.24	97.76	90.53	98.88	95.91
Global system	97.54	98.64	97.77	94.79	100	98.35

We can also notice that the sentences patterns tagger achieved best results: **95.33%** for simple tags and **95.91%** for composed tags. This reflects the efficiency of this new method.

5.2 Out of the Training Data Environment

In the table below, we can observe that the tagging accuracy of the sentences patterns tagger becomes the weakest (except for composed tags), whereas in the ad hoc environment it was the best. This is because we are in an environment out of training

data, and our training data are insufficient for such a method. Therefore, the accuracy rate achieved by this tagger is comprehensible.

However, the accomplished tagging accuracy of the global system, using the simple and composed tags is satisfactory compared to results achieved by the other Arabic tagging systems in a similar environment of experimentation.

Table 4. Tagging accuracy in the out of training data environment

Taggers	Simple tags (%)	Macro tags (%)	Substantives (%)	Verbs (%)	Particles (%)	Composed tags (%)
Unigram	90.92	96.14	97.14	80.87	96.77	90.11
Bigram	90.29	95.02	95.79	78.74	96.61	92.05
Trigram	92.26	95.41	93.10	87.61	96.15	89.62
HMM	91.42	95.4	93.85	86.43	96.66	90.59
Sentences patterns	90.09	94.86	97.36	76.11	92.06	92.38
Global system	92.35	96.68	98.26	85.0	96.77	94.81

6 Conclusion

Our POS-tagging system is based on a combined approach. The efficiency of this approach was proved by the accuracy generated by the global system. In fact, this accuracy is generally higher than the tagging accuracy of each tagger. The new method of tagging based on sentences patterns gives also satisfactory results and proves to be promising. In spite of the lack of our training data and the ambiguous specificities of the Arabic language, the choices that we adopted enabled us to reach our initially drawn up goals. However, the results can be still ameliorated by improving largely the training data. Considering that the majority of the texts available are without vowel marks, we plan to treat this type of texts in a further work.

References

1. Cutting D., Kupiec J., Pedersen J. And Sibun P.: A practical Part-Of-Speech Tagger. In: proceedings of the Third Conference on Applied Natural Language Processing, (1992) 133–140
2. Schmid H. et Stein A. : Etiquetage morphologique de textes français avec un arbre de décision. Le traitement automatique des langues: Traitements probabilistes et corpus, Vol.36, No. 1-2, (1995) 23–35
3. Ratnaparkhi Adwait: A maximum Entropy Model for part of speech tagger. In: proceedings of the first empirical methods in natural language processing conference, Philadelphia, USA, (1996) 133–142
4. Brill E.: Some Advances in Transformation-based part of speech Tagging. In: proceedings of the 12th national conference on artificial intelligence (1992), 722-727
5. Marshall I.: Choice of Grammatical Word-class without Global Syntactic Analysis: Tagging Words in the LOB Corpus. Computers and the Humanities, No.17, (1983) 139–50

6. Brill E. and Wu J.: Classifier combination for improved lexical disambiguation. In: proceedings of the thirty-sixth ACL and seventeenth COLING, Montréal, Canada, (1998) 191–195,
7. Sjöbergh Jonas: Combining POS-Taggers for improved accuracy on Swedish text. NoDaLiDa, Reykjavik, (2003)
8. Debili F., Achour H., Souissi E. : La langue arabe et l'ordinateur : de l'étiquetage grammatical à la voyellation automatique. Correspondances, No. 71, Institut de recherche sur le Maghreb contemporain, CNRS, Tunis, (2002) 10–28
9. Khoja S.: APT: Arabic Part-of-speech Tagger. In: proceedings of the student workshop at the second meeting of the north American chapter of the Association for computational linguistics (NAACL'01), Carnegie Mellon University, Pennsylvania, (2001) 20–26
10. Zemirli Z. et Khabet S. : TAGGAR : un analyseur morphosyntaxique destiné à la synthèse vocale des textes arabes voyellés. JEP-TALN 2004, Traitement Automatique de l'Arabe, Fès, (2004)
11. Ben Othman C. : De la synthèse lexicographique à la détection et la correction des graphies fautes arabes. Thèse de doctorat, Université de Paris XI, Orsay, (1998)
12. Rajman M. et Chappelier J.C. : Chaînes de Markov cachées. Cours TIDT, Département informatique, Ecole Polytechnique de la Lausanne, (2003)

A Comparative Evaluation of a New Unsupervised Sentence Boundary Detection Approach on Documents in English and Portuguese

Jan Strunk¹, Carlos N. Silla Jr.², and Celso A.A. Kaestner²

¹ Sprachwissenschaftliches Institut, Ruhr-Universität Bochum,
44780 Bochum, Germany

`strunk@linguistics.rub.de`

² Pontifical Catholic University of Paraná,
Rua Imaculada Conceição 1155, 80215-901 Curitiba, Brazil
{silla, kaestner}@ppgia.pucpr.br

Abstract. In this paper, we describe a new unsupervised sentence boundary detection system and present a comparative study evaluating its performance against different systems found in the literature that have been used to perform the task of automatic text segmentation into sentences for English and Portuguese documents. The results achieved by this new approach were as good as those of the previous systems, especially considering that the method does not require any additional training resources.

1 Introduction

We are living today in an era of information overload. The web alone contains about 170 terabytes of information, which is roughly 17 times the size of the printed material in the Library of Congress of the USA; cf. [1]. However, it is becoming more and more difficult to use the available information. Many problems such as the retrieval and extraction of information and the automatic summarization of texts have become important research topics in computer science. The use of automatic tools for the treatment of information has become essential to the user because without those tools it is virtually impossible to exploit all the relevant information available on the Web.

One pre-processing component that is essential to most text-based systems is the automatic segmentation of a text into sentences. Existing systems for sentence boundary detection mostly either use a set of heuristics or a supervised machine learning approach. The drawback of both these approaches is that adapting them to new languages can be time and resource intensive. In the first case, it is necessary to adapt the rules to the new language. In the second case, a new training corpus has to be tagged manually for retraining.

In this paper, we compare a new unsupervised approach to sentence boundary detection by Kiss & Strunk [2] with the results of a previous evaluation of three

different systems on English and Portuguese documents [3] carried out by Silla Jr. & Kaestner. The three previous systems are described in the next section.

2 Description of the Systems

2.1 RE System

The first system tested was the RE (Regular Expressions) system¹ developed by Silla Jr. & Kaestner for English; cf. [3]. It was chosen as a representative of the fixed rules approach. The system considers the context where each possible end-of-sentence marker occurs within the document. It uses a database of regular expressions which denote strings that contain punctuation marks but don't indicate the end of a sentence, like abbreviations, e-mail addresses, URLs, etc.

In order to identify sentence boundaries, the system scans the text until it finds a period (.). It then analyzes the preceding string; if this string matches some regular expression, the system concludes that the period is not an end-of-sentence marker and advances to the next period. If the preceding string doesn't match any regular expression, the system considers the string after the period. If it doesn't find any matching regular expression for this string, either, it concludes that the period indicates a sentence boundary. The procedure is repeated until the entire document has been analyzed. The system is also able to deal with ellipses (...).

In order to adapt the system to Brazilian Portuguese, 240 new regular expressions containing abbreviations for the new language had to be added.

2.2 MxTerminator

The MxTerminator system² was developed by Reynar and Ratnaparkhi [4] at the University of Pennsylvania. It uses a supervised machine learning approach called maximum entropy modelling. From a corpus in which the sentences have been identified manually, the model learns to decide for each instance of period (.), exclamation mark (!) and question mark (?) whether it marks the end of a sentence or not.

The training process is robust and doesn't require any additional linguistic information. During training, the system learns probabilistic contextual features from the training corpus that can be used to identify sentence boundaries with high accuracy, such as e.g. the prefix and suffix occurring around a potential sentence boundary symbol, the preceding and following word, capitalization information, etc. It also induces a list of abbreviations from the training corpus by considering as an abbreviation every token in the training set that contains a possible end-of-sentence symbol but does not indicate a sentence boundary.

The system then uses the contextual features and the abbreviation list learned during training to calculate the probability that a possible end-of-sentence marker in a test corpus indeed indicates a sentence boundary or not.

¹ Available from: <http://www.pggia.pucpr.br/~silla/software/yasd.zip>.

² Available from: <ftp://ftp.cis.upenn.edu/pub/adwait/jmx/jmx.tar.gz>.

The procedure to adapt MxTerminator to Brazilian Portuguese was quite simple because the system only requires a text file of any size that must contain one sentence per line as training corpus.

2.3 Satz

The Satz system³ was developed by Palmer and Hearst [5] at the University of California in Berkeley. It is a supervised approach that uses estimates of the part-of-speech distribution of the words surrounding potential end-of-sentence punctuation marks as input to a machine learning algorithm. The part-of-speech information is derived from a lexicon that contains part-of-speech frequency data. In case a word is not in the lexicon, a part-of-speech distribution is estimated by different guessing heuristics. In addition, Satz also uses an abbreviation list and capitalization information. After training the system on a small training and a small cross-validation corpus, it can then be used on new documents to detect sentence boundaries. The system can work with any kind of machine learning approach in principle. Palmer & Hearst's original results [5] were obtained using neural networks and the C4.5 decision tree classifier.

For our own evaluation reported in section 4 we employed a re-implementation of the Satz system in Java by Silla Jr. & Kaestner, which uses J4.8 – a Java version of the C4.5 decision tree induction algorithm. The system had to be re-implemented because of problems with accented characters in Portuguese which had occurred with the original version. However, this re-implementation alone was not enough to adapt the system. Silla Jr. & Kaestner also had to create a small training corpus and a new lexicon with part-of-speech information.

3 The Unsupervised System by Kiss & Strunk

The unsupervised system by Kiss & Strunk (subsequently abbreviated as KS)⁴ combines type-based and token-based classification⁵ in a two-stage approach. It only has to be supplied with the test corpus and does not need further training data, a lexicon, or a list of abbreviations. Instead it uses the test corpus itself as a training corpus on the fly. The system is multilingual in the sense that it is supposed to work for all languages with an alphabetic script in which the period is used to mark both abbreviations and sentence boundaries.

Sentence boundary disambiguation lends itself to a two-stage approach combining type-based and token-based classifiers because in many languages the token-final period (.), the most frequently used sentence boundary marker, is ambiguous in the following way: It can either indicate an abbreviation, a sentence boundary, or an abbreviation at the end of a sentence in which case the

³ Available from: <http://elib.cs.berkeley.edu/src/satz/>

⁴ The KS system is based on an earlier system described in [6] and [7].

⁵ We define a classifier as type-based if it uses global evidence, e.g. the distribution of a type in a corpus, to classify a type as a whole. In contrast, a token-based classifier determines a class for each individual token based on its local context.

period performs a double duty as abbreviation and sentence boundary marker at the same time; cf. [8]. Similar facts hold for ellipses (...), a combination of three or more periods that are used to indicate an omission or an omission followed by a sentence boundary in which case the sentence boundary period is also normally haploglogically omitted. Abbreviations can be detected very well with a type-based classifier because abbreviations are a (productive) class of lexical items, i.e. all instances of abbreviation types such as *e.g.* or *etc.* are abbreviations regardless of what context they occur in as individual tokens. Moreover, any periods that follow instances of types that have been identified as non-abbreviations by the type-based classifier can safely be classified as sentence boundary markers: If we know that a token with a final period is an ordinary word and not an abbreviation, it is clear that the period following it is a sentence boundary marker. The first stage of the KS system therefore consists of a type-based classifier that separates all word types in the test corpus into the three classes: abbreviation, ellipsis, and ordinary word. Most sentence boundaries are already detected by this type-based first stage. It is described in section 3.1.

The token-based second stage of the KS system improves on the initial classification of the periods in the test corpus performed by the type-based first stage. It re-examines the initial annotation and reclassifies certain cases that can only be decided by token-based classification in principle or present difficulties for the type-based classifier. Whether an abbreviations or an ellipsis is followed by a sentence boundary cannot be decided by a type-based algorithm at all because instances of one and the same abbreviation type – such as the English *etc.* – can be followed by a sentence boundary in one case and occur in the middle of a sentence in another case. The token-based stage therefore decides for all abbreviation and ellipsis tokens in the test corpus whether they precede a sentence boundary or not. In addition, the token-based stage is also used to correct the initial classification for certain subclasses of abbreviations, namely initials – such as in *J. Bond* – and ordinal numbers – such as in the German example *3. März* (“third of March”), which are less amenable to a type-based approach because of problems with homography. The token-based second stage of the KS system is described in section 3.2.

3.1 Initial Type-Based Classification

The type-based classification of the KS system is based on the task of abbreviation detection. By finding all abbreviation types in a test corpus, the system is also able to detect a large portion of the sentence boundaries in the corpus by classifying all periods following non-abbreviation types as sentence boundary markers. Kiss & Strunk assume that abbreviation detection is a manageable subproblem of sentence boundary detection and may also be useful in itself in that dynamically generated lists of abbreviations could be used in subsequent natural language processing tasks.

In their approach, Kiss & Strunk concentrate on the following three characteristics of typical abbreviations:

1. *Strong collocational dependence*: Abbreviations always occur with a final period.⁶
2. *Brevity*: Abbreviations tend to be short.
3. *Internal periods*: Many abbreviations contain additional internal periods.

As these three characteristics do not change for each individual instance of a type, they can be combined in a type-based approach to abbreviation detection.

The criterion of *strong collocational dependence* expresses the intuition that an abbreviation and the final period marking it as such form a tight unit in that an ordinary abbreviation should never occur without a following period. Kiss & Strunk implement this intuition using a modification of Dunning’s log-likelihood ratio for collocation detection described in [9]. They use a log-likelihood ratio to compare the probabilities of the following two hypotheses: The null hypothesis H_0 shown in (1) assumes that a type w is not an abbreviation and that therefore the probability of a period occurring after this type is equal to the unconditional probability of occurrence of the period.⁷

$$\text{Null hypothesis } H_0: \quad P(\bullet|w) = P_{MLE}(\bullet) = \frac{\text{count}(\bullet)}{N} \quad (1)$$

The alternative hypothesis assumes that the type w in question is indeed an abbreviation and therefore (almost) always occurs with a following period. The conditional probability of a period given w is therefore taken to be 0.99, i.e. almost one, cf. equation (2).

$$\text{Alternative hypothesis } H_A: \quad P(\bullet|w) = 0.99 \quad (2)$$

The KS system uses the actual number of occurrences of each type in the test corpus with and without a following period to calculate the probabilities for the two hypotheses with the binomial distribution. The two probabilities are compared using the formula in (3).

$$\log \lambda = -2 \log \frac{P_{binom}(H_0)}{P_{binom}(H_A)} \quad (3)$$

The list of candidate types is sorted according to the calculated log-likelihood values. A type with a higher $\log \lambda$ value is more likely to be an abbreviation according to the criterion of *strong collocational dependence* than all types with lower values. The left half of Table 1 shows a section of this sorted list from an English test corpus. Some true abbreviations in this table are either ranked lower than non-abbreviations (written in italics) or receive the same $\log \lambda$ values as non-abbreviations. The criterion of *strong collocational dependence* alone is thus not sufficient to separate abbreviations from non-abbreviations.

⁶ If abbreviations do not have to occur with a final period in a certain language or certain types of abbreviations do not have to, the problem of deciding between the end-of-sentence marker and the abbreviation marker does not occur in this language or for these types of abbreviations.

⁷ *MLE* stands for maximum likelihood estimation. N is the number of tokens in the test corpus.

Table 1. Candidate list from an English test corpus

Candidate type	count(w, ●)	count(w, ¬●)	Original log λ	Final sorting	Final log λ
n.h	5	0	28.08	n.h	7.60
u.s.a	5	0	28.08	a.g	6.08
alex	8	2	26.75	m.j	4.56
<i>ounces</i>	4	0	22.46	u.n	4.56
a.g	4	0	22.46	u.s.a	4.19
ga	4	0	22.46	ga	3.04
vt	4	0	22.46	vt	3.04
ore	5	1	18.99	ore	0.32
<i>1990s</i>	5	1	18.99	reps	0.31
mo	8	3	17.67	mo	0.30
m.j	3	0	16.85	<i>1990s</i>	0.26
<i>depositor</i>	3	0	16.85	<i>ounces</i>	0.06
reps	3	0	16.85	alex	0.03
u.n	3	0	16.85	<i>depositor</i>	0.00

The calculated log-likelihood values are therefore taken as a starting point and multiplied with additional factors to obtain an improved sorting of the candidate types. Table 1 confirms that abbreviations tend to be rather short. The factor F_{length} in (4) expresses this intuition and gives an exponentially growing penalty to longer candidate types.

$$F_{length} = \frac{1}{e^{length(w)}} \quad (4)$$

Kiss & Strunk define $length(w)$ as the length of candidate type w minus the number of internal periods in w because internal periods are actually good evidence in favor of a classification as abbreviation and should not lead to a higher penalty by the length factor. Instead, the KS system rewards internal periods with the factor given in (5).

$$F_{period} = \text{number of internal periods} + 1 \quad (5)$$

The scaled log-likelihood ratio proposed by Kiss & Strunk has the advantage that it makes abbreviation detection more robust. The algorithm does not exclude a candidate from being classified as an abbreviation just because it has occurred without a final period once or twice in the whole corpus when there is otherwise good evidence that it is a true abbreviation. For most languages, this increased robustness is unproblematic because almost all ordinary words occur without a period a sufficient number of times. However, for some languages the log-likelihood ratio in (3) is not restrictive enough. One example are verb-final languages – such as Turkish – where certain very common verbs happen to appear at the end of a sentence most of the time. In such a case, the scaled log-likelihood ratio described so far runs into difficulties because it mistakes the occurrences of these verbs without a period as exceptions. To remedy this

problem, the calculated $\log \lambda$ values are additionally multiplied by a third factor that penalizes occurrences without a final period exponentially, cf. equation (6).

$$F_{penalty} = \frac{1}{length(w)^{count(w, \cdot)}} \quad (6)$$

In order to perform the classification into abbreviations and non-abbreviations, the calculated $\log \lambda$ values for all candidate types are multiplied with all three factors. The resulting final values are then compared with a threshold value. All candidates that attain a value greater or equal to the threshold value are classified as abbreviation types all others as non-abbreviation types, cf. (7).

For each w :

$$\text{If } \log \lambda(w) \times F_{length} \times F_{periods} \times F_{penalty} \geq 0.3 \rightarrow w \text{ is an abbreviation.} \quad (7)$$

$$\text{If } \log \lambda(w) \times F_{length} \times F_{periods} \times F_{penalty} < 0.3 \rightarrow w \text{ is not an abbreviation.}$$

The threshold value 0.3 has been determined experimentally by looking at the sorted list of candidates extracted from a development corpus which was built from a 10 MB part of the Wall Street Journal corpus of American English. Kiss & Strunk assume that the threshold value will not vary much for different languages and corpora and the value 0.3 can thus be used on new corpora and languages without the need for additional manual experiments.⁸ The scaling factors have also been derived in experiments measuring their effect on the goodness of the sorting of the candidate list.

The last two columns in Table 1 show the final scaled $\log \lambda$ values of the candidates and the resulting sorting. Multiplication with the three factors has led to a cleaner separation of the candidates into abbreviations and non-abbreviations.

3.2 Token-Based Reclassification

In the token-based reclassification stage of the KS system, all tokens with a final period are re-examined and possibly reclassified. The evidence for this reclassification comes from the immediate right context of the period that is re-examined.⁹

The token-based stage treats different classes of candidates such as abbreviations, ellipses, initials, and ordinal numbers in different ways. However, the reclassification of all the different classes involves the same kinds of evidence combined in slightly different ways.

One type of evidence that is usually taken to be very fundamental for sentence boundary detection, namely capitalization, is only used as secondary evidence during reclassification in the KS system. Moreover, the *orthographic decision heuristic* used is quite cautious, which makes the system very robust against capitalization errors and enables it to process single-case corpora with almost

⁸ This view is confirmed by a more detailed evaluation in [2].

⁹ If the next token following the period is separated from it by empty lines, up to three new line tokens are ignored, i.e. *etc. \n \n This* is treated as *etc. This*.

the same accuracy as mixed-case corpora; cf. [2]. As data for the *orthographic decision heuristic*, the capitalization behavior of all types in the test corpus is recorded. For each type, it is counted how often it occurs with an uppercase first letter and how often with a lowercase first letter. Moreover, it is also determined on the basis of the initial annotation from the first stage how often every type occurs upper- and lowercased after a sure sentence boundary¹⁰ and within a sentence. The following is the pseudo-code for the *orthographic decision heuristic*:

```
function DECIDE_ORTHOGRAPHIC (TOKEN):
  if TOKEN has uppercase first letter:
    if TOKEN ever occurs with lowercase first letter:
      if TOKEN never occurs with uppercase first letter
        sentence internally:
          Return sentence_boundary
      else
        Return undecided
    else
      Return undecided

  else if TOKEN has lowercase first letter:
    if (TOKEN ever occurs with uppercase first letter)
    or (never occurs with lowercase first letter after
    a sentence boundary):
      Return no_sentence_boundary
    else
      Return undecided
```

The *orthographic decision heuristic* is especially cautious in two cases: First, if a type also occurs with an uppercase first letter within a sentence, as is usually the case with proper names, it is no longer counted as evidence for a preceding sentence boundary if an instance of this type follows a period. Second, if a type also occurs in lower case after a sure sentence boundary, it might be a mathematical symbol or a special word such as *amnesty international* that is always written with a lowercase first letter. This type is then no longer counted as evidence against a sentence boundary if it follows a period.¹¹

The second type of evidence that the system relies on during the token-based stage is collocational data. It is often assumed that there are no strong local dependencies between the end of one and the beginning of the following sentence; cf. e.g. page 195 in [10]. If there is a strong collocational dependence between two types – such as e.g. between an initial and a following last name – this is good evidence against an intervening sentence boundary. The KS system therefore employs the standard log-likelihood ratio for collocation detection described in [9] to calculate the dependence between two types.¹² If the log-likelihood ratio

¹⁰ This means all periods following a type classified as an ordinary word that is longer than one letter, i.e. no possible initial, and is not a number written in digits.

¹¹ This also enables the KS system to classify all-lowercase corpora without bad reclassification by the *orthographic decision heuristic*.

¹² All numbers written in digits are folded into one abstract type *##number##*.

yields a value greater or equal to 7.88, the two types are considered as collocates and as evidence against an intervening sentence boundary.¹³

However, collocational data is also used as evidence in favor of a sentence boundary. For this purpose, the collocational dependence between every type in the test corpus and the abstract type *preceding sentence boundary* is calculated in order to generate a list of frequent sentence starters on the fly. The counts used in these calculations are based on all clear sentence boundaries detected by the type-based first stage. All types for which Dunning's log-likelihood ratio yields a value of at least 30 are considered as frequent sentence starters and regarded as evidence for a sentence boundary if they occur after a period and are written with an uppercase first letter.¹⁴

The main question for all tokens classified as abbreviations by the type-based first stage and all ellipses is whether they precede a sentence boundary. A sentence boundary after these two classes of candidate tokens is assumed by the KS system if the *orthographic decision heuristic* decides in favor of a sentence boundary or the token following the period is a capitalized frequent sentence starter. However, only abbreviations that are longer than one letter and thus not possibly initials are reclassified in this way. Initials present special problems and are therefore reclassified differently.

Initials are a subclass of abbreviations consisting of a *single* letter followed by a period. As there are only about thirty different letters in the average Latin-derived alphabet, the likelihood of being a homograph of a non-abbreviation is very high for initials, consider e.g. the Portuguese articles *o* and *a* or the Swedish preposition *i*. Initials are therefore often not detected by the type-based first stage of the KS system. For this reason, every single letter followed by a token-final period is treated as a possible initial during the token-based reclassification – regardless of whether it has been classified as an abbreviation or not by the type-based stage. Luckily, initials are very often part of a complex name and can be identified using collocational evidence. If a possible initial forms a collocation with the following token and the following token is not a frequent sentence starter, the period in between is reclassified as an abbreviation marker. Alternatively, if the *orthographic decision heuristic* decides against a sentence boundary on the basis of the token following the possible initial, the period is also reclassified as an abbreviation period. Last but not least, if the *orthographic decision heuristic* returns *undecided* and the type following the possible initial always occurs with an uppercase first letter, it is assumed to be a name and the period between the two tokens is again classified as an abbreviation marker.

In many languages such as e.g. German, ordinal numbers written in digits are also marked by a token-final period. However, as every numeric type can also be used as a cardinal number, it cannot be decided by a type-based algorithm whether a period after a number is an abbreviation period or a sentence boundary marker. Numbers are therefore treated in the same way as initials.

¹³ This value was chosen because it represents a confidence degree of 99.95 % according to the χ^2 distribution and worked well on our English development corpus.

¹⁴ The threshold value 30 was determined experimentally on our development corpus.

If the token following a number with a final period forms a collocation with the abstract type `##number##` and is not a frequent sentence starter, the period in between is classified as an abbreviation marker. The same conclusion is reached if the *orthographic decision heuristic* decides against a sentence boundary. In other languages such as English and Portuguese on which we did the evaluation for this paper, ordinal numbers are usually not marked with a period. For these languages, the detection of ordinal numbers can be turned off. The results of the test runs of the KS system reported in section 4 were determined with the detection of ordinal numbers switched off. As the detection of ordinal numbers is a major feature of the KS system we have described it here nonetheless.

4 Experiments and Results

The RE system, MxTerminator, and Satz had already been evaluated by Silla Jr. & Kaestner in a previous comparative study [3] on English and Portuguese documents. For the current paper, we have used the same two test corpora in English and Portuguese to evaluate the unsupervised system by Kiss & Strunk. This allows for a direct comparison of the performance of all four systems.¹⁵

In order to perform the experiments, each of the test documents had its sentence boundaries tagged manually. The different systems were then run on these test documents and the resulting annotation was compared to the reference annotation. We use the following performance measures: *Precision* is calculated as the percentage of correctly classified sentence boundaries, i.e. the number of sentence boundaries *correctly* identified divided by the number of sentence boundaries identified. *Recall* indicates the percentage of sentence boundaries present in the document that were actually found by a particular system, i.e. the number of sentence boundaries correctly identified divided by the number of sentence boundaries present in the reference annotation. The *f-measure* combines precision and recall in a single metric: the harmonic mean. As an indication of the difficulty of the sentence boundary detection task on the two test corpora, we compare the results of the four systems with a simple baseline, which assumes that every token-final period indicates a sentence boundary.

As the test corpus for English, we used part of the TIPSTER document collection from the Text Retrieval Conference, which contains articles from the Wall Street Journal (TREC reference number: WSJ-910130). This corpus comprises 156 documents of different sizes, totaling 3,554 sentences. We performed two different test runs with the unsupervised system by Kiss & Strunk: For the first one, we used the individual files containing the WSJ articles; for the second one, we provided the system with all articles pasted together in a single file. This was necessary to ensure a fair comparison between the different systems because the KS system does not use any additional training data but instead learns from the

¹⁵ Kiss & Strunk have carried out a more extensive evaluation of their system on further languages and genres which is reported in [2].

test corpus on the fly. When the test corpora used are very small, the KS system is likely to suffer from data sparseness.¹⁶

Table 2 shows the results achieved on the English test corpus by the four systems. When the KS system is run on the corpus as a single file, it produces only slightly worse results than Satz – a supervised system which uses a language specific lexicon and abbreviation list – and the RE system – which has been specifically tailored to English newspaper texts – and it is even slightly better than MxTerminator – a more straightforward machine learning system. When the KS system is tested on the individual WSJ articles, which sometimes contain less than ten sentences, performance drops considerably due to data-sparseness but is still much better than the baseline.

Table 2. Results on the TIPSTER document collection (English)

System	Precision	Recall	F-Measure
Baseline	30,29 %	50,61 %	37,89 %
KS (individual files)	80,43 %	83,40 %	81,88 %
MxTerminator	91,19 %	91,25 %	91,22 %
KS (single file)	90,70 %	92,34 %	91,51 %
RE	92,39 %	91,18 %	91,78 %
Satz	98,67 %	85,98 %	91,88 %

For Portuguese, we used the Lacio-Web Corpus [11], which contains 21,822 sentences in all. The systems were tested on this corpus using 10-fold cross-validation. The results achieved are presented in Table 3.

Table 3. Results on the Lacio-Web document collection (Portuguese)

System	Precision	Recall	F-Measure
Baseline	85,40 %	92,25 %	88,69 %
RE	91,80 %	88,02 %	89,87 %
MxTerminator	96,31 %	96,63 %	96,46 %
KS	97,58 %	96,87 %	97,22 %
Satz	99,59 %	98,74 %	99,16 %

The results of the KS system on Portuguese are better than those of MxTerminator and the RE system. The KS system is only second to Satz. It has to be kept in mind, however, that Satz, MxTerminator, and the RE system had to be customized before applying them to the Portuguese corpus, while the KS system was used as is both for English and Portuguese.

¹⁶ This could be remedied by equipping the KS system with a kind of memory function, so that it is able to remember data from previous test runs.

5 Conclusions

We have described an unsupervised approach to sentence boundary detection developed by Kiss & Strunk and have presented the results of a comparative evaluation of this approach and three earlier systems – the RE system, MxTerminator, and Satz – on English and Portuguese corpora. We conclude that the unsupervised approach can be very useful since it can be used out of the box for new languages and genres, while the supervised or rule-based approaches have to be adapted by hand or need retraining, which requires resources that are not always available. Moreover, the performance of the unsupervised KS system is only slightly worse and sometimes even better than that of the other systems.

Acknowledgments

We would like to thank Adwait Ratnaparkhi for sending us the MxTerminator system, and Marti A. Hearst for providing the original files used by Satz.

References

1. Lyman, P., Varian, H.R.: How much information. Retrieved from <http://www.sims.berkeley.edu/how-much-info-2003> on [01/19/2004] (2003)
2. Kiss, T., Strunk, J.: Multilingual unsupervised sentence boundary detection. <http://www.linguistics.rub.de/~strunk/ks2005FINAL.pdf> (Under Review)
3. Silla Jr., C.N., Kaestner, C.A.A.: An analysis of sentence boundary detection systems for English and Portuguese documents. In Gelbukh, A., ed.: *Computational Linguistics and Intelligent Text Processing*. Volume 2945 of *Lecture Notes in Computer Science*, CAU, Seoul, Korea, Springer Verlag (2004) 135–141
4. Reynar, J., Ratnaparkhi, A.: A maximum entropy approach to identifying sentence boundaries. In: *Proceedings of the Fifth Conference on Applied Natural Language Processing*. (1997) 16–19
5. Palmer, D.D., Hearst, M.A.: Adaptive multilingual sentence boundary disambiguation. *Computational Linguistics* **23/2** (1997) 241–267
6. Kiss, T., Strunk, J.: Scaled log likelihood ratios for the detection of abbreviations in text corpora, *Proceedings of COLING 2002, Taipei* (2002) 1228–1232
7. Kiss, T., Strunk, J.: Viewing sentence boundary detection as collocation identification, *Proceedings of KONVENS 2002, Saarbrücken* (2002) 75–82
8. Nunberg, G.: *The Linguistics of Punctuation*. CSLI Lecture Notes Number 18. Center for the Study of Language and Information, Stanford, California (1990)
9. Dunning, T.: Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics* **19/1** (1993) 61–74
10. Manning, C.D., Schütze, H.: *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge/London (1999)
11. Aluisio, S.M., Pinheiro, G.M., Finger, M., Nunes, M.G.V., Tagnin, S.E.: The Lacio-Web Project: Overview and issues in Brazilian Portuguese corpora creation. In: *Proceedings of Corpus Linguistics 2003*. (2003) 14–21

A General and Multi-lingual Phrase Chunking Model Based on Masking Method

Yu-Chieh Wu¹, Chia-Hui Chang¹, and Yue-Shi Lee²

¹ Department of Computer Science and Information Engineering, National Central University,
No.300, Zhong-Da Rd., Zhongli City, Taoyuan County 32001, Taiwan, R.O.C.

bcbb@db.csie.ncu.edu.tw, chia@csie.ncu.edu.tw

² Department of Computer Science and Information Engineering, Ming Chuan University,
No.5, De-Ming Rd, Gweishan District, Taoyuan 333, Taiwan, R.O.C.

leeys@mcu.edu.tw

Abstract. Several phrase chunkers have been proposed over the past few years. Some state-of-the-art chunkers achieved better performance via integrating external resources, e.g., parsers and additional training data, or combining multiple learners. However, in many languages and domains, such external materials are not easily available and the combination of multiple learners will increase the cost of training and testing. In this paper, we propose a mask method to improve the chunking accuracy. The experimental results show that our chunker achieves better performance in comparison with other deep parsers and chunkers. For CoNLL-2000 data set, our system achieves 94.12 in F rate. For the base-chunking task, our system reaches 92.95 in F rate. When porting to Chinese, the performance of the base-chunking task is 92.36 in F rate. Also, our chunker is quite efficient. The complete chunking time of a 50K words document is about 50 seconds.

1 Introduction

Automatic text chunking aims to determine non-overlap phrases structures (chunks) in a given sentence. These phrases are non-recursive, i.e., they cannot be included in other chunks [1]. Generally speaking, there are two phrase chunking tasks, including text chunking (shallow parsing) [15], and noun phrase (NP) chunking [16]. The former aims to find the chunks that perform partial analysis of the syntactic structures in texts [15], while the later aims to identify the initial portions of non-recursive noun phrase, i.e., the first level noun phrase structures of the parsing trees [17] [19]. In this paper, we extend the NP chunking task to arbitrary phrase chunking, i.e., base-chunking. In comparison, shallow parsing extracts not only the first level but also the other level phrase structures of the parsing tree into the flat non-overlap chunks.

Chunk information of a sentence is usually used to present syntactic relations in texts. In many Natural Language Processing (NLP) areas, e.g., chunking-based full parsing [1] [17] [24], clause identification [3] [19], semantic role labeling (SRL) [4], text categorization [15] and machine translation, the phrase structures provide down-stream syntactic features for further analysis. In many cases, an efficient and high-performance chunker is required. In recent years, many high-performance

chunking systems were proposed, such as, SVM-based [9], Winnow [13] [20], voted-perceptrons [3], Maximum Entropy model (ME) [12], Hidden Markov Model (HMM) [11] [14], Memory-based [17] [19], etc. Although some of the outstanding methods gave better results, they were not efficient. In average the chunking speed is about 3-5 sentences per second. Moreover, some of them require external resources, i.e., parser (Winnow[20]), and more training data (HMM[14]), or combining multiple learners (memory-based [19] and SVM-based [9]) to enhance chunking performance. However, the use of multiple learners does not only complicate the original system but also increase chunking time largely. In practice, external resources are not always available in many domains and languages. On the other hand, although some chunkers (e.g., HMM and memory-based chunkers), are quite efficient, they do not have exhilarating performances.

In this paper, we present a novel chunking method to improve the chunking accuracy. The mask method we propose is designed to solve the “unknown word problem” as many chunking errors occur due to unknown words. Imagine the cases when unknown words occur in the testing data, all lexical-related features, for example, unigram, can not be properly represented, thus the chunk type has to be determined by other non-lexical features. To remedy this, we propose a mask method to collect unknown word examples from the original training data. These examples are derived from mapping variant incomplete lexical-related features. By including these instances, the chunker can handle testing data, which contains unknown words. In addition, we also combine a richer feature set to enhance the performance. Based on the two constituents, the mask method and richer feature sets, higher performance is obtained. In the two main chunking tasks, our method outperforms the other famous systems. Besides, this model is portable to other languages. In the Chinese base-chunking task, our chunking system achieves 92.19 in F rate. In terms of time efficiency, our model is satisfactory, and thus able to handle the real-time processes, for example, information retrieval and real-time web-page translation. In a 500K words document, the complete chunking time is about 50 seconds.

The rest of this paper is organized as follows. Section 2 introduces the two main tasks: shallow parsing and base phrase chunking. Section 3 explains our chunking model. The mask method will be described in Section 4. Experimental results are showed in Section 5. Concluding remarks and future work are given in Section 6.

2 Descriptions of the Chunking Tasks

A chunk (phrase) is a syntactic structure, which groups several consecutive words to form a phrase. In this section, we define the two phrase chunking tasks, base-chunking and shallow parsing.

2.1 Base-Chunking

The phrase structures of base-chunking task is similar to that of baseNP chunking [16], but includes all atomic arbitrary phrase chunks in text. In Li and Roth’s works [13], they also compared their chunking system in the base-chunking tasks. Consider the following sentence: “Formed in August, the venture weds AT&T ‘s newly expanded 900 service”. As shown in Fig. 1, the parent node of each word (leaf node) is the

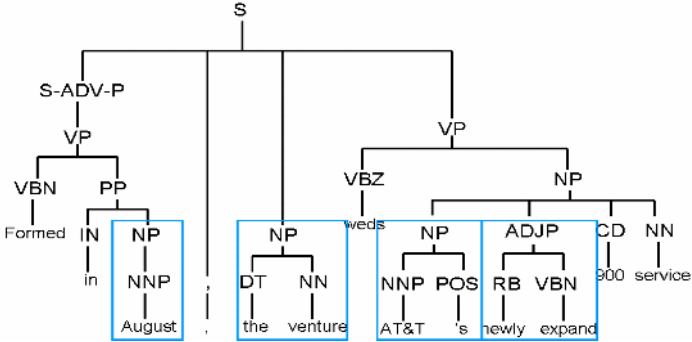


Fig. 1. A Parsing tree for “Formed in August, the venture weds AT&T ‘s newly expanded 900 service”

part-of-speech (POS) tag. The first level chunks of the parsing tree are the pre-terminals that contain no sub-phrases. The base chunks of the above sentence are oval-shaped rectangles including “August”, “the venture”, “AT&T ‘s” and “newly expand”. The other phrase structures can not form the base chunk since they contain sub-phrases, for example, VP (verb phrase).

The phrase structures of the sentence can also be encoded using IOB2 style [18]. The major constituents of the IOB2 style are B/I/O tags and the phrase type, which represent the begin (B) of a phrase, the interior (I) of a phrase and other words (O). For example, the chunk class of each token of the above sentence can be tagged as:

Formed (O) in (O) August (B-NP) , (O) the (B-NP) venture (I-NP) weds (O) AT&T (B-NP) ‘s (I-NP) newly (B-ADJP) expanded (I-ADJP) 900 (O) service (O)

When the chunk structure is encoded as IOB-like style, a chunking problem can be viewed as a word-classification task, i.e., identify IOB chunk tag for each word. Many chunkers [9] [13] [20] learn to label the IOB chunk tags using a classic classification scheme. We also follow the same scheme to design our model. More details of our model can be found in Section 3.

2.2 Shallow Parsing

Shallow parsing is also as known as text chunking which performs partial analysis of the parsing tree. It was the shared task of CoNLL-2000. The phrase structure of the shallow parsing is quite different from the base-chunking. Roughly speaking, in shallow parsing, a chunk contains everything to the left of and including the syntactic head of the constituent of the same name. Shallow parsing focuses on dividing a text into phrases in such a way that syntactically related words become member of the same phrase type. So far, there are no annotated chunk corpora available which contain specific information about dividing sentences into chunks of words of variant phrase types. Thus, CoNLL-2000 [18] defines the shallow parsing phrase structures from parsing trees. Following the chunk definition in [18], the shallow parsing phrase structures are:

Formed (**B-VP**) in (**B-PP**) August (B-NP) , (O) the (B-NP) venture (I-NP) weds (**B-VP**) AT&T (B-NP) 's (I-NP) newly (B-ADJP) expanded (I-ADJP) 900 (**B-NP**) service (**I-NP**)

In this case, even the term “Fomed” does not belong to the first level, it is specified as a verb phrase. A formal detail definition of the phrase structures of the shallow parsing can refer to the web site¹ and literatures [18].

3 Chunking Model

As described in Section 2.1, the chunking problem can be viewed as a series of word-classification [13] [16]. Many common NLP components, for example Part-of-Speech (POS) taggers [7] and deep parsers [17] [19] were represented according to the “word-classification” structure. The proposed general chunking model is also developed following the same scheme. In general, the contextual information is often used as the seed feature type; the other features can then be derived based on the surrounding words. In this paper, we adopt the following feature types.

- **Lexical information (Unigram)**
- **POS tag information (UniPOS)**
- **Affix (2~4 suffix and prefix letters)**
- **Previous chunk information (UniChunk)**
- **Possible chunk classes for current word:** For the current word to be tagged, we recall its possible chunk tags in the training data and use its possible chunk class as a feature.

Additionally, we also add more *N*-gram features, including **Bigram**, **BiPOS**, **BiChunk**, and **TriPOS**. In addition, we design an orthographic feature type called **Token feature**, where each term will be assigned to a token class type via the pre-defined word category mapping. Table 1 lists the defined token feature types. Although this feature type is language dependent, many languages still contain Arabic numbers and symbols.

Table 1. Token feature category list

Feature description	Example text	Feature description	Example text
1-digit number	3	Number contains alpha and slash	1/10 th
2-digit number	30	All capital word	SVM
4-digit number	2004	Capital period (only one)	M.
Year decade	2000s	Capital periods (more than one)	I.B.M.
Only digits	1234	Alpha contains money	US\$
Number contains one slash	3/4	Alpha and periods	Mr.
Number contains two slash	2004/8/10	Capital word	Taiwan
Number contains money	\$199	Number and alpha	F-16
Number contains percent	100%	Initial capitalization	Mr., Jason
Number contains hyphen	1-2	Inner capitalization	WordNet
Number contains comma	19,999	All lower case	am, is, are
Number contains period	3.141	Others	3√4
Number contains colon	08:00		

¹ <http://www.cnts.ua.ac.be/conll2000/chunking/>

We employ SVM^{light} [8] as the classification algorithm, which has been shown to perform well on classification problems [7][8][15]. Since the SVM algorithm is a binary classifier, we have to convert it into several binary problems. Here we use the “One-Against-All” type to solve the problem. To take the time efficiency into account, we choose the linear kernel type. As discussed in [7], working on linear kernel is much more efficient than polynomial kernels. In Section 5, we also demonstrated that the training/testing time of the polynomial kernel is longer than linear kernels while causing a slight improvement.

4 Mask Method

In real world, training data is insufficient, since only a subset of the vocabularies can appear in the testing data. During testing, if a term is an unknown word (or one of its context words is unknown), then the lexical related features, like unigram, and bigram are disabled, because the term information is not found in the training data. In this case, the chunk class of this word is mainly determined by the remaining features. Usually, this will low down the system performance.

The most common way for solving unknown word problem is to use different feature sets for unknown words and divide the training data into several parts to collect unknown word examples (Brill, 1995; Nakagawa et al., 2001; Gimenez & Marquez, 2003). However, the selection of these feature sets for known word and unknown word were often arranged heuristically and it is difficult to select when the feature sets are different. Moreover, they just extract the unknown word examples and miss the instances that contain unknown contextual words.

To solve this problem, the mask method is designed to produce additional examples that contain “weak” lexical information to train. If the classification algorithm can learn these instances, in testing, it is able to classify the examples, which contain insufficient lexical information. The mask method is described in Fig. 2. The method works as follows. First, the training data is divided into k non-overlapped partitions. For each partition, we create a mask, which is used to conceal part of the features for each example in the training set. New representations for each training example are generated, thus increasing the size of training set. The mask is created as follows. Suppose we have derived the feature dictionary, F , from the training set T . By remove partition i , we have a smaller training set with feature dictionary (F_i) smaller than F . We then generate new training examples by mapping the new dictionary set F_i , that is, lexicon-related features that do not occur in F_i are masked. Technically, we create a mask m_i of length $|F|$ where a bit is set for a lexicon in F_i and clear if the lexicon is not in F_i . We then generate new vectors for all examples by logical “AND” it with mask m_i . Thus, items which appear only in part i are regarded as unknown words. The process is repeated for k times and a total of $(k+1)*N$ example vectors are generated (N is the original number of training examples). Computationally, we do not need to generate F_i from scratch. Instead, F_i can be created from F by replace lexicon-related features UD_i , BD_i , and ID_i generated from T' , the training set except from partition i (since other non-lexicon related features are not masked).

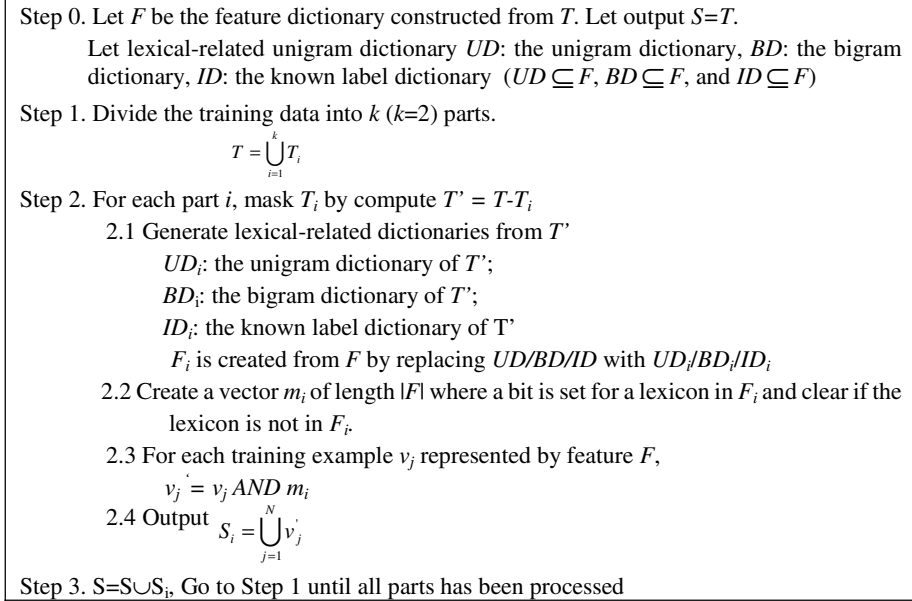


Fig. 2. The mask method to generate incomplete information examples

Let us illustrate a simple example. Assume that the unigram is the only one selected feature, and each training instance is represented via mapping to the unigram dictionary. At the first stage, the whole training data set generates the original unigram dictionary, T : (A, B, \dots, G) . After splitting the training data (Step 1), two disjoint sub-parts are produced. Assume step 2.1 produces new unigram dictionaries, T_1 : (B, C, D, E) by masking the first part, and T_2 : (A, C, D, F, G) by masking the second part. Thus, the mask for the first partition is $(0, 1, 1, 1, 1, 0, 0)$ which reserves the common items in T_0 and T_1 . For a training example with features (A, C, F) , the generated vectors is (C) since A and F are masked (Step 2.3). We use the same way to collect training examples from the second part.

The mask technique can transform the known lexical features into unknown lexical features and add k times training materials from the original training set. Thus, the original data can be reused effectively. As outlined in Fig. 2, new examples are generated through mapping into the masked feature dictionary set F_i . This is quite different from previous approaches, which employed variant feature sets for unknown words. The proposed method aims to emulate examples that do not contain lexical information, since chunking errors often occur due to the lack of lexical features. Traditionally, the learners are given sufficient and complete lexical information; therefore, the trained models cannot generalize examples that contain incomplete lexical information. By including incomplete examples, the learners can take the effect of unknown words into consideration and adjust weights on lexical related features.

5 Experimental Results

In this section, four experiments are presented. First, we report the effects of various feature combinations, and the masking method on three chunking tasks, shallow parsing, base-chunking and Chinese base-chunking. Second, we compare our chunking performance to the other chunking systems using the same benchmarking corpus. The third experiment reports the detail performance on Chinese chunking task. The final experiments report the results using polynomial SVM kernel. The benchmarking corpus of the base-chunking is Wall Street Journal (WSJ) of the English Treebank, sections 02-21 for training and section 23 for testing. For the shallow parsing task, the WSJ sections 15-18 for training and 20 for testing. The POS-tag information of base-chunking and shallow parsing tasks is mainly generated from Brill-tagger [2]. However, in Chinese, there is no benchmarking POS-tagger. Thus, we use the gold POS-tags in Chinese Treebank.

The performance of the chunking task is measured by three rates, recall, precision, and $F_{(\beta=1)}$. CoNLL released a perl-script evaluator that enabled us to estimate the three rates automatically.

5.1 Analysis of Feature Combination and the Mask Method

There are three parameters in our chunking system: the context window size, the frequent threshold for each feature dictionary, and the number of division parts for the unknown word examples (N). We set the first two parameters to 2 as previous chunking systems [9]. Since the training time taken by SVMs scales exponentially with the number of input examples [8], we set N to 2 for all of the following experiments.

The first experiment reports the system performance of different feature combinations. Table 2 lists the chunking results of the added features on the shallow parsing task. For the feature set (4), which combines uni-gram and bi-gram features, it does a great improvement. On the contrary, the proposed ‘‘TokenFeature’’ marginally improves the performance. The best system performance is achieved by combining all of the features, i.e. feature set (5). The feature selection set in the following parts is (5).

In another experiment, we concern the performance of the mask method. Table 3 lists the improvement of the method. The mask method improves system performance in different chunking tasks and different language. Note that there is not a public and well-known Chinese shallow parsing task definition. Thus, we did not perform the shallow parsing task on the Chinese Treebank.

Table 2. System performance on different feature sets in the shallow parsing task

	Features	Recall	Precision	$F_{(\beta)}$
(1)	Unigram+UniPOS+UniChunk+PossibleChunk	92.35	91.87	92.11
(2)	(1)+TokenFeature	92.40	91.95	92.18
(3)	(2)+Affix	92.64	92.18	92.41
(4)	(3)+Bigram+BiPOS+BiChunk	93.52	93.51	93.52
(5)	(4)+TriPOS	93.60	93.53	93.56

Table 3. The improvement by the mask method on Chinese and English corpus

CoNLL-2000	Base-chunking	Shallow-parsing
English	91.96 → 92.95	93.56 → 94.12
Chinese	91.76 → 92.36	N/A

The three statistical tests, s-test, probability distributional test, and McNemar’s test are applied to evaluate the improvement significance. Table 4 lists the results of the three tests. “P<0.01” means the two systems have statistical significant difference under the 99% confidence value. Three tests indicate that the mask method improves the chunking performance on these chunking tasks.

To exploit the detail performance analysis of the mask method, Table 5 shows the chunking performance on the known and unknown phrases. As listed in Table 5, the percentage of unknown phrases in the testing set is in the testing set is about 13.53% and the improvement for unknown words is from 89.84 to 90.84. The mask method improves a lot in the unknown phrase chunking. Moreover, the mask method also improves the chunking performance on known word. As discussed in Section 4, when unknown word appears, its lexical features are not available. The main idea of the mask method is to produce the training examples that contain incomplete lexical features. In testing phase, when the unknown word appears, the classification algorithm can identify the chunk class since it may “similar” to the incomplete training examples.

Table 4. Statistical significance test results

Chunking task	System A	System B	s-test	p-test	M’s-test
Shallow parsing	mask method	Without mask	P<0.01	P<0.01	P<0.01
Base-chunking		method	P<0.01	P<0.01	P<0.01
Chinese base-chunking			P<0.01	P<0.01	P<0.01

Table 5. The improvement by the mask method

Shallow-parsing	Percentage	Improvement
Unknown	13.53%	89.84 → 90.84
Known	86.46%	94.34 → 94.76
Total	100.00%	93.56 → 94.12

5.2 Comparisons to Other Chunking Systems

In this section, we compare our model to currently state-of-the-art parsing systems [5] [6]. Here, we employed the Brill-tagger [2] to generate POS tags for our chunker and Collins’ parser. However, we do not feed the same POS tags to Charniak’s parser, since it takes the global optimize of the parsing tree structure into account. Table 6 lists the experimental results of each chunk type.

Table 7 reports the base-chunking results of the other famous parsing systems, Charniak’s maximum entropy inspired parser [5] and the Collins’ headword-driven parser [6]. Among the three systems, the Charniak’s parser performs the second best on the base-chunking tasks while the Collins parser achieves the third best performance.

Table 6. Base-chunking performance of our model in different phrase type

	Recall	Precision	$F_{(\beta=1)}$		Recall	Precision	$F_{(\beta=1)}$
ADJP	70.00	78.24	73.89	PRN	12.50	100.00	22.22
ADVP	88.54	85.96	87.23	PRT	79.25	79.95	79.50
CONJP	71.43	83.33	76.92	QP	91.22	88.51	89.85
FRAG	0.00	0.00	0.00	UCP	18.18	100.00	30.77
INTJP	81.82	75.00	78.26	VP	84.33	93.72	88.78
LST	0.00	0.00	0.00	WHADJP	66.67	100.00	80.00
NAC	52.63	83.33	64.52	WHADVP	92.59	96.90	94.70
NP	95.70	94.41	95.05	WHNP	96.40	96.17	96.29
NX	8.64	77.78	15.56	X	66.67	66.67	66.67
PP	25.00	40.00	30.77	All	92.93	92.98	92.95

Table 7. Comparisons of chunking performance for base-chunking task

Chunking system	Recall	Precision	$F_{(\beta)}$
This paper	92.93	92.98	92.95
Charniak's full parser [5]	91.81	92.73	92.27
Collins' full parser [6]	89.68	89.70	89.69

Table 8. Statistical significance test results

System A	System B	s-test	p-test	M's-test
This paper	Charniak	P<0.01	P<0.01	P<0.01
This paper	Collins	P<0.01	P<0.01	P<0.01
Charniak	Collins	P<0.01	P<0.01	P<0.01

The three statistical tests, s-test, probability distributional test, and McNemar's test again agree with the significant difference between our model and the two parsers under the 99% confidence score. Table 8 displays the statistical test results.

In the shallow parsing task, we compare our chunker with other chunking systems under the same constraint, i.e. all of the settings should be coincided with the CoNLL-2000 shared task. In other words, the use of external resources or other components is disabled. Table 9 lists our chunking results on each chunk type.

Table 10 reports the chunking results of other systems. In this test, the second best chunking system is the voted-SVMs [9]. As listed in Table 10, our model outperforms the other chunking systems. However, it is difficult to perform the statistical tests on these chunkers. Since the outputs of these systems are not available easily.

Table 9. Shallow parsing performance of our model in different phrase type

	Recall	Precision	$F_{(\beta=1)}$		Recall	Precision	$F_{(\beta=1)}$
ADJP	71.92	81.61	76.46	NP	94.51	94.67	94.59
ADVP	81.64	83.27	82.45	PP	98.27	96.87	97.57
CONJP	55.56	45.45	50.00	PRT	79.25	76.36	77.78
INTJP	50.00	100.00	66.67	SBAR	86.54	87.86	87.19
LST	0.00	0.00	0.00	VP	94.57	94.10	94.34
				All	94.12	94.13	94.12

Table 10. Comparison of chunking performance for text-chunking task

Chunking system	Recall	Precision	$F_{(\beta)}$
This paper	94.12	94.13	94.12
Voted-SVMs [9]	93.89	93.92	93.91
Voted-perceptrons [3]	94.19	93.29	93.74
Generalized Winnow [20]	93.60	93.54	93.57

5.3 Experimental Results on Chinese Base-Chunking Tasks

We port our chunker into Chinese base-chunking task with the same parameter settings. There are about 0.4 million words in the Chinese Treebank. The front part of 0.32 million words forms the training data while the other 0.08 million words as the testing part. The ratio of training and testing data is equivalent to 4:1. Table 11 lists the experimental results of the Chinese base-chunking task.

Table 11. Chinese base-chunking performance of our model in different phrase type

	Recall	Precision	$F_{(\beta=1)}$		Recall	Precision	$F_{(\beta=1)}$
ADJP	98.98	97.86	98.42	PP	0.00	0.00	0.00
ADVP	99.68	99.47	99.58	PRN	0.00	0.00	0.00
CLP	99.88	99.69	99.79	QP	97.61	97.52	97.56
CP	0.00	0.00	0.00	VCD	60.22	63.64	61.88
DNP	0.00	0.00	0.00	VCP	87.50	100.00	93.33
DP	99.40	99.10	99.25	VNV	20.00	100.00	33.33
DVP	0.00	0.00	0.00	VP	91.64	94.92	93.25
FRAG	98.07	98.64	98.36	VPT	100.00	55.56	71.43
LCP	0.00	0.00	0.00	VRD	87.98	88.95	88.46
LST	82.14	100.00	90.20	VSF	14.29	44.00	21.57
NP	88.04	90.31	89.16	All	91.48	93.27	92.36

It is worth to note that the affix feature in the Chinese base-chunking task is not explicitly. Therefore, we use the atomic Chinese characters to form the affix feature. Although, several Chinese shallow parsing systems were proposed in recent years, like HMM-based [11] and maximum entropy-based [12] methods. It is difficult to compare with these chunkers, because there is not a standard benchmark corpus and grammar rules to represent the shallow parsing structures. Nevertheless, they were performed on different training and testing set and employed various pre-defined grammar rules. Thus, we only report the actual results of the proposed chunking model for Chinese base-chunking task.

5.4 Working on Polynomial Kernel

Previous studies [7] indicated that using polynomial kernel to SVM is more accurate than linear kernel but cause much time cost. In this section, we report the performance of our method using polynomial kernel instead of the linear kernel. Table 12 lists the chunking results on the English shallow parsing task. In this experiment, the training time is about 4 days, besides it took 3 hours for chunking. Although the use of polynomial kernel improves the performance, the time cost is largely increased. When working on linear kernel, the training time of our model is less than 2.8 hours and 50

Table 12. Shallow parsing performance of our model using polynomial kernel

	Recall	Precision	$F_{(\beta=1)}$		Recall	Precision	$F_{(\beta=1)}$
ADJP	71.92	78.75	75.18	NP	94.74	94.67	94.71
ADVP	81.29	82.15	81.72	PP	98.32	97.01	97.66
CONJP	55.56	45.45	50.00	PRT	78.30	78.30	78.30
INTJP	100.00	100.00	100.00	SBAR	87.66	89.33	88.49
LST	0.00	0.00	0.00	VP	94.59	94.39	94.49
				All	94.26	94.16	94.21

seconds for testing. The three statistical tests disagree with the significant difference between the two kernels in 95% confidence score.

6 Conclusion

This paper proposes a general and language dependent chunking model based on combining rich features and the proposed mask method. In the two main chunking tasks (shallow parsing and base-chunking), our method outperforms the other systems which employed more training materials or complex models. The statistical tests also report the proposed mask method significantly improves the system performance under a 99% confidence score. The online demonstration of our chunkers can be found at (<http://140.115.155.87/bcbb/chunking.htm>).

Acknowledgement

This work is sponsored by National Science Council, Taiwan under grant NSC94-2524-S-008-002 and NSC94-2622-E-130-001-CC3.

References

1. Abney, S.: Parsing by chunks. In *Principle-Based Parsing. Computation and Psycholinguistics* (1991) 257-278.
2. Brill, E.: Transformation-based error-driven learning and natural language processing: a case study in part of speech tagging. *Computational Linguistics* (1995) 21(4):543-565.
3. Carreras, X. and Marquez, L.: Phrase recognition by filtering and ranking with perceptrons. *Proceedings of the International Conference on Recent Advances in Natural Language Processing* (2003).
4. Carreras X. and Marquez, L.: Introduction to the CoNLL-2004 shared task: semantic role labeling. *Proceedings of Conference on Natural Language Learning* (2004) 89-97.
5. Charniak, E.: A maximum-entropy-inspired parser. *Proceedings of the ANLP-NAACL* (2000) 132-139.
6. Collins, M.: Head-driven statistical models for natural language processing. Ph.D. thesis. University of Pennsylvania (1998).
7. Giménez, J. and Márquez, L.: Fast and accurate Part-of-Speech tagging: the SVM approach revisited. *Proceedings of the International Conference on Recent Advances in Natural Language Processing* (2003) 158-165.

8. Joachims, T.: A statistical learning model of text classification with support vector machines. In Proceedings of the 24th ACM SIGIR Conference on Research and Development in Information Retrieval (2001) 128-136.
9. Kudoh, T. and Matsumoto, Y.: Chunking with support vector machines. Proceedings of the 2nd Meetings of the North American Chapter and the Association for the Computational Linguistics (2001).
10. Li, H., Huang, C. N., Gao, J. and Fan, X.: Chinese chunking with another type of spec. The Third SIGHAN Workshop on Chinese Language Processing (2004).
11. Li, H., Webster, J. J., Kit, C. and Yao, T.: Transductive HMM based chinese text chunking. International Conference on Natural Language Processing and Knowledge Engineering (2003) 257-262.
12. Li, S. Liu, Qun, and Yang, Z.: Chunking based on maximum entropy. Chinese Journal of Computer (2003) 25(12): 1734-1738.
13. Li, X. and Roth, D.: Exploring evidence for shallow parsing. In Proceedings of Conference on Natural Language Learning (2001) 127-132.
14. Molina, A. and Pla, F.: Shallow parsing using specialized HMMs. Journal of Machine Learning Research (2002) 2:595-613.
15. Park, S. B. and Zhang, B. T.: Co-trained support vector machines for large scale unstructured document classification using unlabeled data and syntactic information. Journal of Information Processing and Management (2004) 40: 421-439.
16. Ramshaw, L. A. and Marcus, M. P.: Text chunking using transformation-based learning. Proceedings of the 3rd Workshop on Very Large Corpora (1995) 82-94.
17. Tjong Kim Sang, E. F.: Transforming a chunker to a parser. In Computational Linguistics in the Netherlands (2000) 177-188.
18. Tjong Kim Sang , E. F. and Buchholz, S.: Introduction to the CoNLL-2000 shared task: chunking. In Proceedings of Conference on Natural Language Learning (2000) 127-132.
19. Tjong Kim Sang, E. F.: Memory-based shallow parsing. Journal of Machine Learning Research (2002) 559-594.
20. Zhang, T., Damerau, F., and Johnson, D.: Text Chunking based on a Generalization Winnow. Journal of Machine Learning Research (2002) 2:615-637.

UCSG Shallow Parser

Guntur Bharadwaja Kumar and Kavi Narayana Murthy

Department of Computer and Information Sciences,
University of Hyderabad, India
knmuh@yahoo.com, g_vijayabharadwaj@yahoo.com

Abstract. Recently, there is an increasing interest in integrating rule based methods with statistical techniques for developing robust, wide coverage, high performance parsing systems. In this paper¹, we describe an architecture, called UCSG shallow parser architecture, which combines linguistic constraints expressed in the form of finite state grammars with statistical rating using HMMs built from a POS-tagged corpus and an A* search for global optimization for determining the best shallow parse for a given sentence. The primary aim of the design of the UCSG parsing architecture is developing a judicious combination of linguistic and statistical methods to develop wide coverage robust shallow parsing systems, without the need for large scale manually parsed training corpora. The UCSG architecture uses a grammar to specify all valid structures and a statistical component to rate and rank the possible alternatives, so as to produce the best parse first without compromising on the ability to produce all possible parses. The architecture supports bootstrapping with an aim to reduce the need for parsed training corpora. The complete system has been implemented in Perl under Linux. In this paper we first describe the UCSG shallow parsing architecture and then focus on the evaluation of the UCSG finite state grammar for the chunking task for English. Recall of 91.16% and 93.73% have been obtained on the Susanne parsed corpus and CoNLL 2000 chunking task test data set respectively. Extensive experimentation is under way to evaluate the other modules.

Keywords: Chunking, Shallow Parsing, Finite State Grammar, HMM, A* search, UCSG Architecture.

1 Introduction

Although a lot of work has gone into developing full syntactic parsers, high performance, wide coverage syntactic parsing has remained a difficult challenge [1]. In recent times, there has been an increasing interest in wide coverage and robust but partial or shallow parsing systems. Shallow parsing is the task of recovering only a limited amount of syntactic information from natural language sentences. Often shallow parsing is restricted to finding phrases in sentences, in which case it is also called chunking. Steve Abney[2], has described chunking as

¹ The research work reported here was supported in part by the University Grants Commission under the UPE scheme.

finding syntactically related non-overlapping groups of words. In CoNLL chunking task[3], chunking was defined as *the task of dividing a text into syntactically non-overlapping phrases.* The term *phrase* has come to acquire a very special technical connotation in linguistics and in order to avoid confusion, chunks are also referred to as *word groups*.

As an example, the sentence “He reckons the current account deficit will narrow to only # 1.8 billion in September” could be analyzed as follows by a chunker [3]:

[NP He] [VP reckons] [NP the current account deficit] [VP will narrow] [PP to] [NP only # 1.8 billion] [PP in] [NP September].

Note that prepositional phrases have not yet been built, let alone resolving ambiguities in prepositional phrase attachment. Nor have the thematic roles been assigned to the chunks. Partial parsing systems do a bit more than chunking while still not promising complete syntactic analysis.

Developing computational grammars is a challenging task, even if we restrict to partial parsing. There are broadly two approaches for the development of grammars - the linguistic approach which depends upon hand-crafted rules, and, the machine learning approach where grammars are learned automatically from a parsed training corpus. Developing wide coverage linguistic grammars has proved difficult in practice. Parsed training corpora are also rarely available. Hence the interest in the search for a judicious combination of linguistic and statistical approaches.

In this paper we propose an architecture for shallow parsing, which we call UCSG Shallow Parsing Architecture. The UCSG (Universal Clause Structure Grammar) framework was developed during the early nineties at University of Hyderabad, India. Please see [4, 5] for more on UCSG. In this paper we use only one of the modules of UCSG namely the Finite State Grammar. In the UCSG Shallow Parsing Architecture, the Finite State Grammar is designed to accept *all* valid word groups but not necessarily the *only* those word groups that are appropriate in context for a given sentence. Many additional word groups may also be recognized. The focus in this phase is only on completeness, for, there is a second module consisting of a set of Hidden Markov Models, which will rate and rank the word groups so produced. An A* search algorithm is then used as the final module to obtain globally best chunk sequences for a given sentence. The aim is to produce all possible parses but hopefully in the best first order. The complete system has been implemented in Perl under Linux. The performance of the Finite State Parser has been evaluated on the Susanne parsed corpus as well as CoNLL 2000 chunking task test data set and Recall of 91.16% and 93.73% respectively have been obtained. Extensive experimentation is going on to refine the system through bootstrapping and to evaluate the overall performance.

2 A Brief Survey of Shallow Parsing Systems

Steve Abney [6] proposed finite state cascade models for the chunking task. Grefenstette [7] proposed methods to use finite state transducers for partial

parsing. Parsing with finite state transducers [8] was very popular in the early ninety's. Marc Vilain et al. [9] used rule based sequence processors for the chunking task. Herve Dejean [10] used ALLiS (Architecture for Learning Linguistic Structure), which is a symbolic machine learning system for the chunking task.

Miles Osborne [11] proposed maximum entropy based POS tagger for the chunking task. Veenstra and Bosch [12] used memory based learning for chunking. Zhou et al. [13] proposed error driven HMM based chunk tagger with context dependent lexicon. Rob Koeling [14] applied maximum entropy models for chunking. Christer Johansson [15] proposed context sensitive maximum likelihood approach for chunking task. Tong Zhang et al. [16] proposed generalized winnow algorithm for text chunking. Recently Fei Sha and Pereira [17] used conditional random fields for noun phrase chunking and achieved good performance.

Molina and Pla [18] proposed shallow parsing with specialized HMMs. Carreras et al. [19] used perceptrons for chunking task. Recently, Gondy et al. [20] proposed a shallow parser based on closed-class words to capture relations in biomedical text.

Taku Kudoh et al. [21] proposed SVMs for chunking. This system performed the best in CoNLL-2000 chunking task and achieved an F-measure of 93.48%. Van Halteren [22] proposed Weighted probability distribution voting algorithm (WPDV) for chunking task. Tjong Kim Sang [23] proposed combination of several memory based learning systems for chunking task.

Most of the parsers described in literature have used either only rule based techniques or only machine learning techniques. Hand-crafting rules in the linguistic approach can be very laborious and time consuming. Parsers tend to produce a large number of possible parse outputs and in the absence of suitable rating and ranking mechanisms, selecting the right parse can be very difficult. Statistical learning systems, on the other hand, require large and representative parsed corpora for training. .

Recently, there is an increasing interest on integrating shallow parsers with deep parsing. Berthold Crysmann et al. [24] reported an implemented system called WHITEBOARD which integrates different shallow components with a HPSG based deep parsing system. Ronald M. Kaplan et. al. proposed a hybrid architecture called XLE [25] for combining finite state machine with LFG grammar. In XLE system, first the surface forms are run through the FST morphology to produce the corresponding stems and tags. Stems and tags each have entries in the LFG lexicon. Sub-lexical phrase structure rules produce syntactic nodes covering these stems and tags and standard grammar rules then build larger phrases.

3 UCSG Shallow Parsing Architecture

Purely linguistic approaches have not proved practicable for developing wide coverage grammars and purely machine learning approaches are also impracticable in many cases due to the non-availability of large enough parsed training corpora. Only a judicious combination of the two approaches can perhaps lead to wide coverage grammars and robust parsing systems. In the UCSG Shallow

Parsing Architecture, instead of looking for a grammar that can capture *all and only* valid structures, simultaneous satisfaction of both the requirements having proved very difficult in practice, we employ a Finite State Grammar that is general enough to capture *all* valid word groups without necessarily restricting to *only* those word groups which are appropriate in the context of a given sentence, and a separate statistical component, encoded in HMMs (Hidden Markov Model), to rate and rank the word groups so produced. Note that we are not pruning, we are only rating and ranking the word groups produced. The aim is to produce parse outputs in best first order, without compromising on the ability to produce all possible parses. This system is thus more than a chunker - the word groups produced are often bigger, disambiguated in context to some extent (for example, VVG is disambiguated between a gerund, a present participle and part of a present continuous verb group) and motivated by insights from deeper parsing requirements. This work is part of a larger on-going effort. The UCSG Shallow Parsing Architecture is depicted in Figure 1.

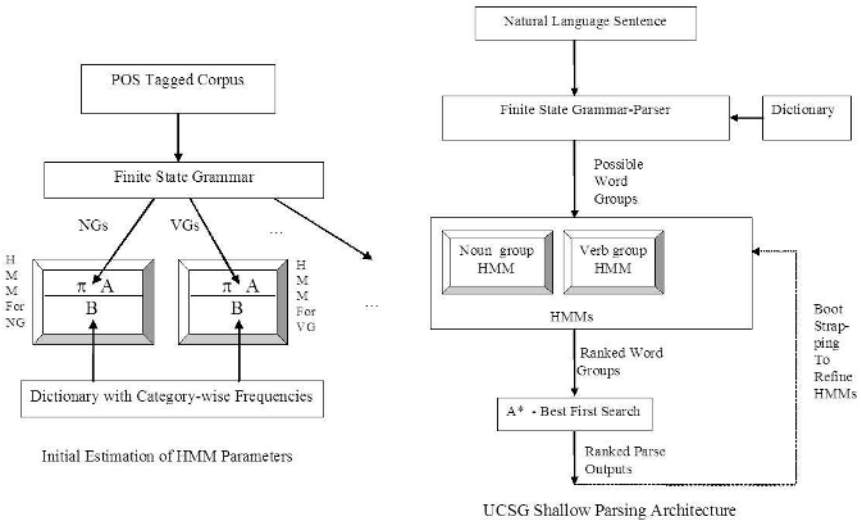


Fig. 1. UCSG Shallow Parsing Architecture

3.1 Finite State Parser

Unlike linguistic grammar formalisms and corresponding full parsers that attempt to capture the full hierarchical and thematic structure within sentences, partial parsing systems and chunkers only need to identify non-overlapping, non-recursive word groups or chunks. Thus the power of Context Free Grammars (also known generally as phrase structure rules in Linguistics) is not required and the simpler Finite State Grammars are sufficient. Finite State Grammars capture simple constraints such as linear precedence, optional items and repetitions but not arbitrarily deep hierarchical nestings or general dependencies

across constituents. Finite State Grammars can be developed with relative ease as compared to developing computational grammars for capturing full syntax. Also word groups can be recognized using these Finite State Machines in linear time [26].

The first module in the UCSG architecture is a Finite State Grammar-Parser. The aim is to develop a general enough grammar that can capture *all* valid word groups. Many additional word groups may be produced due to lexical ambiguities. We do not aim to restrict or prune the various possibilities. Instead we use a separate module to rate and rank these word groups.

The input to the system is one sentence at a time, either plain or POS-tagged. In the former case, the dictionary is consulted to obtain all the possible tags for each word. Our dictionary includes 128,000 root words, grammatical categories in the Claws-5 tag-set format, and frequency of each word in each of the possible categories. This dictionary has been developed over the last many years, cross checked against several large corpora including the British National Corpus and Reuters News Corpus. A coverage of 90 to 97 percent plus has been observed on various corpora. Inflectional morphology is handled by the parser and words still not analyzed are taken by default to be proper nouns. We give a running example to illustrate the working of the system through various stages. The input to the system is a sentence in plain text format: *'He was walking along a quiet street when gunmen shot him several times in the head.'* After Dictionary lookup, we have:

```
<PNPNOM>he <VBD>was <VVG>walking> <AVP_PRP>along <AT0>a
<VVB_AJ0_NN1_VVI>quiet <NN1>street <CJS_AVQ>when
<NN2>gunmen <VVN_VVD_NN1>shot <PNPACC>him
<DT0_PNP_AJ0>several <VVZ_NN2> <times> <PRP_AVP_AJ0>in
<AT0>the <VVB_AJ0_NN1_VVI>head
```

Note the lexical ambiguities. Sample output from the Finite State Parser is given below. Chunks identified are given one per line, starting with the category of the chunk, followed by the words along with the corresponding tags.

```
<vg> <VBD><was>
<vg> <VBD><was><VVG><walking>
<vg> <VBD><was><VVG><walking><AVP><along>
<vgs> <VVG><walking>
<vg> <VVB><quiet>
<ng> <VVG><walking>
<ng> <PRP><along><AT0><a><NN1><quiet>
<ng> <PRP><along><AT0><a><NN1><quiet><NN1><street>
<ng> <PRP><along><AT0><a><AJ0><quiet><NN1><street>
<ng> <AT0><a><NN1><quiet>
<ng> <AT0><a><AJ0><quiet><NN1><street>
<ng> <AT0><a><NN1><quiet><NN1><street>
```

```

<ng> <NN1><quiet>
<ng> <AJ0><quiet><NN1><street>
<ng> <NN1><quiet><NN1><street>
<ng> <NN1><street>
<ng> <VVG><walking><PRP><along><AT0><a>
  <AJ0><quiet><NN1><street>
<ng> <VVG><walking><PRP><along><AT0><a>
  <NN1><quiet><NN1><street>
<ng> <VVG><walking><PRP><along><AT0><a><NN1><quiet>
<ajg> <VVG><walking>
<ajg> <AJ0><quiet>
<part> <AVP><along>

```

3.2 HMM-Module

The second module is a set of Hidden Markov Models (HMM) used for rating and ranking the word groups produced by the Finite State Grammar. A number of word groups would have produced by the first module, the right ones and possibly several additional ones as well. Rating and ranking helps us to prefer the appropriate ones to the others.

One HMM model is built for each major category of word groups. In this work we have used three HMM models, one for noun groups, one for verb groups and one for all other kinds of word groups. Note that in UCSG, prepositional groups are included under noun groups.

States in a HMM correspond to categories. Observation symbols correspond to words. The HMM models $\lambda = (\pi, A, B)$ are defined as follows:

- The number of states of the model N is the number of relevant categories.
- The number of observation symbols M is the number of words.
- The initial state probability $\pi_i = P\{q_i = i\}$ $1 \leq i \leq N$ where q_i is a category (state) starting a particular word group
- State transition probability $a_{ij} = P\{q_{t+1} = j | q_t = i\}$ $1 \leq i, j \leq N$ where q_t denotes the current category (state) and q_{t+1} denotes the next state.
- Observation or emission probability, $b_j(k) = P\{o_t = v_k | q_t = j\}$ $1 \leq j \leq N$, $1 \leq k \leq M$ where v_k denotes the k^{th} word, and q_t the current state.

The HMM parameters A , B and π can be obtained from a training corpus. In case a manually checked and certified chunked corpus is available, these parameters can be estimated from such a training corpus. However, chunked/parsed training corpora are difficult to get and we propose a bootstrapping technique to estimate the HMM parameters when only a POS-tagged corpus is available. In the latter case, we first pass the corpus through our Finite State module and obtain the possible chunks. Taking these chunks to be equi-probable, we can estimate the HMM parameters either using Baum-Welch algorithm or by simply taking the ratios of frequency counts.

In the present work, 2,500,000 randomly selected sentences from the British National Corpus [27] were chunked using our Finite State Grammar. It may be noted that this corpus is POS tagged but not parsed/chunked. The π and A matrix values were estimated from these chunks, taking all chunks as equiprobable and the B matrix values were estimated from our dictionary which includes the frequencies for every category for each word. Note that development of the HMMs is a one-time off-line process. However, as we shall see later, the HMM parameters can be further refined later by bootstrapping.

The HMMs are used only for rating and ranking the word groups already obtained by the Finite State Grammar, not for obtaining the word groups per se. We simply estimate the probability of each chunk in the HMM model for the appropriate category:

$$P(O|\lambda) = \sum_{i=1}^t \pi_{i_1} b_{i_1}(O_1) a_{i_1, i_2} b_{i_2}(O_2) a_{i_2, i_3} \cdots a_{i_{t-1}, i_t} b_{i_t}(O_t)$$

The aim here is to obtain the highest ranks for the correct chunks and to push down other chunks in the ranked order. Contrast this with the more common idea of using of HMMs (say for POS tagging) before parsing. This would require Viterbi search. Also, the HMMs we use in UCSG architecture are specific to different categories of word groups and are local to the neighbourhood of word groups, thereby imposing tighter constraints.

Chunks obtained by the Finite State Grammar can be partitioned into chunk-groups such that chunks across chunk-groups are disjoint. Ranking is performed within chunk-groups, keeping in view the category of the chunks as the example below shows. Performance can be measured in terms of position of the correct chunks in the ranked order. A sample from the running example is given in Table 1 to show the rankings obtained for the various possible word groups. It can be seen that the correct chunks tend to get ranked higher. Extensive experimentation is going on to fine tune the HMMs so that good rankings can be obtained reliably.

3.3 A* Search for Best First Search

It has proved difficult in practice to produce a single parse, or a very small number of parses, and at the same time guarantee correctness of parsing. A large number of possible parse outputs will also be difficult to use if the outputs are not rated and ranked in some way. Our aim is to build parses that can produce parse outputs in best-first order, without compromising on the ability to produce all grammatically valid parses, even when the input sentences are not POS tagged. Depending upon the application, we may either produce all possible parses in ranked order or stop further generation using a suitable thresholding mechanism. In UCSG Architecture, we posit a A* best first search algorithm to select the chunks to produce the best chunk sequence for a given sentence, taking advantage of the ratings provided by the HMM module. Searching involves selecting chunks in best first order to cover the given sentence without overlaps or gaps.

Table 1. Ranking by HMMs

<vg> <VBD><was><VVG><walking>	1
<vg> <VBD><was><VVG><walking><AVP><along>	2
<vg> <VBD><was>	3
<vgs> <VVG><walking>	4
<ng> <VVG><walking><PRP><along><AT0><a>	
<AJ0><quiet><NN1><street>	1
<ng> <VVG><walking><PRP><along><AT0><a>	
<NN1><quiet><NN1><street>	2
<ng> <PRP><along><AT0><a><AJ0><quiet><NN1><street>	3
<ng> <VVG><walking><PRP><along><AT0><a><NN1><quiet>	4
<ng> <PRP><along><AT0><a><NN1><quiet><NN1><street>	5
<ng> <AT0><a><AJ0><quiet><NN1><street>	6
<ng> <PRP><along><AT0><a><NN1><quiet>	7
<ng> <AT0><a><NN1><quiet><NN1><street>	8
<ng> <AT0><a><NN1><quiet>	9
<ng> <AJ0><quiet><NN1><street>	10
<ng> <NN1><quiet><NN1><street>	11
<ng> <NN1><street>	12
<ng> <VVG><walking>	13
<ng> <NN1><quiet>	14

The rating and ranking of the word groups by HMMs is local to the neighbourhood of the word groups, that is, within chunk-groups. Note that while sequences of words within chunks have been considered by the Finite State Parser and HMMs, the sequences of chunks themselves have not been taken into account. Therefore, simply selecting the best rated chunks within each of the chunk-groups will not necessarily form the best parse for the whole sentence.

The A* Best First Search Strategy combines two factors, namely, effort already spent in pursuing the current path (g), and, estimated effort required to reach the goal state (h). A single combined measure of goodness (f) is computed for each node in the search tree and the best node is selected for subsequent expansion: $f(n) = g(n) + h(n)$. The effort already spent is obtained from the probabilities given by the HMM module. The distance to the goal node is estimated in terms of the words yet to be covered in the given sentence. The top parse for our running example is given below:

```

<ng> [<PNPNOM><he>] </ng>
<vg> [<VBD><was><VVG><walking>] </vg>
<ng> [<PRP><along><AT0><a><AJ0><quiet><NN1><street>] </ng>
<sub> [<CJS><when>] </sub>
<ng> [<NN2><gunmen>] </ng>
<vg> [<VVD><shot>] </vg>
<ng> [<PNPACC><him>] </ng>
<ng> [<DT0><several><NN2><times><PRP><in>
  <AT0><the><NN1><head>] </ng>

```

One of the main ideas behind the UCSG architecture is bootstrapping. The final parse outputs produced after A* search will hopefully be more or less in best first order. We can therefore take the top parse, or the top few parses to be correct and re-estimate the HMM parameters using this refined data. We can also manually check the parse outputs and build a dependable partially parsed corpus. We have so far built a manually checked parsed corpus of 2000 plus sentences. Extensive experimentation is on to re-estimate the HMM parameters as also for fine tuning A* search itself.

4 Experiments and Results

The performance of the Finite State module has been evaluated on the Susanne Parsed Corpus as well as CoNLL 2000 Test data set. Since the aim of this module is only completeness, performance is given in terms of Recall.

The Susanne corpus [28] is a manually parsed corpus containing about 130,000 words in more than 6500 sentences. Some preprocessing was necessary. Ambiguities with apostrophes have been resolved. Spelling errors mentioned in the Susanne documentation have been corrected. Since the structure of the parse output in the Susanne corpus differs somewhat from that of UCSG, suitable mapping schemes had to be developed and validated [29]. Plain text sentences were extracted and given as input to the UCSG shallow parser. Results are given below in Table 2 for Noun, Verb, Adjective and Adverb groups.

Table 2. Performance of the Finite State Parser on Susanne Corpus

Word Group Type	No. of Groups in Test Data	No. of Groups Recognized	% Recall
Noun Group	34952	30642	87.67
Verb Group	18134	17975	99.12
Adjective Group	2355	1794	76.18
Adverb Group	5512	5156	93.54
Overall	60953	55567	91.16

Overall, 91.16% of phrases in the Susanne corpus have been correctly identified. 99.12% of all the verb groups could be correctly identified. Failures in the case of verb groups are limited to complex cases such as “have never, or not for a long time, had”.

The CoNLL 2000 test data set consists section 20 of the Wall Street Journal corpus (WSJ) and includes 47377 words and 23852 chunks. In the current evaluation, LST chunks (list items) have been excluded. Also, in the UCSG framework, there are no separate PPs - PPs are included in noun groups. Table 3 gives the performance.

There are a few minor differences in the way chunks are defined in the CoNLL 2000 chunking task and UCSG. Punctuation marks are removed by a pre-processor and handled separately elsewhere in UCSG. Currency symbols

Table 3. Evaluation of Finite State Parser on CoNLL 2000 Test Data Set

CoNLL Chunk Type	UCSG Terms	Chunks in Test Data	Chunks Recognized	% Recall
NP	ng	12422	10588	85.24
VP	vg,inf	4658	3786	81.28
ADVP	avg	866	698	80.60
ADJP	ajg	438	398	90.87
SBAR	sub	535	507	94.77
PRT	part	106	105	99.06
CONJP	sub	9	9	100.00
INTJ	intg	2	1	50.00
Total		19036	16092	84.53

Table 4. Evaluation of the Finite State Parser on CoNLL Data Set after mapping

CoNLL Chunk Type	Chunks in Test Data	Chunks Recognized	% Recall
NP,PP	17233	16158	93.76
VP	4658	4475	96.07

such as \$ and # are considered part of numbers in UCSG while they become separate words in CoNLL. CoNLL splits chunks across the apostrophies in genitives as in *Rockwell International Corporation's tulsa unit* while UCSG does not. To-infinitives as in *continue to plummet* are recognized separately in UCSG while they may form part of a VP in CoNLL. Also, in keeping the UCSG philosophy, PPs are not recognized separately in UCSG, they are included in noun groups. In order to get a better feel for the true performance of the UCSG shallow parser, the above differences were discounted for and performance checked again. The results are given in Table 4 for NP, PP and VPs. There is no change in the performance for other groups. Overall, 22351 out of 23847 chunks have been correctly identified, giving a Recall of 93.73%.

Initial results with ranking by HMMs has shown promise and extensive work on bootstrapping to refine the HMM parameters is currently under way. The overall system is also being evaluated in terms of the percentage of correct chunks found in the top ranked parse as also in terms of the ranking of the fully correct parse in the final output.

5 Conclusions

In this paper we have described an architecture for partial parsing called the UCSG shallow parsing architecture. UCSG combines linguistic constraints expressed in the form of finite state grammars with statistical rating using HMMs built from a POS-tagged corpus and a best first search strategy for global optimization. With appropriate bootstrapping, it would hopefully be possible to develop wide coverage and robust partial parsing systems without the need for parsed corpora which are not easily available in many cases. The UCSG Shallow Parsing Architecture is also computationally efficient. Since large scale plain text

and POS-tagged corpora are becoming available in Indian languages, this approach seems to hold promise for developing parsing systems for these languages as well.

The complete system including all the modules in the architecture has been implemented in Perl under Linux. On the Susanne parsed corpus, an overall Recall of 91.16% has been obtained and on the CoNLL 2000 chunking task test data set, a Recall of 93.73% has been obtained for the Finite State Parser. Further work is under way for refining the system through bootstrapping.

References

1. Doran, C., Egedi, D., Hockey, B.A., Srinivas, B., Zaidel, M.: XTAG system – a wide coverage grammar for english. In: Proceedings of the 15th. International Conference on Computational Linguistics (COLING 94). Volume II., Kyoto, Japan (1994) 922–928
2. Abney, S.P.: Parsing by Chunks. Principle-based parsing: Computation and psycholinguistics edn. Kluwer (1991)
3. Tjong Kim Sang, E.F., Buchholz, S.: Introduction to the conll-2000 shared task: Chunking. In Cardie, C., Daelemans, W., Nedellec, C., Tjong Kim Sang, E., eds.: Proceedings of CoNLL-2000 and LLL-2000, Lisbon, Portugal (2000) 127–132
4. Murthy, K.N.: Universal Clause Structure Grammar. PhD Thesis, University of Hyderabad (1995)
5. Murthy, K.N.: Universal Clause Structure Grammar and the Syntax of Relatively Free Word Order Languages. South Asian Language Review **VII** (1997)
6. Abney, S.: Partial parsing via finite-state cascades. In: Workshop on Robust Parsing, 8th European Summer School in Logic, Language and Information, Prag (1996) 8–15
7. Grefenstette, G.: Light parsing as finite state filtering. In: Workshop on Extended finite state models of language, Budapest, Hungary (1996)
8. Roche, E.: Parsing with finite state transducers. Finite-state language processing edn. MIT Press (1997)
9. Vilain, M., Day, D.: Phrase parsing with rule sequence processors: an application to the shared conll task. In Cardie, C., Daelemans, W., Nedellec, C., Tjong Kim Sang, E., eds.: Proc. of CoNLL-2000 and LLL-2000, Lisbon, Portugal (2000) 160–162
10. Dejean, H.: Learning rules and their exceptions. In: Journal of Machine Learning Research, volume 2. (2002) 669–693
11. Osborne, M.: Shallow parsing as part-of-speech tagging. In Cardie, C., Daelemans, W., Nedellec, C., Tjong Kim Sang, E., eds.: Proceedings of CoNLL-2000 and LLL-2000, Lisbon, Portugal (2000) 145–147
12. Veenstra, J., van den Bosch, A.: Single-classifier memory-based phrase chunking. In Cardie, C., Daelemans, W., Nedellec, C., Tjong Kim Sang, E., eds.: Proceedings of CoNLL-2000 and LLL-2000, Lisbon, Portugal (2000) 157–159
13. Zhou, G., Su, J., Tey, T.: Hybrid text chunking. In Cardie, C., Daelemans, W., Nedellec, C., Tjong Kim Sang, E., eds.: Proceedings of CoNLL-2000 and LLL-2000, Lisbon, Portugal (2000) 163–166
14. Koeling, R.: Chunking with maximum entropy models. In Cardie, C., Daelemans, W., Nedellec, C., Tjong Kim Sang, E., eds.: Proceedings of CoNLL-2000 and LLL-2000, Lisbon, Portugal (2000) 139–141

15. Johansson, C.: A context sensitive maximum likelihood approach to chunking. In Cardie, C., Daelemans, W., Nedellec, C., Tjong Kim Sang, E., eds.: Proceedings of CoNLL-2000 and LLL-2000, Lisbon, Portugal (2000) 136–138
16. Zhang, T., Damerau, F., Johnson, D.: Text chunking based on a generalization of winnow. In: Journal of Machine Learning Research, volume 2. (2002) 615–637
17. Sha, F., Pereira, F.: Shallow parsing with conditional random fields. Technical Report CIS TR MS-CIS-02-35, University of Pennsylvania (2003)
18. Molina, A., Pla, F.: Shallow parsing using specialized hmms. In: Journal of Machine Learning Research, volume 2. (2002) 595–613
19. Carreras, X., Marquez, L.: Phrase recognition by filtering and ranking with perceptrons. In: Proceedings of the International Conference on Recent Advances in Natural Language Processing, RANLP-2003, Borovets, Bulgaria (2003) 127–132
20. Gondy, L., Hsinchun, C., Jesse, M.: A shallow parser based on closed-class words to capture relations in biomedical text. In: Journal of Biomedical Informatics 36. (2003) 145–158
21. Kudoh, T., Matsumoto, Y.: Use of support vector learning for chunk identification. In Cardie, C., Daelemans, W., Nedellec, C., Tjong Kim Sang, E., eds.: Proceedings of CoNLL-2000 and LLL-2000, Lisbon, Portugal (2000) 142–144
22. van Halteren, H.: Chunking with wpdv models. In Cardie, C., Daelemans, W., Nedellec, C., Tjong Kim Sang, E., eds.: Proceedings of CoNLL-2000 and LLL-2000, Lisbon, Portugal (2000) 154–156
23. Erik F. Tjong Kim Sang: Memory-based shallow parsing. In: Journal of Machine Learning Research, volume 2. (2002) 559–594
24. Berthold Crysmann et al.: An integrated architecture for shallow and deep processing systems. In: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL), University of Pennsylvania, Philadelphia (2002)
25. Kaplan, R.M., III, J.T.M., King, T.H., Crouch, R.: Integrating finite-state technology with deep lfg grammars¹. In: Proceedings of the Workshop on Combining Shallow and Deep Processing for NLP(ESLLI). (2004)
26. J. Hopcroft and J. Ullman: Introduction to automata theory, languages, and computation. Addison-Wesley (1979)
27. Burnard, L. In: The users reference guide for the British National Corpus. Oxford University Computing Services, Oxford (1995)
28. Sampson, G.: The susanne treebank: Release 5, Univ.of Sussex, England (2000)
29. Nagesh, K.: Towards a robust shallow parser. Masters thesis, Department of Computer and Information Sciences, University of Hyderabad (2004)

Evaluating the Performance of the Survey Parser with the NIST Scheme

Alex Chengyu Fang

Department of Chinese, Translation and Linguistics,
City University of Hong Kong, Tat Chee Avenue, Kowloon, Hong Kong
acfang@cityu.edu.hk

Abstract. Different metrics have been proposed for the estimation of how good a parser-produced syntactic tree is when judged by a correct tree from the treebank. The emphasis of measurement has been on the number of correct constituents in terms of constituent labels and bracketing accuracy. This article proposes the use of the NIST scheme as a better alternative for the evaluation of parser output in terms of *correct match*, *substitution*, *deletion*, and *insertion*. It describes an experiment to measure the performance of the Survey Parser that was used to complete the syntactic annotation of the International Corpus of English. This article will finally report empirical scores for the performance of the parser and outline some future research.

1 Introduction

Different metrics have been proposed and all aim at the estimation of how good a parse tree is when judged by a correct tree from the Treebank (see [1], [2], [3], [4], and [5]). The emphasis of measurement has been on the number of correct constituents either in terms of constituent labels, such as *labelled match*, *precision*, and *recall*, or in terms of bracketing such as *bracketed match*. Together with *crossing brackets*, these measures indicate the number of correct and wrong matches in the parse tree. However, these measures outlined above do not constitute a satisfactory assessment. We may well imagine a parse tree with only two correct constituents scoring a high rate in terms of labelled and bracketed matches, crossing brackets, precision, and recall while deletions and insertions of nodes and associated labels could render the parse tree totally different from the correct one.

Consider Figures 1 and 2 for a correct tree and a parser-produced tree, both for the same hypothetical input string. The 8-node parse tree largely dissembles the correct one with five correct constituents and three wrongly inserted nodes but would nevertheless score a misleadingly high recall rate of 80% , a precision rate of 50%, and a combined performance (F-score) of 61.5%, calculated as $2 \times \text{Recall} \times \text{Precision} / (\text{Recall} + \text{Precision})$.

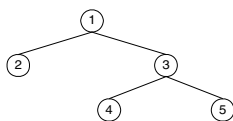


Fig. 1. A correct tree

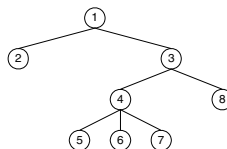


Fig. 2. A parser-produced tree

1.1 The NIST Evaluation Scheme

One obvious omission in the assessment metrics of the parse tree, as we can tell from Figures 1 and 2, comes from the fact that none of the metrics proposes to evaluate the *deletion* and *insertion* of constituents in the parse tree. For the purpose of evaluating the output from speech recognition systems, the National Institute of Science and Technology (NIST) proposed an evaluation scheme that looks at the following properties when comparing recognition results with the correct answer:

$$\text{Correct Match Rate} = \frac{\text{number of correct constituents in } T_P}{\text{number of constituents in } T_C}$$

$$\text{Substitution Rate} = \frac{\text{number of substituted constituents in } T_P}{\text{number of constituents in } T_C}$$

$$\text{Deletion Rate} = \frac{\text{number of deleted constituents in } T_P}{\text{number of constituents in } T_C}$$

$$\text{Insertion} = \text{number of inserted constituents in } T_P$$

$$\text{CombinedRate} = \frac{\text{number of correct constituents in } T_P - \text{number of insertions}}{\text{number of constituents in } T_C}$$

Notably, the correct match rate is identical to the labelled or bracketed recall rate. The insertion score, arguably, subsumes crossing brackets errors since crossing brackets errors are caused by the insertion of constituents even though not every insertion causes an instance of crossing brackets violation by definition. In this respect, the crossing brackets score only implicitly hints at the insertion problem while the insertion rate of the NIST scheme explicitly addresses this issue.

In order to achieve these scores, the NIST scheme performs text alignment between reference (gold standard) and hypothesis word strings. For the experiments to be reported in this article, dynamic programming (DP) was used to align the text strings: the DP algorithm performs a global minimization of a Levenshtein distance function which weights the cost of correct words, insertions, deletions and substitutions as 0, 3, 3 and 4 respectively. The computational complexity of DP is $O(NN)$. See [6], [7] and [8] for a detailed description.

According to this scheme, the parse tree in Figure 2 would have a combined performance rate of only 40%, taking into consideration the three insertions. The NIST metric is thus more informative than the other metrics, allowing for concrete statements as to how many correct constituents there are and, in cases of wrong constituents, how many substitutions, deletions, and insertions are observed in the parse tree. In a desirable way, its combined performance rate is adjusted according to the number of insertions.

Because of the considerations above, this article proposes the use of the NIST scheme for the evaluation of parser performance and reports an experiment to empirically evaluate the performance of the Survey Parser.

1.2 The Survey Parser

The Survey Parser was developed to complete the syntactic annotation of the British component of the International Corpus of English (ICE-GB; see [9] and [10]). Its design and architecture have been previously published. See, for example, [11], [12], and [13]. In order to conduct a precise evaluation of the performance of the parser, the experiments look at the two aspects of the parse tree: labelling accuracy and bracketing accuracy. Labelling accuracy expresses how many correctly labelled constituents there are per hundred constituents and is intended to measure how well the parser labels the constituents when compared to the correct tree. Bracketing accuracy attempts to measure the similarity of the parser tree to that of the correct one by expressing how many correctly bracketed constituents there are per hundred constituents. In this section, the NIST metric scheme will be applied to the two properties separately before an attempt is made to combine the two to assess the overall performance of the Survey Parser.

1.3 Training and Testing Data Sets

The syntactically analysed ICE-GB corpus has a total of 83,554 trees. One tenth of the total was used as test data, from which four sets of 1,000 trees were randomly created and retained as the gold standard. Word strings were then extracted from the trees to create four sets of test data and the Survey Parser was used to create automatically produced trees for the same sets.

2 Labelling Accuracy

To evaluate labelling accuracy with the NIST scheme, the method is to view the labelled constituents as a linear string with attachment bracketing removed. Consider (1), which is associated with a correct tree in Figure 3.

(1) *It was probably used in the Southern States as well.*

After removing the bracketed structure, we then obtain the following linear string of labels:

```
PU CL SU NP NPHD PRON [It] VB VP OP AUX [was] A AVP AVHD ADV [probably]
MVB V [used] A PP P PREP [in] PC NP DT DTP DTCE ART [the] NPHD N [Southern
States] A AVP AVHD ADV [as well]
```

Do the same for both the parser-produced tree and the correct answer, we will be able to characterise the similarity between the two strings in terms of correct and wrong matches, deletions, and insertions and then use the overall score to indicate how well the parser-produced tree matches the correct answer in terms of constituent labels. A program in the fashion of the NIST scoring scheme for speech recognition was modified for this purpose.

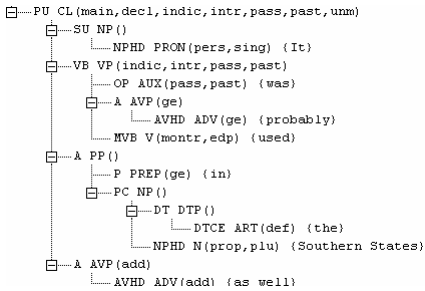


Fig. 3. A correct tree for (1)

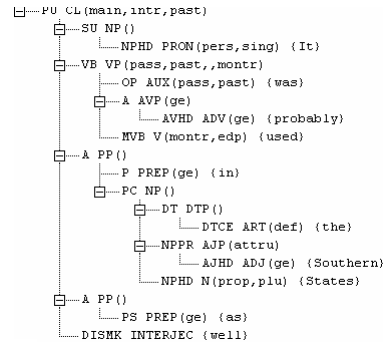


Fig. 4. A parser-produced tree for (1)

Now consider Figure 4, a parse tree automatically produced for (1) by the Survey Parser. In this analysis, *Southern States* is not analysed as a compound noun with *Southern* parsed as the head of an AJP pre-modifying *States*. A second, less desirable difference is the analysis of *as well* as a stranded preposition (PS) *as* followed by a discourse marker *well*. The scoring program therefore produces the following representation for the two trees:

Correct: pu cl su np nphd pron [it] vb vp op aux [was] a avp avhd adv [probably] mvb v [used] a pp p prep [in] pc np dt dtp dtce art [the] ****
 *** NPHD N [southern] **** * [states] a AVP AVHD ADV [as] *****
 [well]

Parser: pu cl su np nphd pron [it] vb vp op aux [was] a avp avhd adv [probably] mvb v [used] a pp p prep [in] pc np dt dtp dtce art [the] NPPR
 AJP AJHD ADJ [southern] NPHD N [states] a PP PS PREP [as] DISMK INTERJEC
 [well]

Table 1. An evaluation of the parser produced tree for [1]

Sent	Constituent	Match	Substitution	Deletion	Insertion	Accuracy
1	42	37 (88.1%)	5 (11.9%)	0 (0.0%)	6	73.8%

In this representation, correct matches are in lower case, substitutions in upper case, insertion by the parser in asterisks in the correct string, and deletion by the parser in asterisks in the parser-produced string. The evaluation program then produces statistics as summarised in Table 1. Accordingly, we may concretely claim that there are 42 constituent labels according to the correct tree, of which 37 (88.1%) are correctly labelled by the parser, with 5 substitutions (11.9%), 0 deletion, and 6 insertions. The overall labelling accuracy is then calculated as 73.8% according to the following formula:

$$\text{Overall labelling accuracy} = \frac{\text{total number of correct matches} - \text{insertion}}{\text{total number of constituent labels}} \times 100$$

2.1 Evaluating the Scoring Program

To ascertain the degree of preciseness of the scoring program when measuring labelling accuracy, errors of various types, including substitutions, deletions, and insertions, were randomly inserted in the correct trees to be compared back with the original ones without automatically inserted errors. This way, we know exactly how many errors there are and how many of them are correctly detected by the scoring program. Table 2 summarises the performance of the scoring program when tested with 4,000 trees divided into two sets of 2,000 each, where *Detected* lists what is reported by the scoring program and *Real* the actual number of errors. For Set I, as an example, the scoring program detected 7,264 substitution errors (11%) while in reality there are only 5,654 substitutions (8.6%).

Table 2. A summary of the performance of the scoring program

	<i>Set I</i>				<i>Set II</i>			
	<i>Detected</i>		<i>Real</i>		<i>Detected</i>		<i>Real</i>	
<i>Tree</i>	2000				2000			
<i>Constituent</i>	65771				61703			
<i>Correct label</i>	56689	86.2%	57847	88.0%	53720	87.1%	54670	88.6%
<i>Substitution</i>	7264	11.0%	5654	8.6%	6329	10.3%	4992	8.1%
<i>Deletion</i>	1818	2.8%	2270	3.5%	1654	2.7%	2041	3.3%
<i>Insertion</i>	2096		2548		1834		2221	
<i>Overall</i>	11178	83.0%	10472	84.1%	9817	84.1%	9254	85.0%

Table 2 shows that the scoring program tends to underestimate the correct labels by about 1.5% and the overall performance by about 1% because of some confusion over what is substitution, deletion and insertion. Take Set I as an example. The scoring program estimates that the overall labelling accuracy is 83% while the real accuracy should be 84.1%. Similarly, the error rate for Set II is overestimated by .9%. From these figures, it can be claimed that the scoring scheme presents accurate but strict estimation of the labelling accuracy of the parser because of its slight underestimate of performance.

2.2 Evaluating the Labelling Accuracy of Parser-Produced Trees

A total of 4,000 trees, divided into four sets of 1,000 each, were selected from the test data to evaluate the labelling accuracy of the parser. Empirical results show that the parser achieved an overall labelling precision of over 80%. Table 3 shows that the Survey Parser scored 86% or better in terms of the labelled recall rate (Correct match) for the four sets. About 10% of the constituent labels are wrong (Substitution) with a deletion rate of about 3.5%. Counting insertions, the overall labelling accuracy by the parser is 79.8%, scoring 86.2% and 86.7% respectively for recall and precision.

Table 3. Labelling precision scores

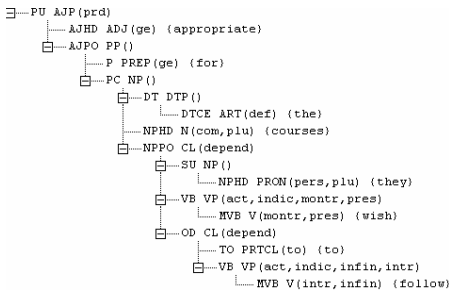
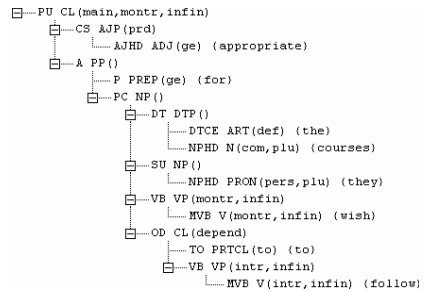
	Test I		Test II		Test III		Test IV	
<i>Tree</i>	1000		1000		1000		1000	
<i>Constituent</i>	31676		34095		31563		30140	
<i>Correct match</i>	27329	86.3%	29263	85.8%	27224	86.3%	26048	86.4%
<i>Substitution</i>	3214	10.1%	3630	10.6%	3253	10.3%	3084	10.2%
<i>Deletion</i>	1133	3.6%	1202	3.5%	1086	3.4%	1008	3.3%
<i>Insertion</i>	2021		2316		1923		1839	
<i>Overall</i>		79.9%		79.0%		80.2%		80.3%

3 Bracketing Accuracy

A second aspect of the evaluation of the Survey Parser involves the measuring of its attachment precision, an attempt to characterise the similarity of the parser-produced hierarchical structure to that of the correct parse tree. To estimate the precision of constituent attachment of a tree, the evaluation scheme needs to bring out statistics regarding the three basic node manipulation errors: substitution, deletion, and insertion. As an example to illustrate these errors, consider (2), which contains an AJP post-modified by a PP:

(2) *appropriate for the courses they wish to follow.*

Figures 5 and 6 show two parse trees for (2): the former is the correct tree structure while the latter is produced automatically by the parser with deficiencies. Ignoring the constituent labels, the pair of trees may be represented as those in Figures 7 and 8. The numbered black dots represent non-terminal nodes while the terminal nodes are represented by corresponding lexical items. For clarity of discussion, the numbers in the parser-produced tree (T_p) correspond to those in the correct tree (T_c). In T_c , the leaf node *appropriate* is attached directly to Node 1 (N_1). In T_p , however, an insertion occurs between the root and the leaf node *appropriate*, resulting in the un-numbered node. An instance of deletion is illustrated by the disappearance of N_5 in T_p . Substitution in the current evaluation scheme refers to the wrong attachment of a node.

**Fig. 5.** A correct tree for (2)**Fig. 6.** A parser-produced tree for (2)

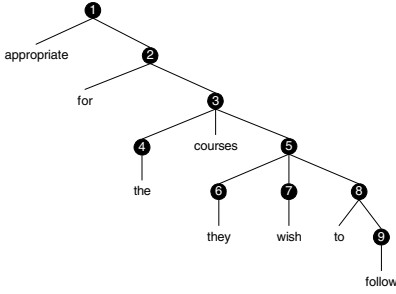


Fig. 7. T_c for (2)

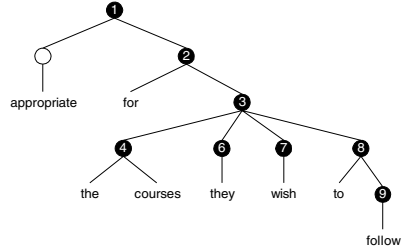


Fig. 8. T_p for (2)

T_c specifies the attachment of *courses* to N_3 while in T_p , however, this leaf node is wrongly attached to N_p , which results in an instance of substitution.

While the task of the evaluation scheme is to present an objective count of substitution, insertion, and deletion errors, it is also desirable that wrongly attached non-terminal nodes are penalised once and not for their correctly attached sister and daughter nodes. Consider T_p in Figure 7. Because of the deletion of N_5 (one instance of deletion), N_6 is counted in addition as an instance of substitution but since nodes N_{7-8} are correctly represented as its sisters, it would be unfair to count these as instances of substitutions.

3.1 A Linear Representation of the Hierarchical Structure

A linear representation of the hierarchical structure of the parse tree is designed such that the same scoring program for the assessment of labelling precision could be used to detect substitution, deletion, and insertion of nodes in parser-produced trees. This representation also ensures that wrongly attached non-terminal nodes are penalised only once if their sister and daughter nodes are correctly aligned. In this representation, the parse tree is converted into a linear string of integers indicating the branching movements:

- 1 if the current node is the daughter of the immediately previous node,
- 0 if the current node is the root or the sister of the immediately previous node,
- 1, -2... if the current node is the sister/mother/grandmother node of the immediately previous node's mother.

This linear conversion is a truthful representation of the hierarchical structure of the parse tree, which can be unambiguously recovered. The pair of trees in Figures 5 and 6, as an example, are thus represented as

T_C	0	1	*	0	1	0	1	1	-1	0	1	1	-1	1	-1	1	0	1
T_P	0	1	1	-1	1	0	2	1	-1	-1	1	*	-1	1	-1	1	0	1
	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑
	C	C	I	S	C	C	S	C	C	S	C	D	C	C	C	C	C	C

with insertion (*I*) by the parser indicated by an asterisk in the correct string and deletion (*D*) an asterisk in the parser-produced string. Substitutions (*S*) are indicated by un-matching integers. The scoring program will then produce the following statistics:

Table 4. An evaluation of the parser produced tree for [2]

Sent	Constituent	Match	Substitution	Deletion	Insertion	Accuracy
1	17	13 (76.5%)	3 (17.6%)	1 (5.9%)	1	70.6%

We may then read that there are a total of 17 constituents in the parse tree, including both terminal and non-terminal nodes, of which 13 (76.5%) are correctly attached. There are 3 substitution errors, accounting for 17.6% of the constituents, with one deletion and one insertion. The overall constituent accuracy is calculated as 70.6% according to the following formula:

$$\text{Overall constituent accuracy} = \frac{\text{total number of correct constituent} - \text{insertion}}{\text{total number of constituents}} \times 100$$

3.2 Evaluating the Scoring Program

To ascertain the degree of preciseness of the scoring program when estimating the constituent accuracy, errors of various types (substitution, deletion, and insertion) were randomly inserted in the correct trees to be compared back with the original ones without automatically inserted errors. This way, we know exactly how many errors there are and how many of them are correctly detected by the scoring program. Table 5 summarises the performance of the scoring program when tested with 4,000 trees, where *Detected* lists what is reported by the scoring program and *Real* the actual number of errors. For Set I, as an example, the scoring program detected 1,967 substitution errors (7.1%) while in reality there are 1,842 substitutions (7.0%).

Table 5 shows that the scoring program tends to slightly underestimate correct attachments and the overall constituent accuracy marginally by .3% for Set I and .4% for Set II.

Table 5. The performance of the scoring program for bracketing accuracy

	<i>Set I</i>				<i>Set II</i>			
	<i>Detected</i>		<i>Real</i>		<i>Detected</i>		<i>Real</i>	
<i>Tree</i>	2000		2000		2000		2000	
<i>Constituent</i>	27841				26269			
<i>Correct label</i>	25268	90.8%	25345	91.0%	23836	90.7%	23893	91.0%
<i>Substitution</i>	1967	7.1%	1907	6.8%	1842	7.0%	1825	6.9%
<i>Deletion</i>	606	2.2%	589	2.1%	591	2.2%	551	2.1%
<i>Insertion</i>	774		757		762		722	
<i>Overall</i>	3347	88.0%	3253	88.3%	3195	87.8%	3098	88.2%

for Set II. From these figures, it can be claimed that the scoring methodology presents accurate but strict estimation of the quality of parse trees in terms of bracketing accuracy since the method gives slight underestimates. The real performance of the parser should be interpreted as better than what the program estimates.

3.3 Evaluating the Bracketing Accuracy of Parser-Produced Trees

Table 6 shows that, considering insertions, the parser scored 76.7%. This indicates that for every 100 bracket pairs 77 are correct, with 8.5% substituted, 5.4% deleted, and the rest inserted. In other words, for a tree of 100 constituents, 23 edits are needed to conform to the correct tree structure. In conventional terms, the parser achieved 85.8% for the bracketed recall rate and 87.2% for precision.

Table 6. Bracketing accuracy scores

	<i>Test I</i>		<i>Test II</i>		<i>Test III</i>		<i>Test IV</i>	
<i>Tree</i>	1000		1000		1000		1000	
<i>Constituent</i>	12451		13390		12411		11858	
<i>Correct match</i>	10679	85.8%	11402	85.2%	10620	85.6%	10271	86.6%
<i>Substitution</i>	1088	8.7%	1249	9.3%	1115	9.0%	968	8.2%
<i>Deletion</i>	648	5.5%	739	5.5%	676	5.4%	619	5.2%
<i>Insertion</i>	1127		1297		1092		1029	
<i>Overall</i>		76.7%		75.5%		76.8%		77.9%

4 Labelling and Bracketing Accuracy Scores Combined

The combined score for both labelling and bracketing accuracy is achieved through representing both constituent labelling and unlabelled bracketing in a linear string. For example, (3) is an input string that contains an adverb phrase pre-modified by another adverb phrase.

(3) *only just*

A correct analysis is represented in Figure 9 while Figure 10 contains a tree structure produced automatically by the parser.

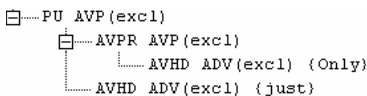


Fig. 9. A correct tree for (3)

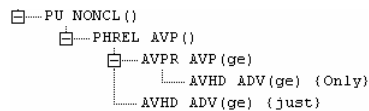


Fig. 10. A parser-produced tree for (3)

While the correct analysis for the string is an AVP with a pre-modified head, the parser analysed it as a non-clause that contains an AVP with a pre-modified head. A combined linear representation of the correct tree is as follows:

0 pu avp 1 avpr avp 1 avhd adv [only] -1 avhd adv [just]

In this representation, the integers represent the bracketing of the labelled constituents. Likewise, the parser-produced tree is represented as

0 pu NONCL 1 PHREL avp 1 avpr avp 1 avhd adv [only] -1 avhd adv [just]

The scoring program will then compare this pair of linear strings, which is to produce the following alignment:

C: 0 pu ***** * ***** avp 1 avpr avp 1 avhd adv [only] -1 avhd adv [just]
P: 0 pu NONCL 1 PHREL avp 1 avpr avp 1 avhd adv [only] -1 avhd adv [just]

For the above example, the scoring program correctly identifies the three insertions in the parse tree: two inserted constituent labels (NONCL and PHREL) and the inserted node. The overall combined performance by the parser is therefore 78.6%:

Table 7. An evaluation of the parser produced tree for [3]

Sent	Constituent	Match	Substitution	Deletion	Insertion	Accuracy
1	14	14 (100.0%)	0 (0.0%)	0 (0.0%)	3	78.6%

4.1 Empirical Scores by the Survey Parser

Labelling and bracketing precision scores are combined and summarised in Table 8, which gives the total number of trees in the four test sets and the total number of constituents. The number of correct matches, substitutions, insertions and deletions are indicated and combined scores computed accordingly. It is shown that the parser achieved an overall performance of 78.9%. Considering that the score program tends to underestimate the success rate, it is reasonable to assume a real overall combined performance of 80%. In terms of recall and precision, the parser would achieve 86.0% and 86.9% respectively.

Table 8. Combined scores for labelling precision and constituent accuracy

	<i>Test I</i>		<i>Test II</i>		<i>Test III</i>		<i>Test IV</i>	
<i>Tree</i>	1000		1000		1000		1000	
<i>Constituent</i>	44127		47485		43974		41998	
<i>Correct match</i>	38008	86.1%	40665	85.6%	37844	86.1%	36319	86.5%
<i>Substitution</i>	4302	9.7%	4879	10.3%	4368	9.9%	4052	9.6%
<i>Deletion</i>	1781	4.0%	1941	4.1%	1762	4.0%	1627	3.9%
<i>Insertion</i>	3148		3613		3015		2868	
<i>Overall</i>	79.0%		78.0%		79.2%		79.6%	

5 Discussion and Conclusion

The NIST scheme was used to evaluate the performance of the grammar when applied in the Survey Parser. An especially advantageous feature of the metric is the

calculation of an overall parser performance rate that takes into account the total number of insertions in the parse tree, an important structural distortion factor when calculating the similarity between two trees. A total of 4,000 trees were used to evaluate the labelling and bracketing accuracies. It is shown that labelling accuracy is 79.8% according to the NIST scheme while the recall and precision measures are more relaxed, giving 86.2% and 86.7% respectively. The bracketing accuracy is 76.7%, 3% lower than labelling, but still scores 85.8% and 87.2% in terms of recall and precision. The combined accuracy for labelling and bracketing is 78.9%, or 86.1% in terms of recall and 86.9% in terms of precision.

It is difficult to draw straightforward comparisons with other systems. [14] reports a maximum entropy inspired parser that scored 90.1% average precision/recall when trained and tested with sentences from the Wall Street Journal corpus (WSJ). While the difference in precision/recall between the two parsers may indicate the difference in terms of performance between the two parsing approaches, there nevertheless remain two issues to be investigated. Firstly, there is the issue of how text types may influence the performance of the grammar and indeed the parsing system as a whole. [14] uses WSJ as both training and testing data and it is reasonable to expect a fairly good overlap in terms of lexical co-occurrences and linguistic structures and hence good performance scores. Indeed, [15] suggests that the standard WSJ task seems to be simplified by its homogenous style. It is thus yet to be verified how well the same system will perform when trained and tested on a more ‘balanced’ corpus such as ICE. Secondly, it is not clear what the performance will be for Charniak’s parsing model when dealing with a much more complex grammar such as ICE, which has almost three times as many non-terminal parsing symbols. The performance of the Survey Parser is very close to that of an unlexicalised PCFG parser reported in [16] but again WSJ was used for training and testing and it is not clear how well their system will scale up to a typologically more varied corpus.

These results show both encouraging and promising performance by the grammar in terms of coverage and accuracy and therefore argue strongly for the case of inducing formal grammars from linguistically annotated corpora. The experiments also demonstrated the advantages of using the NIST scheme for both labelling and bracketing accuracies measured not only in terms of precision and recall, but also in terms of undesirable deletions and insertions of structural constituents. It is shown that syntactic trees may be seen as strings while the number of edits indicates the number of operations needed to obtain the correct tree structure, thus offering a better estimation of structural distortion or accuracy than the conventional recall, precision and f-score indices.

While many other evaluation approaches exist, such as [17], a good evaluation scheme should not only consider how well the parser-produced tree compares to the gold standard but also the size of the grammar and therefore its information content in order to evaluate parser performance across different parsing schemes. A useful development within the area of parser evaluation is to verify the impact of grammar size (in terms of vocabulary) on the performance of the parsing system. It is also important to use a typologically more balanced corpus than WSJ as a workbench for grammar-parser development as well as evaluation.

References

1. Sutcliffe, R.F.E., H.-D. Koch, and A. McElligott, eds. 1996. *Industrial Parsing of Software Manuals*. Amsterdam: Rodopi.
2. Carroll, J.A., E.J. Briscoe, and A. Sanfilippo. 1998. Parser evaluation: a survey and a new proposal. In *Proceedings of the First International Conference on Language Resources and Evaluation, Granada, Spain, 28-30 May 1998*. pp. 447-54.
3. Gaizauskas, R., M. Hepple, and C. Huyck. 1998. A scheme for comparative evaluation of diverse parsing systems. In *Proceedings of the First International Conference on Language Resources and Evaluation, Granada, Spain, 28-30 May 1998*.
4. Lin, D. 1998. A dependency-based method for evaluating broad-coverage parsers. In *Natural Language Engineering 4*. pp 97-114.
5. Srinivasan, B., A. Sarkar, Christine Doran, and Beth Ann Hockey. 1998. Grammar and parser evaluation in the XTAG project. In Carroll, Basili, et al. (eds). In *Proceedings of the Workshop on the Evaluation of Parsing Systems, First International Conference on Language Resources and Evaluation, Granada, Spain, 26 May 1998*.
6. Pallett, D.S., W.M. Fisher, and J.G. Fiscus. 1990. Tools for the Analysis of Benchmark Speech Recognition Tests. In *Proceedings of ICASSP 90*. pp. 97-100.
7. Pallett, D.S., J.S. Garofolo, and J.G. Fiscus. 2000. Measurements in Support of Research Accomplishments. In *Communications of the ACM, Volume 43, Issue 2*. pp 75-79.
8. ftp://jaguar.ncsl.nist.gov/current_docs/sctk/doc/sclite.htm
9. Greenbaum, S. 1992. A new corpus of English: ICE. In *Directions in Corpus Linguistics: Proceedings of Nobel Symposium 82, Stockholm 4-8 August 1991*, ed. by Jan Svartvik. Berlin: Mouton de Gruyter. pp 171-179.
10. Greenbaum, S. 1996. *The International Corpus of English*. Oxford: Oxford University Press.
11. Fang, A.C. 1996. The Survey Parser: Design and development. In S. Greenbaum (ed). pp 142-160.
12. Fang, A.C. 2000. From Cases to Rules and Vice Versa: Robust Practical Parsing with Analogy. In *Proceedings of the Sixth International Workshop on Parsing Technologies, 23-25 February 2000, Trento, Italy*. pp 77-88.
13. Fang, A.C. 2005. The Syntactically Annotated ICE Corpus and the Automatic Induction of a Formal Grammar. In *Proceedings of 6th International Workshop on Linguistically Interpreted Corpora*, Jeju Island, Korea, 15 October 2005.
14. Charniak, E. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st Annual Meeting of the North American Chapter of the ACL, Seattle, Washington*.
15. Gildea, D. 2001. Corpus variation and parser performance. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing, 2001*.
16. Klein, D. and C. Manning, 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the ACL, July 2003*. pp 423-430.
17. Vilares, M., F.J. Ribadas and J. Vilares. 2004. Phrase Similarity through the Edit Distance. In F. Galindo, M. Takizawa and R. Traummüller (eds.), *Database and Expert Systems Applications, Volume 3180, Lecture Notes in Computer Science*, Springer-Verlag, Berlin-Heidelberg-New York, 2004.

Sequences of Part of Speech Tags vs. Sequences of Phrase Labels: How Do They Help in Parsing?

Gabriel Infante-Lopez¹ and Maarten de Rijke²

¹ FaMAF, Universidad Nacional de Córdoba, Córdoba, Argentina
gabriel@famaf.unc.edu.ar

² Informatics Institute, University of Amsterdam, The Netherlands
mdr@science.uva.nl

Abstract. We compare the contributions made by sequences of part of speech tags and sequences of phrase labels for the task of grammatical relation finding. Both are used for grammar induction, and we show that English labels of grammatical relations follow a very strict sequential order, but not as strict as POS tags, resulting in better performance of the latter on the relation finding task.

1 Introduction

Some approaches to parsing can be viewed as a simple context free parser with the special feature that the context free rules of the grammar used by the parser do not exist a priori [1,2,3]. Instead, there is a device for generating bodies of context free rules on demand. Collins [1] and Eisner [2] use Markov chains as the generative device, while Infante-Lopez and De Rijke [3] use the more general class of probabilistic automata. These devices are induced from sample instances obtained from tree-banks. The learning strategy consists of coping all bodies of rules inside the Penn Tree-bank (PTB) to a bodies of rules sample bag which is then treated as the sample bag of an *unknown* regular language. This unknown regular language is to be induced from the sample bag, which is, later on, used for generating new bodies of rules.

Usually, the induced regular language is described by means of a probabilistic automata. The quality of the resulting automata depends on many things; the alphabet of the target regular language being one. At least two such alphabets have been considered in the literature: Part of Speech (POS) tags and grammatical relations (GRs), where the latter are labels describing the relation between the main verb and its dependents; they can be viewed as a kind of non-terminal labels. Using one or the other alphabets for grammar induction might produce different results on the overall parsing task. Which of the two produces “better” automata, that produce “better rules,” which in turn lead to “better” parsing scores? This is our main research question in this paper.

Let us provide some further motivation and explanations. In order to obtain phrase structures like the ones retrieved in [4], the dependents of a POS tag should consist

of pairs of POS tags and non-terminal labels instead of sequences of POS tags alone. Like sequences of POS tags, sequences of pairs of POS tags and non-terminal labels can be viewed as instances of a regular language: one whose alphabet is the product of the set of possible POS tags and the set of possible non-terminal labels. Moreover, they can be viewed as instances of the combination of two regular languages: one modeling sequences of POS tags, and another modeling sequences of non-terminal labels. Infante-Lopez and De Rijke [3] only use the first regular language for grammar induction, while non-lexicalized approaches [5] use the second regular language, and Markovian rules [4] use a combination of the two. Combining the regular language of POS tags and that of non-terminal labels boosts the overall parsing performance, cf., [4,5], but it is not clear why this is the case. Infante-Lopez and De Rijke [3] suggest that lexicalization improves the quality of the automata modeling sequences of POS tags, but they do not provide any insight about the differences or the interplay between these two regular languages.

We design and implement experiments for exploring the differences between the regular language of POS tags and the regular language of non-terminal labels in a parsing setup. Our research aims at quantifying the difference between the two and at understanding their contribution to parsing performance. To address our research question we focus on a task that clearly isolates these two regular languages: detecting and labeling dependents of the main verb of a sentence. We present two approaches to dealing with this task. In the first, we develop two grammars: one for detecting dependents and another for labeling them. The first grammar uses sequences of POS tags as the main feature for detecting dependents, and the second grammar uses sequences of GRs as the main feature for labeling the dependents found by the first grammar. The overall task of detecting and labeling dependents is performed by cascading the two grammars. In the second approach, we build a single grammar that uses sequences of GRs as the main feature for detecting and labeling dependents. The overall task is done in one go by this grammar. The two approaches differ in that the first uses sequences of GRs and sequences of POS tags, while the second only uses sequences of GRs.

English GRs are shown to follow a strict sequential order, but not as strict as POS tags of verbal dependents. Counterintuitively, the latter are more effective for detecting and labeling dependents, and, hence, provide a more reliable instrument for detecting GRs. This feature is responsible for boosting parsing performance.

In Section 2 we detail the task on which we focus; Section 3 builds the grammars used in the experiments. Section 4 argues for the appropriateness of the task on which we focus for our main research questions. Section 5 describes our experiments and answers these questions. We present related work in Section 6 and conclude in Section 7.

2 Task Definition

The task we use for our experiments is to find dependents of main verbs and to determine their GR. Given a sentence, the input for the task consists of the following: (1) the

main verb of the sentence, (2) the head word for each of the chunks into which the sentence has been split, and (3) POS tags for the heads of the chunks. The rest of the information in the sentence is discarded. The information below the line in Figure 1 shows an example of the input data.

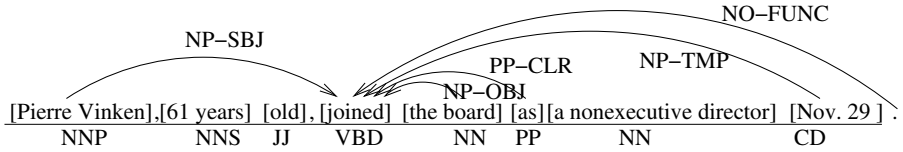


Fig. 1. Information we use from each tree in the PTB

The output consists of a *yes/no* tag for each element in the input string. A POS tag marked *yes* implies that the tag depends on the main verb. If a POS tag is marked *yes*, the output has to specify the GR between the POS tags and the main verb. An example of the desired output is shown in Figure 1. Tags labeled *yes* have been replaced by links between the POS tags and the main verb. *Not* all POS tags in our example sentence bare a relation to the main verb. More generally, there may be POS tags that depend on the main verb but whose relation cannot be labeled by any of the labels we define below. These links receive the *NO-FUNC* label. It is important to distinguish between the POS tags that do not have a relation to the main verb and those that depend syntactically on the main verb but whose relation cannot be labeled. The former are marked with the *no* tag, while the latter are marked with the *yes* tag and the GR is *NO-FUNC*; Figure 1 has an example.

In order to define the regular language of GRs, we codify GRs in pre-terminal symbols. As an example, Figure 2 shows the verb dependents from Figure 1, *nnp* *nn* *pp*, and *cd*, with labels as pictured, while *nns* *jj*, and *nn* are not in any relation to the main verb and, consequently, they are not linked or labeled and not shown in Figure 2. One can clearly distinguish the two regular languages that can be used for detecting

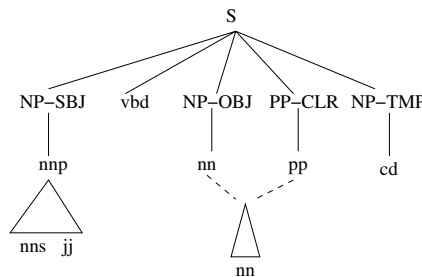


Fig. 2. The desired tree for the input in Figure 1. The subtree pictured as a triangle denotes that it can be adjoined to both points.

dependents of verbs: the sequences NP-SBJ and NP-OBJ PP-CLR NP-TMP are instances of the regular languages whose alphabet is the set of possible GRs, while the sequences nnp and nn pp cd are instances of the regular language whose alphabet is the set of possible POS tags.

3 Building Grammars

We build 3 grammars; each is a PCW-grammars (see Section 3.1 for details): G_D , G_L , and G . The grammar G_D aims to *detect* main verb dependents. It uses automata that model sequences of POS tags. The parser using G_D is fed with all POS tags. For each sentence parsed with this grammar, the parser outputs a dependency structure in which the main verb dependents are found. The grammar G_L aims to *label* dependents. It uses automata that model sequences of GRs. The parser using G_L is fed with the POS tags that are believed to depend on the main verb. The result is a GR name for each POS tag in the input sentence. This grammar assumes that (somehow) the right dependents have been identified, and its task is to assign the correct label to the dependents; it assigns a label to all elements in the the input string. The grammar G aims to *detect* and *label* main dependents. It uses automata that model sequences of GRs together with automata that model sequences of POS tags. The input and output of parsing with G are as for the grammar G_D .

Using G_D , G_L , and G we define two ways to address the relation finding task described in Section 2: (1) We use G_D for detecting dependents, and G_L for labeling the dependents that G_D outputs. (2) We use G for detecting and labeling the main dependents.

The three grammar are PCW-grammars (see Section 3.1). We build them following the same procedure: (1) we build a bodies of rules training set extracted from the PTB (see Section 3.2), (2) we induce an automaton from the training material (see Section 3.4), and, (3) we build a grammar using the automata induced in step 2 (see Section 3.3).

3.1 Grammatical Framework

We need a grammatical framework that models rule bodies as instances of a regular language and that allows us to transform automata to grammars as directly as possible. We use the grammatical framework of CW-grammars [6]. Based on PCFGs, they have a clear and well-understood mathematical background and we do not need to resort to ad-hoc parsing algorithms.

A *probabilistic constrained W-grammar* (PCW-grammar) is a two-level grammar consisting of two sets of PCF-like rules (*pseudo-rules* and *meta-rules*) and three pairwise disjoint sets of symbols (*variables*, *non-terminals* and *terminals*). Pseudo-rules and meta-rules provide mechanisms for building ‘real’ rewrite rules, which are built by

Table 1. Example of a PCW-grammar

meta-rules	pseudo-rules
$\overline{Adj} \xrightarrow{m}_{0.5} \overline{Adj} \overline{Adj}$	$S \xrightarrow{s}_{1} \overline{Adj} \overline{Noun}$
$\overline{Adj} \xrightarrow{m}_{0.5} \overline{Adj}$	$Adj \xrightarrow{s}_{0.1} big$
	$Noun \xrightarrow{s}_{1} ball$
	\vdots
	\vdots

first selecting a pseudo-rule, and then using meta-rules for instantiating the variables in the body of the pseudo-rule.

Parsing PCW-grammars requires two steps: a generation-rule step followed by a tree-building step. Parsing with PCW-grammars can be viewed as parsing with PCF grammars. The main difference is that in PCW-parsing derivations for variables remain hidden in the final tree [6].

3.2 Training Material

The training material we use for building G_D , G_L and G always comes from sections 11–19 of the PTB. We use `chunklink.pl` [7] for transforming the PTB to labeled dependency structures and for marking all the information we use. Briefly, [7] defines a *chunk* to consist of a head, i.e., any word that has a labeled pointer, plus the continuous sequence of all words around it that have an unlabeled pointer to this head. This chunk correspond to the projection of the pre-terminal level in the original tree. *Labels* are defined as concatenation of the non-terminals labels found in the PTB.

Clearly, `chunklink.pl` does *not* define an invertible procedure, i.e., its output dependency trees can *not* be mapped back to the original phrase structure tree, as labels of some intermediate constituents are deleted during pruning [7, p. 60]; some information regarding the original attachment position of grammatical functions is also lost. Despite this, `chunklink.pl` does not appear to discard too much information; the structures it produces are meaningful. All our experiments use the same type of information and the transformation performed with `chunklink.pl` does not favor one experiment over another.

After the transformation, the resulting trees contain information about chunks and labels (see Figure 1). From such trees, two further trees can be extracted, each contain-

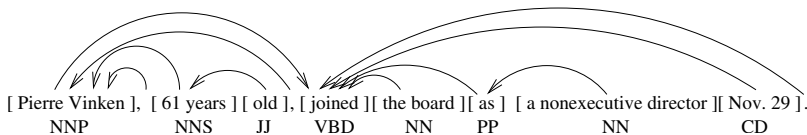


Fig. 3. A dependency tree from which we extracted training material

ing information relevant to the 3 grammars we want to build. For the tree in Figure 1, the trees in Figures 3 and Figure 4 can be obtained. We use these derived trees for obtaining the training material. The precise tree to be used depends on the grammar we want to induce, as we will now explain.

For the grammar for detecting dependants G_D , the dependency trees used are like the one shown in Figure 3, and Table 2 shows the sample sets of right and left dependents we extracted from it.

In contrast, for the grammar G_L we use trees like the one pictured in Figure 4. From such trees, we extract two kinds of information. The first kind is used to model meta-rules yielding GRs, i.e., the first level of the output trees, while the second is used to model pseudo-rules that rewrite names of GRs into POS tags, i.e., the third level of the output tree. Table 3 shows all instances to be added to the training material extracted from the tree in Figure 1.

Probabilities of pseudo-rules in G_D were hand coded, because there is a one to one correspondence with left-hand symbols and the body of rules. For the present grammar, this is no longer the case. Here, left hand symbols of pseudo-rules are GRs, and these

Table 2. Instances of left and right dependents extracted from the tree in Figure 3. The head always starts the string of dependants. Left dependants should be read backwards.

POS	Left	Right
NNP	NNP	NNP COMMA NNS COMMA
COMMA	COMMA	COMMA
NNS	NNS	NNS JJ
JJ	JJ	JJ
COMMA	COMMA	COMMA
VBD	VBD NNP	VBD NN PP CD DOT
NN	NN	NN
PP	PP	PP NN
NN	NN	NN
CD	CD	CD
DOT	DOT	DOT

Table 3. Data extracted from the tree in Fig. 1. Left dependents should be read from right to left.

VBD	
Left	Right
NP-SBJ VBD	VBD NP-OBJ PP-CLR NP-TMP NO-FUNC

Table 4. Pairs of GRs and POS tags extracted from tree in Figure 1

GR	NP-SBJ	NP-OBJ	PP-CLR	NP-TMP	NO-FUNC
POS tag	nnp	nn	pp	cd	dot

names can yield different POS tags. To estimate probabilities, we extracted all pairs of (GR, POS) from the training material and put them aside in only one bag. Table 4 shows the instances of pairs extracted from the tree in Figure 1. The training material used for building G is the union of the training material for G_L and G_D .

3.3 Defining Grammars

We start by building G_D . Once the training material has been extracted, we build two automata per POS tag, one modeling left dependents, the other right dependents. Let POS be the set of possible POS tags, w an element in POS , and A_L^w and A_R^w the two automata associated to it. Let G_L^w and G_R^w be the PCFGs equivalent to A_L^w and A_R^w , respectively, following [8], and let S_L^w and S_R^w be the start symbols of G_L^w and G_R^w , respectively. We build a grammar G_D with start symbol S , by defining its meta-rules as the disjoint union of all rules in G_L^w and G_R^w (for all POS w), its set of pseudo-rules as the union of the sets $\{S \xrightarrow{s} S_L^v v^* S_R^v : v \in \{VB, VBD, VBG, VBN, VBP, VBZ\}\}$. The grammar is designed in such a way that the start symbol S only yields the head words of the sentences which are marked with the $*$ symbol. That is, all sentences that are parsed using these grammars have one word marked with the $*$ symbol indicating that the marked word is the head of the sentence.

For G_L , automata are used to model sequences of GRs instead of POS tags. GRs are at depth one (see Figure 4) and they are modeled with automata and meta-rules. The yield of the tree is at depth two and it is modeled using pseudo-rules. The latter rewrite GR names into a POS tag and they are read from the tree-bank; their probabilities are computed using maximum likelihood estimation [9]. All meta-derivations that took place to produce nodes at depth 1 remain hidden. Hence, the sequence of GRs to the right and to the left of the main verb are instances of the regular languages modeling right or left GRs, respectively.

Once the training material for meta-rules has been extracted, we build two automata per GR, one modeling left sequences of GRs, the other right sequences of GRs. Let VS be the set of possible verb tags, v an element in VS , and A_L^v and A_R^v the two automata associated with it. Let G_L^v and G_R^v be the PCFGs equivalent to A_L^v and A_R^v , respectively, and let S_L^v and S_R^v be the start symbols of G_L^v and G_R^v , respectively. We build a grammar G_L with start symbol S , by defining its meta-rules as the disjoint union of all rules in G_L^v and G_R^v (for all verb POS tags v), and its set of pseudo-rules as the union of the two sets. One set, given by $\{S \xrightarrow{s} S_L^v v^* S_R^v : v \in VS\}$, is connects automata modeling

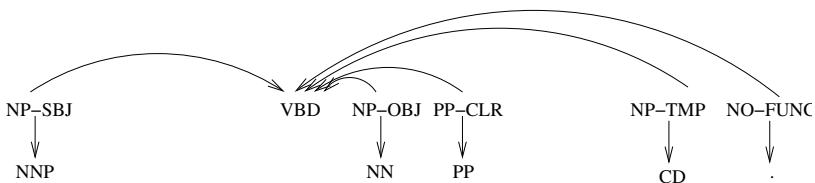


Fig. 4. The tree representation we use, extracted from tree in Figure 1

left sequences of GRs with automata modeling right sequences of GRs. The second set, given by $\{GR \xrightarrow{s}_p w : w \in POS\}$, where GR is the name of a GR, w is a POS tag, and p the probability associated to the rule, is computed using (GR, POS) pairs extracted from the training material, using maximum likelihood estimation.

The automata we use for building G are the same as those used in the previous two grammars, but the set of rules differs. Let POS be the set of possible POS tags, let w be an element in POS ; let A_L^w and A_R^w be the two automata built for each POS tag for the grammar G_D . Let VS be the set of possible verb tags, v an element in VS ; let A_L^v and A_R^v the two automata we built for verb tags for grammar G_L . Let G_L^v , G_R^v , G_L^w , and G_R^w be the PCFGs equivalent to A_L^v , A_R^v , A_L^w and A_R^w , respectively, and let S_L^v , S_R^v , S_L^w and S_R^w be the start symbols of G_L^v and G_R^v , respectively. We build a grammar G with start symbol S , by defining its meta-rules as the disjoint union of all rules in G_L^v , G_R^v , G_L^w and G_R^w , for all POS tags and all verbs tags, while its set of pseudo-rules is the union of the following sets: $\{S \xrightarrow{s}_1 S_L^v v^* S_R^v : v \in VS\}$, $\{W \xrightarrow{s}_1 S_L^w w S_R^w : w \in POS\}$, and $\{GR \xrightarrow{s}_p S_L^w w S_R^w : w \in POS\}$, where p is the probability assigned to the rule $\{GR \xrightarrow{s}_p w : w \in POS\}$ using maximum likelihood estimation.

3.4 Optimizing Automata

Let T be a bag of training material extracted from the transformed tree-bank. The nature of T depends on the grammar we are trying to induce. Since we use the same technique for optimizing all automata, we describe the procedure for a general bag. We use *minimum discrimination information* (MDI) [10] algorithm for inducing the automata, and two different measure for evaluating them: perplexity (PP) and missed samples (MS). A PP value close to 1 indicates that the automaton is almost certain about the next step while reading the string. MS counts the number of strings in the test sample Q that the automaton failed to accept.

The MDI algorithm has one parameter: alpha. We search for the value of alpha that minimizes $q = \sqrt{PP^2 + MS^2}$ (see [6] for motivation), where both PP and MS depend on α . In Figure 5 we have plotted alpha vs. PP, MS and q for the VB tag used

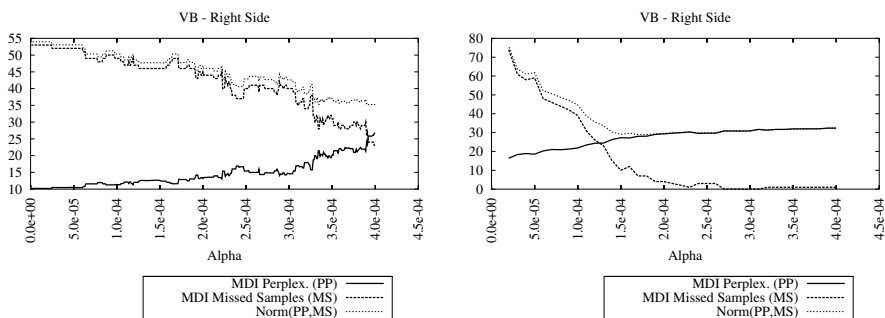


Fig. 5. Left and right plots for automata used in G_L and G_D

in the grammars G_L (left) and G_D (right). Even though the PP values for automata modeling sequences of GRs (left) and the PP values for automata modeling POS tags (right) are close to each other, the difference between their MSs is remarkable. Data sparseness seems to affect the modeling of GRs much more than that of POS tags; it prevents the MDI algorithm from inducing a proper language for GRs.

4 Comparing Probability Distributions

The approach we follow to detect the value of sequences as features is to address the task of detecting and labeling arguments using two different strategies. One is to cascade the grammars G_L and G_D , while the second is to use G in one go. The first approach uses the sequence of POS as a feature while the second one does not. Let us take a closer look. We present the probabilities that each grammar assigns to its tree language. Consider the trees shown in Figure 6. G_D , G_L , and G generate the

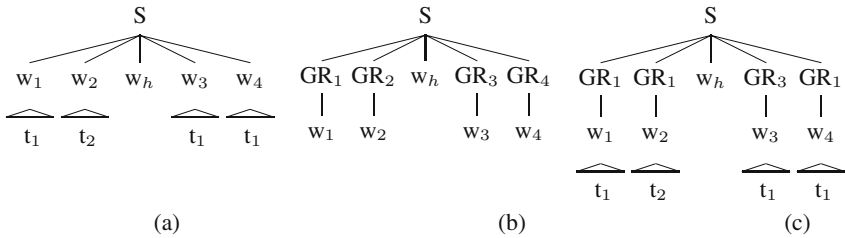


Fig. 6. (a) Example of a structure retrieved by the grammar G_D , (b) An example of a structure retrieved by the grammar G_L , and (c) The result of cascading the grammars for detecting and labeling dependents.

trees in Figure 6 (a), (b) and (c), respectively. The three grammars assigns probabilities $p_{G_D}(t|s)$, $p_L(t|w_1 \dots w_4)$, and $p_G(t|s)$ as defined in Figure 7. There, $p(w_h w_1 w_2)$, $p(w_h w_3 w_4)$, $p(w_h GR_1 GR_2)$ and $p(w_h GR_3 GR_4)$ are the probabilities assigned by the automata to the strings $w_h w_1 w_2$, $w_h w_3 w_4$, $w_h GR_1 GR_2$, and $w_h GR_3 GR_4$, respectively, and similarly for $w_h GR_1 GR_2$ and $w_h GR_3 GR_4$). Further, $p(GR_i \xrightarrow{s} w_i)$ refers to the probability assigned to the rule $GR_i \xrightarrow{s} w_i$ and s is the concatenation of $yield(t_1)yield(t_2)w_h yield(t_3)yield(t_4)$.

If the grammar for labeling dependents is fed with the dependents found by the grammar for detecting dependents, the probability associated to a tree like the one pictured in Figure 6.(c) is as follows

$$\begin{aligned}
 p_{cascading}(t|s) &= p_D(t) \times p_L(t) = & (1) \\
 &= p(GR_1 \dots GR_4) \times \\
 &\quad p(GR_1 \xrightarrow{s} w_1) \dots p(GR_4 \xrightarrow{s} w_4) \times \\
 &\quad p(w_h w_1 w_2)p(w_h w_3 w_4)p(t_1) \dots p(t_4)
 \end{aligned}$$

$$\begin{aligned}
p_{G_D}(t|s) &= p(w_h w_1 w_2) p(w_h w_3 w_4) p(t_1) \dots p(t_4) \\
p_{G_L}(t|w_1 \dots w_4) &= p(w_h GR_1 GR_2) p(w_h GR_3 GR_4) p(GR_1 \rightarrow w_1) \dots p(GR_4 \rightarrow w_4) \\
p_G(t|s)(t) &= p(w_h GR_1 GR_2) p(w_h GR_3 GR_4) p(GR_1 \rightarrow w_1) \dots p(GR_4 \rightarrow w_4) \times \\
&\quad \times p(t_1) \dots p(t_4) \\
p_G(t|s) &= p_{one-go}(t)
\end{aligned}$$

Fig. 7. Probabilities $p_{G_D}(t|s)$, $p_{G_L}(t|w_1 \dots w_4)$, and $p_G(t|s)$ assigned by G_D , G_L and G , respectively

We can now establish the relation between the two probabilities behind the two strategies we defined for solving the task. Let $p_{cascading}$ be the probability distribution generated over trees by cascading the two first grammars, and p_{one-go} the probability distribution generated by G . Both p_{one-go} and $p_{cascading}$ assign probabilities to the same set of trees, and the two are related as follows:

$$p_{cascading}(t) = p_{one-go}(t) \times p(w_h w_1 w_2) p(w_h w_3 w_4). \quad (2)$$

The difference between the two distributions is the probability of the sequence of POS tags $w_1 \dots w_4$.

Summing up, we have two probability distributions for the very same task, one uses an additional feature, namely, the sequence $w_1 \dots w_4$. An empirical comparison of these two distributions will provide us with information about the value of the additional feature; this is what we turn to in next.

5 Experiments

For our experiments we shuffle the PTB sections 10 to 19 into 10 different sets. We run the experiments using set 1 as the test set and sets 2 to 10 as training sets. The tuning samples were extracted from Section 00. All sentences fed to the parser have the main head marked; all sentences whose main head was not tagged as a verb are filtered out. First, we perform the whole task (detecting dependents and labeling their relation with the main verb) according to the two strategies; results are shown in Table 5; we observe a 10% difference in $f_{\beta=1}$ between the cascaded strategy and the “direct” strategy. This helps us answer our main research question (What is the importance of the sequences of POS tags for parsing?). Recall from Equation 2 that the only difference between p_{one-go} and $p_{cascading}$ is that $p_{cascading}$ associates to sequences of POS tags. In other words, the 10% difference in performance between the two strategies is due to the use of this information.

The grammar G_L for labeling dependents allows us to quantify the effectiveness of sequences of GRs together with pseudo-rules $GR \xrightarrow{s} w$ for labeling GRs. To this end, we used grammar the G_L for labeling dependents that are known to be the right dependents. We extracted the correct sequences of dependents from the gold standard and used the grammar G_L for labeling them. Table 6 shows the results of this experiment; the results show that labeling is not a trivial task. The scores obtained are low, especially

Table 5. The results on detecting and labeling main verbs dependents

Approach	Precision	Recall	$f_{\beta=1}$
Cascading	0.73	0.73	0.73
One Go	0.65	0.67	0.66

Table 6. Results of the experiment on labeling gold standard dependents and detecting dependents

Approach	Precision	Recall	$f_{\beta=1}$
Labeling Gold Standard	0.76	0.76	0.76
Detecting Dependents	0.85	0.88	0.86

if we take into account that the sentences fed to the parser consisted only of correct dependents. The poor performance of this grammar is due to the data sparseness problem mentioned above: there is a large number of MS in the automata that model GRs. Moreover, the two grammars in the cascaded approach allow us to quantify how errors percolate from detecting dependents to labeling them. Now, the only aspect of the task that is left to study is the detection of dependents. In Table 6 we see how sensitive the task of labeling dependents is to errors in its input: the labeling precision drops from 0.76 to 0.73 when only the 85% of the arguments fed to the labeling grammar are correct.

6 Related Work

The task of finding GRs has mostly been considered as a classification task [7]. A classifier is trained to find relations and to decide the label of the relations found. The training material consists of sequences of 3-tuples (main verb, label, and context). In contrast to approaches based on classifiers, we view the task of finding GRs as a parsing task. We build grammars that specifically try to find GRs. In order to give an impression of state-of-the-art methods for finding and labeling main dependents, we compare experiments to the approach presented in [7]. She reports 0.86 and 0.80 for precision and recall respectively. These scores are better than ours, and the differences are probably due to the restricted amount of information we used for performing the task. In contrast, Buchholz [7] uses all kinds of features for detecting and labeling dependents.

7 Conclusions

The standard practice in parsing is to use all features that improve parsing performance without clearly stating why they improve. In contrast, we designed grammars and experiments for isolating and explaining two particular types of features: sequences of POS tags and sequences of GRs, both for detecting and labeling and labeling dependents.

We designed and implemented experiments for exploring the differences in contribution to the overall task of parsing between the regular language of POS tags and the regular language of GRs. To assess the contribution of these two features, we

carried out an evaluation in terms of a task that clearly isolates the two regular languages. We used the task of detecting and labeling dependents of the main verb of a sentence. We presented two approaches for addressing this task. For the first, we developed two grammars: one for detecting dependents and another for labeling them. The first grammar used sequences of POS tags as the main feature for detecting dependents, and the second grammar used sequences of GRs as the main feature for labeling the dependents found by the first grammar. The overall task of detecting and labeling dependents was done by cascading these two grammars. In the second approach, we built a single grammar that uses sequences of GRs as the main feature for detecting dependents and for labeling them; here, the overall task was done in one go by this grammar. The first approach used sequences of GRs and sequences of POS tags, while the second only used sequences of GRs.

We showed that English GRs follow a very strict sequential order, but not as strict as POS tags of verbal dependents. The latter are more effective for detecting and labeling dependents, and, hence, provide a more reliable instrument for detecting them. We also showed that sequences of POS tags are fundamental for parsing performance: they provide a reliable source for predicting and detecting dependents.

Acknowledgments. Maarten de Rijke was supported by the Netherlands Organization for Scientific Research (NWO) under project numbers 017.001.190, 220-80-001, 264-70-050, 354-20-005, 612-13-001, 612.000.106, 612.000.207, 612.066.302, 612.069.-006, and 640.001.501.

References

1. Collins, M.: Three generative, lexicalized models for statistical parsing. In: Proc. 35th ACL. (1997)
2. Eisner, J.: Three new probabilistic models for dependency parsing: An exploration. In: Proc. COLING 1996. (1996)
3. Infante-Lopez, G., de Rijke, M.: Alternative approaches for generating bodies of grammar rules. In: Proc. 42nd ACL. (2004)
4. Collins, M.: Head-Driven Statistical Models for Natural Language Parsing. PhD thesis, University of Pennsylvania, PA. (1999)
5. Charniak, E.: Tree-bank Grammars. In: Proceedings AAAI'96, Portland, Oregon (1996)
6. Infante-Lopez, G.: Two-Level Probabilistic Grammars for Natural Language Parsing. PhD thesis, Universiteit van Amsterdam (2005)
7. Buchholz, S.: Memory-Based Grammatical Relation Finding. PhD thesis, Universiteit van Tilburg (2002)
8. Abney, S., McAllester, D., Pereira, F.: Relating probabilistic grammars and automata. In: Proc. 37th ACL. (1999) 542–549
9. Manning, C., Schütze, H.: Foundations of Statistical Natural Language Processing. The MIT Press, Cambridge, MA (1999)
10. Thollard, F., Dupont, P., de la Higuera, C.: Probabilistic DFA inference using Kullback-Leibler divergence and minimality. In: Proc. ICML, Stanford (2000)

Verb Sense Disambiguation Using Support Vector Machines: Impact of WordNet-Extracted Features

Davide Buscaldi, Paolo Rosso, Ferran Pla,
Encarna Segarra, and Emilio Sanchis Arnal

Dpto. Sistemas Informáticos y Computación,
Universidad Politécnica de Valencia, Valencia, Spain
{dbuscaldi, proso, fpla, esegarra, esanchis}@dsic.upv.es

Abstract. The disambiguation of verbs is usually considered to be more difficult with respect to other part-of-speech categories. This is due both to the high polysemy of verbs compared with the other categories, and to the lack of lexical resources providing relations between verbs and nouns. One of such resources is WordNet, which provides plenty of information and relationships for nouns, whereas it is less comprehensive with respect to verbs. In this paper we focus on the disambiguation of verbs by means of Support Vector Machines and the use of WordNet-extracted features, based on the hyperonyms of context nouns.

1 Introduction

Word Sense Disambiguation (WSD) is an open problem in the field of Natural Language Processing (NLP). 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. Supervised approaches to WSD usually perform better than unsupervised ones [4]. Results of the recent Senseval-3¹ contest attest this supremacy; moreover, recent results of the application of Support Vector Machines (SVM), a well-known supervised learning technique, to the Word Sense Disambiguation task seem promising [3].

Some interesting results have been obtained recently in the supervised disambiguation of verbs [1], by using context-extracted features and a multi-class learning architecture. The disambiguation method described in this paper replicates the feature extraction model proposed in [1], with the addition of WordNet [5] extracted features, while using a SVM-based learning architecture. The system was tested over a subset of the Senseval-3 Lexical Sample corpus.

2 Support Vector Machines

The SVM [6] performs optimization to find a hyperplane with the largest margin that separates training examples into two classes. A test example is classified

¹ <http://www.senseval.org>

depending on the side of the hyperplane it lies in. Input features can be mapped into high dimensional space before performing the optimization and classification. A kernel function can be used to reduce the computational cost of training and testing in high dimensional space. If the training examples are nonseparable, a regularization parameter C ($= 1$ by default) can be used to control the trade-off between achieving a large margin and a low training error. We used the implementation of SVM from Thorsten Joachims [2], *SVM-light*. In order to apply the SVM to the WSD task, each nominal feature with possible values was converted into binary (0 or 1) features. If a nominal feature took the i -th value, then the i -th binary feature was set to 1 and all the other binary features were set to 0. The default linear kernel was used. Since SVMs handle only binary (2-class) classification, we built one binary classifier for each sense class.

3 Disambiguation Model

Given a verb in its sentential context $\langle \text{verb}, \text{sentence} \rangle$, the goal is to develop procedures for the automatic labeling of the semantic class it encodes. An important first step is to map the context information of each verb into feature vectors. The following features were selected among the ones proposed in [1]:

- *Word feature*: is the lexical form of each word in a window of size six (three before and three after) surrounding the target verb.
- *Part-of-Speech tag (POS) feature*: is the POS tag of each word in a window of size six surrounding the target verb.
- *Word.POS tag feature*: is the conjunction of each word and its POS tag for each word within a window of size six of the target verb.

Moreover, a feature was added for each noun found in the verb's context: the *l-hyperonym feature*: the hyperonyms extracted from WordNet at depth l , where l indicates the maximum number of levels to be considered, upwards in the WordNet hierarchy.

For instance, let us consider the following POS-tagged sentence: “*Reid/NNP saw/VBD me/PRP looking/VBG at/IN the/DT iron/NN bars/NNS ./.*”, where *looking* is the verb to be disambiguated. Therefore, the following feature vectors are associated to this instance of the verb: (*Reid, saw, me, at, the, iron*) as word feature vector, (*NNP, VBD, PRP, IN,DT,NN*) as the POS feature vector, (*Reid.NNP, saw.VBD, me.PRP, at.IN, the.DT, iron.NN*) as the Word.POS feature vector. Finally, WordNet is used in order to collect the l -hyperonyms of *iron*, the only noun in the context.

Iron has 5 senses in WordNet; for readability reasons, we limit to the first two: *iron, Fe, atomic number 26*: (a heavy ductile magnetic metallic element;), and *iron*: (a golf club). The hyperonyms trees obtained from WordNet for these senses are:

Sense 1: iron, Fe, atomic number 26 ⇒ metallic element, metal ⇒ chemical element, element ⇒ substance, matter ⇒ entity	Sense 2: iron ⇒ golf club, golf-club, club ⇒ golf equipment ⇒ sports equipment ⇒ equipment ⇒ instrumentation ⇒ ... ⇒ <i>entity</i>
--	---

When 1-hyperonyms are used as features, only the first hyperonym is added to the feature vector; in this case, then only the offsets (numeric IDs that identify in an unique way the WordNet synsets) corresponding to (*metallic element, metal*) and (*golf club, golf-club, club*) are added to the features. Otherwise, if, for instance, 5-hyperonyms are used, therefore all the hyperonyms of sense 1 and the hyperonyms up to *instrumentality, instrumentation* for sense 2 are added to the feature vector. The hyperonyms are extracted from all the senses of the noun, without any assumption on the right sense of that noun in its context.

4 Experiments

The experiments have been carried out over the verbs in the Senseval-3 Lexical Sample corpus; for each verb a training set of xml-tagged sentences is provided together with a smaller test set of sentences in the same format. The averaged number of training samples for each verb is 123.53, roughly doubling the averaged number of test samples (61.81). Eight SVM models were trained for every sense of each verb, one without considering the WordNet extracted features, and the remaining seven with l -hyperonyms features, with $l \in \{1, 2, 3, 4, 5, 6, 7\}$. Therefore, for each sense $s_i(v)$ of verb v , a SVM was trained to classify verb instances of sense $s_i(v)$ against the others.

In the testing phase, the $|s(v)|$ SVMs, where $|s(v)|$ is the number of senses of verb v , are used to classify the verb instance. Although *SVM_light* is not a

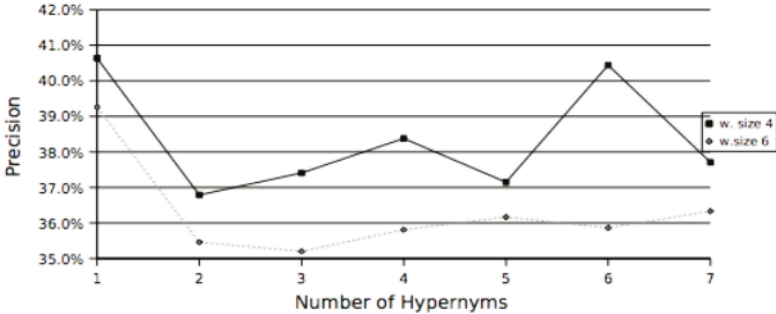


Fig. 1. Results obtained at the different hypernymy levels, considering a window of size 4 and 6

probabilistic SVM, we used its output value w as a confidence weight in the range $[-1, 1]$, $w \in R$, where 1 means 100% confidence in v having the sense $s_i(v)$, and -1 means 100% confidence in v *not* having sense $s_i(v)$. The sense assigned to the verb corresponds to the sense related to the SVM which returns the highest w .

We carried out a study of the window size, obtaining an optimal size of 4 words (2 before and 2 after the verb to disambiguate). The calculated baseline precision, obtained by assigning the most frequent sense to each test instance, is 47.5%. A comparison with Senseval-3 is not feasible, given that the results are not calculated separately for each Part-Of-Speech. The results obtained at the different hypernymy levels are displayed in Fig. 1.

In our experiments the use of all context hypernyms did not allow to improve those obtained by using the base SVM configuration. Nevertheless, we suppose that better results can be achieved if only the hypernyms of the right sense of the context nouns are considered. In fact, we obtained 49.7% in precision by taking into account only the hypernyms from the 3 most common senses of context nouns.

5 Conclusions and Further Work

Our experiments show that although the use of SVMs allowed to obtain better results than the baseline, WordNet-extracted features did not prove so useful. Precision dropped dramatically when using only one hypernym. Size of training samples may be also too small to draw definitive conclusions. Further work may include the possibility to take into account also the distance of context features from the verb (as proposed in [3]) and use a weighting proportional to the depth of the hyperonyms in the hierarchy.

Acknowledgments

We would like to thank R2D2 CICYT (TIC2003-07158-C04-03) and ICT EU-India (ALA/95/23/2003/077-054) research projects for partially supporting this work.

References

1. Girju, R., Roth, D., Sammons, M.: Token-level Disambiguation of VerbNet classes. Proc. of the Interdisciplinary Workshop on Verb Features and Verb Classes, Saarbrücken, Germany (2005)
2. Joachims, T.: Making large-scale SVM Learning Practical. Advances in Kernel Methods. MIT-press, 1999.
3. Lee, Y.K., Ng, H.T: Supervised Word Sense Disambiguation with Support Vector Machines and Multiple Knowledge Sources. Proc. of the SENSEVAL-3 workshop. Barcelona, Spain, 2004
4. Mihalcea, R., Moldovan, D.I.: A Method for Word Sense Disambiguation of Unrestricted Text. Proc. of the ACL-99 Conference. Maryland, NY, U.S.A., 1999
5. Miller, G.: WordNet: a lexical database for english. CACM, 38(11):39-41, 1995.
6. Vapnik, V.N.: The Nature of Statistical Learning Theory. Springer-Verlag, New York (1995)

Preposition Senses: Generalized Disambiguation Model

Chutima Boonthum, Shunichi Toida, and Irwin Levinstein

Department of Computer Science, Old Dominion University,
Norfolk, Virginia 23529, USA
{cboont, toida, ibl}@cs.odu.edu

Abstract. Our previous study on disambiguating the preposition “with” (using WordNet for hypernym and meronym relations, LCS for verb and preposition lexical information, and features of head and complement) looked promising enough to warrant study for other prepositions. Through investigation of ten frequently used prepositions, this paper describes general senses of prepositions and sense-case definitions, introduces a novel generalized sense disambiguation model, and demonstrates how this benefits a paraphrase recognition system.

1 Introduction

Why is preposition sense disambiguation important in a paraphrase recognition system? When two expressions describe the same situation, each is considered to be a paraphrase of the other. Various authorities have mentioned the following paraphrase patterns: using synonyms, changing part-of-speech, reordering ideas, breaking a sentence into smaller ones, substituting a word with its definition, and using different sentence structures. Prepositions play a significant role in changing sentence structures more than other paraphrase patterns. Consider the following sentences:

- (a) “John builds a house *with* a hammer.”
- (b) “John *uses* a hammer *to* build a house.”
- (c) “John builds a house *by using* a hammer.”
- (d) “A house is built *by* John who *uses* a hammer.”
- (e) “A house is built *by* John *using* a hammer.”

Although these sentences convey the same meaning, they have different syntactic structures and use different prepositions. Sentence (a) uses ‘with’ to indicate an instrument used to complete an action while (b), (c), (d), and (e) have the verb ‘use’ to indicate a use of an instrument. Sentences (d) and (e) are in the passive voice and they use the preposition ‘by’ to indicate an agent (who performs the action.) Sentence (c) uses ‘by’ to indicate a secondary action of this agent in completing the primary action. ‘By’ can be omitted in (c) and the sentence still has the same meaning.

- (f) “John builds a house *with* a kitchen.”
- (g) “John builds a house *that has* a kitchen.”
- (h) “John builds a house *having* a kitchen.”
- (i) “A house is built *by* John *with* a kitchen.”

The preposition ‘with’ in (f) indicates that a ‘kitchen’ is a property or a part of a ‘house.’ Given that a complement (a noun attached to a preposition) indicates a

property of a head (a noun to which a preposition is attached), the verb ‘have’ is used in the absence of a preposition, as shown in sentences (g) and (h). Sentence (i) is in the passive voice and ‘by’ indicates the agent.

In some cases, prepositions can be used interchangeably, for example “Mary covers the baby *with* blankets” vs. “Mary covers the baby *in* blankets.” In many cases; however, changing prepositions means changing sentence structures, for example “There are sixteen ounces *for* every pound” vs. “Each pound consists *of* sixteen ounces.” In addition to changing a sentence structure, an absence of a preposition can result in changing a part-of-speech, for example “a book *with* a green cover” vs. “a *green-covered* book”. In this example, since ‘with’ indicates a property of a book, a property (‘green cover’) can change its part-of-speech from a *noun* to an *adjective*.

As can be seen from the examples above, to recognize paraphrases more efficiently and accurately, senses of prepositions need to be identified. Hence, preposition disambiguation is indeed essential.

As generally with word sense disambiguation, preposition sense disambiguation requires lexical and world knowledge, and contextual information. In sentence (a), ‘with’ indicates an instrument used in building a house whereas ‘with’ in (f) indicates a property of a house that has a kitchen. To distinguish these two senses of ‘with’, we need to understand the difference between ‘hammer’ and ‘kitchen’. This requires world knowledge as well as the understanding of word meanings.

2 Related Works on Preposition Disambiguation

Recently preposition disambiguation has been studied by a number of researchers including Alam [1], Harabagiu [15], O’Hara and Wiebe [32, 33], Litkowski [21, 22], Mohanty et al. [28, 30], Bannard et al. [2], Saint-Dizier et al. [36], and Sopena et al. [37].

Alam [1] has worked on the disambiguation of ‘over’ by considering its meaning with respect to two main categories: one is the meaning that can be identified by its complement noun phrases and the other is the one that derives from its head (verb or noun phrase). Alam defined the subcategories of ‘over’ in terms of the various features of head and complement. For both head and complement, ontological categories (hypernyms) are used, e.g. furniture is a *physical object*, coffee is a *drink*. Two decision trees were proposed: one for the head and another for the complement. To determine the meaning of ‘over’, the complement decision tree is examined first, since most of the meanings of ‘over’ can be identified from its complement. If the sense of ‘over’ cannot be identified by this tree, then the head decision tree is checked. Alam performed this evaluation manually, though she claimed that the algorithm should be easy to implement.

Harabagiu [15] used WordNet to disambiguate prepositional phrase attachments. She used the hypernym/hyponym relation of either verbs or nouns or both from WordNet to categorize the arguments of preposition relations. This approach is based on inferential heuristics. Three heuristic rules were defined for the preposition ‘of’ in order to understand different types of valid prepositional structures.

Other work using WordNet for word sense disambiguation include Li, Szpakowicz and Matwin [19]; Voorhees [40]; Nastase and Szpakowics [31]; Peh and Ng [34], but they are primarily for nouns and verbs.

Mohanty et al. [28, 30] have used preposition syntactic frames to define prepositional semantic. A number of rules are defined to analyze each frame type (attribute of a verb, attribute of a noun before a preposition and a noun after a preposition) to disambiguate prepositional phrase attachment as well as to identify a semantic relation of this attachment. They used a system called UNL (Universal Networking Language), which has its own lexical knowledge. Its English Analyzer uses both the existing English grammar rules in UNL itself and user-defined rules of preposition attachment and semantics, and then generates a UNL expression. They started on the preposition ‘of’ and are expanding this concept for other prepositions (*for, from, in, on, to, with*).

Litkowski [21, 22] has designed “The Preposition Project” (TPP) which provides a comprehensive characterization of preposition senses. This is a well-defined sense inventory with FrameNet [4] instances; however, the properties of complement and attachment are given as English phrases (e.g. “permeable or breakable physical object”, “a perceived object; sometimes complement of a verb of perception”). Therefore, an automated disambiguation system would have to understand them to use that information. He completed the prepositions ‘by’ and ‘through’, and has been working on ‘with’, ‘for’, and ‘of’.

Our model expands upon Alam’s work in disambiguating preposition senses by identifying the features of the head and complement in that it adds WordNet’s meronym/holonym relation to the set of disambiguating resources. It also draws upon Harabagiu’s idea of using WordNet ontological categories although for a different purpose. We are unable to use the preposition definitions from either Litkowski or Mohanty et al. because they are still under development.

3 Generalized Disambiguation Model

As a first step for generalizing our approach, we investigated the ten most frequently used prepositions in the Brown corpus [13]: *of, to, in, for, with, on, at, by, from, and over*. These prepositions cover 85.63% of all the occurrences of the 46 prepositions used in this corpus. The Brown corpus consists of 1,015,945 words, of which 14.2% are prepositions.

In this section we will describe general senses of prepositions, sense-case definitions, and a generalized sense disambiguation model.

3.1 General Senses of Prepositions

For each preposition, we initially used LCS’ preposition lexical information [11, 12] to define preposition senses. Unfortunately, we found that the LCS’ senses of many prepositions are inadequate. Because LCS lexical information was mainly developed for verbs, the meanings of prepositions were defined only if they were used in the verbs’ definition.

We then explored general positions (describing a role played by the preposition). Jackendoff [16, 17] defined six positions while Dorr’s LCS [12] has ten positions (four new positions have been added to Jackendoff’s). Only six positions are appropriate for preposition senses: *Location*, *Possession*, *Intention*, *Instrument*, *Identification*, and *Temporal*. After some effort in finding the features of head and complement for each sense of these prepositions, we discovered that these six general senses are still not enough. Finally, the preposition meanings by Quirk [35] and the preposition classification by Barker [3] have been studied and were used to finalize the following seven general senses of prepositions:

Participant – the preposition indicates that its head or complement participates in the event or action. This includes agents, accompaniments, objects, recipients, beneficiaries, and experiencers.

Location – the preposition indicates that its complement is the location where the event or action occurs. This includes directions, sources, intermediate locations, and destinations.

Time – the preposition indicates that its complement is temporally related to the event or action. This includes durations, specific date or days, time, and frequencies.

Intention – the preposition indicates that its complement is a purpose, a cause, a manner of the event or action.

Instrument – the preposition indicates that its complement is a tool used to complete the event or action. This also includes a use of materials.

Identification – the preposition indicates that its head is a part or a property of its complement (and vice versa).

Quality – the preposition indicates that its complement represents content, measure, and order. The content includes physically filling a container.

We validated our general senses by mapping meanings of prepositions (except idiomatic expressions) defined in other dictionaries, such as Longman’s Dictionary [23], Merriam-Webster online [28], and dictionary.com [19], to our general senses. Using prepositions’ meaning from Longman’s Dictionary, we found that our seven general senses cover all meanings.

Should we need finer-grained senses for prepositions, these seven general senses can be specialized. In our initial attempt in utilizing these seven senses in a paraphrase recognition system, they proved sufficiently adequate.

3.2 Preposition Sense-Case Definition

For each preposition, we defined a number of rules for each sense. Each rule (called a *sense-case* or *case*) consists of the preposition name, general sense, sense-case identification, verb attribute, features of head and complement, relation between head and

complement, syntactic role of head and complement, and mapping rule to the CG relation. Table 1 provides examples of case definitions in a concise format¹.

We divided these cases into two categories: if the features of head and complement can be clearly defined, we tag this case as a *specific case*; otherwise, as a *general case*. Ontology categories used in general cases are either the top-level ontology in WordNet (e.g., *entity*, *act*, *state*) or immediate children of the top-level ontology (*thing*, *physical object*, *location*, *sky* are immediate children of *entity*). The general case is used to cover broader and unclearly-identified categories defined in WordNet.

Table 1. Sense-Case Definition. Examples of ‘with’ and ‘in’ case definition: specific and general (general sense-cases include parentheses).

Prep	Sense	Sense-Case	Head	Complement	HC Relation
With	Part	WithPartAgtAcmp	Person (Subj)	Person	
With	Part	WithPartObjAcmp	Person (Obj)	Person	
With	Loc	WithLoc_L		Location	
With	Loc	WithLoc_A		Area	
With	Inten	WithIntenGen_F		Feeling	
With	Inst	WithInstr		Instrumentality	
With	Iden	WithIdentIsPart	physical_obj	physical_obj	Is-Part_Head
With	Qual	WithQualContSubs	container	Substance	
With	Loc	WithLoc_(S)		Space	
With	Inten	WithInten_(A)		Act	
With	Iden	WithIdent_(A)		Attribute	
In	Loc	InLocAt_St		State	
In	Time	InTimeAt_S		season	
In	Time	InTimeAt_M		month	
In	Time	InTimeDur_U		time_unit	
In	Inst	InInstr_C		communication	
In	Inst	InInstrMatr_T		material	
In	Iden	InIdentIsPart	physical_obj	physical_obj	Is-Part_Comp
In	Part	InPart_Rel		relation	
In	Loc	InLoc_(S)		space	
In	Inten	InInten_(M)		motivation	
In	Quan	InQuan_(Q)		quantity	

Currently, all cases are equally weighted. In future, we will provide a weight for each case based on commonly used senses and case categories (specific or general).

3.3 Sentence Parsing and Meaning Representation Construction

This research has been conducted in connection with a paraphrase recognition system to be used in a web-based automated reading strategy trainer called iSTART (Interactive

¹ Verb attributes and syntactic rules are less used in the current implementation. Mapping rules are used in transformation process described in section 5.

Strategy Trainer for Active Reading and Thinking). iSTART is motivated by a live training called SERT (Self-Explanation Reading Training), developed by McNamara and her colleagues [23, 27] as a way to train adolescents to use active reading strategies to self-explain difficult texts more effectively. Details of the reading strategies can be found in [23] and of iSTART in [26].

In the trainer, a student’s explanation of a sentence is compared with the given sentence to determine (among other things) whether it is a paraphrase of the sentence and of which paraphrase type it is. The current system uses statistical methods [25] (that is simple word matching and latent semantic analysis) which have limited reliability, especially in distinguishing paraphrases from explanations that contain more than a paraphrase. We therefore propose a new system for handling the student’s explanation more effectively by constructing an internal representation and then recognizing the use of paraphrase.

In the proposed system, the formalism of Conceptual Graph (CG) [38, 39] is used as an internal meaning representation; hence the recognition process is to match the component CG triplets of both sources and identify matching categories. Details of paraphrase definition and recognition model can be found in [5] and [6].

To construct a meaning representation, a sentence is first parsed by the Link Grammar (which may produce several alternative parses) [8]. A parse consists of triplets: starting word, ending word, and a connector type between these two words. The sentence “John builds a house with a hammer” is tokenized and parsed as follows:

```
[ (0=LEFT-WALL) (1=John) (2=builds.v) (3=a) (4=house.n)
(5=with) (6=a) (7=hammer.n) (8=.) ]
[[ [0 8 (Xp)] [0 1 (Wd)] [1 2 (Ss)] [2 5 (MVp)] [2 4 (Os)]
[3 4 (Ds)] [5 7 (Js)] [6 7 (Ds)] ]]
```

[1 2 (Sp)] means ‘John’ is the subject of ‘build’, [2 5 (MVp)] means ‘build’ is connecting to ‘with’ prepositional phrase and [5 7 (Js)] means the preposition ‘with’ has ‘hammer’ as its object. We then convert each Link triplet into a corresponding CG triplet. The two words in the Link triplet are converted into two concepts of the CG. To decide whether to put a word on the left or the right side of the CG triplet, we define a mapping rule for each Link connector type. For example, a Link triplet [1 2 (S*)] will be mapped to the ‘Agent’ relation, with word-2 as the left-concept and word-1 as the right-concept: [Word-2] → (Agent) → [Word-1]. The CG triplets for this sentence are follows:

```
2 [1 2 (Ss)] -> #S# ->
   [builds.v] -> (Agent) -> [John]
3 [2 5 (MVp)] -> #M# MVp + J (6) # ->
   [builds.v] -> (Verb_Prep) -> [hammer.n] (with)
4 [2 4 (Os)] -> #S# ->
   [builds.v] -> (Patient) -> [house.n]
```

Each case (numbered 2-4) shows a Link triplet and its corresponding CG triplet. These will be used in the recognition process. The ‘#S#’ and ‘#M#’ indicate single and multiple mapping rules. Details are omitted here, but can be found in [5] and [6].

3.4 Generalized Sense Disambiguation Model

To disambiguate the meaning of a preposition, the following general steps are taken:

1. A sentence is parsed by Link Grammar and a preliminary CG for the sentence is generated.
2. For each preposition found in the sentence and within a parse produced by Link Grammar, a CG triplet containing the preposition (called a *target*) is selected.
3. The head and complement of the target are identified.
4. For each sense-case of a target preposition (a row defined in Table 1), the head and complement are analyzed using WordNet to determine the hypernym of each, the hypernym between the two (head is a *kind of* complement and vice versa), and the meronym relation between the two (head is a *part of* complement and vice versa).
5. Because of our inclusive approach for the paraphrase recognition, all sense-cases that meet the criteria are selected as possible senses of this target preposition.

For example, let us consider the sentence “John builds a house with a hammer.” The preposition in this sentence is ‘with’. One of the linkages from Link Grammar shows that the complement of ‘with’ is ‘hammer’ and the head is ‘house’. For each sense-case of ‘with’, the features of the head and complement are checked. For instance for the ‘WithInstr’ case, only the complement is required and it should be an instrumentality category. In this case, ‘hammer’ is a kind of instrument; hence this ‘WithInstr’ sense-case is selected. From the current implementation, a portion of the output of this example is shown below. This is a concise version of the result showing the main sense, sense-case, head, complement, and a verb of the sentence.

= S: John builds a house with a hammer. =

Preposition Senses:

- Participant	WithPartObjs ## [house,hammer] - build	
- Participant	WithPart_(T) ## [,hammer] - build	<i>T=Thing</i>
- Instrument	WithInstr ## [,hammer] - build	
- Intention	WithInten_(A) ## [,hammer] - build	<i>A=Act</i>

The detailed versions of results can be found in [7]. That includes which Link Grammar linkage they are derived from, the sense of the complement of ‘with’, the tree of the hypernym relation, the node of the WordNet SynSet, and the hierarchy level (if there is a meronym relation). Currently we search WordNet for words in their original lexical form or with minimal stemming (only -s and -ed suffixes are removed). In our final paraphrase recognition system; however, different word forms will be considered.

4 Evaluation

For each preposition (except ‘with’ for which the procedure of sentence selection can be found in [7]), 120 sentences² were hand-selected from either Link Grammar

² Our existing sentence selection may bias our results. The sentences were manually selected as described to serve as a preliminary test set. The sentences had to be parseable by the Link Grammar to be useable in the system. In future, we are planning to use other annotated corpora, such as SENSEVAL or MSPC (Microsoft Paraphrase Corpus) to reduce authors’ influences.

sample sentences, one of the existing corpora, online resources, or manually created sentences. Due to the limitation of the Link Grammar parser and the current implementation of the Conceptual Graph generator, it is necessary for us to construct or simplify some sentences manually to illustrate how the algorithm works.

Each of the 120 sentences was manually analyzed by the authors to identify possible senses. (In future, persons other than the authors will provide the analysis in order to reduce the authors' influence.) Then, this evaluation was used to compare against the results obtained by our implementation.

For each sentence, the sense result can be classified into one of the following categories:

1. *Exactly Correct*: the preposition specific sense-case provided by our implementation is exactly the same as the expected sense.
2. *Specific-case Partially Correct*: at least one of the resultant preposition specific sense-cases is the expected sense.
3. *General-case Partially Correct*: at least one of the resultant preposition general sense-cases is the expected sense, and it was not listed under the specific cases.
4. *Specific-case Incorrect*: the specific-case result does not include the expected sense.
5. *General-case Incorrect*: the general-case result does not include the expected sense
6. *No Result*: there is at least one Link Grammar linkage, but the categories of head or complement or both existing in WordNet do not meet the criteria defined. This includes cases when prepositions are used as verb-particles or in idiomatic expressions.

The results are shown in Table 2. Overall, the precision of our disambiguation model is 79% of sentences with resultant senses and 76% of all sentences.

Table 2. Preposition Sense Disambiguation Results. Ordered by the frequency of use in the Brown Corpus, this indicates the number of sentences that have been classified into each of the 6 categories. (A= No. of sentences contain correct senses, B = No. of sentences that could find preposition senses, C = Percent correctness of those that could find a result, D = Percent correctness of all 120 sentences, E = Percent correct from specific cases, F = Percent correct from general cases. [C = E + F], Z = The best percent correctness from previous work, if any).

Result	of	to	in	for	with	on	at	by	from	over
1	29	47	8	42	23	19	28	18	48	23
2	27	34	69	10	61	23	22	4	11	21
3	52	18	20	39	19	41	41	50	33	33
4	5	4	8	16	14	15	9	10	5	5
5	7	17	15	6	3	18	15	29	23	18
6	0	0	0	7	0	4	5	9	0	20
Total	120	120	120	120	120	120	120	120	120	120
A	108	99	97	91	103	83	91	72	92	77
B	120	120	120	113	120	116	115	111	120	100
C	0.90	0.83	0.81	0.81	0.86	0.72	0.79	0.65	0.77	0.77
D	0.90	0.83	0.81	0.76	0.86	0.69	0.76	0.60	0.77	0.64
E	0.47	0.68	0.64	0.46	0.70	0.36	0.43	0.20	0.49	0.44
F	0.43	0.15	0.17	0.35	0.16	0.35	0.36	0.45	0.28	0.33

Our result for the preposition ‘over’ is superior to Alam’s work (a manual evaluation) and for ‘of’ it is as good as the result by Manhanty et al. Our result for ‘by’ is low due to a usage of ‘by’ in the passive voice. This can be disambiguated in the CG generation process which currently is separated from the preposition disambiguation model. We are confident that our generalized disambiguation model is adequate for the paraphrase recognition system.

The incorrect results are due to limitations in one or more of the following components: *Link Grammar Parser* - its parse algorithm, words contained in its dictionary, and part-of-speech categories of words; *WordNet* - word senses, word categories (hypernym/meronym); *CG Generator* - mapping rules from Link Grammar to CG for identifying a target preposition and its head and complement; or *Human Analysis* - expected preposition sense.

5 Paraphrase Recognition

After all plausible preposition senses are identified, the meaning of the preposition should be incorporated into the CG to be used in the paraphrase recognition system. From the example “John builds a house with a hammer” in section 3, the CG triplet containing the preposition ‘with’ is shown below:

```
3 [2 5 (MVp)] -> #M# MVp + J (6) # ->
   [builds.v] -> (Verb_Prep) -> [hammer.n] (with)
```

A subset of disambiguated senses of this example is shown in section 3.4. To utilize these resultant senses in the paraphrase recognition system, the following tentative steps are taken:

1. For each sense-case, a mapping rule is defined to transform this case into a proper CG relation, e.g. a ‘WithInstr’ case will be mapped to an ‘Instrument’ relation.

```
WithInstr :
    [Left-Concept] -> (Instrument) -> [Right-Concept]
```

2. For each resultant sense-case, the CG triplet will be transformed to the corresponding CG relation. For this example, the transformed CG triplet for the ‘WithInstr’ case is as shown below:

```
3 [2 5 (MVp)] -> #M# MVp + J (6) # ->
   [builds.v] -> (Instrument) -> [hammer.n]
```

3. Therefore, recognizing a paraphrase can be as simple as matching CG triplets between transformed CGs of two sentences. In the sentence “John uses a hammer to build the house,” the transformed CG of one of the two linkages is as shown below:

```
[uses.v] -> (Agent) -> [John]
[uses.v] -> (TO) -> [build.v]
[uses.v] -> (Patient) -> [hammer.n]
[build.v] -> (Patient) -> [house.n]
```

In this example, the following paraphrase recognition rule is defined and will be used during recognition process.

```

[uses.v] -> (TO) -> [VERB]
[uses.v] -> (Patient) -> [INSTRUMENT]
≡      [VERB] -> (Instrument) -> [INSTRUMENT]

```

Briefly, to utilize the sense disambiguation result in the paraphrase recognition process, we need (1) mapping rules that transfer the disambiguated sense-cases to their corresponding CG relations and (2) paraphrasing rules that match a group of CG triplets beyond the simple matching process described in [5] and [6].

6 Discussion and Future Work

After applying the same technique used for ‘with’ for other prepositions and having generalized the disambiguation model, the preliminary results are encouraging but various issues still need to be explored include improving the result by disambiguating noun senses or using world knowledge or context information, ranking the result for future use in the paraphrase recognition process, handling cases of prepositions in metaphors and verb particles, and considering other factors besides heads and complements. Some of these issues are further explained in this section.

We currently give all possible senses of nouns equal weighting. To narrow down choices for the sense of prepositions, we plan to add noun sense disambiguation to the system. Some of the heuristic methods referred to in [9] will be examined, e.g. most frequently used senses and a default sense. Theoretically, disambiguating nouns will add more precision to the preposition sense disambiguation.

We are using all plausible senses of prepositions in two ways: one is when we prepare the text that is to be paraphrased, in which case these plausible senses (highest rank first) are presented to an expert to select the correct sense, and another is when we attempt paraphrase recognition where we use an inclusive approach, that is if one of the plausible senses matches the given paraphrase, then that is selected.

Even with the noun disambiguation or context information or both, the sense of a preposition may not be uniquely determined. To benefit from this disambiguation model in the paraphrase recognition process, the ranking of preposition senses is essential. Based on head and complement information, the most frequently used sense could be rated higher than ones which rarely occur. Similarly, the specific sense-cases should be rated higher than the general ones. This ranked result will also be presented to the user during the identification process of the appropriate sense.

Prepositions are also used in metaphoric expressions, idioms (e.g. *with it* – dressing in fashionable clothes; *with you* – understand someone’s explanation; *over with* – completely finished), or verb particles (e.g. *come up with*, *deal with*, *relate to*, *tie in*). The sense of prepositions in such cases is idiosyncratic and no general rule can be given. Therefore, we currently are not considering the disambiguation of such uses.

Besides features of heads and complements, other information may be needed in the prepositional sense disambiguation; they are yet to be explored. Even so, our generalized disambiguation model has proved adequate to benefit the paraphrase recognition system.

References

1. Alam, Y.: Decision Trees for Sense Disambiguation of Prepositions: Case of Over. HLT-NAACL, Computational Lexical Semantic Workshop, Boston, MA (2004) 52-59
2. Bannard, C., Baldwin, T.: Distributional Models of Preposition Semantics. ACL-SIGSEM, Workshop on "The Linguistic Dimensions of Prepositions and their Use in Computational Linguistics Formalisms and Applications", Toulouse, France (2003) 169-80
3. Barker, K.: The Assessment of Semantic Cases Using English Positional, Prepositional and Adverbial Case Markers. TR-96-08, Computer Science, University of Ottawa. (1996)
4. University of California, Berkeley: FrameNET <http://framenet.icsi.berkeley.edu/>
5. Boonthum, C.: iSTART: Paraphrasing Recognition. Doctoral Dissertation Proposal, Computer Science Department, Old Dominion University, Norfolk, VA (2004)
6. Boonthum, C.: iSTART: Paraphrase Recognition. ACL, Student Research Workshop, Barcelona, Spain (2004) 31-36
7. Boonthum, C., Toida, S., Levinstein, I.B.: Sense Disambiguation for Preposition 'with'. ACL-SIGSEM, Workshop on "The Linguistic Dimensions of Prepositions and their Use in Computational Linguistic Formalisms and Applications", University of Essex - Colchester, United Kingdom (2005) 153-162
8. Carnegie Mellon University: Link Grammar. <http://www.link.cs.cmu.edu/link/> (2000)
9. Castillo, M., Real, F., Atserias, J., Rigau, G.: The TALP Systems for Disambiguating WordNet Glosses. ACL, SENSEVAL-3 Workshop, Barcelona, Spain (2004) 93-96
10. Cycorp.: Cyc. <http://www.cyc.com/> (2002)
11. Dorr, B.: LCS Verb Database, Online Software Database of Lexical Conceptual Structures and Documentation, UMCP (2001)
12. Dorr, B., Hendler, J., Blanksteen, S., Migdalof, B.: Use of LCS and Discourse for Intelligent Tutoring: On Beyond Syntax. In M. Holland, J. Kaplan, and M. Sams (eds.), Intelligent Language Tutors: Balancing Theory and Technology, Lawrence Erlbaum Associates, Hillsdale, NJ (1995) 288-309
13. Edict VLC: Word Frequency Lists <http://www.edict.com.hk/textanalyser/wordlists.htm>
14. Fellbaum, C.: WordNet: an electronic lexical database. The MIT Press: MA (1998)
15. Harabagiu, S.: An Application of WordNet to Prepositional Attachment. ACL, Santa Cruz (1996) 360-363
16. Jackendoff, R.: Semantics and Cognition. MIT Press, Cambridge: MA (1983)
17. Jackendoff, R.: Semantics Structures. MIT Press, Cambridge: MA (1990)
18. Levin, B.: English Verb Classes and Alternations: A Preliminary Investigation, University of Chicago Press, Chicago: IL (1993)
19. Lexico Publishing Group, LLC. Dictionary.com (1995)
20. Li, X., Szapkowicz, S., Matwin, S.: A WordNet-based Algorithm for Word Sense Disambiguation. IJCAI, Montreal, Canada (1995)
21. Litkowski, K.: Digraph Analysis of Dictionary Preposition Definition. ACL-SIGLEX, SENSEVAL Workshop on "Word Sense Disambiguation: Recent Success and Future Directions", Philadelphia, PA (2002) 9-16
22. Litkowski, K., Hargraves, O.: The Preposition Project. ACL-SIGSEM, Workshop on "The Linguistic Dimensions of Prepositions and their Use in Computational Linguistic Formalisms and Applications", Colchester, United Kingdom (2005) 171-179
23. Longman Group Ltd. Longman Dictionary of Contemporary English (3rd Edition). Longman, Harlow: UK (1995)
24. McNamara, D.S.: SERT: Self-Explanation Reading Training. Discourse Processes, Vol. 38 No. 1 (2004) 1-30

25. McNamara, D.S., Boonthum, C., Levinstein, I.B., Millis, K. K. (in press) Using LSA and word-based measures to assess self-explanations in iSTART. In T. Landauer, D.S. McNamara, S. Dennis, & W. Kintsch (Eds.), *LSA: A Road to Meaning*, Mahwah, NJ: Erlbaum.
26. McNamara, D.S., Levinstein, I.B., Boonthum, C.: iSTART: Interactive strategy training for active reading and thinking. *Behavior Research Methods, Instruments, & Computers*, Vol. 36 No. 2 (2004) 222-233
27. McNamara, D.S., Scott, J.L.: Training reading strategies. *Proceedings of the Twenty first Annual Meeting of the Cognitive Science Society*. Hillsdale, NJ: Erlbaum (1999)
28. Merriam-Webster, Inc. Merriam-Webster Online. In <http://www.m-w.com/> (1997)
29. Mohanty, R., Almeida, A., Samala, S., Bhattacharyya, P.: The Complexity of OF in English. *ICON*, Hyderabad, India (2004)
30. Mohanty, R., Almeida, A., Bhattacharyya, P.: Prepositional Phrase Attachment and Interlingua, *International Conference on Intelligent Text Processing and Computational Linguistics (CCLING-2005) Workshop on UNL and other Interlingua and their Applications*, Mexico City, Mexico (2005)
31. Nastase, V., Szpakowics, S.: Word Sense Disambiguation in Roget's Thesaurus Using WordNet. *NAACL, WordNet&Other Lexical Resources Workshop*. Pittsburgh (2001) 17-22
32. O'Hara, T., Wiebe, J.: Classifying Preposition Semantic Roles using Class-based Lexical Associations. *TR NMSU-CS-2002-013*, New Mexico State University (2002)
33. O'Hara, T., Wiebe, J.: Preposition Semantic Classification via Penn Treebank and Framenet. *CoNLL*, Edmonton, Canada (2003) 79-86
34. Peh, L., Ng, H.: Domain-Specific Semantic Class Disambiguation Using WordNet. *ACL, Workshop on Very Large Corpora*, Beijing, China (1997) 56-65
35. Quirk, R., Greenbaum, S., Leech, G., Svartvik, J.: *A comprehensive grammar of the English language*. London: Longman (1985)
36. Saint-Dizier, P., Vazquez, G.: A Compositional Framework for Prepositions. *ACL-SIGSEM, International Workshop on Computational Semantic*, Tilburg, Netherlands (2001)
37. Sopena, J., Lloberas, A., Moliner, J.: A Connectionist Approach to Prepositional Phrase Attachment for Real World Text. *ACL, Montreal, Quebec, Canada* (1998) 1233-1237
38. Sowa, J.: *Conceptual Structures: Information Processing in Mind and Machine*. Addison-Wesley: MA (1983)
39. Sowa, J.: Conceptual Graphs as a Universal Knowledge Representation. *Computers Math. Application*, Vol. 23 No. 2-5 (1992) 75-93
40. Voorhees, E.: Using WordNet to Disambiguate Word Sense for text Retrieval. *ACM-SIGIR*, Pittsburgh, PA (1993) 171-180

An Unsupervised Language Independent Method of Name Discrimination Using Second Order Co-occurrence Features

Ted Pedersen¹, Anagha Kulkarni¹, Roxana Angheluta²,
Zornitsa Kozareva³, and Tamar Solorio⁴

¹ University of Minnesota, Duluth, USA

² Katholieke Universiteit Leuven, Belgium

³ University of Alicante, Spain

⁴ University of Texas at El Paso, USA

Abstract. Previous work by Pedersen, Purandare and Kulkarni (2005) has resulted in an unsupervised method of name discrimination that represents the context in which an ambiguous name occurs using second order co-occurrence features. These contexts are then clustered in order to identify which are associated with different underlying named entities. It also extracts descriptive and discriminating bigrams from each of the discovered clusters in order to serve as identifying labels. These methods have been shown to perform well with English text, although we believe them to be language independent since they rely on lexical features and use no syntactic features or external knowledge sources. In this paper we apply this methodology in exactly the same way to Bulgarian, English, Romanian, and Spanish corpora. We find that it attains discrimination accuracy that is consistently well above that of a majority classifier, thus providing support for the hypothesis that the method is language independent.

1 Introduction

Purandare and Pedersen (e.g., [9], [10]) previously developed an unsupervised method of word sense discrimination that has also been applied to name discrimination by Pedersen, Purandare, and Kulkarni [8]. This method is characterized by a reliance on lexical features, and avoids the use of syntactic or other language dependent information. This is by design, since the method is intended to port easily and effectively to a range of languages. However, all previous results with this method have been reported for English only.

In this paper, we evaluate the hypothesis that this method of name discrimination is language independent by applying it to name discrimination problems in Bulgarian, Romanian, and Spanish, as well as in English.

Ambiguity in names of people, places and organizations is an increasingly common problem as online sources of information grow in size and coverage. For example, Web searches for names frequently locate different entities that share

the same name, and this can lead to considerable confusion or misunderstanding on the part of users.

This method assumes that Named Entity Recognition (NER) has already been performed on the text. Thus, our goal is not to identify named entities in text, but rather to disambiguate among those that have already been identified and determine the number of underlying entities that might share the same name.

This paper continues with an overview of our method of clustering similar contexts in order to perform name discrimination. Then we describe the experimental data for each of the four languages included in this study in some detail. We go on to present our experimental results, focusing on the overall accuracy of the automatic discrimination, and giving examples of the labels that are created for clusters. We close with a discussion of related work and some brief thoughts on future work.

2 Discrimination by Clustering Similar Contexts

The method of clustering similar contexts developed by Purandare and Pedersen is well described elsewhere (e.g., [9], [10]) and is implemented in the freely available SenseClusters package¹.

In this paper we employ one variation of their general approach, which results in a second order co-occurrence representation of the contexts to be clustered. We begin with a collection of contexts to be clustered. In general a context can be an unit of text from a few words to a paragraph or entire document. In these experiments, each context contains one or two sentences that contain a single occurrence of an ambiguous name.

If there are a small number of such contexts to cluster, it might be necessary to select the features to represent these contexts from a separate corpus (assuming it is relevant to the contexts to be clustered). However, in this case there are a sufficient number of contexts to cluster such that features can be identified within that data. Thus, in these experiments we say that the test or evaluation data is the same data as the feature selection data. These methods are completely unsupervised and the true senses of the ambiguous name are not used in the feature selection phase or at any stage of the method apart from the evaluation phase. Thus even if one uses the test data as the feature selection data this does not unfairly influence our results as it would for supervised methods.

We identify bigram features from the contexts to be clustered using the log-likelihood ratio. We define bigrams to be two word sequences where no intervening words are allowed. We conducted our experiments both with and without a stop-list, which is a list of closed-class words such as articles, conjunctions, prepositions, etc. When using stop-lists, any bigram made up of one or two stop words is rejected as a feature.

In addition, any bigram that occurs only one time was rejected as a feature, as would be any bigram that has a log-likelihood ratio score less than 3.841. Bi-

¹ <http://senseclusters.sourceforge.net>

grams with values under this threshold have a 95% chance of being independent of each other, that is they are occurring together as if by chance.

The bigram features are represented as a co-occurrence matrix, where the rows represent the first word in the bigram, and the columns represent the second word. The cell values are the corresponding log-likelihood ratios. Singular Value Decomposition (SVD) is performed on this matrix, reducing it down to 10% of the original number of columns, or to 300 dimensions, whichever is smaller. Each row in the resulting matrix is viewed as a co-occurrence vector for the word associated with that row.

We represent the contexts to be clustered using second order context vectors. These vectors are created by considering a *test scope* of five words to the left and five words to the right of the ambiguous target word. The words found in this window for which we have a co-occurrence vector are replaced by their vector. Then the context is represented by averaging together all the vectors found for the words in the test scope.

The resulting contexts were then clustered using the k-means clustering algorithm (referred to as direct clustering in CLUTO², which is the package SenseClusters uses for clustering). We used the I2 criterion function for clustering, and we must specify the number of clusters we wish to find prior to clustering. The I2 criterion function finds the clustering solution that minimizes the distance of the members of a cluster to the centroid of its cluster.

In the experiments in this paper we know what the “correct” clustering should be, since we will create ambiguous names by conflating together relatively unambiguous names and replacing each occurrence of each name with the newly ambiguous conflated form. Then, the effectiveness of the clustering can be evaluated by measuring how well the discovered clusters have separated the entities associated with the name we have conflated together. We report results in terms of accuracy, that is what percentage of the contexts are correctly clustered.

Finally, cluster labels are created for each cluster by identifying the top ten descriptive and discriminating bigrams according to the log-likelihood ratio found in the contexts of each cluster. These bigrams are found by treating the contexts in each cluster as a corpus, and applying the measure in exactly the same way as we did during feature identification. However, for labeling, we allow up to three intervening words between the words that make up the bigram. The descriptive bigrams are the top ten bigrams found in the contexts associated with a cluster, and the discriminating bigrams are the top ten bigrams that are unique in the contexts associated with a cluster. Thus, it’s possible that the descriptive and discriminating labels will overlap.

3 Second Order Co-occurrence Features

At the heart of this method lies the idea of a second order co-occurrence vector. In general, a second order co-occurrence exists between two words that do not occur together, but both tend to occur with some other word. For example, *fire*

² <http://www-users.cs.umn.edu/~karypis/cluto/>

and *garden* might not occur together often, but both may occur frequently with *hose*, as in *fire hose* and *garden hose*. Thus, there is an indirect relationship between *fire* and *garden* through *hose*. This can be thought of as a friend of a friend relation.

Our method for creating second order features was originally proposed by Schütze[11]. It does not directly search for second order co-occurrences in the contexts to be clustered (by creating a network of word associations, for example). Rather, they are identified as a by-product of the method used for representing the contexts to be clustered. Recall that a word by word co-occurrence matrix is created from the feature selection data. This is a matrix containing information about first order co-occurrences, that is showing which words occur together. Each word in a context to be clustered is represented by a vector from this word by word matrix (if one exists for that word), which indicates the first order co-occurrences of that word.

Once collected, all of the available word vectors associated with a context are averaged together to form a representation of that context. Remember that the context contains an occurrence of the ambiguous name, whose underlying identity is what we seek to base our clustering upon. Thus, the name to be disambiguated is represented not by the words that it occurs with, but rather by the average of the first order vectors of the words that co-occur with the target name. Thus, the name to be disambiguated is represented by second order co-occurrences.

We believe second order features are a suitable representation for this problem, since they allow us to find more matching features when confronted with relatively sparse or noisy data. While our data occurs in fairly large quantities, it is from newspaper corpora and as such can be somewhat unfocused or rapidly changing.

4 Experimental Data

In order to evaluate the language independence of our method, we utilized four languages in our experiments: Bulgarian, English, Spanish, and Romanian. We have at least one native speaker of each language among the authors of the paper.

We located large news corpora for each of the four languages, and then identified named entities automatically, or based on our own knowledge of current events and regional history. In order to facilitate evaluation, we created ambiguities in the data by conflating together names that are largely unambiguous. For example, we took all occurrences of *Bill Clinton* and all occurrences of *Tony Blair* and made their names ambiguous by replacing them with *Bill Clinton-Tony Blair*³.

These conflated names appear ambiguous to our method, but we of course know their true underlying identity (pre-conflation) which will allow for auto-

³ The actual conflation of the data was done with version 0.14 of the freely available nameconflate program (<http://www.d.umn.edu/~kulka020/kanaghaName.html>).

matic evaluation. The methodology and motivation behind creating conflated names are identical to pseudo-words as used in the word sense disambiguation literature. 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 [2]. However, our formulation is similar to that of Nakov and Hearst [7], who suggest creating pseudo words of words that are individually unambiguous, and yet still related in some way.

For each language we created five sets of confluations for use in our experiments. Two sets contained the names of people, two contained country or city names, and then one set included organization names. Thus we are making distinctions between names that are of the same general class, making these less obvious distinctions than those between a city and a person, for example. We did not use the same names for all of the languages, since some of the names were specific to a particular language or region and would not appear in sufficient quantity for experimenting in all languages. However, the fact that the words share general categories makes the results somewhat comparable.

For each language also we found or manually constructed a stop-list of commonly used words, consisting mostly of function words such as articles, conjunctions, and so forth. The stop-lists were of comparable size, except for Bulgarian which was somewhat larger with 806 stop words. English had 426, Romanian 438, and Spanish 499. In fact, the stop-lists are not derived in a language independent way for these experiments, and represent the only language dependent part of the process. However, we believe that it will be possible to develop methods that derive stop-lists automatically. This remains an important area of future work.

Below we describe the names used in our experiments, and the corpora from which they were derived. We provide a brief description of each named entity. Note that the distribution of names prior to conflation is shown in Table 1.

4.1 Bulgarian

The Bulgarian experiments relied on the Sega2002 news corpus, which was originally prepared for the CLEF⁴ competition. This is a corpus of news articles from the Newspaper Sega⁵, which is based in Sofia, Bulgaria.

The version of the corpus used in our experiments was created with the help of the CLARK system⁶. Initially individual articles were found in different XML files depending on the year, month, and day of their publication. We merged these into a single file and only utilized the content between the text tags. The sentences that contained the names to be used in the experiments were extracted, and the Cyrillic characters were transliterated. Most Cyrillic characters are mapped one to one to the Latin alphabet, however several Cyrillic characters had to be represented by combination of two Latin symbols as the transliteration was phonetically based.

⁴ <http://www.clef-campaign.org>

⁵ <http://www.segabg.com>

⁶ <http://bultreebank.org/clark/>

The Bulgarian stop-list was taken from the resources distributed with the HPSG-based Syntactic Treebank of Bulgarian⁷.

Countries. Germaniya (Germany), Franciya (France), and Rusiya (Russia) are major European countries. Their occurrences were conflated into a single three way ambiguous name Fr-Ge-Ru.

Organizations. The organization names in this experiment are the abbreviations of the two leading political parties in Bulgaria. BSP (Balgarska Socialisticheska Partija, or Bulgarian Socialist Party) is the left leaning party and the successor to the Bulgarian Communist Party. It was formed in 1990 in post-communist Bulgaria. SDS (Saúz na demokratichnite sili, or The Union of Democratic Forces) is the right leaning political party. It was formed at the end of 1989 as a union of non-governmental organizations and reinvigorated old parties who had historically opposed the Communist government. These two names were conflated into a single ambiguity, BSP-SDS.

Cities. Varna and Burgas are the largest cities on the Bulgarian Black Sea Coast, and are the third and fourth largest cities overall in Bulgaria. Their names were combined into a single ambiguity, Va-Bu.

People. Ivan Kostov was Prime Minister of Bulgaria from May 1997 to June 2001 and leader of the Union of Democratic Forces (SDS, see above) between December 1994 and June 2001. Presently he is leader of the political party he formed, Democrats for a Strong Bulgaria. Petar Stoyanov was the President of Bulgaria from 1998 until 2002. He is now chairman of the Union of Democratic Forces (SDS, see above). Georgi Parvanov has been the President of Bulgaria since January 22, 2002. He is member of the Bulgarian Communist Party. The three names above were conflated to form the ambiguous name PS-IK-GP.

Nadejda Mihaylova is politician from the Democratic party and was Minister of Exterior from 1997 to 2001. Nikolay Vasilev is a politician from the National Movement Simeon II party. He was Vice-Premier and Minister of Economics during 2001, and Vice-Premier and Minister of Transport and Communications during 2003. Simeon Sakskoburggotski was the last King of Bulgaria, and was Prime Minister of Bulgaria from 2001 until August 2005. These three names were conflated to form the ambiguous name NM-NV-SS.

4.2 Romanian

The Romanian data was taken from the 2004 archives of the newspaper Adevarul (The Truth)⁸. This is a daily newspaper that is among the most popular in Romania. Named entities of interest were extracted via the grep command, and then any remaining html tags were removed. While Romanian typically contains diacritical markings, Adevarul does not publish their text with diacritics, so it was not necessary to account for them in processing.

⁷ <http://www.bultreebank.org/Resources.html>

⁸ <http://www.adevarulonline.ro/arhiva>

We initially used a stop-list created by Rada Mihalcea⁹, but observed that it was somewhat smaller than the stop-lists we were using for the other languages. It had approximately 250 entries, whereas the English and Spanish stop-lists had more than 400 entries, and Bulgarian approximately 800. Thus, we augmented the stop-list to make it more comparable with the other languages, so that the version we used in our experiments has 438 stop words.

The original Mihalcea stop-list followed pre-Revolution spelling conventions. For example, prior to 1989 verbs like *a mânca* (to eat) were spelled *mânca* (*minca* after removing diacritics) while now they are spelled *mânca* (*manca* after removing diacritics). Another example is the verb *to be* which, for first person, was spelled *sînt* (I am) while now it is spelled *sunt*. The words following post-Revolution conventions have been added to the list. Another source of new words was an online Romanian dictionary¹⁰, which offered all the inflected forms for pronouns. As a general remark, since Romanian is a language with a rich morphology, when adding a new word to the stop-list generally all the inflected forms have been added as well. Finally, the list was enriched also by translating words from the English stop-list, when appropriate.

Organizations. Partidul Democrat (PD) is the Romanian Democratic Party. For the 2004 elections they joined forces with the National Liberal Party to create the Justice and Truth (Dreptate si Adevar) political alliance, whose main purpose was to compete against PSD. They were successful in this election, and now hold power in Romania. The Partidul Social Democrat (PSD) is currently the main opposition party in Romania. These two names were conflated into the ambiguous name PD-PSD.

People. Traian Basescu is the current president of Romania, elected in 2004. His principal rival for the presidency was Adrian Nastase. Between 2000 and 2004 Basescu was the mayor of Bucharest. His political party is Partidul Democrat (PD, see above). Adrian Nastase is currently the President of Chamber of Deputies. In 2004 he competed for the presidential elections but he was defeated by Traian Basescu. He was Prime Minister between 2000 and 2004. He is a member of the Partidul Social Democrat (PSD) (see above). These names were conflated to create a two way ambiguity, TB-AN.

Ion Iliescu is the former Romanian president. He was president for 11 years, from 1990 to 1996 and from 2000 to 2004. Currently he is a senator for the PSD. This name was added to the two above to create a three way ambiguity, TB-II-AN.

Cities. Bucuresti (Bucarest) is the Romanian capital. It is the largest city in Romania, located in the southeast of the country. Brasov is a popular tourist destination in central Romania, located in the Carpathian Mountains of Transylvania. These two names were conflated into a single ambiguity Br-Bu.

⁹ http://nlp.cs.nyu.edu/GMA_files/resources/romanian.stoplist

¹⁰ <http://dexonline.ro/>

Countries. The country names included in the Romanian experiment include Romania, SUA (Statele Unite ale Americii, or the United States), and Franta (France). Their names were conflated into a single ambiguity, Fr-SUA-Ro.

4.3 English

The source of the English data was the English GigaWord Corpus, available from the Linguistic Data Consortium. In total this contains 1.7 billion words from four different news services. Our data was selected from either the 900 million word New York Times (nyt) portion or the 170 million word Agence France Presse English Service (afe) portion of the corpus. This text comes from the period 1994 through 2002.

Organizations. Microsoft is the world's largest software company. It was founded in 1975 by Bill Gates and Paul Allen. IBM is a large computer hardware and software company that has existed since 1888. These names were conflated into IBM-Mi.

Locations. There were three countries and one state included in these experiments. Mexico is the largest Spanish-speaking country in the world. It is located in North America, directly south of the United States. Uganda is a country in East Africa. While it is landlocked, it has access to Lake Victoria, the largest lake in Africa. These two country names were conflated into Me-Ug.

India is a South Asian country which is the second most populous in the world. California is the most populous state in the United States. It is on the west coast. Peru is a Spanish speaking country in western South America. These four names were conflated into Me-In-Ca-Pe.

People. Tony Blair is the current Prime Minister of England. He has held this office since 1997. He is the leader of the Labour Party. Bill Clinton was the 42nd President of the United States, and was in office from 1993 to 2001. Prior to serving as President, he was the Governor of Arkansas. He is a member of the Democratic Party. These two names were conflated into BC-TB.

Ehud Barak was the 10th Prime Minister of Israel, serving from 1999 to 2001. He was the leader of the Labor Party. This name was added to the two above to create the three way ambiguity, BC-TB-EB.

4.4 Spanish

The Spanish corpora comes from the Spanish news agency EFE from the year 1994. It contains a total of 215,738 documents. This collection was used in the Question Answering Track at CLEF-2003, and also for CLEF-2005.

A Named Entity Recognizer was used, and then the frequencies of entities was manually examined to determine the list of candidates for the experiment. The stop-list for Spanish was the same as used in the CLEF-2005 competition.¹¹

¹¹ <http://www.unine.ch/info/clef>

People. Yaser Arafat was the Chairman of the Palestine Liberation Organization from 1969 until his death in 2004. Bill Clinton is a former US president, as mentioned above. These two names were conflated to the ambiguous form YA-BC.

Juan Pablo II (John Paul II) was pope of the Roman Catholic Church from 1987 until his death in 2005. Boris Yeltsin was the President of Russia from 1991 to 1999. These were conflated to JP-BY.

Organizations. OTAN is the Spanish abbreviation for NATO, the North Atlantic Treaty Organization. This is an alliance between the United States, Canada and many European nations. EZLN is the Ejército Zapatista de Liberación Nacional, known in English as the Zapatista Army of National Liberation. It is based in Chiapas, Mexico and seeks to make revolutionary changes to Mexican society. These two names were conflated to form OTAN-EZLN.

Cities. Nueva York (New York) and Washington are major cities in the United States. Washington may also refer to a state on the West coast of the USA, so there is some ambiguity. These were conflated to form NY-Wa.

Brasil (Brazil) is the largest country in South America, both in terms of land mass and population. This was added to the names above to form the conflation NY-Br-Wa.

5 Experimental Results and Discussion

Our experimental results are summarized in Table 1. The conflated names are shown in the first column, and then the distribution of the instances for each underlying entity in the name are shown. For example, we can see that the conflated Bulgarian name Va-Bu occurred 2,501 times, and that 1,240 of these were the underlying entity Varna, and 1,261 were for Burgas.

Please note that the names are organized for each language such that the first two entries are for people, the third entry for organizations, and the last two are for locations.

In the third column the percentage of the instances that belong to the most frequent underlying entity are shown. This value is associated with a simple baseline clustering method that would simply assign all of the contexts to one cluster. Then columns 4 and 5 show the accuracy associated with the clustering without and with a stop-list. The number of contexts that are clustered correctly are shown next to the accuracy percentage. Finally, the last column shows the difference between the best result obtained for a name with the majority percentage.

Generally we can observe that nearly all of the experiments show a positive increase from the majority classifier. In nearly all cases the best results are shown when using a stop-list. Of the 20 conflated names, 4 show a significant increase above the majority class without using a stop-list, and 13 show an significant increase beyond the majority class with a stop-list.

Table 1. Experimental Results

Name	Distribution	Majority	No Stop-list	With Stop-list	Diff.
Bulgarian:					
PS-IK-GP	318+524+811=1653	49.06%	40.53% 670	58.68% 970	+9.62
NM-NV-SS	645+849+976=2470	39.51%	35.79% 884	59.39% 1467	+19.88
BSP-SDS	2921+4680=7601	61.57%	51.97% 3950	57.31% 4356	-4.26
Fr-Ge-Ru	1726+2095+2645=6466	40.91%	39.07% 2526	41.60% 2690	+0.69
Va-Bu	1240+1261=2501	50.42%	66.09% 1653	50.38% 1260	+15.71
English:					
BC-TB	1900+1900=3800	50.00%	80.89% 3074	80.95% 3076	+30.95
BC-TB-EB	1900+1900+1900=5700	33.33%	46.68% 2661	47.93% 2732	+14.60
IBM-Mi	2406+3401=5807	58.57%	50.59% 2938	63.70% 3699	+5.13
Me-Ug	1256+1256=2512	50.00%	50.76% 1275	59.16% 1486	+9.16
Me-In-Ca-Pe	1500+1500+1500+1500=6000	25.00%	28.75% 1725	28.78% 1727	+3.78
Romanian:					
TB-AN	1804+1932=3736	51.34%	50.59% 1890	51.34% 1918	+0.00
TB-II-AN	1948+1966+2301=6215	37.02%	34.16% 2123	39.31% 2443	+2.29
PD-PSD	2037+3264=5301	61.57%	52.08% 2761	77.70% 4119	+16.13
Br-Bu	2310+2559=4869	52.56%	51.22% 2494	63.67% 3100	+11.11
Fr-SUA-Ro	1370+2396+3890=7656	50.81%	40.73% 3118	52.66% 4032	+1.85
Spanish:					
YA-BC	1004+2340=3344	69.98%	50.24% 1680	77.72% 2599	+7.74
JP-BY	1447-1450=2897	50.05%	63.62% 1843	87.75% 2897	+37.70
OTAN-EZLN	1093+1093=2186	50.00%	50.09% 1095	69.81% 1526	+19.81
NY-Wa	1517+2418=3935	61.45%	54.69% 2152	54.66% 2151	-6.76
NY-Br-Wa	1517+1748+2418=5683	42.55%	39.24% 2230	42.88% 2437	+0.33

In addition, in nearly all cases the use of a stop-list either results in better or equally good accuracy as when not using a stop-list. The only exception to this is the Bulgarian name Va-Bu, which has two underlying entities, the cities of Varna and Burgas. This is the only case where the use of a stop-list has actually hurt performance rather badly. However, we also note that the location names in general seem to show relatively little improvement with stop-lists in at least one of the two cases for all of the languages.

We theorize that location names can be used in a wide range of contexts, and that it is harder to find discriminating features for them. However, locations also have many unique names associated with them, so it is still unclear to us why the location names often pose the most significant challenges for this approach.

6 Cluster Label Examples

In addition to grouping the contexts into clusters that reflect the underlying or true entities, we generate a label for each cluster based on its content. This is intended to act as a simple identifier for the underlying entity.

The top ten bigrams found in the contexts in a cluster that do not include stop words and have a log-likelihood score above 3.841 are chosen as the descriptive labels, regardless of how many other clusters they may occur in. The discriminating labels are the top ten bigrams that are unique to a cluster, according to

Table 2. Cluster Label Examples

True Entity	Created Labels
Bulgarian (2 political parties)	
BSP	Nikolay Mladenov, liderat Sergey , Visshiya savet, liderat Stanishev , vot nedoverie, Ekaterina Mihaylova, Sergey Stanishev, G n, Ivan Kostov, Nadejda Mihaylova
SDS	mestnite izbori, d r, Rakovski 134 , politicheska sila, vot nedoverie, Ekaterina Mihaylova, Sergey Stanishev, G n, Ivan Kostov, Nadejda Mihaylova
English (2 companies)	
IBM	5 8, BW GEN, 3 4 , interest rates, Texas Instruments, Hewlett Packard, 30 Treasury, 7 8 , Wall Street, billion dollars
MICROSOFT	vice president, million dollars, Windows 95 , operating system United States, Bill Gates, Justice Department, personal computers , Wall Street, billion dollars
Romanian (2 political parties)	
PD	Popescu Tariceanu, Theodor Stolojan, Alianta PNL, Calin Tariceanu , Camera Deputatilor, PNL PD, Adrian Nastase, Traian Basescu
PSD	Camera Uniunea, Deputatilor Uniunea, partidul guvernament, Cozmin Gusa, Ion Iliescu, Emil Boc , Camera Deputatilor, PNL PD, Adrian Nastase, Traian Basescu
Spanish (2 leaders)	
BILL CLINTON	presidente estadounidense, EFE presidente, presidente OLP, Casa Blanca, Washington EFE, presidente Unidos
YASER ARAFAT	Exteriores Peres, ministro israeli, Palestina OLP, Gaza Jerico, Hafez Asad, Isaac Rabin, proceso paz, Asuntos Exteriores

these same criteria. In the cluster labels we allow the words that form a bigram to be separated by up to three intervening words.

As yet we do not have a reliable means of evaluating these labels, so we simply show examples of the labels found for each language in Table 2. For each language we show a two sense distinction, where the true underlying entity for a cluster is on the left, and the automatically generated labels are on the right. The descriptive labels are shown in normal text, and labels that are both discriminating and descriptive are in bold. Note that descriptive labels may be shared by the two clusters, and can be thought of as providing some indication of the general topic or subject that pertains to both clusters. The discriminating labels are meant to distinguish between the different clusters.

In these examples there is no discriminating label that is not a descriptive label as well. This simply indicates that all of the discriminating labels occurred in the top ten bigrams overall.

For Bulgarian and Romanian, we show the cases where two political parties are discriminated. The labels consist mainly of names, and in general these names are commonly understood to be associated with the party mentioned.

In Bulgarian the BSP cluster shows discriminating labels that include *liderat Sergey* and *liderat Stanishev*, which is quite reasonable since *liderat* means *leader*, and *Sergey Stanishev* is the leader of the Socialist Party in Bulgaria (BSP). Also note that he appears as a descriptive label for SDS. This can be understood by pointing out that the leader of an opposing party could well be mentioned in contexts that are about the SDS. It is encouraging to note that references to him as *leader* were unique to the BSP cluster.

In Romanian, the PSD cluster includes *Ion Iliescu* and *Cozmin Gusa* as discriminating features, both who are members of the PSD. The PSD cluster also has *partidul guvernament* as a discriminating feature, which means *government party*, which describes the PSD in 2004. The PD cluster includes discriminating labels *Popescu Tariceanu*, *Theodor Stolojan*, and *Calin Tariceanu*, who are all members of the Liberal Party, which formed an alliance with the PD. And in fact that alliance has been included as a discriminating label via *Alianta PNL*. Note that the full name of the alliance is *Alianta PNL PD*, but since we rely on bigrams this has been split into two (where *PNL PD* is included as a descriptive label of PD).

In English we show the labels for the clusters associated with IBM and Microsoft. We note that these labels are somewhat noisier than those of the political parties. For example, there are a number of unusual looking pairs of numbers in the IBM cluster. However, these are the result of a tokenization scheme that simply removed non-alphanumeric characters (e.g., so *3/8* become *3 8*). These fractions refer to movements in the stock price. The companies *Texas Instruments* and *Hewlett Packard* are shown as discriminating labels for IBM, and may reflect the fact that these companies are often mentioned together when discussing stock market activity.

The Microsoft cluster has a discriminating label *Bill Gates*, who is the co-founder of the company. It also includes *Justice Department* as a discriminating label, which is appropriate given the great attention paid to the legal case against Microsoft. The discriminating labels *Windows 95*, *operating system*, and *personal computers* are certainly useful in identifying Microsoft, whereas those for *vice president* and *million dollars* are less so.

The inclusion of *Wall Street* and *billion dollars* as descriptive labels for IBM and Microsoft suggests that these are companies that are traded on the stock market (which is a reasonable description) but does not offer any unique discriminating information about either company.

In the Spanish data, all of the labels shown are both descriptive and discriminating, meaning that the top ten bigrams in each cluster were unique to that cluster. The labels for Bill Clinton include *presidente estadounidense*, which translates as *President of the United States*, and *Casa Blanca*, which is the White House. It also has a certain amount of noise, for example various labels that mention EFE, which is a Spanish news agency and in fact the source of this corpora. We believe that this is due to the presence of datelines in the contexts, as in *Washington, Jan 2 (EFE) - President Clinton said ...*

The labels for Yaser Arafat include several that are quite discriminating, including *proceso paz* (*peace process*), and *Palestina OLP*, which refers to the Palestinian Liberation Organization (OLP in Spanish). However, the cluster for Bill Clinton also includes *presidente OLP*, due to his frequent meetings with Arafat during this time. *Hafez Asad* was the president of Syria, and *Isaac Rabin* was the Prime Minister of Israel (known as *Yitzhak* in English).

In general we can see that these labels provide relevant and useful information about the underlying entities, but that they are somewhat noisy and perhaps

not obvious indicators of that entity. Please note that the descriptive labels are not intended to uniquely describe the cluster, but rather to give an overall gist of what the cluster is about, while the discriminating labels are those that are meant to provide the unique information about an underlying identity.

7 Related Work

Bagga and Baldwin [1] propose a method for resolving cross document references (such as recognizing that John Smith and Mr. Smith refer to the same person) 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 [4] 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 [6] 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 then 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.

Name disambiguation is also a problem in the medical domain. For example, Hatzivassiloglou, et. al. [5] 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. [3] develop an algorithm for disambiguation of protein names based on weighted features vectors derived from surface lexical features and achieve equally good results.

8 Future Work

There are two language dependent aspects to this method. The first is that it does assume that the words in the language have been segmented. In the case of the languages used in this study, we have simply assumed words to be alphabetic strings that are white space separated. However, in some languages segmentation is a more difficult issue, and that would need to be resolved before this method was applied.

Second, we have utilized pre-existing or manually derived stop-lists, which introduces a language dependence on our method. We are confident that we can develop a language independent method of finding stop words in the corpora we are clustering. Some variant of term frequency/inverse document frequency (TF/IDF) might be appropriate, or we could simply identify those words that occur in a majority of all contexts and consider those as stop words.

9 Conclusions

The experiments and results in this paper show that our hypothesis that these methods are language independent has some validity. Results well in excess of the majority class baseline are obtained for four different languages using exactly the same methodology. The fact that these methods are completely unsupervised and yet they could be successfully applied to the discrimination problem from different domains like politics, geographical locations, and organizations also suggests that the methods are also domain-independent.

Acknowledgments

Ted Pedersen and Anagha Kulkarni are supported by a National Science Foundation Faculty Early CAREER Development Award (#0092784).

All of the experiments in this paper were carried out with version 0.71 of the SenseClusters package, freely available from <http://senseclusters.sourceforge.net>.

All of the data and stop-lists for the four languages used in these experiments are available at <http://www.d.umn.edu/~tpederse/pubs.html>.

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. T. Gaustad. Statistical corpus-based word sense disambiguation: Pseudowords vs. real ambiguous words. In *Proceedings of the Student Research Workshop at ACL-2001*, pages 61–66, Toulouse, France, 2001.
3. 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.

4. 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.
5. 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.
6. 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.
7. 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.
8. T. Pedersen, A. Purandare, and A. Kulkarni. Name discrimination by clustering similar contexts. In *Proceedings of the Sixth International Conference on Intelligent Text Processing and Computational Linguistics*, pages 220–231, Mexico City, February 2005.
9. A. Purandare. Discriminating among word senses using McQuitty’s similarity analysis. In *Proceedings of the Student Research Workshop at HLT-NAACL 2003*, pages 19–24, Edmonton, Alberta, Canada, May 27 - June 1 2003.
10. 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.
11. H. Schütze. Automatic word sense discrimination. *Computational Linguistics*, 24(1):97–123, 1998.

Extracting Key Phrases to Disambiguate Personal Names on the Web

Danushka Bollegala¹, Yutaka Matsuo², and Mitsuru Ishizuka¹

¹ University of Tokyo

{danushka, ishizuka}@miv.t.u-tokyo.ac.jp

² AIST

y.matsuo@carc.aist.go.jp

Abstract. When you search for information regarding a particular person on the web, a search engine returns many pages. Some of these pages may be for people with the same name. How can we disambiguate these different people with the same name? This paper presents an unsupervised algorithm which produces key phrases for the different people with the same name. These key phrases could be used to further narrow down the search, leading to more person specific unambiguous information. The algorithm we propose does not require any biographical or social information regarding the person. Although there are some previous work in personal name disambiguation on the web, to our knowledge, this is the first attempt to extract key phrases to disambiguate the different persons with the same name. To evaluate our algorithm, we collected and hand labeled a dataset of over 1000 Web pages retrieved from Google using personal name queries. Our experimental results shows an improvement over the existing methods for namesake disambiguation.

1 Introduction

The Internet has grown into a collection of billions of web pages. One of the most important interfaces to this vast information are web search engines. We send simple text queries to search engines and retrieve web pages. However, due to the ambiguities in the queries and the documents, search engines return lots of irrelevant pages. In the case of personal names, we may receive web pages to other people with the same name (*namesakes*). However, the the different namesakes appear in quite different contexts. For example if we search for *Michael Jackson* in Google, among the top hundred hits we get a beer expert and a gun dealer along with the famous singer. However, the context in which the singer appears is quite different from his namesakes. However, context associated with a personal name is difficult to identify. In cases where the entire web page is about the person under consideration, the context could be the complete page. On the other hand the context could be few sentences having the specified name. In this paper we explore a method which uses terms extracted from web pages to represent the context of namesakes. For example, in the case of Michael Jackson, terms such as *music*, *album*, *trial* associate with the famous singer, whereas we

get *beer*, *travel*, *hunter* as terms for the other (beer expert) namesake of Michael Jackson. These term sets appear to be defining different contexts. We could use this difference in context to discriminate the namesakes.

Disambiguating namesakes on the Web is a difficult task due to the diversity of web pages. We do not know in advance the exact number of namesakes for a name on the Web. In many cases there are two or three famous namesakes which have lots of pages regarding them and all other namesakes have just one or two pages on them. Some of the web pages are not exclusively about a person, but just mention the name on passing (ex: book reviews on Amazon mentioning an author of a book, conference programs mentioning names of the authors of papers, etc). This paper presents an unsupervised clustering framework, which uses a robust similarity metric to overcome these difficulties.

On the other hand there are cases where an individual has various web appearances. For example the renowned linguist Noam Chomsky appears as a linguist and also as a critic of American foreign policy. It would be interesting to see how a namesake disambiguation method responds to such complications. In Chomsky's example one would like to extract terms such as *Generative Grammar*, *Linguistic Theory*, *Transformational Grammar*, etc from pages which describe Chomsky's linguistic work whereas *American foreign policy*, *Iraq*, *critic*, etc from pages which describe Chomsky's political views. Although our main focus is on disambiguating people with one specific web appearance, we also explore the possibilities of our algorithm to identify the different web appearances of individuals.

This paper is structured as follows. First we give an overview of the related work in this area. Then we explain the different components in our system. Namely; term extraction, similarity calculation, clustering, determining the number of namesakes and term ranking. Finally we show experimental results for the proposed method and conclude this paper.

2 Related Work and Problem Setting

There is little previous work we know of that directly addresses the problem of extracting key phrases to disambiguate personal names on the web, but some related problems have been studied. Disambiguation of namesakes is similar to tuple matching in databases—the problem of deciding whether multiple relational tuples from heterogeneous sources refer to the same real-world entity [7, 1].

From a natural language perspective, there has been a lot of work on the related problem of co-reference resolution [2, 11]. The goal in co-reference resolution is to link occurrences of noun phrases and pronouns, typically occurring in a close proximity, within a few sentences or a paragraph, based on their appearance and local context. Co-reference resolution is vital for many natural language tasks such as text summarization and question answering. Various algorithms have been proposed for co-reference resolution. Fundamentally, these algorithms map the local information around a pronoun to a set of features and use a machine learning technique to determine whether a given pronoun corresponds to a given noun phrase.

A few works address the problem of personal name disambiguation across a collection of documents. Mann, et al [10] considers the problem of distinguishing occurrences of a personal name in different documents. They propose an unsupervised algorithm which extracts *people-specific* biographical information such as birth date, birth place, occupation etc using a set of regular expressions to cluster the documents to their namesakes. However, such person-specific information is not always available for all the namesakes on the web. Even in cases where such information is available, a set of fixed regular expressions as used by Mann et al [10] is not sufficient to extract them. Bekkerman, et al [4] proposes a link structure model and an agglomerative-conglomerative double clustering (A/CDC) based algorithm to disambiguate a group of people on the web. The algorithm assumes our ability to obtain information regarding the social network (associates) of the person to be disambiguated. The method can be readily used when we have such information. However, in most of the situations we do not know well enough about the associates of the person which we want to disambiguate. Pedersen et. al. [13] proposes a method for discriminating names by clustering the instances of a given name into groups. They extract the context of each instance of the ambiguous name and generate second order context vectors using significant bigrams. The vectors are clustered such that instances that are similar to each other are placed into same clusters. Li, et al [9] suggests an algorithm which could be used to disambiguate not only personal names but other named entities such as organizations and locations. They propose a discriminative model based on agglomerative clustering and a generative model which uses a language model combined with EM algorithm. Their experimental results show that the generative model out performs the discriminative model. However, they do not discuss the topic of extracting key phrases to distinguish the different entities. In this paper we try to extract key phrases to distinguish each of the different namesakes in our document collection. In this paper we will assume that each document in the collection represents only one of the namesakes (i.e. no document covers two or more namesakes). We will first cluster the set of documents and then select key phrases from these clusters to distinguish the different namesakes.

3 Method

The outline of our method is illustrated in in figure 1. Our method takes the name to be disambiguated as the input and outputs a list of key phrases for each of the different namesakes. As shown in figure 1, the algorithm we propose for this task consists of eight steps. We first introduce each of the steps in our method and details are left for the sections to follow.

First we send the name to be disambiguated to a web search engine and download a set of web pages. We used Google ¹ and download the top 100 pages for the given name. These pages will be processed in the next steps in our method. Downloaded pages are not required to be exclusively on a certain

¹ <http://www.google.com/apis>

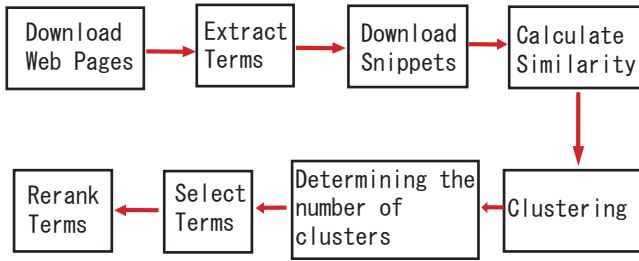


Fig. 1. Outline of the method

person. However, we assume one page to be associated with only one of the namesakes. We extract a set of terms from each one of the pages in our document collection (which was downloaded in the previous step). The term extraction algorithm we use for this task is explained in section 3.1. We then cluster the document collection based on the terms we extracted. To cluster documents we define a pairwise similarity measure. We use *Snippet Similarity* to measure the similarity between two terms. Section 3.2 explains snippet similarity. We utilize an agglomerative clustering method to cluster the document collection as described in section 3.3. Ideally, the clusters yielded by the clustering algorithm should represent a different namesake. However, in reality we do not know the exact number of namesakes for a given name in advance. Therefore, we define a measure which we will call *Cluster Quality* in this paper based on the internal and external correlation of the clusters, and decide the number of clusters. Finally, we select representative terms from each of the clusters and rank them according to their relevance to the name under consideration.

3.1 Term Extraction

Our method is based on the fact that different namesakes appear under different contexts on the web. We assume that each document in our downloaded web page collection represents some namesake of the given name. **Contextual Hypothesis for Senses** [15] states that two occurrences of an ambiguous word belong to the same sense to the extent that their contextual representations are similar. According to this hypothesis, if two pages are similar in context, then we could assume that these pages are likely to be on the same namesake. However, a document may not totally focus on the namesake, but also contain lots of irrelevant information. Therefore, we need to represent the documents in a model that captures the essence of the document and ignores the irrelevant facts. We use *C-value* [5, 6], an automatic term recognition algorithm, to extract multi-word terms from the documents and represent each document by the set of terms extracted from it.

The *C-value* approach combines linguistic and statistical information, emphasis being placed on the statistical part. The linguistic information consists of the part-of-speech tagging of the document being processed, the linguistic filter

constraining the type of terms extracted and the stop lists. The statistical part combines statistical features of the candidate string. The linguistic filter contains a predefined set of patterns of nouns, adjectives and prepositions that are likely to be terms. The stop list is a list of words which are not expected to occur as term words in a given domain. Having a stop list improves the precision. However, in our experiments we did not use a stop list because it is not possible to determine in advance the domains which a namesake belongs to.

The combinations of nouns, adjectives and prepositions that are allowed by the linguistic filter and the stop list are considered as the potential candidates as terms. The *termhood* (likeliness of a candidate to be a term) is evaluated using C-value. C-value is built using statistical characteristics of the candidate string, such as, the total frequency of occurrence of the candidate string in the document, the frequency of the candidate string as part of other longer candidate strings, the number of these longer candidate terms and the length of the candidate string (in number of words). C-value is defined as follows,

$$C - value(a) = \begin{cases} \log_2 |a| \cdot f(a) & a \text{ is not nested,} \\ \log_2 |a| (f(a) - \frac{1}{P(T_a)} \sum_{b \in T_a} f(b)) & \text{otherwise} \end{cases} . \quad (1)$$

where, a is the candidate string, $f(a)$ is its frequency of occurrence in the document, $|a|$ is the length of the candidate string, T_a is the set of extracted candidate terms that contain a , $P(T_a)$ is the number of these candidate terms.

We prefer the candidate terms with higher c-values to terms with lower c-values. However, there are cases where the terms extracted from Frantzi's [6] c-value method tend to be exceedingly longer and meaningless. For example, we get a term *Search Archives Contact Us Table Talk Ad* from a page about the netscape founder, Jim Clark. This term is a combination of words extracted from a navigation menu and not a genuine term. Using such terms to represent the context of a namesake cannot be acceptable. To avoid such terms we use two heuristics. First we ignore any term which is longer than four words. Then, for the remaining terms, we check the number of hits we get for the term in a web search engine. Our assumption here is if a term is a meaningful one it is likely to be used in many web pages. We ignore any terms with less than five hits. Using these heuristics does not only allow us to extract more expressive and genuine terms but also prevents data sparseness when calculating snippet based similarity between terms as explained in the next section.

3.2 Similarity Calculation

Exact matches of terms extracted from different documents are rare. Therefore, we would require a similarity metric which is capable of comparing the terms at a semantic level. For example, the two terms *George Bush* and *The president of the United States* are closely related but do not have any words in common. Word Net ² based similarity metrics have been widely used as semantic similarity measures between words in sense disambiguating tasks [12, 3]. However, the

² <http://wordnet.princeton.edu/perl/webwn>

<i>“George Bush”</i>	<i>“The president of the United States”</i>
<ul style="list-style-type: none"> (1) Official White House site presents issue positions, news, Cabinet, appointments, offices and major speeches. Includes biography, video tour and photo ... (2) Official Internet home of the Republican National Committee. Updated daily with news and commentary from the RNC. (3) George Bush (41st President: 1988–1992). (4) Zack Exley’s satirical site of George W. Bush campaign, now quite overt. (5) Parody of official White House web site. Includes spoof news and gossip. 	<ul style="list-style-type: none"> (1) Whitehouse.gov is the official web site for the White House and President George W. Bush, the 43rd President of the United States. (2) Background information, election results, cabinet members, notable events, and some points of interest on each of the presidents. (3) For a list of persons who served as the President of the United States following the ratification of the United States Constitution see the list of ... (4) A history of presidents, the presidency, politics and related subjects. Includes biographies for every president. (5) ... Presidents of the Continental Congress as well as information about David Rice Atchison who some believe was the 12th President of the United States. ...

Fig. 2. Top five snippets extracted for two terms

major problem with such approaches is the low coverage of words. For example, we would not find proper nouns such as *George Bush* in WordNet. However, such proper nouns (specially human names) are useful to disambiguate namesakes (see section 4). We use the World Wide Web as our knowledge source and define similarity between terms using web snippets. Mehra [14] proposes a method to calculate similarity between words (also can be used with terms) using snippets retrieved by a web search engine. A Snippet is a small piece of text, containing two or three sentences extracted from the document around the query term. Most web search engines provide snippets along with the links to the source pages. A user can read the snippet and decide whether the linked page is relevant to the query, thereby avoiding the time to download and read the complete page. The snippet gives the context in which the searched term appear in the page. We use Google as our web search engine and extract the top 100 snippets for each term we extract.

For example, consider the first five snippets we get for *George Bush* and *The president of the United States* shown in figure 2. Even among the first five snippets for these two terms, we find many common words such as *President, White House, Official, and, site*, etc. However, some of these words (ex: and, of) have a purely grammatical functionality and do not carry any semantic information regarding the searched terms. We use a predefined list of stop words and remove such words from the snippets. We then merge the top hundred snippets together (here on, we will call this merged result as the snippet text) and compare the distribution of words to calculate the similarity between the terms. In order to calculate the word distribution we count the frequency of each word in the

snippet text. We divide the frequency counts by the total number of words to convert the frequency distribution into a probability distribution. These normalized word distributions, calculated using snippets retrieved for different terms, are compared using Kullback-Liebler (KL) divergence. KL-divergence is a popular metric used in measuring the distance between two probability distributions. For two probability distributions $p(x)$ and $q(x)$, which are defined over a random variable $x \in X$, their KL-divergence $D(p||q)$ is defined as follows,

$$D(p||q) = \sum_{x \in X} p(x) \log \frac{p(x)}{q(x)}. \quad (2)$$

Where, X is the set of values that random variable x takes. Since we are concerned on word distributions, X is the vocabulary of words used in the snippets. However, due to the sparseness of data, some words may not appear in both distributions. KL-divergence becomes undefined when there are words with zero probabilities. Skew divergence is used to overcome this problem [8]. Skew divergence $S_\alpha(p, q)$ is defined as follows,

$$S_\alpha(p, q) = D(q||\alpha p + (1 - \alpha)q). \quad (3)$$

Therein: $\alpha \in [0, 1]$ is the degree of skewness between the two distributions p and q . It has been shown that ([8]) skew divergence best expresses the divergence between two distributions when the value of α is closer to 1. In our experiments we set $\alpha = 0.99$.

However, both KL-divergence and skew divergence are not symmetric and does not satisfy the properties of distance metrics. We define a distance function $d(p, q)$ by considering the skew divergence on both ways. Thus, the distance $d(p, q)$ between two distributions p, q is defined as follows,

$$d(p, q) = \frac{1}{2}(s_\alpha(p, q) + s_\alpha(q, p)). \quad (4)$$

We further convert the distance values given by equation 4 to similarity values $\text{sim}(p, q)$ by taking their negative exponential values as follows,

$$\text{sim}(p, q) = \exp(-d(p, q)). \quad (5)$$

Equation 5 defines the similarity between two terms using the probability distributions calculated for each of the terms.

However, to cluster the documents, we need a pairwise similarity measure which is defined upon the documents. For this we extend the similarity function defined by equation 5 to two documents. We take the average of similarity for all the pairs of terms extracted by the two documents. The similarity, $\text{DocSim}(A, B)$, between two documents A and B is defined as follows,

$$\text{DocSim}(A, B) = \frac{1}{|A||B|} \sum_{(a,b) \in (A \times B)} \text{sim}(a, b). \quad (6)$$

Therein: A and B are the sets of terms extracted from the corresponding documents. $|A|$ denotes the cardinality of the set A . $A \times B$ gives the set of pairs taken from the two sets. $a \in A$ and $b \in B$ are terms in the sets.

3.3 Clustering

Having defined a similarity metric in section 3.2, our next task is to cluster the documents using this similarity metric. In this paper, we use group-average agglomerative clustering (GAAC) as our clustering algorithm, a hybrid of single-link and complete-link clustering. We begin by assigning a separate cluster for each document in the collection. GAAC in each iteration executes the merger that gives rise to the cluster Γ with the largest average correlation $C(\Gamma)$ where,

$$C(\Gamma) = \begin{cases} 1 & |\Gamma| = 1, \\ \frac{1}{2} \frac{1}{|\Gamma|(|\Gamma|-1)} \sum_{u \in \Gamma} \sum_{v \in \Gamma} \text{DocSim}(u, v) & \text{otherwise.} \end{cases} \quad (7)$$

Therein: $|\Gamma|$ denotes the number of documents in the merged cluster Γ ; u and v are two documents in Γ and $\text{DocSim}(u, v)$ is given by equation 6. Ideally, the clustering process should terminate when there is exactly one cluster representing each of the namesakes in the collection. However, in practice, the exact number of different namesakes that exist in the collection is not known. Therefore, we define a measure which we will call *Cluster Quality* in section 3.4 and terminate the above mentioned GAAC process when the cluster quality falls below a predefined threshold. In section 4 we show empirical evidence to the fact that the cluster quality measure approximates well the disambiguation accuracy and stops the clustering when there are sufficient clusters to represent the different namesakes.

3.4 Cluster Quality

Clustering in general can be considered as an optimizing problem. In clustering we try to;

1. maximize the similarity of items (documents) within a cluster,
2. minimize the similarity of items (documents) between clusters.

We prefer our clusters to be well correlated internally and each of the clusters to be different among themselves. The quality (goodness) of the formed clusters can be evaluated based on how well the clusters satisfy these two conditions. We define *internal correlation* as a measure of how well the first condition is satisfied (i.e. the degree of similarity of documents within clusters). Internal correlation, $\text{IntCor}(A)$, of a set A of n clusters c_1, c_2, \dots, c_n is defined as follows,

$$\text{IntCor}(A) = \frac{1}{n} \sum_{\Gamma \in A} C(\Gamma). \quad (8)$$

Where, $C(\Gamma)$ is the average correlation defined in equation 7.

We define *external correlation* as a measure of how well the second condition is satisfied (i.e. the degree of dis-similarity between clusters). Using the above notation, external correlation, $\text{ExtCor}(A)$, is defined as the dis-similarity between the two most similar clusters in A as follows,

$$\text{ExtCor}(A) = 1 - \frac{1}{|A_a||A_b|} \sum_{u \in A_a} \sum_{v \in A_b} \text{DocSim}(u, v). \quad (9)$$

Where,

$$(\Gamma_a, \Gamma_b) = \arg_{\Gamma_i, \Gamma_j \in \Lambda} \min C(\Gamma_i \oplus \Gamma_j) \quad (10)$$

and the operator \oplus denotes the merging operation between two clusters. Using equations 8 and 9 we define *Cluster Quality*, $Q(\Lambda)$, as follows,

$$Q(\Lambda) = \frac{1}{2}(\text{IntCor}(\Lambda) + \text{ExtCor}(\Lambda)). \quad (11)$$

We terminate GAAC when cluster quality drops below a predefined threshold and assign the remaining documents to the already formed clusters.

3.5 Term Selection and Ranking

GAAC (section 3.3) creates clusters for different namesakes. The next step is to select a set of terms from these clusters that describes each namesake. We select all the terms that appear in a cluster for a certain namesake but does not appear in other clusters. We then rank these terms by the relevancy of the term to the ambiguous name. We use snippets based similarity measure described in section 3.2 to calculate relevancy.

4 Results and Discussion

4.1 Test Data

We evaluated our algorithm on pseudo names as well as naturally ambiguous names. For automated pseudo name evaluation purposes, we collected 50 documents from the web for three different people for conflation. Our collection contains documents for *Maria Sharapova* the tennis player, *Bill Gates* chairman and chief software architect of Microsoft corporation and *Bill Clinton* former president of the United States. We then replace every occurrence of these names in the documents with *person-x*.

To evaluate our algorithm on naturally ambiguous names we selected names that appear in previous work([10, 4]) in this field, such as *Jim Clark*, *William Cohen*, *Tom Mitchell*, *Michael Jackson*. To evaluate our algorithm on people with different web appearances we tested for *Noam Chomsky*.

4.2 Disambiguation Accuracy

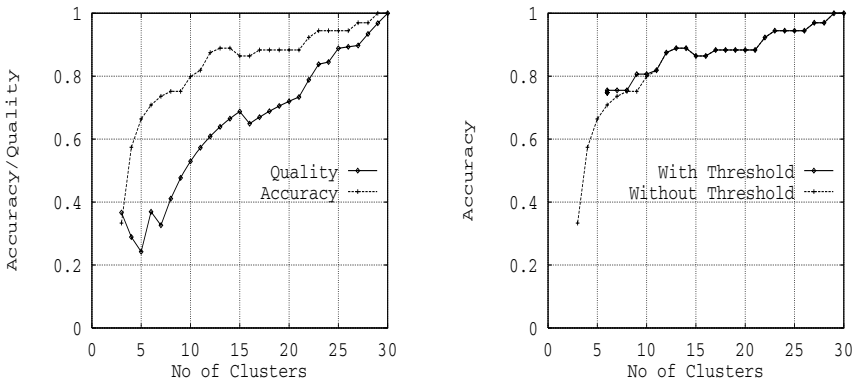
To evaluate the clusters produced by the proposed algorithm, we first assign each cluster to the namesake that appears most (we will call this namesake the *holder* of the cluster) in that cluster. We then count the number of documents in the collection for each of the different namesakes. Precision, $P(C)$, of cluster C is calculated as follows;

$$P(C) = \frac{\text{Number of documents in } C, \text{ representing the holder of } C}{\text{Total number of documents in } C}. \quad (12)$$

However, the distribution of documents for each of the different namesakes in the collection is not even. Some namesakes have lots of pages representing them, where as for some namesakes they are mentioned only in one or two documents. To reflect this fact in our evaluation metric we define *Disambiguation Accuracy*, as the weighted sum of each cluster’s precision. Accuracy (disambiguation accuracy) is defined as follows,

$$\text{Accuracy} = \sum_{C \in \mathcal{A}} P(C) \frac{\text{Number of documents in the collection for the holder of } C}{\text{Total number of documents in the collection}}. \tag{13}$$

Where, \mathcal{A} is the set of clusters and $P(C)$ is given by equation 12.



(a) Accuracy/Quality Vs No of Clusters (b) Accuracy Vs No of Clusters, with and without Quality Threshold

Fig. 3. Effect of the quality threshold

Figure 3(a) depicts the accuracy/quality vs the number of clusters in the experiment with pseudo names. From figure 3(a) it can be seen that when we do not impose a threshold on quality, there exists a steep drop in accuracy when ten clusters are formed. This is due to the outliers that get attached to the otherwise pure (representing the dominant namesakes) clusters. The value of quality when this happens is 0.6. Although the number of clusters when this happens is different for different names, our experiments show that the value of quality is around 0.6 in all the experiments. Therefore, we set the threshold of quality to 0.6. When the threshold is imposed, accuracy does not drop as it can be seen from figure 3(b).

Table 1 shows results of our experiments. We implemented a TF-IDF based similarity metric and compared our algorithm against it. In the TF-IDF based method, we consider all the words in each document (except stop words) and represent documents with TF-IDF weighted vectors. Then we take the cosine similarity between the vectors as the similarity measure in the group average

Table 1. Accuracy for ambiguous names

Name	Number of namesakes	Proposed	TF IDF
Jim Clark	8	71.95	59.20
Michael Jackson	3	94.96	88.76
William Cohen	10	72.71	57.96
person-X	3	81.10	39.88
Noam Chomsky	2	94.19	82.79

agglomerative clustering in section 3.3. However, note that the TF-IDF based method does not produce any key phrases. Table 1 reports higher accuracy values for the proposed method compared to this TF-IDF based method.

Our algorithm finds key phrases such as *racing driver Jim Clark, Formula One World Championships and motor racing* for the racing car driver-Jim Clark and *Silicon Valley, netscape* for the founder of netscape -Jim Clark. In the case of Michael Jackson, the top ranking terms for the singer are *Fan Club, World network, news, Micheal Jackson case and pop star*. The proposed method had the lowest accuracy for william cohen as it found only three out of the ten namesakes in the collection. In the person-X experiments, we find key phrases such as *first set, US open, Wimbledon, Venus Williams and Grand Slam title* for Maria Sharapova, *wealthiest person, Microsoft* for Bill Gates and *White house, former president* for Bill Gates. Although, Noam Chomsky is not an ambiguous name, we tested on it to evaluate the algorithm on individuals with different web appearances. Interestingly, the algorithm ranks key phrases such as *preventive war, government complicity, George Bush, Tony Blair* in the Chomsky the critic cluster and *universal grammar, linguistic theory* in the Chomsky the linguist cluster.

5 Conclusion

We proposed and evaluated an algorithm to extract key phrases from the web, to disambiguate personal names. In future, we intend to explore the possibilities to extend the proposed method to disambiguate other types of entities such as location names and organization names.

References

1. P. Andritsos, R. Miller, and P. Tsapars. Information-theoretic tools for mining database structure from large data sets. In *Proceedings of the ACM SIGMOD Conference*, 2004.
2. A. Bagga and B. Baldwin. Entity-based cross-document coreferencing using the vector space model. In *Proceedings of COLING*, pages 79–85, 1998.
3. S. Banerjee and T. Pedersen. An adapted lesk algorithm for word sense disambiguation using word net. In *Proceedings of the third international conference on computational linguistics and intelligent text processing*, pages 136–145, 2002.

4. R. Bekkerman and A. McCallum. Disambiguating web appearances of people in a social network. In *Proceedings of the 14th international conference on World Wide Web*, pages 463–470, 2005.
5. K. Frantzi and S. Ananiadou. Extracting nested collocations. In *16th Conference on Computational Linguistics*, pages 41–46, 1996.
6. K. Frantzi and S. Ananiadou. The c-value/nc-value domain independent method for multi-word term extraction. *Journal of Natural Language Processing*, 6(3):145–179, 1999.
7. M. Hernandez and S. Stolfo. The merge/purge problem for large databases. In *SIGMOD Conference*, pages 127–138, 1995.
8. L. Lee. On the effectiveness of the skew divergence for statistical language analysis. *Artificial Intelligence and Statistics*, pages 65–5, 2001.
9. X. Li, P. Morie, and D. Roth. Semantic integration in text, from ambiguous names to identifiable entities. *AI Magazine, American Association for Artificial Intelligence*, Spring:45–58, 2005.
10. G. S. Mann and D. Yarowsky. Unsupervised personal name disambiguation. In *Proceedings of CoNLL-2003*, pages 33–40, 2003.
11. A. McCallum and B. Wellner. Toward conditional models of identity uncertainty with application to proper noun coreference. In *IJCAI Workshop on Information Integration on the Web, 2003*, 2003.
12. D. McCarthy, R. Koeling, J. Weeds, and J. Carroll. Finding predominant word senses in untagged text. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04)*, pages 279–286, 2004.
13. T. Pedersen, A. Purandare, and A. Kulkarni. Name discrimination by clustering similar contexts. In *Proceedings of the Sixth International Conference on Intelligent Text Processing and Computational Linguistics*, 2005.
14. M. Sahami and T. Heilman. A web-based kernel function for matching short text snippets. In *International Workshop located at the 22nd International Conference on Machine Learning (ICML 2005)*, 2005.
15. H. Schutze. Automatic word sense discrimination. *Computational Linguistics*, 24(1):97–123, 1998.

Chinese Noun Phrase Metaphor Recognition with Maximum Entropy Approach*

Zhimin Wang, Houfeng Wang, Huiming Duan, Shuang Han, and Shiwen Yu

Department of Computer Science and Technology,
Institute of Computational Linguistics, Peking University, 100871, Beijing, China
{wangzm, wanghf, duenhm, yusw}@pku.edu.cn

Abstract. This paper presents a maximum entropy (ME)-based model for Chinese noun phrase metaphor recognition. The metaphor recognizing process will be viewed as a classification task between metaphor and literal meaning. Our experiments show that the metaphor recognizer based on the ME method is significantly better than the Example-based methods within the same context windows. In addition, performance is further improved by introducing additional features into the ME model and achieves good results in window (-2,+2).

1 Introduction

The task of identifying metaphors for a large-scale corpus has received an increasing amount of attention in the computational linguistics literature. Metaphors, one of figurative languages or tropes, can lead to inaccurate translation in Machine Translation systems and irrelevant document retrieval in Information Retrieval systems. For example, the Chinese word for “翅膀” means literally “wing of an animal”. However, when this word appears in a particular context, it has metaphorical expressions. For example,

张开 理想 的 翅膀 (meaning “explore fantasies”)
Spread fantasies of wings

where “翅膀” was not denoted the former literal meaning of “wing”, but has a metaphorical expression of “explore fantasies”. Information Retrieval systems should exclude this metaphorical expression while searching for “翅膀”.

Much research has gone into the processing of metaphors and provides some metaphor understanding systems such as the Met5, which is the first system to recognize examples of metaphors and metonymy under the guidance of preference constraint view[4], the Structure-Mapping Engine (SME), a program for studying analogical progressing [7], and the MIDAS system based on knowledge and analogy approach [13].

* Supported by the National Grand Fundamental Research 973 Program of China under Grant No. 2004CB318102; the National High-Tech Research and Development Plan of China under Grant Nos. 2001AA114210, 2002AA117010 (863); the National Natural Science Foundation of China Under Grant No.60473138.

Unfortunately, most of current systems use small, hand-coded semantic knowledge bases. It is quite difficult and time-consuming to manually provide such tremendous bases. In order to solve this problem, Mason [12] built the CorMet system with a statistical approach, which is a corpus-based system for discovering metaphorical mapping between concepts. It does this by finding systematic variations in domain-specific preferences, which is inferred from large dynamically mined Internet corpora and is tested on its ability to find a subset of the Master Metaphor List [12]. Mason's work offers us a wonderful idea of statistical approach.

Another constraint of metaphor computational models discussed above focuses more on the semantic preference violation of “verb + noun” or “noun + verb” in a sentence. Metaphors constitute a violation of selection restrictions. For instance, in this sentence “My car drinks gasoline” [6], the system identifies metaphorical relation by seeking violated preferences “drink” and “car”. According to their semantic violation relationship between “verb + noun”, however, we fail to deal with the following metaphorical expressions:

奔向/v 知识/n 的/u海洋/n(meaning “go into the oceans of knowledge”)
go into knowledge of oceans
陷入/v 思想/n 漩涡/n (meaning “think deeply”)
sink into think whirlpool

Where “奔向/v(go into) 海洋/n(oceans)” and “陷入/v(sink into) 漩涡/n(whirlpool)” have no semantic violation. But due to the insertion of “知识(knowledge)”and “思想(think)”, both sentences are changed metaphorical expressions. Above examples show that the metaphors do not appear on the “verb + noun” level, but on a phrase level like “noun + 的(of) + noun” or “noun + noun”. There is almost no published work on this kind of nominal metaphorical phrase. In this paper, we focus on such metaphors. The metaphor recognition process may be viewed as a classification task between metaphors and literal meanings. For example, “海洋” is a metaphorical expression that means “a large number” in the sentence “飞向知识的海洋(fly toward the oceans of knowledge)”. But for “中国的海洋生物(the oceanic life-form of China)”, “海洋” is only viewed as a literal meaning.

In this paper, we introduce the Maximum Entropy algorithm (ME) in the metaphor recognizing process, since it has achieved good results in many classification tasks such as part-of-speech tagging, Word Sense Disambiguation (WSD)(Berger et al [3], and Named Entity Recognition (Chieu et al [18]) and it has attained a good level of performance in this experiment.

The layout of this paper is as follows. In Section 2 we outline the ME framework and specify the features that we used. The method of feature selection is presented in Section 3; Section 4 addresses our experiment. Finally, future improvements will be discussed in Section 5.

2 Maximum Entropy Approach

Ratnaparkhi [2] noted that many problems in natural language processing (NLP) can be re-formulated as classification problems. The metaphor recognition in NLP can be

described as a classification task between metaphors and literal meanings, in which the task is to observe some linguistic “context”(c) and predict the correct linguistic “metaphor” (m). It can be denoted with a conditional probability distribution “p”, such that $p(m|c)$ is the probability of “metaphor” (m) given some “context” (c). where **【**m=metaphor denotes metaphorical expressions, m=no-metaphor denotes literal meaning**】**. The context (c) may consist of one or several words with parts of speech tags. We can not get all possible (m, c) pairs, so we need to seek a method for using the partial tagging corpus about (m) and (c) to estimate the probability model “p”.

The probability distribution used here is a maximum entropy model in the following equation.

$$p(m|c) = \frac{1}{Z(c)} \exp\left(\sum_{i=1}^n \lambda_i f_i(c, m)\right) \tag{1}$$

$$Z(c) = \sum_m \exp\left(\sum_{i=1}^n \lambda_i f_i(c, m)\right) \tag{2}$$

Where

f_i is an arbitrary feature function and is obtained from the training data. λ_i is a learned weight for each feature function. We adopt the GIS algorithm 20 to calculate the values for the parameter λ_i .

We may introduce the following feature for the example “思想的漩涡 (whirlpool of think)”.

$$f(c, m) = \begin{cases} 1 & \text{if } m = \text{'metaphor'} \text{ and its preceding word} = \text{"思想"} \\ 0 & \end{cases} \tag{3}$$

3 Features Selection

The features we used in this paper consist of two classes: vocabulary features and additional features. Vocabulary features are features that “ W_i ” appears in context. Additional features are some information derived from location of words, punctuation, stop words of target word etc.

3.1 Vocabulary Features

Vocabulary features we defined are the same as those used in MENE [1,12], taken from context windows W-2 to W+2 or W-1 to W+1. In this paper, we used more experiments with the context windows, including “W-1 to W+1” (-1, +1), “W-2 to W+1”(-2,+1) , “W-2 to W+2” (-2,+2), hoping to find an ideal window with a good result. The case of (-1,+2) is not considered just because the metaphorical expressions are more relevant to the previous words of W_i . The vocabulary features are given in Table 1.

Table 1. Vocabulary features

Location	Features
W_{i-2}	Word and its post of tagging
W_{i-1}	Word and its post of tagging
W_i	Word and its post of tagging
W_{i+1}	Word and its post of tagging
W_{i+2}	Word and its post of tagging

Each “ W_i ” predicts the Metaphor using its lexical and position tagging of the previous two words and the next two words.

3.2 Additional Features

Besides the selection of vocabulary features, the additional feature is another factor that might affect performance. We used the following additional positive features:

Left -Right information

A Left -Right feature indicates whether the word is to the left or right of the target word. For example,

知识的海洋 (oceans of knowledge)
海洋的知识 (knowledge of the ocean)

The “oceans of knowledge” is a metaphorical expression, while “knowledge of the ocean” is a literal expression, which is similar to an example such as “oceanic science”.

Punctuation candidate information

From the observation of metaphorical expressions, we find that a punctuation mark might follow “ W_i ”. We adopted punctuation as a kind of additional feature.

Stop words information

Within our context windows ,there usually consists some words with a high frequency such as the Chinese character ,“的(of)”, which often appears in both the metaphor expression “知识的海洋 (oceans of knowledge)” and the literal expression “国家的海洋资源(country’s oceanic resource)”. We consider some words like“的(of)” as a stop word.

Person’s name information

A target word sometimes is viewed as person’s name such as “李/n 海洋/nr”, where 海洋/nr is tagged a person’s name (given name) token, which is a remarkable tagging feature different from a metaphor.

4 Experiments and Results

The metaphor recognition algorithm is used to determine whether or not an example is a metaphor. We obtain the Vocabulary features through searching for the position of W_i in documents. After determining windows, we extract all kinds of features within the windows.

4.1 Metaphor Training Corpus and Test Data

In this experiment, we used the *People's Daily* corpus as training and testing data. In order to obtain the amount of vocabulary necessary for the model, we first extracted all sentences with target words from raw *People's Daily*. The sentences then were automatically segmented and tagged [17]. In addition, we manually separated all these sentences into two files as the Metaphor Tagging Corpus (MTC); the metaphorical expressions and the literal expressions of target words respectively. We firstly chose five target words to be metaphor-tagged. They are“海洋(ocean)”, “潮水(tidewater)”, “漩涡(whirlpool)”, “泥沼(swamp)”, “港湾(harbor)”. The MTC consisting of 5747 sentences from *People's Daily* (1993-2002 year, excluded 1998 year), is used as the training corpus.

In a manner similar to the training corpus extraction approach, we built Metaphor Test Data (MTD) which included 498 sentences of target words also from *People's Daily* (1998(1-6)). Every example in MTD was given a human class tag, after which the machine evaluated automatically each of the two output clusters according to MTC. The distribution of examples in the training and testing datasets are given in table 2.

Table 2. The distribution of examples in MTC and MTD

Words	Metaphor Tagging Corpus (MTC) =5747			Metaphor Test Data (MTD)=498		
	Distribution	m	no-m	Distribution	m	no-m
海洋	4751	448	4303	431	51	380
潮水	314	243	71	16	15	1
港湾	411	139	272	33	3	30
漩涡	213	131	82	11	10	1
泥沼	58	43	15	7	6	1

Where ‘m’ means the metaphorical examples, ‘no-m’ means the no metaphorical examples.

4.2 Example-Based Method for Baseline

In order to test the performance with maximum entropy method, the example-based method was introduced firstly as our baseline, which is a simple string-match. Firstly, we extracted all words between the windows of (-1,+1),(-2,+1),(-2,+2) from the training data and formed a word set. When we predicted the sentences from test data, we only checked whether the new words in the windows were contained in the corresponding word sets. Based on the observations of Chinese noun phrase metaphors, we chose the Chinese character,“的(of)”,and “punctuation”marks as stop words in our experiment.

We evaluated them on Precision, Recall and F-Measure. The precision is the number of correct metaphor examples divided by the total number of detected metaphor examples. The recall is the number of correct metaphor examples divided by the total number of noun phrase metaphors in the manual evaluation data.

The F-measure is calculated by using recall R and precision P

$$F = \frac{2 \times P \times R}{P + R}$$

The baseline system in following Table 3 refers to the example-based system that uses only vocabulary features. We experimented with different window between (-1,+1),(-2,+1) and (-2,+2).

Table 3. Example-based method with only positive examples

Windows	泥沼	海洋	港湾	漩涡	潮水
(1,1)	P=100.0% R=66.7% F=80.0%	P=21.5% R=90.2% F=34.7%	P=0.0% R=0.0% F=0.0%	P=90.0% R=90.0% F=90.0%	P=93.8% R=100.0% F=96.8%
(2,1)	P=100.0% R=66.7% F=80.0%	P=17.6% R=92.2% F=29.6%	P=42.9% R=100.0% F=60.0%	P=90.9% R=100.0% F=95.2%	P=93.8% R=100.0% F=96.8%
(2,2)	P=100.0% R=66.7% F=80.0%	P=14.2% R=94.1% F=24.7%	P=37.5% R=100.0% F=54.5%	P=90.9% R=100.0% F=95.2%	P=93.8% R=100.0% F=96.8%

There are several reasons for the poor performance of the example-based system. First, we only introduced positive word sets with metaphor vocabulary features in our model, not using the negative word set with metaphor features. Also, the example-based method system introduced noise, due to the simple string-match. Table 4 shows that adding negative features yielded a significant improvement. Moreover, we extended or narrowed the context window when dealing with positive word sets and negative word sets at same time.

Table 4. Example-based method with both positive and negative examples

Windows	泥沼	海洋	港湾	漩涡	潮水
(-1,+1)	P=100.0% R=83.3% F=90.9%	P=88.2% R=29.4% F=44.1%	P=50.0% R=33.3% F=40.0%	P=90.0% R=90.0% F=90.0%	P=93.8% R=100.0% F=96.8%
(-2,+1)	P=100.0% R=83.3% F=90.9%	P=100.0% R=25.5% F=40.6%	P=50.0% R=66.7% F=57.1%	P=90.0% R=90.0% F=90.0%	P=100.0% R=93.3% F=96.6%
(-2,+2)	P=100.0% R=83.3% F=90.9%	P=87.5% R=27.5% F=41.8%	P=66.7% R=66.7% F=66.7%	P=90.0% R=90.0% F=90.0%	P=100.0% R=100.0% F=100.0%

From the above table, we can see the baseline we obtained with both positive and negative features is quite high, The main reason is the examples of ‘no-m’ we obtained in MTD are limited, especially in the cases of “泥沼,漩涡,潮水”, which only have one ‘no-m’ example respectively.

Some people may think that overlapping examples existed in the training data and the testing data. In order to clarify this question, we have carried out the following experiments. From the sentence level, we have not found identity examples between MTC and MTD. Meanwhile, we compared the overlap cases of all examples of the above three words within the window (-2,+2). Only “漩涡” has one identical example, while others have no overlap phenomenon.

4.3 Performance for Maximum Entropy-Based Method

Experiments were also conducted on noun phrase metaphor recognition based on the ME method. Table 5 shows the performance of the ME-based system using only vocabulary features, which achieved good results. The words “泥沼, 潮水” have an excellent performance rating of 100%.

Table 5. Results of ME method with different windows

Windows	泥沼	海洋	港湾	漩涡	潮水
(-1,+1)	P=100.0% R=100.0% F=100.0%	P=54.4% R=60.8% F=57.4%	P=13.3% R=66.7% F=22.2%	P=90.0% R=90.0% F=90.0%	P=93.3% R=93.3% F=93.3%
(-2,+1)	P=100.0% R=100.0% F=100.0%	P=78.9% R=58.8% F=67.4%	P=50.0% R=100.0% F=66.7%	P=90.9% R=100.0% F=95.2%	P=100.0% R=86.7% F=92.9%
(-2,+2)	P=100.0% R=100.0% F=100.0%	P=78.9% R=58.8% F=67.4%	P=50.0% R=66.7% F=57.1%	P=90.0% R=90.0% F=90.0%	P=100.0% R=100.0% F=100.0%

The experiment showed that the system’s precision, recall, F measure had a significantly better result than the baseline. Meanwhile the case of (-2, +1) was higher than the case of (-1, +1), (-2, +2).

Table 6. Results of ME method combing with additional features

Windows	泥沼	海洋	港湾	漩涡	潮水
(-1,+1)	P=100.0% R=83.3% F=90.9%	P=60.0% R=58.8% F=59.4%	P=7.7% R=33.3% F=12.5%	P=90.9% R=100.0% F=95.2%	P=93.3% R=93.3% F=93.3%
(-2,+1)	P=100.0% R=83.3% F=90.9%	P=73.2% R=58.8% F=65.2%	P=40.0% R=66.7% F=50.0%	P=90.9% R=100.0% F=95.2%	P=93.3% R=93.3% F=93.3%
(-2,+2) good results	P=100.0% R=83.3% F=90.9%	P=78.0% R=62.7% F=69.6%	P=66.7% R=66.7% F=66.7%	P=90.9% R=100.0% F=95.2%	P=100.0% R=100.0% F=100.0%

Table 6 shows that the performance of the system using additional features and vocabulary features was better than using only vocabulary features. All the above examples performed better than those in the first experiment except “泥沼”. Good results are achieved when the window is set at (-2, +2).

4.4 The Cross-Validation Experiments for Maximum Entropy-Based Method

The cross-validation experiments we adopted firstly put all examples of MTC and MTD together in one data set, and split them into five parts, then used each in turn for testing, and the rest for training. We performed five cross-validation runs. Table 7 shows that overall the results were significantly better than the previous tests.

Table 7. Results of cross-validation experiments

Division	泥沼	海洋	港湾	漩涡	潮水
Division 1	P=90.0% R=100.0% F=94.7%	P=80.5% R=65.3% F=72.1%	P=45.5% R=78.9% F=57.7%	P=78.6% R=84.6% F=81.5%	P=82.5% R=97.9% F=89.5%
Division 2	P=100.0% R=100.0% F=100.0%	P=76.0% R=72.4% F=74.1%	P=74.3% R=86.7% F=80.0%	P=87.1% R=90.0% F=88.5%	P=73.4% R=100.0% F=84.7%
Division 3	P=80.0% R=80.0% F=80.0%	P=73.6% R=79.4% F=76.4%	P=69.2% R=84.4% F=76.1%	P=90.3% R=84.8% F=87.5%	P=92.6% R=96.2% F=94.3%
Division 4	P=80.0% R=88.9% F=84.2%	P=69.6% R=71.7% F=70.6%	P=67.4% R=93.5% F=78.4%	P=82.8% R=80.0% F=81.4%	P=82.5% R=97.9% F=89.5%
Division 5	P=80.0% R=100.0% F=88.9%	P=76.3% R=74.0% F=75.1%	P=50.0% R=86.4% F=63.3%	P=66.7% R=96.0% F=78.7%	P=93.3% R=98.2% F=95.7%
Overall average	P=86% R=83.8% F=84.9%	P=75.3% R=71.8% F=73.5%	P=61.3% R=86.0% F=71.6%	P=81.1% R=87.1% F=84.0%	P=84.9% R=90.0% F=87.4%

5 Conclusions

Metaphor recognition is a new research field in Chinese Information Processing. In this paper, we have demonstrated the high performance of the ME model for Chinese nominal metaphor recognition. Our experiments show that the selection of context windows is important for metaphor recognition. In addition, we have also outlined experiments, performed and compared their performance when introducing additional features to the ME model, which have achieved a good result in window (-2,+2). An example-base method is also tested as a baseline.

The plan for further research includes combining the semantic information into the ME model. Moreover, we will also try to make use of the analogy between metaphorical expressions to automatically generalize other metaphors in corpus through the semantic relative analogy. Examples include the expressions such as “历史的长河(the river of history)”, which are currently not in our MTC through the semantic relative analogy with “oceans of knowledge” which have been annotated in MTC. Building a relative mapping between different domains is crucial to successfully detect noun phrase metaphors. Additionally, because many sentences in the training corpus lack structural information, the current model will take full advantage of the syntactic features.

From the examples collected, we also found important topical information such that literal or metaphorical expressions usually come from different domains. For instance, the former reported the topic of a bloody accident or environmental protection, while the latter discussed political topics or literature descriptions such as “eddy of politics”, “tide of reform” and “fleet of life”. Future work will be done to add this type of information into the model.

Finally, we also plan to explore new methods to detect “noun+的(of)+noun” metaphorical patterns, rather than several target words. This will be done by assigning a CCD (the Chinese Concept Dictionary) noun top-level semantic category (Liu et al [16], Yu et al [15]) to build the nominal metaphor analogy. We believe that the underlying analogy between noun words is crucial information for future metaphor recognition.

References

1. Borthwick, A., Sterling, J., Agichtein, E., and Grishman R., Exploiting Diverse Knowledge Sources via Maximum Entropy in Named Entity Recognition. In Proceedings of the Sixth Workshop on Very Large Corpora, Montreal, Canada, 1998.152-160.
2. Ratnaparkhi, A. A Maximum Entropy Model for Part-of-Speech Tagging. In Proceedings of Conference on Empirical Methods in Natural Language Processing, University of Pennsylvania, 1996. 133-141.
3. Berger, A. Della Pietra, S. A., and Della Pietra, V. J. A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics*, 1996.22(1):39-71.
4. Fass, D.C. An Account of Coherence, Semantic Relations, Metonymy, and Lexical Ambiguity Resolution, in *Lexical Ambiguity Resolution: Perspectives from Psycholinguistics, Neuro psychology and Artificial Intelligence*. ed. Small, S. I, G. W. Cottrell & M. K. Tanenhaus. San Mateo, CA: Morgan Kaufmann. 1988.
5. Fass, D.C. Metonymy and Metaphor: What's the Difference? In Proceedings, 12th International Conference on Computational Linguistics (COLING-88), Budapest, Hungary, 1988.P177-181.
6. Fass, D.C. met*: A Method for Discriminating Metonymy and Metaphor by Computer, Association for Computational Linguistics, 1991.
7. Gentner, D. Structure-Mapping: A Theoretical Framework for Analogy . *Cognitive Science*, 1983. 7: 155-170.
8. Gentner, D. Falkenhainer, B. and Skorstad, J. Viewing metaphor as analogy. In Helman, D. Editor, *Analogical Reasoning*. Kluwer Academic Publishers.1988.
9. Indurkha, B. Approximate Semantic Transference: A computational Theory of Metaphors and Analogy. *Cognitive Science*, 1987.11:445-480.

10. Holyoak, K. J. Thagard, P. Analogical Mapping by Constraint Satisfaction, *Cognitive Science*, 1989.Vol.13: 295-355.
11. Lakoff, G. Johnson, M. *Metaphors We Live By*. Chicago: University of Chicago Press, 1980.
12. Mason, Z. CorMet: A Computational, Corpus-Based Conventional Metaphor Extraction System. *Computational Linguistics*. 2004.30(1): P 23-44.
13. Martin, J. H. *A Computational Model of Metaphor Interpretation*. NY: Academic Press. 1990.
14. Wilks Y. A Preferential Pattern-Seeking Semantics for Natural Language Inference. *Artificial Intelligence*, 1975.Vol.6:53-74,
15. Yu,JS., liu,Yang. and Yu, Shiwen. The Specification of the Chinese Concept Dictionary. *Journal of Chinese Language and Computing (in Singapore)*. 2003.Vol.13, No.2, P177-194.
16. Liu,Y., Yu Jiangsheng. and Yu ,Shiwen. A Tree-Structure Solution for the Development of ChineseNet. In *Proceedings of ICG2002*, 2002.P51-56.
17. Yu,SW. *The Grammatical Knowledge-Based of Contemporary Chinese — A Complete Specification*, Tsinghua University Press, 2003.
18. Chieu HL., Hwee Tou Ng. A Maximum Entropy Approach to Information Extraction from Semi-Structured and Free Text. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence*, 2002a.P786-791.
19. Chieu HL., Hwee Tou Ng. Named Entity Recognition: A Maximum Entropy Approach Using Global Information. In *Proceedings of the Nine-teenth International Conference on Computational Linguistics*, 2002b.P190-196.
20. Darroch, J.N and Ratcliff, D. Generalized Iterative Scaling for Log-linear Models. *Annals of Mathematical Statistics*, 1972.43:1470-1480.

Zero Anaphora Resolution in Chinese Discourse

Yuzhen Cui, Qinan Hu, Haihua Pan, and Jianhua Hu

Department of Chinese, Translation and Linguistics,
City University of Hong Kong, Hong Kong
{50007840, qinan.hu}@student.cityu.edu.hk
{cthpan, ctjhu}@cityu.edu.hk

Abstract. This paper explores various factors involved in the resolution of zero anaphora in Chinese discourse. Our study differs from previous ones in distinguishing three types of utterances and using clauses as the unit of resolution. The hierarchical structures of utterances enable us to process inter- and intra-utterance anaphora uniformly. Experimental results show that (1) clauses function significantly better than sentences as the unit of resolution, providing an improvement of precision from 36.0% to 63.4%; (2) the inclusion of cataphors and the use of NP forms as a criterion in Cf ranking do not lead to significant improvement of precision; and (3) when assigning antecedents to more than one zero pronoun in the same utterance, the criterion based on grammatical functions gives rise to better performance than that with linear orders.

1 Introduction

Several studies were conducted on zero anaphora for languages like Chinese [1], Japanese [2], Italian [3] and Turkish [4]. The Chinese study resolves zero pronouns in a part-of-speech tagged and shallow-parsed corpus, focusing on pronouns in topic, subject, or object positions in main clauses. All these studies employ Centering Theory (CT) [5, 6] as their framework.

Several problems are found in previous studies. First, it is not clear what counts as an utterance in Chinese discourse. Previous studies either provide no specification or simply use commas and periods as the indicators of utterance ending. Second, the resolution of zero pronouns in subordinate clauses has not been well studied. Third, when two zero pronouns or more occur in the same utterance, it is unclear when they share the same antecedent and when they do not. Finally, cataphora is often not discussed in previous studies.

2 Zero Anaphora in Chinese Discourse

In this study, the term *utterance* refers to an instance of a sentence which is delimited by periods, exclamations, or question marks, and three types of utterances are distinguished, i.e. simple, compound, and complex utterances. A simple utterance consists of only one clause, a compound utterance, several coordinated clauses, and a complex utterance, clauses of different phrasal levels. We notice that (1) zero pronouns could appear in either topic, subject or object position in each utterance type; (2) cataphora

occurs only in complex utterances where matrix clauses precede subordinate clauses; and (3) inter-utterance zero anaphora usually occurs in simple and complex utterances, and intra-utterance zero anaphora, in compound and complex utterances.

Like previous studies (cf. [7]) this study also adopts CT as the framework. CT was originally designed to model the center of attention of discourse participants that is relative to the relationship of attentional state, inferential complexity and the form of referring expressions. It defines a set of centers (Cf: forward-looking center, Cb: backward-looking center, Cp: preferred forward-looking center), rules and constraints. Each utterance has a list of Cfs which are ranked according to their salience as being the link between utterances. Some regularity is found in our study concerning the relation between antecedents and Cf ranking in Chinese discourse, as shown in Table 1. Nearly 80% of the antecedents has the highest rank in their utterances; about 20% has second highest; only about 2% has other ranks.

Table 1. The Cf ranking order of antecedents in their utterances

Genre	Highest ranked	Second highest ranked	Other ranked
Essay	76.9%	21.1%	2.0%
Expository article	79.0%	18.4%	2.6%

If a zero pronoun and its antecedent do not co-occur in the same utterance, they may occur in two (non-)consecutive utterances. The distribution of anaphoric distance is given in Table 2. Over 70% occurs in two consecutive utterances, over 20% in the same utterance, and only a small portion in two non-consecutive utterances.

Table 2. The anaphoric distances between zero pronouns and their antecedents

Genre	In same utterance	In consecutive utterances	In non-consecutive utterances
Essay	21.3%	76.1%	2.6%
Expository article	22.4%	77.6%	0%

3 Algorithm and Experiments

Our algorithm includes two parts. One generates a Cf list for each clause by using their immediate constituent NPs, and the other determines in which clause the antecedents appear. If a clause contains sub-clauses, its Cf list will be conjoined in the order of breadth-width traversal. Zero pronouns are resolved in terms of clauses, and their current and previous clauses are identified. To identify the previous clause of a zero pronoun, we consider whether there is any clause at the same phrasal/sentential level before the current clause, and if not, go to its parent clause. This process will be iterated when the Cf list of the previous clause is empty or contains no higher ranked Cfs other than the zero pronoun itself until a qualified Cf is found. Since matrix clauses and subordinate clauses are not distinguished in our study, inter-utterance and intra-utterance anaphora are processed in the same way.

The corpus used in our experiments contains articles of various genres. It is partially parsed, with embedded sentential structures identified, and grammatical func-

tions of noun phrases and antecedents of pronouns annotated. It contains more than 1000 zero pronouns in total. The following questions are addressed in our study:

- At what level should zero pronouns be resolved, sentence or clause levels?
- To what extent does the resolution of cataphors affect the overall performance?
- What should be used, e.g. grammatical functions or NP forms, to establish ranking criteria for ranking Cfs in constructing a Cf list?
- What determines the antecedent assignment, e.g. linear orders or grammatical functions, when two zero pronouns or more occur in the same utterance?

Table 3. System configurations and precisions

Configuration	Cataphors included	Unit of resolution		Cf ranking		Antecedent assignment strategies			Precision
		Sentence	Clause	Forms of Cfs	Grammatical functions	Same antecedent	Linear order	Grammatical functions	
Config1		Y			Y	Y			35.7%
Config2	Y	Y			Y	Y			36.0%
Config3	Y		Y		Y	Y			63.4%
Config4	Y		Y		Y		Y		64.9%
Config5	Y		Y		Y			Y	65.9%
Config6	Y		Y	Y	Y			Y	66.1%

To answer the above questions, we have implemented systems with various configurations, as listed in Table 3. Config-1 is the baseline; it excludes cataphors. Config-2 differs from Config-1 in including cataphors, and from Config-3 at the unit of resolution: Config-2, sentences, and Config-3, clauses. Config-3, 4 and 5 differ in their strategies of antecedent assignment. Config-3 assigns the highest ranked Cf to all zero pronouns in the same utterance, and Config-4, the highest ranked Cf to the first zero pronoun, the second highest, the second, and so on. Config-5 assigns highest ranked Cfs to subject pronouns, and second highest to object pronouns, and the same for Config-6. Grammatical functions are used as the only Cf ranking criterion [8] in all the configurations except for Config-6 which employs the forms of NPs as the primary criterion (zero pronouns are ranked higher than other kinds of pronouns which in turn are ranked higher than other forms of NPs), with grammatical functions being only the secondary criterion to rank Cfs with the same forms.

Experimental results show that the biggest improvement of precision appears from Config-2 to Config-3, and they differ in the units of resolution employed: Config-2, sentences, and Config-3, clauses. This result confirms Poesio et al.'s [9] findings on the parameters of CT, namely that, when sentences rather than finite clauses are used as utterances, more violations of the pronominal rule (Rule 2) are observed, though no distinction is made between finite and non-finite clauses in our study, as there is no such distinction in Chinese (cf. Hu et al. [10]). This also shows that, although using sentences as the resolution unit could represent inter-utterance consistency to some extent, they are problematic in representing intra-utterance consistency.

Config-3, 4, 5 and 6 present comparable results, with Config-6 being the best. Error analysis shows that Cf ranking is the main source of errors. The fact that the precision of Config-6 is slightly lower than that reported in Yeh and Chen [1] (70%) is due to different corpora used and the fact that our study also includes zero pronouns in embedded clauses, while theirs only includes zero pronouns in main clauses.

4 Conclusions

From our experiments we can see that both the ordering among utterances and the hierarchical structures of utterances play an important role in Chinese zero anaphor resolution. Distinguishing three types of utterances and using clauses as the units of resolution enables us to process inter- and intra-utterance anaphora uniformly. Experimental results have shown that (1) zero pronoun resolution at the clause-level significantly outperforms that at the sentence-level, with an increase of precision by 27.4%; (2) the inclusion of cataphors and the use of NP forms as a criterion in Cf ranking do not lead to significant improvement of precision; and (3) when assigning antecedents to two zero pronouns or more in the same utterance, the criterion based on grammatical functions gives rise to better performance than that with linear orders.

References

1. Yeh C.L., Chen Y.C., 2005, Zero Anaphora Resolution in Chinese with Shallow Parsing, To appear in: *Journal of Chinese Language and Computing*.
2. Kameyama M., 1985, Zero Anaphora: The Case of Japanese. PhD Dissertation, Stanford University.
3. Di Eugenio B., 1998, Centering in Italian, in Walker M. et al. eds, *Centering Theory in Discourse*, pp 115-137, Oxford University Press.
4. Turan U. D., 1996, Null vs. Overt Subjects in Turkish Discourse: A Centering Analysis. PhD Dissertation, University of Pennsylvania.
5. Grosz B.J., Joshi A.K., Weinstein S., 1995, Centering: A Framework for Modelling the Local Coherence of Discourse, *Computational Linguistics*, 21 (2), pp.203-225.
6. Walker M., Joshi A. and Prince E., 1998, *Centering Theory in Discourse*, Oxford University Press.
7. Miltsakaki E., 2002. Toward an Aposynthesis of Topic Continuity and Intrasentential Anaphora, *Computational Linguistics*, Vol 28. No. 3.
8. Brennan S., Friedman M.W., and Pollard C., 1987, A Centering Approach to Pronouns, In *Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics*, pp. 155-162, Stanford, California.
9. Poesio M., Stevenson R., Di Eugenio B. and Hitzeman J., 2004, Centering: A Parametric Theory and its Instantiations. *Computational Linguistics*. Vol 30. No. 3.
10. Hu J.H., Pan H.H., Xu L.J, 2001, Is There a Finite vs. Nonfinite Distinction in Chinese? In *Linguistics*. Vol 39. No. 6.

Random Walks on Text Structures

Rada Mihalcea

University of North Texas,
Computer Science Department
rada@cs.unt.edu

Abstract. Since the early ages of artificial intelligence, associative or semantic networks have been proposed as representations that enable the storage of language units and the relationships that interconnect them, allowing for a variety of inference and reasoning processes, and simulating some of the functionalities of the human mind. The symbolic structures that emerge from these representations correspond naturally to graphs – relational structures capable of encoding the meaning and structure of a cohesive text, following closely the associative or semantic memory representations. The activation or ranking of nodes in such graph structures mimics to some extent the functioning of human memory, and can be turned into a rich source of knowledge useful for several language processing applications. In this paper, we suggest a framework for the application of graph-based ranking algorithms to natural language processing, and illustrate the application of this framework to two traditionally difficult text processing tasks: word sense disambiguation and text summarization.

1 Introduction

Many language processing applications can be modeled by means of a graph. These data structures have the ability to encode in a natural way the meaning and structure of a cohesive text, and follow closely the associative or semantic memory representations. For instance, Figure 1 shows examples of graph representations of textual units¹ and the relationships that interconnect them: 1(a) (adapted from [6]) shows a network of concepts related by semantic relations – simulating a fragment of human memory, on which reasoning and inferences about various concepts represented in the network can be performed; 1(b) shows a network with similar structure, this time automatically derived via definitional links in a dictionary; finally, 1(c) is a graph representation of the cohesive structure of a text, by encoding similarity relationships between textual units.

Provided a graph representation of the text, algorithms for the activation or ranking of nodes in such structures can be used to simulate the functioning of human memory, consequently resulting in solutions for a variety of natural language processing tasks that can be modeled by means of a graph. In this paper, we suggest a framework for the application of graph-based ranking algorithms to text-based graph structures, and show how two text processing applications: word sense disambiguation and text summarization, can find successful solutions within this framework.

¹ We use the term *textual unit* to refer to the textual representation of a *cognitive unit* as defined by Anderson [1]. It can be a word, a concept, a sentence, or any other unit that can find a representation in language.

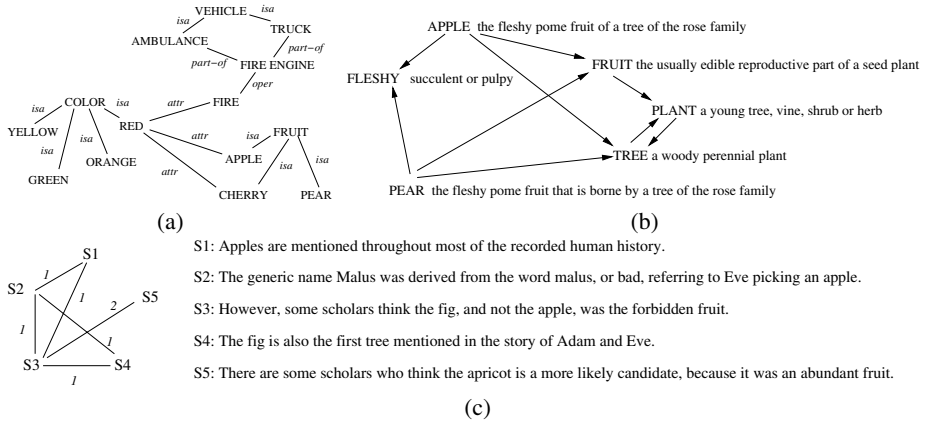


Fig. 1. Graph representations of textual units and relationships that interconnect them

2 Background

Although initiated from different theoretical backgrounds, there is nonetheless a close relation between the current graph-based ranking algorithms (first introduced in graph-theory), and the earlier models of spreading activation (due to cognitive psychology), which stands primarily in their common underlying graph representations and their fundamental idea of exploiting flow over a network.

The idea of associative networks as mental representations for cognitive units and the relationships between them goes back to early work in psychology [9] and psycholinguistics [27], [29]. Theories of semantic or associative memory [26] have initially emerged in cognitive psychology as models for human language representation and reasoning, and since then have been applied to a variety of computer-based applications.

Spreading activation can be regarded as an earlier version of current graph-based ranking algorithms. It refers to network²-based models where activation started from *one or more* source nodes is propagated over the network, activating more and more nodes, until a certain termination condition is met (usually the distance from the source nodes). In these models, it is the *activation* itself that matters, and thus the information recorded at node level is whether the node is *active* or *passive*.

In the more recently introduced graph-based ranking models (or node-ranking models), the propagation of flow over the network starts from *all* the nodes simultaneously, and runs repeatedly throughout the network until a stable state is achieved (convergence). In these models, the information recorded at node level is the *rank* of the node relative to all other nodes in the network.

Despite their potential appealing connection to models of human memory, the large scale application of spreading activation and graph-based ranking models to text processing tasks has been limited for various reasons: Early work in spreading activation methods was hindered by the complexity of underlying structures (e.g. entire semantic

² The terms *network* and *graph* are used interchangeably, a graph being the computer-based representation of a network.

networks), and as a result only few applications with small scale evaluations have been attempted. On the other hand, recent research in graph-based ranking algorithms has focused on social networks and Web-link analysis, and their application to text processing has not been explored.

3 Related Work in Natural Language Processing

Starting with the seminal work of Quillian on theories of semantic memory [26], associative or semantic networks and spreading activation processes have been used as the underlying model for several applications in language processing, including word sense disambiguation, automated reasoning, text generation, information retrieval, text summarization, and others.

A major impediment faced by early work in this area was the complexity of the resulting structures, which sometimes was not supported by the underlying computer hardware (although attempts were made to overcome these hardware limitations with parallel architectures for semantic processing such as SNAP [24]). This fact has consequently resulted in limitations on the size of applications attempted with these approaches. For similar reasons, the evaluation of such algorithms was most of the times performed on toy-size problems (e.g. Quillian has evaluated his proposed word sense disambiguation algorithm on nineteen ambiguous words [26]), and the scalability of the models was rarely, if ever, evaluated.

Spreading activation was previously used as a method for solving lexical ambiguity of words [11] through simultaneous identification of word senses and frame case slots. Another related line of work is the algorithm proposed in [31], where a large neural network is built by relating words through their dictionary definitions. Spreading activation was also suggested as a means for dictionary access [33], using a method that simulates processes of human mind, thus improving over other traditional ways for dictionary look-up. The application of spreading activation algorithms was also tested in information retrieval, in a monolingual domain specific environment [5], or more general multilingual environments for cross language information retrieval [2].

More recently, graph-based ranking algorithms (e.g. HITS [13] or PageRank [3]) have been successfully used in citation analysis, social networks [7], and the analysis of the link-structure of the World Wide Web [3]. A node ranking algorithm relying on PageRank [3] and the ArcRank extension [12] was used as a method for thesaurus construction starting with electronic dictionaries [12].

In recent work, we have shown how graph-based ranking algorithms designed for content-independent Web link analysis can be turned into a useful source of information for language processing tasks when applied to graphs extracted from natural language texts – with encouraging results on the problem of word sense disambiguation [21], sentence ranking and extraction for text summarization [17], and selection of important terms in a text [19]. The PageRank algorithm was also evaluated in a comparative study of coherence algorithms [32], where it was found to exceed the performance of other paragraph and word oriented algorithms for sentence ranking. Finally, the same ranking algorithm was integrated in an event-centric approach to summarization [30], where it was used to identify important elements (events or entities) in a text.

4 General Framework

We suggest a framework targeted to the application of graph-based ranking models to text processing tasks. Several new methods and representations are described, specifically tailored for the application of this framework to natural language processing. In Section 6, we describe two applications that can be successfully addressed within this framework.

4.1 A New Graph-Based Representation of Text

To enable the application of graph-based ranking algorithms to natural language texts, we have to build a graph that represents the text, and interconnects words or other text entities with meaningful relations. The graphs constructed in this way are centered around the target text, but can be extended with external graphs, such as off-the-shelf semantic or associative networks (e.g. WordNet [22]), or other similar structures automatically derived from large corpora.

Representation

Graph Nodes: Depending on the application at hand, text units of various sizes and characteristics can be added as vertices in the graph, e.g. words, collocations, word-senses, entire sentences, entire documents, or others. Note that the graph-nodes do not have to belong to the same category.

Graph Edges: Similarly, it is the application that dictates the type of relations that are used to draw connections between any two such vertices, e.g. lexical or semantic relations, measures of text cohesiveness, contextual overlap, membership of a word in a sentence, and others.

Algorithm

Regardless of the type and characteristics of the elements added to the graph, the application of the ranking algorithms to natural language texts consists of the following main steps:

1. Identify text units that best define the task at hand, and add them as vertices in the graph.
2. Identify relations that connect such text units, and use these relations to draw edges between vertices in the graph. Edges can be directed or undirected, weighted or unweighted.
3. Apply a graph-based ranking algorithm to find a ranking over the nodes in the graph. Iterate the graph-based ranking algorithm until convergence. Sort vertices based on their final score. Use the values attached to each vertex for ranking/selection decisions.

5 Graph-Based Ranking Algorithms

Graph-based ranking algorithms are essentially a way of deciding the importance (or “power”) of a vertex within a graph, based on information drawn from the graph structure. The basic idea implemented by a graph-based ranking model is that of “voting”

or “recommendation”. When one vertex links to another one, it is basically casting a vote for that other vertex. The higher the number of votes that are cast for a vertex, the higher the importance of the vertex. Graph-based ranking algorithms have been successfully used in citation analysis, social networks [7], and content-independent Web-link analysis [3].

These graph ranking algorithms are based on a random walk model, where a walker takes random steps on the graph G , with the walk being modeled as a Markov process – that is, the decision on what edge to follow is solely based on the vertex where the walker is currently located. Under certain conditions, this model converges to a stationary distribution of probabilities, associated with vertices in the graph, representing the probability of finding the walker at a certain vertex in the graph. Based on the Ergodic theorem for Markov chains [10], the algorithms are guaranteed to converge if the graph is both aperiodic and irreducible. The first condition is achieved for any graph that is a non-bipartite graph, while the second condition holds for any strongly connected graph. Both these properties are typically achieved for the text-based graphs constructed for the language processing applications considered in this work.

Let $G = (V, E)$ be a directed graph with the set of vertices V and set of edges E , where E is a subset of $V \times V$. For a given vertex V_i , let $In(V_i)$ be the set of vertices that point to it (predecessors), and let $Out(V_i)$ be the set of vertices that vertex V_i points to (successors). We describe below two graph-based ranking algorithms:

HITS (Hyperlinked Induced Topic Search) [13] is an iterative algorithm that was designed for ranking Web pages according to their degree of “authority”. The HITS algorithm makes a distinction between “authorities” (pages with a large number of incoming links) and “hubs” (pages with a large number of outgoing links). For each vertex, HITS produces two sets of scores – an “authority” score, and a “hub” score:

$$HITS_A(V_i) = \sum_{V_j \in In(V_i)} HITS_H(V_j) \quad (1)$$

$$HITS_H(V_i) = \sum_{V_j \in Out(V_i)} HITS_A(V_j) \quad (2)$$

PageRank [3] is perhaps one of the most popular ranking algorithms, and was designed as a method for Web link analysis. Unlike other ranking algorithms, PageRank integrates the impact of both incoming and outgoing links into one single model, and therefore it produces only one set of scores:

$$PR(V_i) = (1 - d) + d * \sum_{V_j \in In(V_i)} \frac{PR(V_j)}{|Out(V_j)|} \quad (3)$$

where d is a parameter that can be set between 0 and 1.³ In matrix notation, the *PageRank* vector of stationary probabilities is the principal eigenvector for the matrix A_{row} , which is obtained from the adjacency matrix A representing the graph, with all rows normalized to sum to 1: ($P = A_{row}^T P$).

³ The damping factor d has the role of integrating into the model the probability of jumping from a given vertex to another random vertex in the graph. The factor d is usually set at 0.85 [3].

A ranking process starts by assigning arbitrary values to each node in the graph, followed by several iterations until convergence below a given threshold is achieved. Convergence is achieved when the error rate for any vertex in the graph falls below a given threshold, where the error rate of a vertex V_i is approximated with the difference between the scores computed at two successive iterations: $S^{k+1}(V_i) - S^k(V_i)$ (usually after 25-35 iteration steps). After running the algorithm, a score is associated with each vertex, which represents the “importance” (*rank*) of the vertex within the graph. Note that for such iterative algorithms, the final value obtained for each vertex is not affected by the choice of the initial value, only the number of iterations to convergence may be different.

The basic graph-based ranking framework can be improved with representations specifically tailored to language processing tasks. We describe below two such improvements, consisting of the application of graph-based ranking to *undirected* and *weighted* graphs.

Undirected Graphs. Although traditionally applied on directed graphs, algorithms for node activation or ranking can be also applied to undirected graphs. In such graphs, convergence is usually achieved after a larger number of iterations, and the final ranking can differ significantly compared to the ranking obtained on directed graphs.

Weighted Graphs. When the graphs are built from natural language texts, they may include multiple or partial links between the units (vertices) that are extracted from text. It may be therefore useful to indicate and incorporate into the model the “strength” of the connection between two vertices V_i and V_j as a weight w_{ij} added to the corresponding edge that connects the two vertices. Consequently, we introduce new formulae for graph-based ranking that take into account edge weights when computing the score associated with a vertex in the graph, e.g.

$$PR^W(V_i) = (1 - d) + d * \sum_{V_j \in In(V_i)} w_{ji} \frac{PR^W(V_j)}{\sum_{V_k \in Out(V_j)} w_{jk}} \quad (4)$$

Similar weighted versions can be defined for all other ranking algorithms, as also shown in our previous work [17]. The final vertex scores (and therefore rankings) for weighted graphs can differ significantly as compared to their unweighted alternatives.

6 Graph-Based Ranking Algorithms for Text Processing

Many natural language processing applications can be modeled by means of a graph, and thus the framework suggested in this paper can provide solutions for many important text processing problems. In this paper, we specifically address the application of graph-based ranking algorithms to text processing at two different levels of granularity: *sentence level* and *document level*. We show how the graph-based ranking algorithms can be applied to: (1) text processing at sentence level, where we specifically address the problem of word sense disambiguation; and (2) text processing at document level, with a focus on the problem of extractive summarization.

6.1 Text Processing at Sentence Level: Unsupervised Word Sense Disambiguation

The task of word sense disambiguation consists of assigning the most appropriate meaning to a polysemous word within a given context. To enable the application of algorithms for graph-based ranking to the disambiguation of all words in unrestricted text, we have to build a graph that represents the text and interconnects the words with meaningful relations.

We start by first formulating the word sense disambiguation problem as a sequence data labeling problem. Note that this formulation applies not only to word sense disambiguation, but also to other labeling problems, e.g. part-of-speech tagging, named entity resolution, etc. Given a sequence of words $W = \{w_1, w_2, \dots, w_n\}$, each word w_i with corresponding admissible labels $L_{w_i} = \{l_{w_i}^1, l_{w_i}^2, \dots, l_{w_i}^{N_{w_i}}\}$, we define a label graph $G = (V, E)$ such that there is a vertex $v \in V$ for every possible label $l_{w_i}^j$, $i = 1..n$, $j = 1..N_{w_i}$. Dependencies between pairs of labels are represented as directed or undirected edges $e \in E$, defined over the set of vertex pairs $V \times V$. Such label dependencies can be learned from annotated data, or derived by other means, e.g. by measuring similarity between instances (see Figure 2 for an example). Note that the graph does not have to be fully connected, as not all label pairs can be related by a dependency.

Given such a label graph associated with a sequence of words, the likelihood of each label can be determined using an iterative graph-based ranking algorithm, which runs over the graph of labels and identifies the importance of each label (vertex) in the graph. We use the weighted version of the ranking algorithms, as they prove particularly useful for sequence data labeling, since the dependencies between pairs of sense labels are more naturally modeled through weights indicating their strength, rather than using binary 0/1 values. Intuitively, the stationary probability associated with a vertex in the graph represents the probability of finding the walker at that vertex during the random walk, and thus it represents the importance of the vertex within the graph. In the context of sequence data labeling, the random walk is performed on the label graph associated with a sequence of words, and thus the resulting stationary distribution of probabilities can be used to decide on the most probable set of senses for the given sequence. Through the label graphs it builds for a given sequence of words, the algorithm exploits relations between word labels, and implements a concept of *recommendation*. A label recommends other related labels, and the strength of the recommendation is recursively computed based on the importance of the labels making the recommendation. In this way, the algorithm simultaneously annotates all the words in an input sequence, by identifying the most probable (most *recommended*) set of labels.

Given a sequence of words with their corresponding admissible labels, the algorithm for sequence data labeling seeks to identify a graph of label dependencies on which a random walk can be performed, resulting into a set of scores that can be used for label assignment. The algorithm consists of three main steps: (1) construction of label dependencies graph; (2) label scoring using graph-based ranking algorithms; (3) label assignment. First, a weighted graph of label dependencies is built, by adding a vertex for each admissible label, and an edge for each pair of labels for which a dependency is identified. A maximum allowable distance can be set (*MaxDist*), indicating a constraint over the distance between words for which a label dependency is sought. For

instance, if *MaxDist* is set to 3, no edges will be drawn between labels corresponding to words that are more than three words apart. Label dependencies are determined through a *Dependency* function, whose definition depends on the application and type of resources available (e.g. dictionary definitions, annotated corpora, etc.). Next, scores are assigned to vertices using a graph-based ranking algorithm. Finally, the most probable set of labels is determined by identifying for each word the label that has the highest score. Note that all admissible labels corresponding to the words in the input sequence are assigned with a score, and thus the selection of two or more most probable labels for a word is also possible.

For the particular application of word sense disambiguation, we need information on labels (word senses) and dependencies (word sense dependencies). Word senses can be easily obtained from any sense inventory, e.g. WordNet or LDOCE, or any other machine readable dictionary.

Sense dependencies can be derived in various ways, depending on the type of resources available for the language and/or domain at hand. If only a dictionary is available, a sense dependency can be defined as a measure of similarity between word senses. There are several metrics that can be used for this purpose, see for instance [4] for an overview. If sense annotated corpora are available, the similarity between two word senses can be measured as a similarity between their corresponding feature vectors, where a feature vector could include features traditionally used for this task, e.g. surrounding words and their parts of speech, syntactic dependencies, keywords in context, etc. Word sense similarities can be also derived starting with raw corpora, by bootstrapping starting with a small set of labeled examples, or in a completely unsupervised fashion, through latent semantic analysis [14].

An Example. Consider the task of assigning senses to the words in the text *The church bells no longer rung on Sundays*⁴. For the purpose of illustration, we assume at most three senses for each word, which are shown in Figure 2. Word senses and definitions are obtained from the WordNet sense inventory [22]. All word senses are added as vertices in the label graph, and weighted edges are drawn as dependencies among word senses, derived using a definition-based similarity measure inspired from the Lesk algorithm [15]. The resulting label graph is an undirected weighted graph, as shown in Figure 2. After running the ranking algorithm, scores are identified for each word-sense in the graph, indicated between brackets next to each node. Selecting for each word the sense with the largest score results into the following sense assignment: *The church#2 bells#1 no longer rung#3 on Sundays#1*, which is correct according to annotations performed by professional lexicographers.

To evaluate the application of the graph ranking algorithms to word sense disambiguation, we implemented a sense relatedness measure based on definition overlap, used as an indicator of the dependency between sense labels. Given two word senses and their corresponding definitions, the sense similarity is determined as a function of definition overlap, measured as the number of common tokens between the two definitions, after running them through a simple filter that eliminates all stop-words. To avoid

⁴ Example drawn from the data set provided during the SENSEVAL-2 English all-words task [25]. Manual sense annotations were also made available for this data.

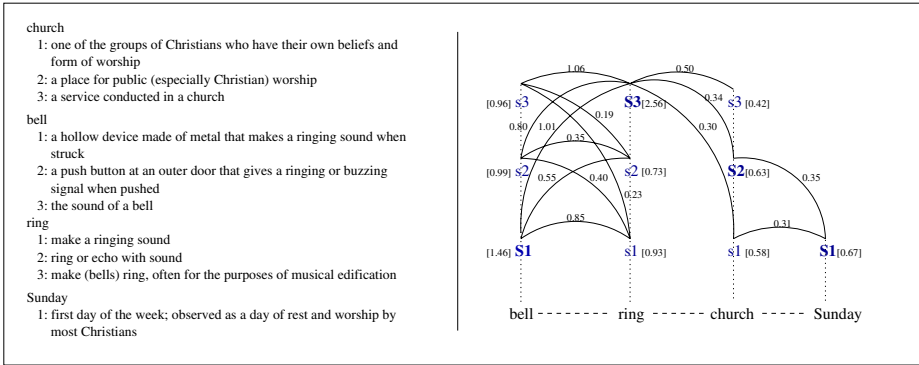


Fig. 2. Label graph for assigning senses to words in the sentence *The church bells no longer rung on Sundays*

promoting long definitions, we also use a normalization factor, and divide the content overlap of the two definitions with the length of each definition. This sense similarity measure is inspired by the definition of the Lesk algorithm [15].

The algorithm was primarily evaluated on the SENSEVAL-2 English all-words data set (and therefore sense distinctions are performed with respect to WordNet [22]), but evaluations were also run on other data sets. The performance of the algorithm is compared with the disambiguation accuracy obtained with a variation of the Lesk algorithm [15], which selects the meaning of an open-class word by finding the word sense that leads to the highest overlap between the corresponding dictionary definition and the current context. We thus compare the performance of sequence data labeling, which takes into account label dependencies and the flow of “importance” over them as implemented by the graph-based ranking algorithms, with individual data labeling, where a label is selected independent of the other labels in the text. Note that both algorithms rely on the same knowledge source, i.e. dictionary definitions, and thus they are directly comparable. Moreover, none of the algorithms take into account the dictionary sense order (e.g. the most frequent sense information provided in WordNet), and therefore they are both fully unsupervised.

Table 1 shows disambiguation results using: (a) sequence data labeling with iterative graph-based algorithms; (b) individual data labeling with a version of Lesk al-

Table 1. Disambiguation accuracy for graph-based sequence data labeling (SENSEGRAPH) and individual data labeling (LESK) on the SENSEVAL-2 data set

Part-of-speech	GRAPH RANKING		SENSEGRAPH	
	Precision	Recall	Precision	Recall
Noun	61.53%	28.86%	54.75%	25.68%
Verb	38.97%	8.75%	32.90%	7.39%
Adjective	61.05%	11.51%	53.39%	10.07%
Adverb	66.66%	7.84%	63.50%	7.47%
ALL	56.97%	56.97%	50.61%	50.61%

gorithm; and (c) random baseline. A baseline for this fully unsupervised setting consists of a random selection of senses, which results into a precision and recall of 37.9%.

The accuracy of the graph-based sequence data labeling algorithm exceeds by a large margin the individual data labeling algorithm, resulting into 12.87% error rate reduction, which is statistically significant ($p < 0.0001$, paired t-test). Performance improvements are equally distributed across all parts-of-speech, with comparable improvements obtained for nouns, verbs, and adjectives. Additional details on the algorithm, as well as evaluations on other data sets are reported in [18].

6.2 Text Processing at Document Level: Extractive Summarization

Another text processing application that finds an elegant solution within the graph-based ranking framework is the selection of sentences that are informative for the overall understanding of a given text. Iterative graph-based algorithms have the ability to identify important sentences based on the cohesive structure of a text, by taking into account the connections between sentences in a text.

For this task, the goal is to rank entire sentences, and therefore a *vertex* is added to the graph for each sentence in the text. To draw *edges* between vertices, we are defining a similarity relation, where “similarity” can be defined in various ways. In the experiments described in this paper, we use a simple cosine similarity based on a measure of text overlap. Such a relation between two sentences can be seen as a process of recommendation: a sentence that addresses certain concepts in a text, gives the reader a recommendation to refer to other sentences in the text that address the same or similar concepts. The resulting graph is highly connected, with a weight associated with each edge, and thus we use again the weighted version of the graph algorithms. The graph can be represented as: (a) simple *undirected* graph; (b) directed weighted graph with the orientation of edges set from a sentence to sentences that follow in the text (*directed forward*); or (c) directed weighted graph with the orientation of edges set from a sentence to previous sentences in the text (*directed backward*).

An Example. Figure 3 shows a text sample, and the associated weighted graph constructed for this text. The figure also shows sample weights attached to the edges connected to vertex 9, and the final score computed for each vertex, using the PageRank algorithm, applied on an undirected graph. The sentences with the highest rank are selected for inclusion in the abstract. For this sample article, sentences with id-s 9, 15, 16, 18 are extracted, resulting in a summary of about 100 words, which according to automatic evaluation measures, is ranked the second among summaries produced by 15 other systems.

We evaluate the sentence extraction algorithm in the context of a single-document summarization task [17], using 567 news articles provided during the Document Understanding Evaluations 2002 [8]. For each article, we generate a 100-words summary, by taking the sentences with the highest rank according to the graph-based ranking algorithm. For evaluation, we use the ROUGE toolkit, which is a method based on Ngram

- 3: BC-HurricaneGilbert, 09-11 339
- 4: BC-Hurricane Gilbert, 0348
- 5: Hurricane Gilbert heads toward Dominican Coast
- 6: By Ruddy Gonzalez
- 7: Associated Press Writer
- 8: Santo Domingo, Dominican Republic (AP)
- 9: Hurricane Gilbert swept toward the Dominican Republic Sunday, and the Civil Defense alerted its heavily populated south coast to prepare for high winds, heavy rains, and high seas.
- 10: The storm was approaching from the southeast with sustained winds of 75 mph gusting to 92 mph.
- 11: "There is no need for alarm," Civil Defense Director Eugenio Cabral said in a television alert shortly after midnight Saturday.
- 12: Cabral said residents of the province of Barahona should closely follow Gilbert's movement.
- 13: An estimated 100,000 people live in the province, including 70,000 in the city of Barahona, about 125 miles west of Santo Domingo.
- 14: Tropical storm Gilbert formed in the eastern Caribbean and strengthened into a hurricane Saturday night.
- 15: The National Hurricane Center in Miami reported its position at 2 a.m. Sunday at latitude 16.1 north, longitude 67.5 west, about 140 miles south of Ponce, Puerto Rico, and 200 miles southeast of Santo Domingo.
- 16: The National Weather Service in San Juan, Puerto Rico, said Gilbert was moving westward at 15 mph with a "broad area of cloudiness and heavy weather" rotating around the center of the storm.
- 17: The weather service issued a flash flood watch for Puerto Rico and the Virgin Islands until at least 6 p.m. Sunday.
- 18: Strong winds associated with the Gilbert brought coastal flooding, strong southeast winds, and up to 12 feet to Puerto Rico's south coast.
- 19: There were no reports on casualties.
- 20: San Juan, on the north coast, had heavy rains and gusts Saturday, but they subsided during the night.
- 21: On Saturday, Hurricane Florence was downgraded to a tropical storm, and its remnants pushed inland from the U.S. Gulf Coast.
- 22: Residents returned home, happy to find little damage from 90 mph winds and sheets of rain.
- 23: Florence, the sixth named storm of the 1988 Atlantic storm season, was the second hurricane.
- 24: The first, Debby, reached minimal hurricane strength briefly before hitting the Mexican coast last month.

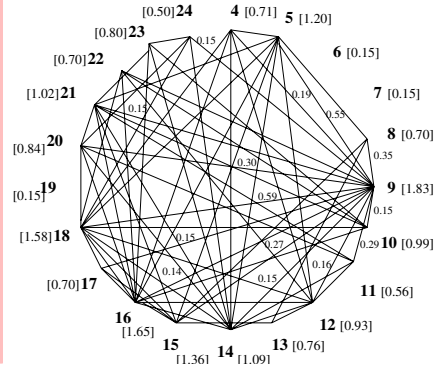


Fig. 3. Sample graph build for extractive summarization from a newspaper article

Table 2. Summarization results using a graph-based ranking approach to sentence extraction

Algorithm	Graph		
	Undirected	Dir.Forward	Dir.Backward
$HITS_A^W$	0.4912	0.4584	0.5023
$HITS_H^W$	0.4912	0.5023	0.4584
PageRank	0.4904	0.4202	0.5008

Top five DUC 2002 systems [8]					Baseline
S27	S31	S28	S21	S29	
0.5011	0.4914	0.4890	0.4869	0.4681	0.4799

statistics, found to be highly correlated with human evaluations [16]. Two manually produced reference summaries are provided, and used in the evaluation process⁵.

The summaries are evaluated using two graph-based ranking algorithms: HITS and PageRank, and Table 2 shows the results obtained with each algorithm, when using graphs that are: (a) undirected, (b) directed forward (with the orientation of edges set from a given sentence to sentences that follow in the text), or (c) directed backward (reversed orientation). For a comparative evaluation, the table also shows the results obtained on this data set by the top five (out of 15) performing systems participating in the single document summarization task at DUC 2002 [8]. It also lists the baseline performance, computed for 100-word summaries generated by taking the first sentences in each article.

The graph-based algorithms succeed in identifying the most important sentences in a text based on information exclusively drawn from the text itself. Unlike other su-

⁵ The evaluation is done using the Ngram(1,1) setting of ROUGE, which was found to have the highest correlation with human judgments, at a confidence level of 95%. Only the first 100 words in each summary are considered.

ervised systems, which attempt to learn what makes a good summary by training on collections of summaries built for other articles, these algorithms are fully unsupervised, and rely only on the given text to derive an extractive summary, which represents a summarization model closer to what humans are doing when producing an abstract for a given document.

Another interesting aspect is that these algorithms provide a ranking over all sentences in a text – which means they can be easily adapted to extracting very short summaries (headlines consisting of one sentence), or longer more explicative summaries, consisting of more than 100 words. Finally, since the algorithms do not require any training corpora, they can be adapted to other languages and domains. In fact, we have recently shown they can be successfully applied to the summarization of texts in Portuguese, without any changes in the algorithm [20].

7 Conclusions

In this paper, we suggested a framework for the application of graph-based ranking algorithms to natural language processing problems. Inspired by early work on spreading activation, random walk algorithms have been traditionally and successfully applied to structured data, such as graphs of Web links or social networks, and much less to graphs derived from unstructured texts.

We described two text processing applications that were shown to find successful solutions within the suggested framework: (1) a sentence level text processing application targeting the resolution of the semantic ambiguity of all words in unrestricted text (word sense disambiguation), and (2) a document level text processing application, targeting the ranking of sentences in a text based on their importance for the overall understanding of the text (extractive summarization). Through evaluations performed on standard benchmarks, the accuracy achieved on both applications using the graph-based ranking algorithms was shown to be competitive with that of previously proposed state-of-the-art methods. An important aspect of these algorithms is that they do not require deep linguistic knowledge, nor domain or language specific annotated corpora, which makes them highly portable to other domains, genres, or languages.

References

1. ANDERSON, J. A spreading activation theory of memory. *Journal of Verbal Learning and Verbal Behavior* 22 (1983).
2. BERGER, H., DITTENBACH, M., AND MERKL, D. An adaptive information retrieval system based on associative networks. In *Proceedings of the first Asian-Pacific conference on Conceptual modelling* (Dunedin, New Zealand, 2004).
3. BRIN, S., AND PAGE, L. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems* 30, 1–7 (1998).
4. BUDANITSKY, A., AND HIRST, G. Semantic distance in wordnet: An experimental, application-oriented evaluation of five measures. In *Proceedings of the NAACL Workshop on WordNet and Other Lexical Resources* (Pittsburgh, June 2001).
5. COHEN, P., AND KJELDSSEN, R. Information retrieval by constrained spreading activation in semantic networks. *Information Processing and Management* 23, 4 (July 1987).

6. COLLINS, A. M., AND LOFTUS, E. A spreading-activation theory of semantic processing. *Psychological Review* 82, 6 (1975).
7. DOM, B., EIRON, I., COZZI, A., AND SHANG, Y. Graph-based ranking algorithms for e-mail expertise analysis. In *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery* (San Diego, California, 2003).
8. DUC. Document understanding conference 2002, 2002. <http://www-nlpir.nist.gov/projects/duc/>.
9. FREUD, S. *Psychopathology of everyday life*. Payot, 1901.
10. GRIMMETT, G., AND STIRZAKER, D. *Probability and Random Processes*. Oxford University Press, 1989.
11. HIRST, G. Resolving lexical ambiguity computationally with spreading activation and Polaroid words. In *Lexical Ambiguity Resolution*, S. Small, G. Cottrell, and M. Tanenhaus, Eds. Morgan Kaufmann, 1988.
12. JANNINK, J. *A Word Nexus for Systematic Interoperation of Semantically Heterogeneous Data Sources*. PhD thesis, Stanford University, 2001.
13. KLEINBERG, J. Authoritative sources in a hyperlinked environment. *Journal of the ACM* 46, 5 (1999), 604–632.
14. LANDAUER, T. K., FOLTZ, P., AND LAHAM, D. Introduction to latent semantic analysis. *Discourse Processes* 25 (1998).
15. LESK, M. Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone. In *Proceedings of the SIGDOC Conference 1986* (Toronto, June 1986).
16. LIN, C., AND HOVY, E. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of Human Language Technology Conference (HLT-NAACL 2003)* (Edmonton, Canada, May 2003).
17. MIHALCEA, R. Graph-based ranking algorithms for sentence extraction, applied to text summarization. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL 2004) (companion volume)* (Barcelona, Spain, 2004).
18. MIHALCEA, R. Large vocabulary unsupervised word sense disambiguation with graph-based algorithms for sequence data labeling. In *Proceedings of the Human Language Technology / Empirical Methods in Natural Language Processing conference* (Vancouver, 2005).
19. MIHALCEA, R., AND TARAU, P. TextRank – bringing order into texts. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2004)* (Barcelona, Spain, 2004).
20. MIHALCEA, R., AND TARAU, P. An algorithm for language independent single and multiple document summarization. In *Proceedings of the International Joint Conference on Natural Language Processing (IJCNLP-2005)* (Korea, 2005).
21. MIHALCEA, R., TARAU, P., AND FIGA, E. PageRank on semantic networks, with application to word sense disambiguation. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING 2004)* (Geneva, Switzerland, 2004).
22. MILLER, G. Wordnet: A lexical database. *Communication of the ACM* 38, 11 (1995), 39–41.
23. MILLER, G., LEACOCK, C., RANDEE, T., AND BUNKER, R. A semantic concordance. In *Proceedings of the 3rd DARPA Workshop on Human Language Technology* (Plainsboro, New Jersey, 1993).
24. MOLDOVAN, D., LEE, W., AND LIN, C. Parallel knowledge processing on SNAP. *IEEE Transactions on Knowledge and Data Engineering* 5, 1 (1993).
25. PALMER, M., FELLBAUM, C., COTTON, S., DELFS, L., AND DANG, H. English tasks: all-words and verb lexical sample. In *Proceedings of ACL/SIGLEX Senseval-2* (Toulouse, France, 2001).
26. QUILLIAN, M. Semantic memory. In *Semantic Information Processing*, M. Minsky, Ed. MIT Press, 1968.

27. SCHVANEVELDT, R. *Pathfinder Associative networks: studies in knowledge organization*. Norwood, 1989.
28. SNYDER, B., AND PALMER, M. The English all-words task. In *Proceedings of ACL/SIGLEX Senseval-3* (Barcelona, Spain, July 2004).
29. SPITZER, M. *The mind within the net: models of learning, thinking, and acting*. MIT Press, 1999.
30. VANDERWENDE, L., BANKO, M., AND MENEZES, A. Event-centric summary generation. In *Proceedings of the Document Understanding Conference* (2004).
31. VERONIS, J., AND IDE, N. Word sense disambiguation with very large neural networks extracted from machine readable dictionaries. In *Proceedings of the 13th International Conference on Computational Linguistics (COLING 1990)* (Helsinki, Finland, August 1990).
32. WOLF, F., AND GIBSON, E. Paragraph-, word-, and coherence-based approaches to sentence ranking: A comparison of algorithm and human performance. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics* (Barcelona, Spain, July 2004).
33. ZOCK, M., AND BILAC, S. Word lookup on the basis of associations: from an idea to a roadmap. In *Proceedings of the Coling 2004 workshop on "Enhancing and Using Electronic Dictionaries"* (Geneva, Switzerland, August 2004).

Shallow Case Role Annotation Using Two-Stage Feature-Enhanced String Matching

Samuel W.K. Chan

Dept. of Decision Sciences, The Chinese University of Hong Kong,
Hong Kong SAR, China
swkchan@cuhk.edu.hk

Abstract. A two-stage annotation method for identification of case roles in Chinese sentences is proposed. The approach makes use of a feature-enhanced string matching technique which takes full advantage of a huge number of sentence patterns in a Treebank. The first stage of the approach is a coarse-grained syntactic parsing which is complementary to a semantic dissimilarities analysis in its latter stage. The approach goes beyond shallow parsing to a deeper level of case role identification, while preserving robustness, without being bogged down into a complete linguistic analysis. The ideas described have been implemented and an evaluation of 5,000 Chinese sentences is examined in order to justify its significances.

1 Introduction

Automatic information extraction is an area that has received a great deal of attention in recent development of computational linguistics. While a plethora of issues relating to questions of efficiency, flexibility, and portability, amongst others, have been thoroughly discussed, the problem of extracting meaning from natural texts has scarcely been addressed. When the size and quantity of documents available on the Internet are considered, the demand for a highly efficient system that identifies the semantic meaning is clear. Case frame¹, as proposed by most linguists, is one of the most important structures that can be used to represent the meaning of sentences [9]. One could consider a case frame to be a special, or distinguishing, form of knowledge structure about sentences. Although several criteria for recognizing case frames in sentences have been considered in the past, none of the criteria serves as a completely adequate decision procedure. Most of the studies in computational linguistics do not provide any hints on how to map input sentences into case frames automatically, particularly in Chinese. As a result, both the efficiency and robustness of the techniques used in information extraction is highly in doubt when they are applied to real world applications.

The objective of this research is twofold. First, a shallow but effective sentence chunking process is developed. This sentence chunking process is to extract all the

¹ Due to the lack of conciseness or conformity that authors have shown in using this and other terms, in this paper, a *case frame* is to be understood as an array of slots, each of which is labelled with a case name, and eventually possibly filled with a case filler, the whole system representing the underlying structure of an input sentence.

phrases from the input sentences, without being bogged down into deep semantic parsing and understanding. Second, a novel case role annotation technique which is based on the syntactic and semantic tags of the latest Chinese Sinica Treebank is being developed [6]. One of our primary goals in this research is to design a shallow but robust mechanism which can analyze sentences in Chinese [13, 16]. Even though the classical syntactic and semantic analysis in Chinese is extremely difficult, if not impossible, to systematize in the current computational linguistics research, our approach does not require any deep linguistic analysis to be formalized. Consequently, the annotated sentences will give piecemeal the underlying semantic representation, without being mired into the formalism.

The organization of the paper is as follows. The related work in case role identification and sentence chunking are first described in Section 2. The characteristics of the Treebank which supports our approach described in this paper will also be explained. In this research, each Chinese token will have two attributes, i.e., Part-of-Speech (*POS*) and Semantic Classes (*SC*). Any input sentence is first transformed into a feature-enhanced string. The detailed discussion on how the two-stage feature-enhanced string matching algorithm can be applied in the case role annotation is shown in Section 3. The system has already been implemented using Java language. In order to demonstrate the capability of our system, an experiment with 5,000 sentences is conducted. It is explained in Section 4 followed by a conclusion.

2 Related Work

Following the framework of case grammar which was originally proposed by Fillmore in 1968, many researchers in linguistics and philosophy have accepted that every nominal constituent in every language bears a single syntactic–semantic case relation [14, 8]. Computational techniques can be found in many earlier systems [21, 27]. Nagao *et al.* developed a powerful parser for Japanese sentences based on the case frames encoded in a verb dictionary [18]. Somers described a prototype computer program which attempts to map surface strings of English onto a formalism representing one level of a deep structure [22]. It was suggested that semantic features inherent in the main verb of a sentence can be used to infer a potential case frame for that sentence. Weischedel *et al.* [26] predicted the intended interpretation of an utterance when more than one interpretation satisfies all known syntactic and semantic constraints, and ascertained its case frames. Utsuro, Matsumoto and Nagao [24] described a method for acquiring surface case frames of Japanese verbs from bilingual corpora. They made use of translation examples in two distinct languages that have quite different syntactic structures and word meanings. Kurohashi and Nagao [15] used the case frame dictionary, which has some typical example sentences for each case frame, to select a proper case frame for an input sentence. More recently, Cook developed a matrix model and applied it to an in-depth analysis of 5,000 English clauses [7].

On the other hand, any high level language understanding process, such as case role annotation, must involve chunking sentences into segments. Motivated by the psycholinguistic evidence which demonstrates that intonation changes or pauses would affect the language understanding processes in humans [10], Abney proposes

the concept of text chunking as a first step in the full parsing [1]. A typical chunk of a text is defined as consisting of a single content word surrounded by a constellation of function words, matching a fixed template. Church also uses a simple model for finding base (non-recursive) NPs in sequence of POS tags [5]. Turning sentence chunking into a bracketing problem, Church calculates the probability of inserting both the open and close brackets between POS tags. Each chunking alternative is ranked and the best alternative is selected. Using transformation-based learning with rule-template referring to neighboring words, POS tags and chunk tags, Ramshaw and Marcus identify essentially the initial portions of non-recursive noun phrases up to the head, including determiners [19]. These chunks were extracted from the Treebank parses, by selecting NPs that contained no nested NPs. While the above approaches have been proposed to recognize common subsequences and to produce some forms of chunked representation of an input sentence, the recognized structures do not include any recursively embedded NPs. As the result, the resultant fragments bear little resemblance to the kind of phrase structures that normally appear in linguistics.

The state of the art in computational linguistics is to make use of the knowledge encoded in Treebank to analyze sentence structures. Two major issues have to be solved. First, what formalism do we assume to annotate the corpus utterances? Second, how can the trees or sub-trees be identified and how do we combine the recognized subtrees to form a complete tree for the sentence? In this paper, we address the first issue using the Sinica Chinese Treebank [3]. In contrast to the English and Chinese Penn Treebank which took a straightforward syntactic approach [17, 28], the Information-based Case Grammar (ICG) in Sinica Chinese Treebank stipulates that each lexical entry contains both semantic and syntactic features. The grammar indicates the way that lexical tokens in the sentences are related to each other. Grammatical constraints are expressed in terms of linear order of thematic roles hinted by syntactic and semantic clues. This tree structure has the advantage of maintaining phrase structure rules as well as the syntactic and semantic dependency relations. The latest version of Sinica Treebank (v.2.1), released in early 2004, contains about 55,000 trees with 300,000 words. The Treebank contains a compact bundle of syntactic and semantic information, with more than 150 different types of POS and 50 semantic roles.

On the other hand, while it may be too computationally demanding to have a full syntactic and semantic analysis of every sentence in every text, Sima'an addressed the second issue and presented a *Tree-gram* model which integrated bilexical dependencies, and conditions its substitutions based on the structural relations of the trees that are involved [20]. The *Tree-gram* model is a typical example of data-oriented parsing (DOP) advocated by Bod *et al.* [2]. The basic ideas of the *Tree-gram* model are to (i) take a corpus of utterances annotated with labeled trees; (ii) decompose every corpus tree into the bag of all its subtrees; (iii) treat the union of all these subtree bags as a stochastic tree substitution grammar, where the substitution probability of each subtree is estimated as the relative frequency of this subtree among the subtrees with the same root label. Inspired by the *Tree-gram* model, in this research, we propose a mechanism in sentence chunking and shallow case role annotation by matching any input Chinese sentence with the trees in the Treebank through a two-stage approximate pattern matching technique. Different from the stochastic tree substitution grammar proposed in the *Tree-gram* model, our approach, characterized by an optimization technique, looks for a transformation with a minimum cost, or called *edit dis-*

tance. While the concept of edit distance is commonly found in the conventional pattern matching techniques [12, 23], we take a step further in applying the technique in shallow language parsing. The detailed discussion of the algorithm is shown as follows.

3 Two-Stage Feature-Enhanced String Matching Algorithm

In this section, we will first outline the concepts of edit operations which are essential components of our feature-enhanced string matching algorithm. The two-stage shallow case role annotation will be discussed thoroughly in Section 3.2.

3.1 Edit Operations

The algorithm is essentially accomplished by applying a series of edit operations to an input sentence to change it to every tree in the Treebank. Every edit operation has been associated with a cost and the total cost of the transformation can be calculated by summing up the costs of all the operations. This edit distance reflects the dissimilarity between the input sentence and the trees. Instead of analyzing the exact Chinese tokens appearing in the sentence, extended attributes of each token in both input sentence and the trees, with their POS and semantic classes, are used. The closely matched tree, i.e., the one with minimum cost or edit distance, is selected and the corresponding phrase structures and semantic role tags delineated in the tree are unified with the input sentence.

Let two given feature-enhanced strings A and B denoted as $A = a_1a_2a_3\dots a_m$ and $B = b_1b_2b_3\dots b_n$, where are a_i, b_j the i th and j th attributed symbols of A and B respectively. Each attributed symbol represents a primitive of A or B . Generally speaking, to match a feature-enhanced string A with another B means to transform or edit the symbols in A into those in B with a minimum-cost sequence of allowable edit operations. In general, the following three types of edit operations are available for attributed symbol transformation.

- (a) *Change*: to replace an attributed symbol a_i with another b_j , denoted as $a_i \rightarrow b_j$.
- (b) *Insert*: to insert an attributed symbol b_j into a feature-enhanced string, denoted as $\lambda \rightarrow b_j$ where λ denotes a null string.
- (c) *Delete*: to delete an attributed symbol a_i from a feature-enhanced string, denoted as $a_i \rightarrow \lambda$.

[Definition 1]

An *edit sequence* is a sequence of ordered edit operations, s_1, s_2, \dots, s_p where s_i is any of the following three types of edit operations, *Change, Insert, Delete*.

[Definition 2]

Let R be an arbitrary nonnegative real cost function which defines a cost $R(a_i \rightarrow b_j)$ for each edit operation $a_i \rightarrow b_j$. The cost of an edit sequence $S = s_1, s_2, \dots, s_p$ to be

$$R(S) = \sum_{i=1}^p R(s_i) \quad (1)$$

[Definition 3]

For two strings A and B with length m and n respectively, $D(i, j)$ denotes the edit distance, which is the minimum number of edit operations, needed to transform the first i characters of A into first j characters of B , where $1 \leq i \leq m$ and $1 \leq j \leq n$.

In other words, if A has m letters and B has n letters, then the edit distance of A and B is precisely the value $D(m, n)$. Wagner & Fischer had proposed the following algorithm for computing every edit distances $D(i, j)$ [25].

[Algorithm A]

```

D(0, 0) := 0;
for i := 1 to len(A) do D(i, 0) := D(i-1, 0) + R(ai→λ);
for j := 1 to len(B) do D(0, j) := D(0, j-1) + R(λ→bj);
for i := 1 to len(A) do
  for j := 1 to len(B) do
    begin
      m1 := D(i, j-1) + R(λ→bj);
      m2 := D(i-1, j) + R(ai→λ);
      m3 := D(i-1, j-1) + R(ai→bj);
      D(i, j) := min (m1, m2, m3);
    end
  end
end

```

Our feature-enhanced string matching in case role annotation is to make use of the algorithm above and modify the cost function $R(\cdot)$ for various edit operations.

3.2 Shallow Case Role Annotation as Two-Stage Feature-Enhanced String Matching

Our shallow sentence parsing is defined as a two-stage feature-enhanced string matching using the edit operations. For every input sentence, a coarse-grained syntactic matching is conducted in our first stage of matching. The matching relies on a set of coarse-grained but global Part-Of-Speech (POS) tags. The major objective of this stage is to shortlist all the potential trees, which are relevant to the input sentence, without getting bogged down into computational complexity with other linguistic details. The second stage of the matching is followed to compute the dissimilarity measure between the input sentence and every short-listed candidate that is identified in the first stage. Detailed POS and semantic class (SC) tags will be employed. As a result, a candidate tree which has the minimum dissimilarity with the input sentence will be identified. The underlying case roles and phrases of the candidate tree are used to determine the shallow language patterns of the input sentence. The details of the two-stage matching are explained in the following.

3.2.1 Coarse-Grained Syntactic Matching

In the first stage of matching, each Chinese token are represented by their corresponding part-of-speech (POS). Let S be an input sentence and the T be a tree in the Sinica Treebank, s_i and t_j be two tokens in S and T with attribute $\langle POS_i \rangle$ and $\langle POS_j \rangle$ respectively. We define the cost function for a *change* operation $s_i \rightarrow t_j$ to be

$$R(s_i \rightarrow t_j) = u(POS_i, POS_j) \quad (2)$$

where $u(POS_i, POS_j)$ defines the cost due to the difference between the POS of the tokens respectively. The POS tags from the Chinese Knowledge Information Processing Group (CKIP) of Academia Sinica are employed [4]. The tags are subdivided into 46 main POS classes which are further refined into more than 150 subtypes. However, in this coarse-grained matching, only the main POS classes will be used. In order to figure out the cost function $u(\cdot, \cdot)$ in the coarse-grained matching, all the main POS tags are organized into a tree structure with an associated hard-coded cost function. Figure 1 shows a subtree of notional words and describes the relative distances between the adjectives (A), verbs (V), status-verbs (VH), measure-words (Nf), nouns (N), position-words (Ng), time-words (Nd) and place-words (Nc). All notional words have definite meanings in the language. The cost function is based on their interchangeability, the degree of flexibility in placement in the syntax, and the similarity of their acceptable modifiers [11, 29]. For example, in Chinese, verbs and adjectives share a lot of common features, i.e., both can be predicates, or can be modified by adverbs and the word, *not*. All these features fail to appear in nouns. The abbreviations in bracket indicate the original POS tags marked by the CKIP. The corresponding tree structure of the XML is shown in Figure 2 as below.

```

<Head toll="5">
  <NodeB toll="2">
    <NodeC toll="2">
      <Adjective toll="5"/>
      <Verb toll="5"/>
    </NodeC>
    <Status-Verb toll="7"/>
  </NodeB>
  <NodeD toll="2">
    <Measure-Word toll="7"/>
    <NodeE toll="2">
      <Noun toll="5"/>
      <NodeF toll="2">
        <Position-word toll="3"/>
        <NodeG toll="1">
          <Time-word toll="2"/>
          <Place-word toll="2"/>
        </NodeG>
      </NodeF>
    </NodeE>
  </NodeD>
  ...
</Head>

```

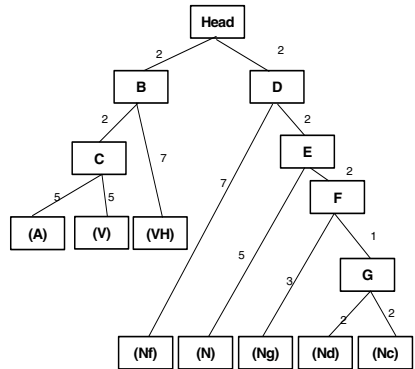


Fig. 1. XML illustrating the relative distances between 8 different types of POS

Fig. 2. Corresponding tree structure of the XML shown in Fig.1

The cost function $u(\cdot, \cdot)$ will reflect the difference based on the tag toll encoded in the XML as shown in Figure 1. For example, the cost for changing a word having POS from Adjective to Verb,

$$u(\text{Adjective}, \text{Verb}) = \text{toll}(\text{Adjective} \rightarrow \text{NodeC}) + \text{toll}(\text{NodeC} \rightarrow \text{Verb}) = 5 + 5 = 10$$

Similarly, $u(\text{Verb}, \text{Noun}) = \text{toll}(\text{Verb} \rightarrow \text{NodeC}) + \text{toll}(\text{NodeC} \rightarrow \text{NodeB}) + \text{toll}(\text{NodeB} \rightarrow \text{Head}) + \text{toll}(\text{Head} \rightarrow \text{NodeD}) + \text{toll}(\text{NodeD} \rightarrow \text{NodeE}) + \text{toll}(\text{NodeE} \rightarrow \text{Noun}) = 5 + 2 + 2 + 2 + 2 + 5 = 18$

The function $u(\cdot, \cdot)$ partially indicates the alignment of the syntactic structure of the input sentence and the trees in the Treebank. Although two feature-enhanced strings with the same POS sequence do not imply they will share the same syntactic structure, this coarse-grained syntactic matching shortlists the potential trees by imposing a necessary, even not sufficient, constraint on its syntactic structure and limits the potential search space in the subsequent stage of semantic matching.

3.2.2 Computation of Semantic Dissimilarity

What this second stage matching basically does is to make a detailed comparison between the input sentence with the short-listed trees in the earlier stage. In this stage, each Chinese token has two attributes, i.e., a detailed part-of-speech (POS) and semantic class (SC). Similar to the approach in Section 3.2.1, we define the cost function for a *change* operation $s_i \rightarrow t_j$ to be

$$R(s_i \rightarrow t_j) = u(POS_i, POS_j) + v(SC_i, SC_j) \tag{3}$$

where the second term $v(SC_i, SC_j)$ measures their semantic differences and $u(POS_i, POS_j)$ defines the partial cost due to the difference between the detailed POS of the tokens. The detailed POS tags are organized in XML format, similar to the approach demonstrated in Figure 1. Figure 3 shows the further breakdown of the nouns (Na) which is divided into in-collective (Nae) and collective (Na1) nouns. The collective nouns are then subdivided into in-collective concrete uncountable nouns (Naa), in-collective concrete countable nouns (Nab), in-collective abstract countable nouns (Nac), in-collective abstract uncountable nouns (Nad). The figure associated with the arcs in the Figure 3 illustrates the cost function.

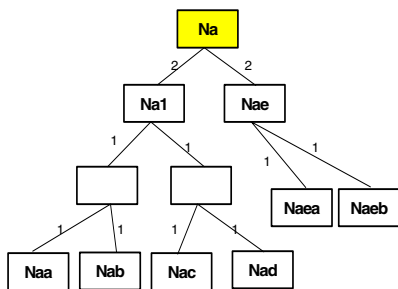


Fig. 3. Tree structure of Nouns (Na) based on the CKIP Academia Sinica

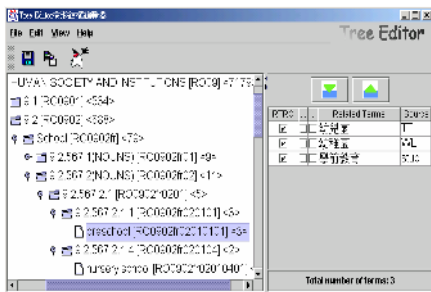


Fig. 4. *Is-a* hierarchy in the bilingual thesaurus

The second term in equation (3) defines the other partial cost due to the semantic differences. In our approach, the lexical tokens in the input sentences and the trees are identified using a lexical source similar to the Roget’s Thesaurus. Our semantic classes are defined using the bilingual thesaurus with an *is-a* hierarchy. The *is-a* hierarchy, shown the underlying ontology, can be viewed as a directed acyclic graph with a single root. Figure 4 shows one of our *is-a* hierarchies in our bilingual thesaurus using our Tree Editor. While the upward links correspond to generalization, the spe-

cialization is represented in the downward links. The hierarchies demonstrated in the thesaurus are based on the idea that linguists classify lexical items in terms of similarities and differences. They are used to structure or rank lexical items from more general to the more special. Based on the *is-a* hierarchy in the thesaurus, we define the conceptual distance d between two notional words by their shortest path lengths. Given two tokens t_1 and t_2 in an *is-a* hierarchy of the thesaurus, the distance d between the tokens is defined as follows:

$$d(t_1, t_2) = \text{minimal number of } is-a \text{ relationships in the shortest path between } t_1 \text{ and } t_2 \tag{4}$$

The shortest path lengths in *is-a* hierarchies are calculated. Initially, a search fans out through the *is-a* relationships from the original two nodes to all nodes pointed to by the originals, until a point of intersection is found. The paths from the original two nodes are concatenated to form a continuous path, which must be a shortest path between the originals. The number of links in the shortest path is counted. Since $d(t_1, t_2)$ is positive and symmetric, $d(t_1, t_2)$ is a metric which means (i) $d(t_1, t_1) = 0$; (ii) $d(t_1, t_2) = d(t_2, t_1)$; (iii) $d(t_1, t_2) + d(t_2, t_3) \geq d(t_1, t_3)$. At the same time, the semantic similarity measure between the items is defined by:

$$v(t_i, t_j) := \begin{cases} d(t_i, t_j) & \text{if } d(t_i, t_j) \leq d_{max} \\ MaxInt & \text{otherwise} \end{cases} \tag{5}$$

where d_{max} is proportional to the number of lexical items in the system and $MaxInt$ is a maximum integer of the system. This semantic similarity measure defines the degree of relatedness between tokens. Obviously, strong degree of relatedness exists between the lexical tokens under the same nodes.

For the cost of the insert and delete operations, we make use the concept of *collocation* which measures how likely two tokens are to co-occur in a window of text. To better distinguish statistics based ratios, work in this area is often presented in terms of the mutual information, which is defined as

$$MI(t_{j-1}, t_j) = \log_2 \frac{P(t_{j-1}, t_j)}{P(t_{j-1}) \times P(t_j)} \tag{6}$$

where t_{j-1} and t_j are two adjacent tokens. While $P(x, y)$ is the probability of observing x and y together, $P(x)$ and $P(y)$ are the probabilities of observing x and y anywhere in the text, whether individually or in conjunction. Note that tokens that have no association with each other and co-occur together according to chance will have a mutual information number close to zero. This leads to the cost function for insertion and deletion shown in equation (7) and (8) respectively.

$$R(\lambda \rightarrow t_j) = \begin{cases} K \times |z| & \text{if } 0 \geq z > \epsilon \\ MaxInt & \text{otherwise} \end{cases} \tag{7}$$

where $z = \min \{MI(t_{j-1}, t_j), MI(t_j, t_{j+1})\}$

$$R(t_j \rightarrow \lambda) = \begin{cases} L \times |MI(t_{j-1}, t_{j+1})| & \text{if } 0 \geq MI(t_{j-1}, t_{j+1}) > \epsilon \\ MaxInt & \text{otherwise} \end{cases} \tag{8}$$

where K, L, ϵ are three constants relied on the size of the active corpus.

Obviously, the insertion operation will be penalized if the co-occurrence between the newly inserted token and its neighbors is low. Similarly, the deletion operation is most likely to happen if there is a high co-occurrence between the adjacent pairs after the deletion. Using the above cost functions for the three types of edit operations, the tree in the Treebank with minimum cost is identified to best approximation of the input sentence *S* and its relevant case roles tags and phrase structures will be adopted. Shallow language patterns are then extracted based on the recursive structures and case role tags appeared in the Treebank. The experimental results and an illustration of the patterns extracted are shown in the following section.

4 An Illustration and Experimental Results

We have implemented the system using Java JDK1.4.2 under Sun Microsystems. The whole system development is designed under Unified Modeling Language (UML). In our design, for every input sentence, the best matching tree with minimum edit distance in the Treebank is calculated. The Information Case Grammar (ICG) of the best matching tree in the Treebank will be adopted. In order to clarify what we have discussed in the above sections, an illustration is shown as follows.

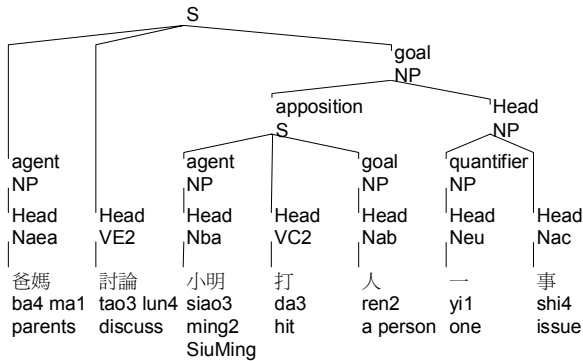


Fig. 5. Tree in the Treebank which closely matches, edit distance equal to 15, with the input sentence shown in (S)

The sentence

議員商討總統出兵一事

(in English, *The senators discuss the issue on sending troops initiated by the president*) (S)

has a small edit distance, equal to 15, with the tree shown in Figure 5, the sentence is then chunked into phrases, 〈議員〉 (*The senators*), 〈商討〉 (*discuss*), and 〈總統出兵一事〉 (*the issue on sending troops initiated by the president*), which are further tagged with *agent*, *act*, and *goal* respectively by taking the advantage of annotation in the Treebank. Certainly, the phrase 〈總統出兵一事〉 (*the issue on sending troops initiated by the president*) can be further chunked into more details 〈總統出兵〉 (*sending troops initiated by the president*), 〈一事〉 (*the issue*). This chunking not only provides the basic semantic tag for each constituent, it also reflects the language patterns of the

input sentence. As shown in Table 1, the shallow language patterns extracted are indicated by the square brackets together with the explicit semantic tags. While the sentence pattern is marked with @SP [...], the embedded phrases are marked by different tags, such @AP [...] for apposition phrase, or @PP [...] for position phrase.

Table 1. Shallow language patterns extracted from the input sentence (S)

@SP[議員商討總統出兵一事	<i>The senators discuss the issue on sending troops initiated by the president</i>
Agent	〈議員〉	<i>The senators</i>
Act	〈商討〉	<i>Discuss</i>
Goal	@AP[〈總統出兵〉]AP, @NP[〈一事〉]NP	@AP[(<i>sending troops initiated by the president</i>)]AP, @NP[(<i>the issue</i>)]NP
]SP		
@AP[總統出兵	<i>sending troops initiated by the president</i>
Agent	總統	<i>the president</i>
Act	出	<i>Sending</i>
Goal	兵	<i>Troops</i>
]AP		

We have tested our shallow case role annotation with 5,000 input sentences. The detailed results are shown in Table 2. The average sentence length is around 10.5 characters per sentence. It is worthwhile to mention that, as shown in column (e) of Table 2, more than 500 sentences have incomplete information which mainly comes from unknown words and proper nouns. Both of them have neither defined POS nor semantic class in our bilingual thesaurus. While the boundaries between words and phrases in Chinese are not easy to differentiate, the performance, due to the coverage of POS and semantic classes, does not deteriorate much in our system. This tolerance ability provides the graceful degradation in our case role annotation. While other systems are brittle and working only in all-or-none basis, the robustness of our system is guaranteed.

Table 2. Analysis of 5,000 sentences in the experiment. Edit distance is defined as a minimum cost in transforming the input sentence with the closest sentence pattern in the Treebank. The smaller the distance, the higher similarity they have.

Edit distance (a)	# of sentences (b)	Average # of tokens (c)	Average edit distance (d)	# of sentences with incomplete information (e)
0-25	336	5.24	21.06	8
26-50	1220	6.40	38.10	132
51-75	1285	7.30	62.10	175
76-100	2159	6.55	90.94	282

As with other text analysis, the effectiveness of the system appears to be dictated by recall and precision parameters where recall (R) is a percentage of how many correct case roles can be identified while precision (P) is the percentage of case roles, tackled by the system, which are actually correct. In addition, a common parameter F is used as a single-figure measure of performance as in follows,

$$F = \frac{(\beta^2 + 1) \times P \times R}{\beta^2 \times P + R} \quad (9)$$

We set $\beta=1$ to give no special preference to either recall or precision. The recall, precision and F value are 0.84, 0.92 and 0.878 respectively.

5 Conclusion

In this paper, we have illustrated a shallow technique in which sentence patterns are extracted in forms of chunks of phrases or words using a two-stage feature-enhanced string matching algorithm. While the first stage is to shortlist the potential trees in the Treebank, chunks are further tagged with case roles in the second stage. Our approach does not require a full syntactic parse to pursue semantic analysis and the recursively embedded phrases can also be identified without pain. This shallow case role annotation is inspired by the research in the area of bio-molecular sequences analysis which advocates *high sequence similarity usually implies significant function or structural similarity*. It is characteristic of biological systems that objects have a certain form that has arisen by evolution from related objects of similar but not identical form. This *sequence-to-structure* mapping is a tractable, though partly heuristic, way to search for functional or structural universality in biological systems. With the support from the results as shown in this paper, we conjecture this *sequence-to-structure* phenomenon appears in our sentences. The sentence sequence encodes and reflects the more complex linguistic structures and mechanisms described by linguists. While our system does not claim to deal with all aspects of language, we suggest an alternate, but plausible, way to handle the real corpus.

Acknowledgement

The work described in this paper was partially supported by the grants from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project Nos. CUHK4438/04H and CUHK4706/05H).

References

- [1] Abney, S. (1991). Parsing by chunks. In Berwick, R., Abney, S. & Tenny, C. (Eds.), *Principle-Based Parsing*. Kluwer Academic.
- [2] Bod, R., Scha, R., & Sima'an, K. (2003). *Data-Oriented Parsing*. Stanford: California, CSLI.
- [3] Chen, F.-Y., Tsai, P.-F., Chen, K.-J., & Huang, C.-R. (2000). Sinica Treebank. [in Chinese] *Computational Linguistics and Chinese Language Processing*, 4, 2, 87-103.
- [4] Chen, K.-J., Huang, C.-R., Chang, L.-P., & Hsu, H.-L. (1996). Sinica Corpus: Design Methodology for Balanced Corpora. *Proceedings of the 11th Pacific Asia Conference on Language, Information, and Computation (PACLIC II)*, Seoul Korea, 167-176.
- [5] Church, K. (1988). A stochastic parts program and noun phrase parser for unrestricted text. *Proceedings of Second Conference on Applied Natural Language Processing*, Austin, Texas.
- [6] CKIP (2004). *Sinica Chinese Treebank: An Introduction of Design Methodology*. Academic Sinica.
- [7] Cook, W.A. (1998). *Case Grammar Applied*. Summer Institute Linguistics.

- [8] Dowty, D. (1991). Thematic proto-roles and argument selection. *Language*, 67, 547-619.
- [9] Fillmore, C.J. (1968). The case for case. In E. Bach & R.T. Harms (Eds.), *Universals in Linguistic Theory*, 1-90. Holt, Rinehart & Winston.
- [10] Gee, J., & Grosjean, F. (1983). Performance structures: A psycholinguistic and linguistic appraisal. *Cognitive Psychology*, 15, 4, 411-458.
- [11] Guo, R. (2002). *Xiandai Hanyu Cilei Yanjiu*. Commercial Press. [In Chinese: 郭銳 (2002) 《現代漢語詞類研究》商務印書館]
- [12] Gusfield, D. (1997). *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge University Press.
- [13] Her, O. S. (1990). *Grammatical Functions and Verb Subcategorization in Mandarin Chinese*. The Crane publishing Co.
- [14] Jackendoff, R. (1983). *Semantics and Cognition*. MIT Press.
- [15] Kurohashi, S., and Nagao, M. (1994). A method of case structure analysis for Japanese sentences based on examples in case frame dictionary. *IEICE Transactions on Information and Systems*, vol. E77-D, no. 2, 227-239.
- [16] Li, Y. C. (1971). *An investigation of Case in Chinese Grammar*. Set On Hall University Press.
- [17] Marcus, M., Santorini, B. and Marcinkiewicz, M. (1993). Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19, 2, 313-330.
- [18] Nagao, M., Tsujii, J., and Tanaka, K. (1976). Analysis of Japanese sentences by using semantic and contextual information-semantic analysis. *Information Processing Society of Japan*, 17, 1, 10-18.
- [19] Ramshaw, L. A., & Marcus, M.P. (1995). Text chunking using transformation-based learning. *Proceedings of the Third Workshop on Very Large Corpora*, 82-94.
- [20] Sima'an, K. (2000). Tree-gram parsing: lexical dependencies and structural relations. *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, 53-60, Hong Kong.
- [21] Simmons, R.F. (1970). Natural language question answering systems. *Communications of ACM*, 13, 15-30.
- [22] Somers, H.L. (1982). The use of verb features in arriving at a 'meaning representation'. *Linguistics*, 20, 237-265.
- [23] Tsay, Y.T., & Tsai, W.H. (1989). Model-guided attributed string matching by split-and-merge for shape recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, 3, 2, 159-179.
- [24] Utsuro, T., Matsumoto, Y., and Nagao, M. (1993). Verbal case frame acquisition from bilingual corpora. *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, vol. 2, 1150-1156.
- [25] Wagner, R.A., & Fischer, M.J. (1974). The string-to-string correction problem. *Journal of the Association for Computing Machinery*, 21, 1, 168-173.
- [26] Weischedel, R., Meteer, M., Schwartz, R., Ramshaw, L., Palmucci, J. (1993). Coping with ambiguity and unknown words through probabilistic models. *Computational Linguistics*, 19, 2, 359-382.
- [27] Wilks, Y.A. (1972). *Grammar, Meaning and the Machine Analysis of Language*. Routledge.
- [28] Xia, F., Palmer, M., Xue, N., Okurowski, M.E., Kovarik, J., Chiou, F.-D., Huang, S., Kroch, T., & Marcus, M. (2000). Developing Guidelines and Ensuring Consistency for Chinese Text Annotation. *Proceedings of the second International Conference on Language Resources and Evaluation (LREC-2000)*, Athens, Greece.
- [29] Yu, S. (2003). *The Grammatical Knowledge-Base of Contemporary Chinese – A Complete Specification*. TsingHua University Press. [In Chinese: 俞士汶 (2003) 《現代漢語語法信息詞典詳解》清華大學出版社]

SPARTE, a Test Suite for Recognising Textual Entailment in Spanish

Anselmo Peñas, Álvaro Rodrigo, and Felisa Verdejo

Dpto. Lenguajes y Sistemas Informáticos, UNED
{anselmo, alvarory, felisa}@lsi.uned.es

Abstract. The aim of Recognising Textual Entailment (RTE) is to determine whether the meaning of a text entails the meaning of another text named hypothesis. RTE systems can be applied to validate the answers of Question Answering (QA) systems. Once the answer to a question is given by the QA system, a hypothesis is built turning the question plus the answer into an affirmative form. If the text (a given document) entails this hypothesis, then the answer is expected to be correct. Thus, a RTE system becomes an Answer Validation system. Within this framework the first problem is to find collections for training and testing RTE systems. We present here the SPARTE corpus aimed at evaluating RTE systems in Spanish. The paper presents the methodology to build SPARTE from the Spanish QA assessments performed at the Cross-Language Evaluation Forum (CLEF) during the last three editions. The paper also describes the test suite and discusses the appropriate evaluation measures together with their baselines.

1 Introduction

The task of Recognising Textual Entailment (RTE) [3] aims at deciding whether the truth of a text entails the truth of another text named hypothesis or, in other words, if the meaning of the hypothesis is enclosed in the meaning of the text. The entailment relation between texts is useful for a variety of tasks as, for example, Automatic Summarisation, where a system could eliminate the passages whose meaning is already entailed by other passages; or Question Answering (QA), where the answer of a question must be entailed by the text that supports the correctness of the answer.

Since RTE task has been defined recently, there exists only few corpora for training and testing RTE systems, and none of them are in Spanish. Thus, we planned the development of SPARTE, a corpus for training and testing RTE systems in Spanish, and specially, systems aimed at validating the correctness of the answers given by QA systems. This automatic Answer Validation would be useful for improving QA systems performance and also for helping humans in the assessment of QA systems output.

SPARTE has been built from the Spanish corpora used at Cross-Language Evaluation Forum (CLEF) for evaluating QA systems during 2003, 2004 and 2005. At the end of development, SPARTE contains 2962 hypothesis with a document label and a TRUE/FALSE value indicating whether the document entails the hypothesis or not.

Section 2 describes the development of SPARTE in detail. Section 3 evaluates some features of the corpus. Section 4 discusses and suggests the way of using SPARTE for evaluation purposes. Section 5 is devoted to some other corpora related to RTE. Finally, we give some conclusions and future work.

2 Development of SPARTE

SPARTE is a training and testing corpus for RTE systems, containing text and hypothesis pairs together with a TRUE/FALSE value indicating whether the text entails the hypothesis or not. The hypothesis have been built from the questions and answers used in the evaluation of QA systems at CLEF. Next subchapters describes in detail the methodology followed.

2.1 Original Corpus

The starting point for development of SPARTE were the Spanish corpora used at the Cross-Language Evaluation Forum (CLEF) for evaluating Spanish Question Answering (QA) systems during the last three years [7] [9] [8] [5] [11]. The organization provided the participants with the set of questions they had to answer, and a large document collection where the systems had to find and extract the answers. The Spanish collection contains 454,045 news in Spanish from the EFE News Agency for the years 1994 and 1995.

Each year, the participant systems submitted up to two runs responding 200 questions per year. Together with the answer string, the systems must indicate the document that supports the correctness of the answer. Occasionally, systems give NIL to indicate that the question had no answer in the collection. Table 1 gives an idea of the corpus size, showing the number of question and answer pairs available at the beginning of the development. Each answer was assessed by humans in order to decide whether it was correct, exact and supported by the given document or not.

Table 1. Number of question answer pairs for SPARTE development

Year	#questions	#answers	#runs	#participants	#q-a pairs
2003	200	3	2	1	1200
2004	200	1	2	5	1600
pilot 2004	100	-	1	1	100
2005	200	1	2	9	3598
					6498

2.2 Building the Hypothesis

Since textual entailment was defined between statements, the first step was to turn the questions into an affirmative form. For example, the question “*Which is the capital of Croatia?*” was transformed into “*The capital of Croatia is <answer/>*”, where the mark “<answer/>” has to be instantiated with any answer given to that question by any system. In this way, we prepared the corpus to build all the hypothesis automatically by substituting the mark with the corresponding answers.

Figure 1 shows the xml format used at this stage. The *answer* mark inside the hypothesis is not instantiated yet. The *instance* marks contain an answer that will substitute the *answer* mark to complete the hypothesis. Instances include the document identification

```

<case id="1">
  <question>
    ¿Cuál es la capital de Croacia?
  </question>
  <hypothesis>
    La capital de Croacia es <answer/>
  </hypothesis>
  <instance id="1" text="EFE19940127-14481" eval="R">
    Zagreb
  </instance>
  <instance id="2" text="EFE19941119-11475" eval="W">
    Fuerzas de la ONU
  </instance>
  <instance id="3" text="EFE19940907-03455" eval="W">
    ONU
  </instance>
  <instance id="4" text="EFE19940907-03366" eval="W">
    Bosnia-Herzegovina
  </instance>
</case>

...

<case id="88">
  <question>
    ¿En qué año Kuwait fue invadido por Irak?
  </question>
  <hypothesis>
    Kuwait fue invadido por Irak en el año <answer/>
  </hypothesis>
  <instance id="1" text="EFE19950202-00912" eval="X">
    2 de agosto de 1990
  </instance>
  <instance id="2" text="EFE19940326-16845" eval="R">
    1990
  </instance>
  <instance id="3" text="EFE19950411-06234" eval="U">
    1990
  </instance>
  <instance id="4" text="EFE19950731-19147" eval="W">
    Bagdad
  </instance>
</case>

```

Fig. 1. XML for the hypothesis templates

and the assessment given to the answer by a human: correct (R), incorrect (W), inexact (X) or unsupported (U).

Repeated answers (instances) and NIL answers have been removed. NIL answers might be correct or not, but in any case, NIL stands for the absence of answer and therefore, there is no answer to validate.

```

<pair id="1" value="TRUE" task="QA">
  <q>
    ¿Cuál es la capital de Croacia?
  </q>
  <t doc="EFE19940127-14481"> </t>
  <h>
    La capital de Croacia es Zagreb
  </h>
</pair>
...
<pair id="614" value="TRUE" task="QA">
  <q>
    ¿Qué torneo ganó Andrei Medvedev?
  </q>
  <t doc="EFE19940424-13985"> </t>
  <h>
    Andrei Medvedev ganó el torneo de Montecarlo
  </h>
</pair>
...
<pair id="26" value="FALSE" task="QA">
  <q>
    ¿Qué país ganó la Copa Davis?
  </q>
  <t doc="EFE19940406-02726"> </t>
  <h>
    Roland Garros ganó la Copa Davis
  </h>
</pair>
...
<pair id="58" value="FALSE" task="QA">
  <q>
    ¿Quién era conocido como el "Zorro del Desierto"?
  </q>
  <t doc="EFE19940205-02731"> </t>
  <h>
    Laguna del Desierto era conocido como el
    "Zorro del Desierto"
  </h>
</pair>

```

Fig. 2. SPARTE corpus excerpt

Notice that no snippet, short passage or sentence is explicitly given for the text, but the identification of a whole document. The reason is that the current assessment at CLEF requests a whole document for supporting the answer. Thus, the answer can be supported by a conjunction of sentences not necessarily in consecutive order inside the document. In other words, verifying the truth of the hypothesis could require more than one inter-related sentences from the document. From our point of view, this approach is realistic, leaving to the RTE system developers the decision of managing the whole text or only a passage extracted previously, containing the answer string.

2.3 Building the Text-Hypothesis Pairs

Once the answers are grouped as possible instances for building the hypothesis, the next step is to build the text-hypothesis pairs with the entailment TRUE/FALSE value. Figure 2 shows the xml format that conforms the final SPARTE structure. The mark `<pair>` includes the TRUE/FALSE value to indicate whether the text entails the hypothesis or not. Inside `<pair>` there are three marks: one containing the original question, a second containing the document identification, and a third containing the hypothesis to be validated with the document.

The hypothesis has been generated automatically from the hypothesis template instantiated with each answer. Thus, some wrong answers could give to the hypothesis not only a wrong semantics but also a wrong syntactic structure (see Figure 3). In our opinion this is a desirable feature for the corpus, allowing the development of syntactic criteria for RTE and also promoting the development of systems robust to some formal and syntactic errors.

```
<pair id="51" value="FALSE" task="QA">
  <q>
    ¿Qué es UNICEF?
  </q>
  <t doc="EFE19950126-152091"'> </t>
  <h>
    UNICEF es China con
  </h>
</pair>
```

Fig. 3. SPARTE sample with syntactic errors

2.4 Determining the Entailment Value

Each text-hypothesis pair in the corpus has associated an entailment value to indicate if the meaning of the hypothesis can be derived from the document. However, the result of the QA assessment was not a binary value and, therefore, a simple mapping is necessary for converting QA assessments into entailment values. We followed this criteria:

- *Correct answer (R)*: the answer to the question is correct and the document support its correctness. Then the text entails the hypothesis and the entailment value is TRUE.

- *Unsupported answer (U)*: although the answer is correct, the document does not permit to affirm the correctness of the answer. The text does not entail the hypothesis and the entailment value is FALSE.
- *Inexact answer (X)*: This is a difficult case also for the human assessors. There is no additional information to decide whether the answer contains too much information or, in the contrary, the answer string is too short to be considered as a correct one. Both cases were tagged as inexact. In the short cases, the resulting hypothesis could be true in the document, although they are not valid answers. In the long cases, although the answer string contains a correct answer the resulting hypothesis might have a different meaning and be false because of the extra information. Thus, we do not have a clear criteria to determine the entailment values in these cases without a human assessment. Fortunately, there are only few cases (6% of the pairs) and we opted for excluding them from the final SPARTE corpus.
- *Incorrect answer (W)*: the answer to the question is wrong. Although the answer could be directly extracted from the text, the joint reformulation of question and answer as a statement (hypothesis) makes very difficult the entailment between text and hypothesis. Therefore, the entailment value in this cases is FALSE. However, we detected few cases where the answer is not responsive but it becomes a hypothesis true in the text. For example, “Japan” was considered a not responsive answer (wrong answer) to the question “Where did explode the first atomic bomb?”. However, the resulting hypothesis “The first atomic bomb exploded in Japan” is true in the text. We have studied a 10% of the FALSE pairs in the final corpus finding that a 4% of the hypothesis FALSE (3% of all hypothesis) could be considered TRUE in the text. In other words, although the source answers can be considered not responsive, and in fact the corresponding pairs have been tagged with an entailment value FALSE, they can be found TRUE in the corresponding texts. However, this source of errors is in the same range than the disagreement between the human annotators that made the QA assessments [11].

3 Evaluation of SPARTE

We performed a partial human evaluation of the corpus in order to assess the quality of SPARTE. We took randomly the 10% of TRUE pairs and the 5% of the FALSE ones (see Table 2).

Table 2. Manual evaluation of SPARTE

	Considered	Correct	Incorrect
Pairs TRUE	70 (10% of TRUEs)	67 (96%)	3 (4%)
Pairs FALSE	113 (5% of FALSEs)	111 (98%)	2 (2%)
Total	183 (6%)	178 (97%)	5 (3%)

We found that errors are in the same range as inter-annotator disagreement in the QA assessments (less than 5%). In fact, the errors we found come from some wrong

```

<pair id="9" value="TRUE" task="QA">
  <q>
    ¿Cuál es el nombre de pila del juez Borsellino?
  </q>
  <t doc="EFE19940718-10595"> </t>
  <h>
    Paolo Borsellino es el nombre de pila del
    juez Borsellino
  </h>
</pair>
...
<pair id="398" value="TRUE" task="QA">
  <q>
    ¿Qué iglesia aprobó los nuevos cánones para la
    ordenación de mujeres?
  </q>
  <t doc="EFE19941202-00867"> </t>
  <h>
    La iglesia Sínodo de la Iglesia Anglicana aprobó los
    nuevos cánones para la ordenación de mujeres
  </h>
</pair>

```

Fig. 4. Sample with an incorrect TRUE

```

<pair id="144" value="FALSE" task="QA">
  <q>
    ¿A cuántos años de prisión fue sentenciado Bettino
    Craxi?
  </q>
  <t doc="EFE19951103-01683"> </t>
  <h>
    Bettino Craxi fue condenado a ocho años de prisión
  </h>
</pair>

```

Fig. 5. Sample with an incorrect FALSE

QA assessments. Some examples are given in Figures 4 and 5. In the first case, incorrect TRUE pairs are due to *Inexact* answers that were judged as *Right*. In the second case, the incorrect FALSE pair is due to a *Right* answer that was incorrectly judged as *Wrong*. We also verified that errors are independent of the entity type requested by the question.

Another interesting feature of SPARTE is that the hypothesis expressions do not appear in the documents. From the 183 pairs studied, only four hypothesis can be found in the text (see one of them in figure 6). This good feature is the result of building the hypothesis as an affirmative expression of the question which conforms a statement not present in the text. However, we found that in 100% of the cases one sentence of the text is enough to support the answer or, in other words, to entail the hypothesis.

```

<pair id="418" value="TRUE" task="QA">
  <q>
    ¿Cuándo ocurrió la catástrofe de Chernobil?
  </q>
  <t doc="EFE19940626-16005"> </t>
  <h>
    La catástrofe de Chernobil ocurrió en abril de 1986
  </h>
</pair>

```

DOC "EFE19940626-16005" : "... Entre los países que acogerán a los niños afectados por radiaciones están también Bélgica, Alemania, Finlandia y Eslovaquia. **La catástrofe de Chernobil ocurrió en abril de 1986.** Unas 7.000 personas murieron inmediatamente o poco después a causa de las radiaciones, y varios centenares de miles sufren aún sus consecuencias..."

Fig. 6. Sample of hypothesis contained in the text

4 Evaluating RTE Systems with SPARTE

The final SPARTE corpus has 2962 text-hypothesis pairs from 635 different questions (4.66 average number of pairs per question). Table 3 shows also the number of pairs with an entailment value equals to TRUE (695) and the number of pairs FALSE (2267).

Table 3. Number of text-hypothesis pairs in SPARTE

	Number	Percentage
Pairs TRUE	695	23%
Pairs FALSE	2267	77%
Total	2962	

The evaluation of a RTE system must quantify its ability to predict the TRUE/FALSE entailment value. Notice that the percentage of pairs FALSE (77%) is much larger than the percentage of pairs TRUE (23%). We decided to keep this proportion since this is the result of real QA systems submission. However, this fact introduces some issues in the evaluation of RTE systems with SPARTE. For example, a baseline RTE system that gives always FALSE would obtain an accuracy equal to 77%, been a very high baseline (see Table 4).

From the RTE evaluation point of view, it would be enough to balance the corpus selecting randomly a 30% of the FALSE pairs and using only these in conjunction with the TRUE pairs. This would yield a corpus big enough and perfectly balanced.

However, from the Answer Validation point of view, a system that validates QA responses does not receive correct and incorrect answers in the same proportion. Thus we think that is useful to maintain the same percentage of TRUE and FALSE pairs that

Table 4. Baselines accuracy in SPARTE

Baselines	Accuracy
Give always TRUE	0.23
Random 50% FALSE 50% TRUE	0.5
Random 77% FALSE 23% TRUE	0.65
Give always FALSE	0.77

Table 5. Baselines for the evaluation proposed

Baselines	Precision TRUEs	Recall TRUEs	F-measure TRUEs
Answer always TRUE	0.23	1	0.37
Random 50% FALSE 50% TRUE	0.5	0.5	0.5
Random 77% FALSE 23% TRUE	0.23	0.23	0.23

systems will receive in an Answer Validation exercise. We think this leads to different development strategies closer to the real exercise that, anyway, must be evaluated with this unbalanced nature.

For this reason, we propose an evaluation based on the detection of pairs with entailment (entailment value equals to TRUE). Turning this into the Answer Validation problem, the proposed evaluation with SPARTE would be focused on the detection of correct answers. This approach has sense since, at the moment, QA systems give more incorrect answers than correct ones, and working towards the detection of the correct ones would be very useful both, for the improvement of systems and for the combination of results coming from different systems.

With this approach we also give a partial solution to the problem of the pairs FALSE that could be considered TRUE in SPARTE (4% of pairs FALSE) (see section 2.4). Although they affect to the training phase, they do not affect to the testing since they are ignored (as they are still FALSE pairs in the corpus).

Therefore, instead of using an overall accuracy as the evaluation measure, we propose to use precision (1), recall (2) and a F measure (3) over pairs with entailment TRUE. In other words, to quantify systems ability to detect the pairs with entailment. Table 5 shows the three baselines with this setting: a system responding always TRUE, a system responding 50% of TRUEs and 50% of FALSEs, and a system responding TRUEs and FALSEs in the same proportion they appear in SPARTE. As shown in the table, the F measure over pairs TRUE becomes a good measure for comparing different systems under the same evaluation conditions.

$$precision = \frac{\#predicted\ as\ TRUE\ correctly}{\#predicted\ as\ TRUE} \quad (1)$$

$$recall = \frac{\#predicted\ as\ TRUE\ correctly}{\#TRUE\ pairs} \quad (2)$$

$$F = \frac{2 * recall * precision}{recall + precision} \quad (3)$$

5 Related Work

SPARTE corpus is inspired in the corpus used for training and testing RTE systems at the PASCAL RTE Challenge 2004 [3]. This is a corpus available¹ in English containing 567 text-hypothesis pairs for the development and training phase, and 800 pairs for the testing. In these collections the number of pairs with entailment (TRUE) is equal to the number of pairs without entailment (FALSE). All these pairs were selected, filtered and adapted manually from a different number of sources related to different NLP and IR areas such as Information Retrieval, Machine Translation, Information Extraction, Paraphrase Acquisition or Question Answering among others. All the text-hypothesis pairs have a tag indicating the corresponding type of source. The evaluation showed that all systems perform better over some types of tasks than others. Due to the manual effort, this is a general corpus that covers a wide range of linguistic phenomena and makes very difficult the RTE task for automatic systems.

In the first PASCAL RTE Challenge, MITRE decided to cast the RTE problem as one of statistical alignment and for this, they needed a corpus larger than the one provided by the organization [2]. From MITRE point of view, most of the TRUE pairs exhibit a paraphrase relationship in which the hypothesis is a paraphrase of a subset of the text. Therefore, they took a news corpus in which the headline of a news article is often a partial paraphrase of the lead paragraph. After a semi-automatic processing they selected the more promising 100,000 pairs, estimating that 74% of them have an entailment relationship. This corpus lead MITRE to one of the best results in the PASCAL Challenge. Although this is a corpus useful for training statistical RTE systems, the absence of human assessment for every pair impede its use for evaluation purposes.

There are some other works aimed at acquiring paraphrases without the notion of entailment [1] [10] [6]. One corpus available in this direction is the Microsoft Research Paraphrase Corpus² [4]. Based on the idea that an event generates hundreds of different news articles in a closed period of time, they decided to apply unsupervised techniques over news clusters for acquiring sentence-level paraphrases. Again, this corpus could be useful for training RTE systems working with English rather than for evaluation purposes.

6 Conclusions and Future Work

SPARTE is the first corpus aimed at evaluating RTE systems in Spanish, containing 2962 text-hypothesis pairs with TRUE/FALSE entailment values. It is specially oriented to the development and evaluation of Answer Validation systems, that is to say, systems aimed at deciding whether the responses of a QA system are correct or not. For this reason, SPARTE deals with the particularities of QA at CLEF: Answers must be supported by documents, and systems give more incorrect than correct answers. We showed here the methodology for developing SPARTE taking advantage of the human assessments made in the evaluation of QA systems at CLEF. We also suggest the appropriate evaluation methodology using SPARTE.

¹ Available at <http://www.pascal-network.org/Challenges/RTE/>

² Available at <http://research.microsoft.com/research/downloads/default.aspx>

Future work is oriented to derive some subcollections of SPARTE where TRUE and FALSE pairs appear in the same proportion.

The second research line is to automatically extract the sentence from the document that contains and supports the answer, in order to give it as the text in the entailment pairs.

Finally, we would like to extend this work to the rest of languages used in the QA Track at CLEF, in order to make available a training corpus for an Answer Validation exercise in multiple languages.

Acknowledgments

This work has been partially supported by the Spanish Ministry of Science and Technology within the following project: TIC-2003-07158-C04-02, R2D2-SyEMBRA. We are grateful to Víctor Peinado, Jesús Herrera and Valentín Sama of the UNED NLP Group, for the QA assessments.

References

1. R. Barzilay and L. Lee. Learning to Paraphrase: An Unsupervised Approach Using Multiple-Sequence Alignment. In *HLT-NAACL 2003: Main Proceedings*, pages 16–23, 2003.
2. J. Burger and L. Ferro. Generating an Entailment Corpus from News Headlines. In *Proceedings of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment*, pages 49–54, June 2005.
3. I. Dagan, O. Glickman, and B. Magnini. The PASCAL Recognising Textual Entailment Challenge. In *Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment, Southampton, UK*, pages 1–8, April 2005.
4. B. Dolan, C. Quirk, and C. Brockett. Unsupervised Construction of Large Paraphrase Corpora: Exploiting Massively Parallel News Sources. In *Proceedings of COLING 2004, Geneva, Switzerland, 2004*.
5. J. Herrera, A. Peñas, and F. Verdejo. Question Answering Pilot Task at CLEF 2004. In C. Peters, J. Gonzalo, M. Kluck, P. Clough, G.J.F. Jones, and B. Magnini, editors, *Multilingual Information Access for Text, Speech and Images. CLEF 2004.*, volume 3491 of *Lecture Notes in Computer Science*, pages 581–590, 2005.
6. D. Lin and P. Pantel. DIRT Discovery of inference rules from text. In *KDD '01: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 323–328. ACM Press, 2001.
7. B. Magnini, S. Romagnoli, A. Vallin, J. Herrera, A. Peñas, V. Peinado, F. Verdejo, and M. de Rijke. The Multiple Language Question Answering Track at CLEF 2003. In C. Peters, J. Gonzalo, M. Braschler, and M. Kluck, editors, *Comparative Evaluation of Multilingual Information Access Systems. CLEF 2003.*, volume 3237 of *Lecture Notes in Computer Science*, pages 471–486, 2004.
8. B. Magnini, A. Vallin, C. Ayache, G. Erbach, A. Peñas, M. de Rijke, P. Rocha, K. Simov, and R. Sutcliffe. Overview of the CLEF 2004 Multilingual Question Answering Track. In C. Peters, P. Clough, J. Gonzalo, G. J. F. Jones, M. Kluck, and B. Magnini, editors, *Multilingual Information Access for Text, Speech and Images. CLEF 2003.*, volume 3491 of *Lecture Notes in Computer Science*, pages 371–391, 2004.

9. A. Peñas, F. Verdejo, and J. Herrera. Spanish Question Answering Evaluation. In A. Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing. CICLing 2004.*, volume 2945 of *Lecture Notes in Computer Science*, pages 472–483, 2004.
10. Y. Shinyama, S. Sekine, K. Sudo, and R. Grishman. Automatic Paraphrase Acquisition from News Articles. In *Automatic Paraphrase Acquisition from News Articles. Proceedings of Human Language Technology Conference, San Diego, USA.*, 2002.
11. A. Vallin, B. Magnini, D. Giampiccolo, L. Aunimo, C. Ayache, P. Osenova, A. Peñas, M. de Rijke, B. Sacaleanu, D. Santos, and R. Sutcliffe. Overview of the CLEF 2005 Multilingual Question Answering Track. In *Proceedings of CLEF 2005*, 2005.

Analysis of a Textual Entailer

Vasile Rus¹, Philip M. McCarthy², and Arthur C. Graesser²

¹ Department of Computer Science

² Department of Psychology, Institute for Intelligent Systems,
The University of Memphis, Memphis, TN 38120, USA
{vrus, pmmccrth, a-graesser}@memphis.edu

Abstract. We present in this paper the structure of a textual entailer, offer a detailed view of lexical aspects of entailment and study the impact of syntactic information on the overall performance of the textual entailer. It is shown that lemmatization has a big impact on the lexical component of our approach and that syntax leads to accurate entailment decisions for a subset of the test data.

1 Introduction

The task of textual entailment is to decide whether a text fragment the size of a sentence, called the Text (T), can logically infer another text of same or smaller size, called the Hypothesis (H).

Entailment has received a great deal of attention since it was proposed (in 2004) under the Recognizing Textual Entailment (RTE) Challenge [7]. In our experiments presented here, we use the standard data set that RTE offers for development and comparison purposes.

The purpose of this paper is to perform an analysis of the textual entailer presented in [8]. In particular, we consider the three main subsystems of the entailer: the lexical component, the syntactic component and the negation handling component. We study each element's contribution to the performance of the system or a part of it. Different aspects of entailment have been analyzed by different groups. The Related Work section describes previous work on entailment analysis. Here, we analyze the task from a systemic, component angle. For instance, we report the impact of lemmatization for entailment, which, as far as we are aware, has yet to be reported. This type of analysis is important to better understand the interaction among different processing modules and to improve decisions as to whether the inclusion of a particular component is advantageous.

In our study, we conduct two levels of analysis. First, we look at how a particular feature impacts the component to which it belongs. For instance, lemmatization is part of the lexical component and we report how the lexical score changes depending upon its presence. Second, we present how a particular feature affects the overall entailment performance. The reader should note that our solution to entailment is based on a limited number of external resources and thus components such as world knowledge are not investigated: we use lexical information, syntactic information provided by a parser, synonymy relations

from a thesaurus, negation, and antonymy relations. WordNet [6] is our source of synonymy and antonymy.

Our paper is structured as follows. The next section presents related work on entailment. Section 3 describes major issues in entailment, and Section 4 presents our approach to solving the entailment task. In Section 5, we report our analyses, describing the importance that lemmatization plays and showing that syntactic information leads to highly accurate entailment decisions for data with high lexical overlap between Hypothesis and Text. The paper ends with Conclusions.

2 Related Work

A great number of approaches to entailment have been taken since the RTE [7] data was made available. They range from shallow approaches such as the weighted-bag of words, to deeper approaches that rely on theorem proving and world knowledge. For example, using T-H pairs from the RTE data set, Vanderwende and colleagues [11] claim that a “large proportion of the data” (37%) can be handled through syntax alone. When a general-purpose thesaurus is added, performance rose to 49%. Vanderwende and colleagues’ claims are based on two human annotators who examined the data manually. More importantly, their “soley” syntactic approach included many linguistic areas (e.g., pronoun resolution) traditionally viewed outside the realm of syntax.

Bar-Heim and colleagues [12] present an interesting conceptual analysis of entailment at the lexical and syntactic level. They found that paraphrases are important and that a lexico-syntactic model outperforms a lexical model; however, both the lexico-syntactic and lexical models offered low recall. Like Vanderwende and colleagues, however, Bar-Haim and colleagues used manual testing and a broad definition of syntax. In contrast, we analyze a running software system that implements a lexico-syntactic approach to entailment. We also conduct a systemic analysis of the textual entailment. A systemic analysis stands to evaluate the impact of each module on the overall performance of the system.

3 A Conceptual Analysis of Entailment

The task of entailment requires a diverse array of language processing modules. Table 1 shows some examples from the RTE data sets and what type of linguistic knowledge would be necessary to solve them. The first column indicates the pair ID as assigned by RTE Challenge organizers. The second column contains the type of the text fragment which is pasted in the third column. The last column provides clues as to what we need (beyond bag-of-words) so as to recognize the entailment relation for the pair. We do not show examples of pairs for which other types of knowledge, e.g. world knowledge, is required since our approach does not take advantage of them and thus it is beyond the scope of this paper. The point of our approach is to see how far we can get with a light system, i.e. a system that uses a minimal number of resources. A light system would be

Table 1. Examples of text-hypothesis pairs from Recognizing Textual Entailment (RTE) Challenge

Pair ID	Type	Content	Solution
2132	Text Hypo	Ralph Fiennes, who has played memorable villains in such films as 'Red Dragon' and 'Schindler's List,' is to portray Voldemort, the wicked warlock, in the next Harry Potter movie.	remote dependencies, lexical relations, paraphrasing
1981	Text Hypo	The bombers had not managed to enter the embassy compounds.	remote dependencies, negation
878	Text Hypo	A British oil executive was one of 16 people killed in the Saudi Arabian terror attack.	remote dependencies

needed if a real time response is required. This is especially true for interactive application such as Intelligent Tutoring Systems [2], e.g. AutoTutor [3].

4 The Approach and Anatomy of the Textual Entailer

The approach for recognizing textual entailment which we consider, described in [8], is based on the idea of subsumption. In general, an object X subsumes an object Y if X is more general than or identical to Y, or alternatively we say Y is more specific than X. The same idea applies to more complex objects, such as structures of interrelated objects. Applied to textual entailment, subsumption translates into the following: hypothesis H is entailed from T if and only if T subsumes H.

The solution has two phases: (I) map both T and H into graph structures and (II) perform a subsumption operation between the T-graph and H-graph.

The first phase includes tokenization (separation of punctuation from words), lemmatization (map morphological variations of words to their base or root form), part-of-speech tagging (assign parts of speech to each word) and parsing (discover major phrases and how they relate to each other; the phrases are grouped in a parse tree). Another important step, part of the first phase, identifies major concepts in the input (e.g., named entities, compound nouns and collocations, postmodifiers, and existentials). This step is important because entities may appear as composed of multiple words in T, (e.g., *venture Services nc*), and as a single word concept in H, (e.g., *venture*). To assure a proper treatment of those cases only common collocations in the input are represented as a single concept by replacing the consecutive words forming a collocation with a new concept composed of the individual words glued with an underscore. A dictionary of collocations (compiled from WordNet) and a simple algorithm helps

detect common collocations in the input. The first phase continues with a step in which parse trees are transformed in a way that helps the graph generation process in the next phase. For example, auxiliaries and passive voice are eliminated but their important information is kept: voices are marked as additional labels to the tag that identifies the verb while aspect information for the verb that a modal acts upon is recorded as an extra marker of the node generated for the verb.

The actual mapping from text to the graph representation is based on information from parse trees. A parse tree groups words in a sentence into phrases and organizes phrases into hierarchical tree structures from where we can easily detect syntactic dependencies among concepts. Charniak's [1] parser (Charniak's proved to be the best according to an across-genre evaluation [9]) is used to obtain parse trees and head-detection rules [5] to obtain the head of each phrase. A dependency tree is generated by linking the head of each phrase to its modifiers in a straightforward mapping step. The problem with the dependency tree is that it only encodes local dependencies (head-modifiers). Remote dependencies are not marked in such dependency trees. An extra step transforms the previous dependency tree into a dependency graph in which remote dependencies are explicitly marked. The dependency graph is then mapped into a final graph in which direct relations among content words are coded. The remote dependencies are obtained using a naive-Bayes functional tagger. The Bayesian model relies on more than a dozen linguistic features automatically extracted from parse trees (phrase label, head, part of speech, parent's head, parent's label, etc.). The model was trained on annotated data from the Wall Street Journal section of Penn Treebank [4]. The accuracy of the functional tagger is in the 90-th percentile [8].

4.1 Phase II: Graph Subsumption

The subsumption algorithm for textual entailment has three major steps: (1) find an isomorphism between H_v (set of vertices of the Hypothesis graph) and T_v (2) check whether the labelled edges in H , H_e , have correspondents in T_e and (3) compute score. Step 1 is more than a simple word-matching method since if a node in H does not have a direct correspondent in T a thesaurus is used to find all possible synonyms for nodes in T . Nodes in H have different priorities: head words are most important, followed by modifiers. Modifiers that indicate negation are handled separately from the bare lexico-syntactic subsumption since if H is largely subsumed by T , and T is not negated but H is or viceversa, the overall score should be dropped, with high confidence, to indicate no entailment. Step 2 takes each relation in H and checks its presence in T . It is augmented with relation equivalences among appositions, possessives and linking verbs (*be have*). Lastly, a normalized score for node and edge mapping is computed. The score for the entire entailment is the sum of each individual node matching and relation matching score. The node match consists of lexical matching and aspect matching (for verbs). The overall score is sanctioned by negation relations.

4.2 Negation and Hypothetical Sentences

Negation is important for our approach. Consider the ideal case when all nodes in H and their relations can be perfectly mapped in T . According to the presented approach, one would decide, with maximum confidence, that T entails H . The decision would be wrong, however, if T were negated and H were not. To handle such cases negation treatment is necessary.

Special attention is paid to two broad types of negation: explicit and implicit. Explicit negation is indicated by particles such as: *no not neither ... nor* and *nt*. Implicit negation is present in text via deeper lexico-semantic relations among different linguistic expressions, the most obvious example is the *antonymy* relation among lemmas which can be retrieved from WordNet. Negation is regarded as a feature of both Text and Hypothesis and it is accounted for in the score after the entailment decision for the Text-Hypothesis pair without negation is made. If one of the text fragments is negated the decision is reversed while if both are negated the decision is retained (double-negation). In Equation 1 the term $\#neg_rel$ represents the number of negation relations between T and H .

4.3 The Scoring

The formula to obtain an overall score aims to deliver both a numerical value for the degree of entailment between T and H and a degree of confidence in our decision. The scores range from 0 to 1, with 1 meaning TRUE entailment with maximum confidence and 0 meaning FALSE entailment with maximum confidence. The score is so defined to be non-reflexive, i.e. $entail(T, H) \neq entail(H, T)$.

$$\begin{aligned}
 entscore(T, H) = & (\alpha \times \frac{\sum_{V_h \in H_v} \max_{V_t \in T_v} match(V_h, V_t)}{|V_h|} + \\
 & \beta \times \frac{\sum_{E_h \in H_e} \max_{E_t \in T_e} synt_match(E_h, E_t)}{|E_h|} + \gamma) \times \\
 & \frac{(1 + (-1)^{\#neg_rel})}{2} \quad (1)
 \end{aligned}$$

The formula to compute the overall score is provided by Equation 1. There are three important components of the score: lexical or node matching, syntactic or relational matching, and negation. The weights of lexical and syntactic matching are given by parameters α and β , respectively. The effect of negation on entailment decision is captured by the last term of the equation. An odd number of negation relations between T and H , denoted $\#neg_rel$, would lead to an entailment score of 0 while an even number will not change the bare lexico-semantic score. The choice of α , β and γ can have a great impact on the overall score. Logistic regression was used to estimate the parameters and also a balanced scheme in which both the lexical component and the syntactic component are given the same weights ($\alpha = \beta = 0.5$, $\gamma = 0$). The results presented throughout the paper are for the balanced scheme that produced better results on development data. From the way the score is defined it is obvious that $entscore(H, T) \neq entscore(T, H)$.

4.4 The Structure of the Textual Entailer

The major subsystems of the textual entailment are the lexical matching, syntactic matching and negation (see Figure 1). Each subsystem is composed of several modules. The lexical matching subsystem includes tokenization, lemmatization, collocations, part of speech tagging, and synonymy extraction component (from a thesaurus). The syntactic matching subsystem is composed of parsing, local dependency relations extraction and remote dependency relations extraction. The negation subsystem contains the explicit negation handling component and implicit negation (antonymy identification with WordNet) handling component.

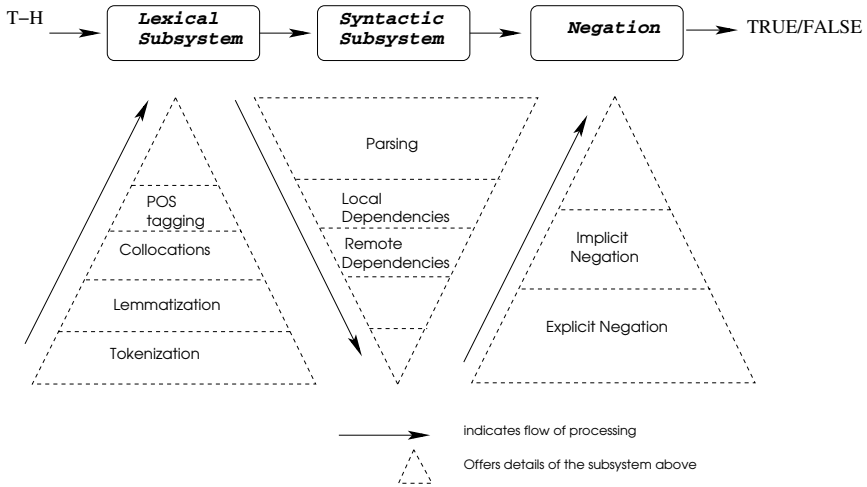


Fig. 1. The Architecture of the Textual Entailer

5 The Analysis

In this section we conduct a systematic analysis of each of the major subsystems of the entailment. This analysis is imposed by the structure of the scoring formula where the three terms account for the above mentioned three subsystems. The analysis is separated along two dimensions. First, we analyze the impact of a particular feature on one of the three components of the score. For instance, we analyze the impact of antonymy in detecting negation. Second, we analyze the impact of a subsystem on the overall score.

Let us look at the experimental setup and performance measures before exploring the details of the analysis. The results reported throughout the paper, unless otherwise specified, are on the RTE test data, containing 800 pairs of Text-Hypothesis collected by human annotators. The data consists of seven subsets, which correspond to typical success and failure settings in different applications: Question Answering, Information Retrieval, Comparable Documents,

Reading Comprehension, Paraphrase Acquisition, Information Extraction, Machine Translation. Within each application setting the annotators selected both positive entailment examples (judged as TRUE), where T entails H, and negative examples (FALSE), where entailment does not hold (50%-50% split).

The evaluation is automatic. The judgments (classifications) returned by the system are compared to those manually assigned by the human annotators (the gold standard). The percentage of matching judgments provides the accuracy of the run, i.e. the fraction of correct responses. As a second measure, a Confidence-Weighted Score (CWS, also known as average precision) is computed (see [7] for details). The CWS varies from 0 (no correct judgements at all) to 1 (perfect score), and rewards the systems' ability to assign a higher confidence score to the correct judgments.

5.1 Lexical Component Analysis

The lexical subsystem is one of the three major components of the scoring formula. Part of the lexical subsystem are all the modules that may alter the lexical component of the score. Those modules are: tokenization, lemmatization, collocations, part of speech tagging, and synonymy. Due to space constraints, we report here only the effects of the lemmatization on the lexical component of the score and on the overall performance of the textual entailment.

For a better understanding of our study, a number of terms we use must first be defined. *lexical matchin*, between H and T is the number of common words divided by the total number of words in H. This is basically the lexical component of the overall score. From the formula, we can see that the higher the lexical component the higher the score (except for pairs with negation). We also define *de ree o lexical mismatchin*, as the number of words in the Hypothesis that do not have a direct correspondent in the Text. For instance, a Hypothesis that has all its words matched to a word in the Text would have 0 degrees of lexical mismatching while a Hypothesis with one word not being matched would have 1 degree of lexical mismatching, and so on.

Now, we look at how lemmatization affects lexical matching. We computed the distribution of the degree of lexical mismatching with and without lemmatization. The number of pairs with perfect lexical score jumps from 58 to 98 (no synonymy at all is used in order to minimize its effects). The same effects can be observed on pairs with higher degrees of mismatching in that there is a moving mass of pairs from higher levels of mismatching to lower levels. One way to quantify the effects of these movements is to compare the number of pairs at a certain lexical degree of mismatching before and after using lemmatization. The problem with this method lies in the composed effects of pairs leaving a level (to an upper level) and pairs entering a level (from a level below). For instance the number of pairs at level 1 (one concept in H not being matched in T) before lemmatization is 238. The same number is 218 after activating lemmatization. Since the number decreases we might conclude that the effects of lemmatization at this level is harmful. However, considering that 40 pairs have moved from level 1 to level 0, such a conclusion must be rejected. The 218 pairs obtained after lemmatization

should be compared to 198 (238-40 - the number of pairs that do not move up). This figure indicates a net advantage of lemmatization for this level. A more global measure is needed to avoid the pitfalls of the method just described and we propose to compute the ratio of the number of moving-up pairs to the total number of pairs. We call the new measure *degree of lexical matching improvement*. The ideal method to compute the degree of lexical matching improvement is to monitor for each pair its level before and after the use of lemmatization. We found a mass of 17.03% pairs having a higher lexical matching score when lemmatization was used.

Now, let us look at the impact of lemmatization on the overall entailment score. One of the most appropriate ways to quantify the contribution of the lexical component to the entailment decision is to compare a purely lexical approach to a blind approach. Another possible way is to compare a bare bones system with the purely lexical approach. Let us look at what constitutes a blind, bare bones, and purely lexical approach.

In [7], the first suggested baseline is the method of blindly and consistently guessing TRUE or FALSE for all test pairs. Since the test data was balanced between FALSE and TRUE outcomes this blind baseline would provide an average accuracy of 0.50. Randomly predicting TRUE or FALSE is another blind method that leads to a run being better than chance for (cws>0.540)/(accuracy>0.535) at the 0.05 level or for a run with (cws>0.558)/(accuracy>0.546) at the 0.01 level.

A bare bones approach is to just take the unlemmatized words and compute the degree of overlap between H and T. If the overlap is greater than .5 then solve to TRUE entailment, otherwise to FALSE. Such a bare bones system yields cws=.5023 and accuracy=.4947.

Pure lexical overlap would be a more informed baseline: tokenize, lemmatize (using *wnstemma* in *wn* library), ignore punctuation and compute the degree of lexical overlap between H and T. We normalize the result by dividing the lexical overlap by the total number of words in H. Then if the normalized score is greater than 0.5 we assign a TRUE value meaning T entails H, otherwise we assign FALSE. The normalized score also plays the role of the confidence score necessary to compute the CWS metric. The results for CWS(.5433) and accuracy (.5375) are better than the consistently guessing TRUE method. They are close to chance though, a possible suggestion that the test corpus is balanced in terms of lexical overlap. The precision (only accounting for positive entailment cases) of 0.6111 on this lexical baseline method may indicate that higher lexical matching may be a good indicator of positive entailment. By comparing the results of the purely lexical method with the bare bones system we can conclude lemmatization definitely helps.

5.2 Syntactic Component Analysis

First, we estimate the impact of syntactic information on a subset of RTE test data to better illustrate the contribution of syntax on top of lexical matching.

Second, we compare the performance of the system with and without the syntactic component.

To study the impact of syntactic information we isolated a subset of the test data that contains pairs with perfect lexical overlap - that is, all the concepts in H had a correspondent in T. We then manually checked whether syntactic information, of the kind we gather from parse trees, could help solve the entailment for this particular T-H pairing. The isolation yielded 106 pairs with perfect lexical overlap of which 101 were selected for further analysis. Five pairs were discarded as the annotator disagreed with the RTE answer. Finally, we measure the performance of our system on the subset that was manually determined to benefit from syntax. Because lexical information cannot help in those cases it is a good testbed to check how much syntax helps.

We judged the lexically overlapped pairs to fall into one of three syntactical categories. These categories require some explanation which we have supplemented with condensed examples taken from the RTE.

S1-type pairs were deemed those where relatively simple syntactical rules were judged likely to bring a significant number of correctly identified entailments. For example, consider the following text and hypothesis.

Text: *The Alameda Central, west of the Zocalo, was created in 1592.*

Hypothesis: *The Alameda Central is west of the Zocalo.*

Replacing the comma that separates two consecutive NPs in the Text-half of a pair with the verb *be*, is likely to result in entailment. The rule is concise, simple to apply, and predicted to be generally correct. We therefore labeled examples of this kind as type-S1.

S2-type pairs differed from S1 pairs in two ways. First, S2-type pairs were deemed those where describing the syntax rule was moderately complex. Second, and assuming the complexity of the rule did not prevent it from being computationally realized, a S2-type pair was deemed one where the likelihood of the rule identifying correct entailments was significantly lower than that of S1-types. For example, consider the following text and hypothesis.

Text: *In 1541, the Turks took Buda and held it until 1686; the city changed very little during this time.*

Hypothesis: *The Turks held Buda between 1541 and 1686.*

In this example, the prepositions *in* and *until* in the text-half correspond to the prepositions *between* and *and* in the hypothesis. However, the overlap of these prepositions is not enough by itself. Issues such as the presence and position of each token, and corresponding dates must also be considered. There is also the complicating matter of the presence of a coordinating conjunction in the text-half, meaning that the rule is complex and unlikely to bring as significant a number of correctly identified entailments as the S1-type.

S3-type pairs fell into two categories. First, pairs were deemed S3 if the hypothesis required extra textual information. For example, consider the following pair.

Text: *A state of emergency was declared in Guatemala City.*

Hypothesis: *A state of emergency was declared in Guatemala.*

Although the entailment is correct in this hypothesis, there is no syntactical way to know that Guatemala City is in Guatemala.

Pairs were also deemed S3 if a potential syntactical rule was deemed highly complex and unlikely, even if constituted, to bring a significant number of correctly identified entailments. For example, consider the following pair.

Text: *The demonstrators were calling for a trial of the right-wing politician and on seeing them he gave them a rude gesture and the police then had to stop four demonstrators who tried to chase the senator.*

Hypothesis: *Demonstrators gave the right-wing senator a rude gesture.*

This hypothesis calls for both pronoun resolution and syntactical agreement over distant sentence elements. Even if a rule could be formulated, its likelihood of correctly identifying a significant number of entailments was deemed small.

Let us look at the figures. Of the original 106 pairs, 44 solved to TRUE and 62 to FALSE according to RTE. Of the final 101 pairs, 41 led to TRUE and 60 to FALSE meaning a blind method for this category could deliver .5940 accuracy. This is in concordance with [7] which reports that pairs with high lexical match are biased towards FALSE entailment. About 47 (47%) of the 101 could be answered by syntax as defined by us (S1) and 69 (72%) by a broader definition of syntax (S1+S2). This is a little higher than results reported by Vanderwende and colleagues [11].

Following the manual check, we evaluated the performance of our system on the 47 pairs that could be answered by syntax. These pairs were split as: 19-TRUE and 28-FALSE according to RTE annotation. Since those pairs have perfect lexical overlap a lexical baseline approach is blind to this subset. Possible solutions would be to either not give the lexical component the same weight as the syntactic component or, if weighted the same, the threshold above which TRUE entailment holds needs to be set higher.

From our studies, we found that for pairs with full lexical matching, the syntactic overlap should also be high (>90%) in order to have true entailment and everything below should be solved to FALSE. If such a mechanism is built into the system, the accuracy for fully lexically matched pairs increases from 38.89% to 80.85%.

The argument that setting a high threshold automatically leads to higher accuracy is only half true as it only works for positive (TRUE entailment) cases. For negative cases (FALSE entailment) the lower the threshold the higher the

accuracy. Moreover, setting a high threshold only means high accuracy with high confidence and we still need to decide what to do below a high threshold: decreasing confidence and keeping the same decision as above the threshold, or changing the decision (from TRUE to FALSE). For regular pairs we would just lower the confidence for high levels of lexical and syntactic matchings. For fully lexically matched pairs we claim one can confidently change the decision to FALSE below the high threshold of 90%.

Now, let us consider the performance of the system with and without the syntactic component on all RTE data. The full system, with syntax, yields $cws=0.6029$ and $accuracy=0.5537$. If we turn off the syntactic component that leads to $cws=0.5635$ and $accuracy=0.5463$. The last figures are different than the figures provided by purely lexical approach since negation is also used now. By comparing the before and after figures we can conclude that overall the impact of syntax is not significant. However, this maybe true only on the balanced RTE data set. As our previous experiment showed, syntax makes a difference on particular data subsets.

5.3 Negation

As mentioned earlier we consider two categories of negation. First we handle explicit negation by way of clue phrases. Second, we use antonymy relations in WordNet to detect antonymy relations between a word in H and a word in T.

There are 78 pairs that exhibit some form of negation as detected by the system: explicit (66), implicit (17), simple negation (66), and double negation (12). There are 5 pairs that exhibit both explicit and implicit negation. In the current implementation the system uses negation only for pairs with full lexical overlap because it is not yet clear how to make the decisions for the other cases. Of the 78 cases it is used only on 4 cases with an accuracy of 75%. The results on the 78 pairs with negation are 0.6073 (cws) and 0.5256 (accuracy). Results without negation are: 0.5502 (cws) and 0.5000 (accuracy). Results on all test data (800 pairs) without negation are: 0.5951 (cws), 0.5513 (accuracy). Results on all test data with negation are: 0.6029 (cws) and 0.5537 (accuracy). In both cases we can see that negation handling is beneficial. The antonymy relations help detecting 17 possible negations between H and T which accounts for 21% of the pairs that are detected exhibiting some form of negation.

6 Conclusions

This paper presented the structure and component analysis of a textual entailment. It was shown that lemmatization is important for the lexical component of our approach and that syntax can have a significant impact on pairs with perfect lexical score. Negation is also useful and although it does not have a significant impact on the RTE data it may be significant in other data sets, more realistic (as opposed to balanced) or more negation oriented data sets.

References

1. Charniak, E. A Maximum-Entropy-Inspired Parser, Proceedings of the North-American Chapter of Association for Computational Linguistics, 2000, Seattle, WA, United States.
2. Graesser, A.C. and VanLehn, K. and Rose, C. and Jordan, P., and Harter, D. Intelligent tutoring systems with conversational dialogue, *AI Magazine*, 2001, volume 22, pages 39-51
3. Graesser, A.C. and Lu, S. and Jackson, G.T. and Mitchell and H. and Ventura, M. and Olney, A., and Louwerse, M.M. AutoTutor: A tutor with dialogue in natural language, *Behavioral Research Methods, Instruments, and Computers*, 2004, pages 180-193
4. Marcus, M.P., Santorini, B., Marcinkiewicz, M.A. Building a Large Annotated Corpus of English: The Penn Treebank, *Computational Linguistics*, 19-2, pages 313-330.
5. Magerman, D.M. Natural Language Parsing as Statistical Pattern Recognition, PhD Thesis, Stanford University, February, 1994
6. Miller, G. WordNet: a lexical database for English, *Communications of the ACM*, 1995, volume 38, number 11, pages 39-41
7. Dagan, I., Glickman, O., Magnini, B. The PASCAL Recognising Textual Entailment Challenge Proceedings of the Recognizing Textual Entailment Challenge Workshop, 2005, Southampton, U.K., April 11 - 13
8. Rus, V., Graesser, A., Desai, K. Lexico-Syntactic Subsumption for Textual Entailment, Proceedings of Recent Advances in Natural Language Processing (RANLP), Borovets, Bulgaria, September 17-19, 2005
9. Rus, V., Hempelmann, C. Across-Genre and Empirical Evaluation of State-of-the-Art Treebank-style Parsers Proceedings of the 2nd Language Technologies Conference, Poznan, Poland, April 21-23, 2005,
10. Bies, A., Ferguson, M., Katz, K., MacIntyre, R. Bracketing Guidelines for Treebank II Style. Penn Treebank Project.
11. Vanderwende, L., Coughlin, D., Dolan, B. What Syntax can Contribute in Entailment Task Proceedings of the Recognizing Textual Entailment Challenge Workshop, 2005, Southampton, U.K., April 11 - 13
12. Bar-Haim, R., Szpektor, I., Glickman, O. Definition and Analysis of Intermediate Entailment Levels Proceedings of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment, 43rd Annual Meeting of The Association for Computational Linguistics, 2005, July 25-30, Ann Arbor, MI

Referring Via Document Parts

Ivandr  Paraboni¹ and Kees van Deemter²

¹ Instituto de Ci ncias Matem ticas e de Computa  o – ICMC,
Universidade de S o Paulo - USP
Av Trabalhador S o-Carlense, 400, 13560-970 - S o Carlos SP, Brazil
ivandre@icmc.usp.br

² Department of Computing Science, King's College, University of Aberdeen,
Aberdeen AB24 3UE, Scotland, UK
kvdeemte@csd.abdn.ac.uk

Abstract. Documents in a wide range of genres often contain references to their own sections, pictures etc. We call such referring expressions instances of *Document Deixis*. The present work focuses on the generation of Document Deixis in the context of a particular kind of natural language generation system in which these descriptions are not specified as part of the input, i.e., when it is up to the system to decide whether a reference is called for and, if so, which document entity it should refer to. We ask under what circumstances it is advantageous to describe domain objects in terms of the document parts where they are mentioned (as in “the insulin described in section 2”). We report on an experiment suggesting that such indirect descriptions are preferred by human readers whenever they cause the generated descriptions to be shorter than they would otherwise be.

1 Introduction

Document parts such as sections, subsections, pictures, paragraphs etc may be referred to for various purposes, for example to point to additional information on the current topic of the text, e.g., “see also section 7”. References to document parts will often be *deictic*, in the sense that the realisation of the expression depends on the place in the document where the referring expression is uttered (e.g., “this section” versus “section 1.1.”). Accordingly, we will call the references to parts of the *same* document instances of Document Deixis (DDX).

We are interested in a particular kind of DDX, which we have previously called *object-level* instances of Document Deixis [9]. These are usually part of a larger expression which refers to a domain entity. The entity in question may be concrete (e.g., the medicines in Example 1) or abstract (e.g., the advice in Example 2). In the corpora that we investigated – patient information leaflets [1] - references to abstract entities or sets of them are far more common.

Example 1 (...) if you are taking any of *the medicines e.g. antibiotics listed under the section "Are you taking other medicines?"* (Cilest).

Example 2 You will notice that *the advice contained in this leaflet* may vary depending on whether you are (...) (Neoral Oral Solution)

Unlike most concrete domain entities (e.g., objects in the physical world), such abstract referents often lack a concise textual description, and perhaps for that reason are referred to via document parts. Interestingly, many of these document-deictic referents can be also discourse deictic referents [4,12].

Document-deictic descriptions are ubiquitously found in many document genres, making their generation a practically relevant research topic on its own right, in connection with a certain class of natural language generation (NLG) systems for aiding in the authoring of complex documents [2]. In a system of this kind, the user, or author, creates a document by specifying its content (and sometimes providing also a high-level specification of its form) leaving to the system the responsibility over linguistic form and layout of the document.

Different systems may require different degrees of participation from the author. In most cases, however, low-level decisions such as lexical choice and referring expression generation are not under the author's control. For example, the author may specify that a certain domain entity has a certain property P , but it is up to the system to determine whether the entity will be referred to via a proper name, a pronoun, or a definite description, perhaps quoting a document part as in the previous example. The remainder of this paper investigates under what circumstances object-level Document Deixis is useful.

2 Document Deixis as ‘Ordinary’ Definite Descriptions

In this section we discuss how a ‘traditional’ algorithm for generating descriptions of domain entities in a given NLG application can be extended to generate certain instances of DDX as well. More specifically, we discuss how to adapt the well-known Dale & Reiter Incremental algorithm [3] to the task of generating ordinary and DDX descriptions alike. Briefly, the Incremental algorithm takes as its input a set of domain entities (the intended referent r and the context from which r has to be told apart) and their referable properties (pairs attribute-value such as ‘colour-black’ or ‘type-dog’) and a (domain-dependent) list of preferred attributes P specifying the order in which the algorithm will attempt to add properties to the description under generation. Properties that are *restrictive* (i.e., those that help ruling out distractors) are selected one by one, until a unique description is obtained (e.g., “the black dog”).

We will limit ourselves to restrictive uses of document-deictic information. In this case, DDX can be viewed as an abbreviation device not unlike discourse anaphora, employed to avoid the repetition of a long description which has already been introduced in the discourse. We will assume that the choice between an anaphoric reference and a full definite description has already been made [6] and that an opportunity for a description (potentially making use of DDX) has been identified.

The semantic function of a restrictive DDX does not differ from that of ‘ordinary’ referring expressions, namely to single out a domain entity. One obvious approach is to assume that these expressions are constructed in the same way as ordinary referring expressions, e.g., by using the Incremental algorithm. Thus, document parts can be viewed as document-deictic properties of some of the domain entities they describe. For example, a domain entity realised as a picture may have, besides all its domain properties, the *document-related* property of ‘being described by’ the picture.

For concreteness, we will consider the following example of document structure, which we call our *target document*. The document consists of two sections (1 and 2) divided into three subsections (A, B and C) each. Each of the six subsections has a unique and well-defined topic (an insulin product), i.e., each topic is ‘described by’ a different document part. Although not truly representative of the documents found in the PILLS corpus (whose leaflets are far less structured and do not usually describe different medicines in such ordered fashion), this organisation represents the minimal complexity required for discussing the most interesting instances of cross-section DDX such as “the insulin described in subsection C”.

The following is a simplified representation of the target document, in which the domain entities representing the topic of each section are shown in brackets.

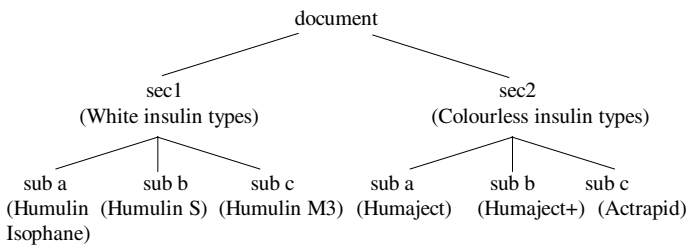


Fig. 1. Example of document structure

In this example, domain entities (i.e., different kinds of insulin products such as Humulin Isophane etc) have ordinary domain-related properties such as *type* (insulin), *colour* (white or colourless), *origin* (human or animal) etc. In addition to that, each domain entity has the document-related property of being *described_by* a particular document part. Thus, the description “the insulin mentioned in subsection C” may distinguish one product from all others, analogous to an ‘ordinary’ description like “the white animal insulin containing folic acid”.

In order to avoid the problem of overlapping values (i.e., the problem of being described by section 1 and section 1a simultaneously, as reported in [10]) we assume that only the leaves of the document tree such as subsections (which correspond to the most specific values of the attribute) are modelled as the possible values of the *described_by* attribute. This will have also the effect of producing references whose referents are arguably easier to identify [8].

3 When to Generate Document-Deictic Descriptions

Object-level DDX represents a non-standard means of referring to domain entities, and perhaps for that reason these descriptions remain little investigated in the NLG field. In this section we discuss some circumstances under which it may be appropriate, or perhaps even necessary, to generate them.

Work on multimodal systems suggest a few possible triggers for reference to parts of a multimodal presentation. For example, in the COMET system [7] a reference

such as “Remove the holding battery cover plate, highlighted in the right picture” can be generated when the system determines that the user does not know the term commonly used to describe an object depicted in the presentation. However, it is likely that a document-authoring system will have to take many other (linguistically-motivated) factors into account. For example, an entity may be referred to via a document part as in Example 3 simply because not referring to the document part would lead to a more complex description as in Example 4:

Example 3 Are you allergic to any of *the ingredients listed under section 5*?

Example 4 Are you allergic to diclofenac, mannitol, sodium metabisulphite (E223), benzyl alcohol, propylene glycol or sodium hydroxide?

We will focus on the cases in which document-related properties are viewed as a repair strategy: ordinary descriptions (i.e., those not using any document-related properties) are the method of choice, and document-related properties are used only to avoid overly long, or otherwise awkward descriptions, reflecting a Gricean-style brevity maxim [5]. Suppose, for example, the generator has to refer to a particular side effect, and that the description \mathbb{L} produced by the Incremental algorithm makes use of the following domain-related properties:

‘being associated with medicine 1’

‘being associated with medicine 2’

‘being a long-term side effect’

Suppose that the side effect in question is the only one to be described in, say, section 1. In a case like this, the sheer length of \mathbb{L} might be sufficient reason for the generation of DDX as in “The side effects described in section 1”, rather than “The long-term side effects that are associated with both medicine 1 and medicine 2”. In other words, the length of the description can be regarded as a factor for triggering instances of DDX.

Taking the Incremental algorithm as a basis, the idea of using description length as a factor to trigger the generation of DDX could be implemented in various ways. One such strategy would be varying the priority assigned to the *described_by* attribute, using it for example either before the inclusion of any domain property, or only after a number of domain properties have already been added. We call the former a *ddxFirst* strategy, and the latter *ddxLast*.

When *described_by* properties are highly discriminating (as in our target document), *ddxFirst* may produce fairly short descriptions such as “The insulin described in subsection A”. On the other hand, by regarding *described_by* as a last resort to disambiguate the reference, *ddxLast* may produce lengthy descriptions such as “The liquid insulin containing phenol described in subsection A of section 2”.

Using the length of the description as a trigger for DDX is comparable to the strategy adopted by van der Sluis [11] to generate *pointing gestures* for replacing overly long descriptions in the sense that both are strategies for facilitating reference. There is however one major difference between the two approaches: unlike pointing gestures in [11], document-related properties cannot be assumed to be uniquely distinguishing. After the inclusion of the *described_by* attribute it may be still necessary to include further (domain-related) properties to obtain a uniquely distinguishing de-

scription. By contrast, van der Sluis simply discards the generated description and generate a (uniquely distinguishing) pointing gesture instead.

The use of DDX as a means of rephrasing a ‘bad’ description can be implemented as follows. First, a description using only domain-related properties is attempted. If the length of the description rises above a certain level, then the description is discarded and a new reference is generated, this time using *described_by* as the most preferred attribute in the list *P*. We will call this the *regen(eration)* strategy.

The length of the description can be measured in different ways. We follow Dale & Reiter [3] in defining descriptions in terms of semantic rather than syntactic components, i.e., in terms of properties rather than words or other syntactic structures, even though the number of properties may not reflect the length of the surface string. This may be especially true for document-related properties. For example, a given set of symptoms referred to via a picture as in “the symptoms shown in pic.5” may convey only two attributes (*type* and *described_by*), depending on the surface realisation of the document-deictic description (which may require reference to more than one document entities this reference could be realised as “the symptoms shown in picture 3 in part B of section 2”).

Both *ddxFirst* and *ddxLast* can be easily implemented as part of the Incremental algorithm by varying the position of the attribute *described_by* in the list of preferred attributes *P*. The *regen* strategy can be implemented as follows. In this representation, *P* is assumed to be the list of preferred attributes used by the *MakeReferringExpression* function, which is the core of the Dale & Reiter Incremental algorithm.

```

L ← MakeReferringExpression(r, C)
if (Length(L) > maxlength) or (L = ∅) then
    P ← <described_by> ∪ P
    L ← MakeReferringExpression(r, C)
return (L)

```

The following are examples of reference according to the above strategies (plus the *original* version, which does not make use of document-related properties). Note that the output of the *regen* strategy coincides with either *original* or (as in the example) *ddxFirst*. A list of descriptions generated by the four strategies in the context of our target document is presented in the appendix.

Table 1. Examples of descriptions produced by different generation strategies

Strategy	domain entity (insulin product)
<i>original</i>	“The animal liquid insulin containing phenol”
<i>ddxFirst</i>	“The insulin described in subsection A of section 2”
<i>ddxLast</i>	“The liquid insulin containing phenol described in subsection A of section 2”
<i>regen</i>	“The insulin described in subsection A of section 2”

To put into practice some of these ideas we implemented a simple DDX generator. The program displays four versions of our target document in which all the information except the referring expressions is canned text. In each version, the referring expressions are generated according to a different strategy that may include the use of document-related properties or not. In order to prevent the generation of an overly

large number of these references, we assumed that an instance of DDX is only possible if (a) the reference is the first mention of the domain entity in the corresponding subsection and (b) in the case of reference to individual domain entities, the referent is described by a subsection other than the subsection containing the referring expression (i.e., a situation of non-local reference). We do not claim however any plausibility on these constraints. As we pointed out in the previous sections, adequacy constraints of this kind seem to be style- or genre-dependent, and our present choice is strongly based on the content and structure of the target document.

The generated documents are necessarily sketchy and repetitive, and their descriptions are sometimes clumsy owing to the large number of semantic properties conveyed (especially when not using document-related properties). Nevertheless, it is tempting to ask how the use of document-related properties affects their acceptability as referring expressions, as we will discuss in the next section.

4 An Experiment on the Use of Document Deixis

We carried out an experiment on the acceptability of selected instances of reference to individual domain entities generated according to the *original*, *ddxFirst*, *ddxLast* and *regen* strategies discussed above. Our goal was to find out (a) whether the predictions made by the *regen* strategy are accurate and (b) how the *ddxFirst* strategy compares to *ddxLast* and *original*. Our hypotheses were the following:

- h.1a: when *regen* predicts *ddxFirst*, *ddxFirst* is preferred to *original*;
- h.1b: when *regen* predicts *original*, *original* is preferred to *ddxFirst*;
- h.2: *ddxFirst* is always preferred to *ddxLast*;
- h.3: *ddxFirst* is always preferred to *original*.

Hypotheses h.1a and h.1b test the outcome of the *regen* strategy (which may produce descriptions making use of a document-related property or not). Hypothesis h.2 compares the strategy *ddxFirst* with *ddxLast*, and h.3 compares *ddxFirst* with *original*. Note that h.1b potentially contradicts h.3.

Method

Subjects were asked to rate the acceptability of instances of DDX generated by the implemented program described in the previous section.

Subjects: 20 graduate students.


Materials: We chose to evaluate five contexts of reference (cf. appendix) found in the target document used in our implementation. Each context consisted of a different place of utterance and a different referent. In cases of DDX, the direction (backwards or forwards) and region (local or non-local) of reference were as evenly distributed as possible, although these factors were not expected to affect the results since the subjects did not have to identify the referents, but simply evaluate the wording of the referring expressions.

Because in some cases two different strategies produced the same output, it sufficed to test 14 descriptions covering the above cases, namely, five instances of *original*, five instances of *ddxFirst* and the instances of *ddxLast* in contexts 1-4. The evaluation of *ddxLast* in context 5 (which coincides with *original*) and the evaluation of *regen* (which coincides with *ddxFirst* in contexts 1-3, and with *original* in contexts 4-5) were simply derived from the evaluation of the main set of 14 descriptions.

The 14 references were presented in a printed format, keeping their original context (i.e., as in the target document used in the implementation). Minor changes were made in the format and content of the document (mainly involving text aggregation) to make it more appealing. In order to reduce the scope for comparison between different strategies that could lead to a bias against DDX (which seemed to occur in a preliminary pilot experiment), the three strategies of each situation of reference were split across three different versions of the same document (i.e., one document for each strategy, comprising five references in each of the first two documents and four references in the third one). The documents were identical except for the wording of the referring expressions.

Procedure: The subjects were presented a printed version (one page-long, with no page numbers) of the target document (called document 1) with no references to be rated, and they were asked questions about its content and form to guarantee that they were familiar with the setting of the experiment. Next, the subjects were given three versions of the same document (documents 2-4) containing the 14 references to be evaluated¹, and they were asked to judge each case on its own. The following is a fragment of one such question corresponding to *ddxFirst* in context 1 (underlined).

This medicine is a liquid form of insulin.
Because it contains phenol this medicine is
not suitable for children under 12.
The insulin described in subsection C is
more appropriate for children under 12

Acceptability
(-)  (+)
0 1 2 3 4

The subjects rated the referring expressions from 0 (unacceptable) to 4 (highly acceptable). The greatest difference in results was obtained in the *ddxLast* strategy. Unlike *ddxFirst* and *original*, this strategy was never rated as highly acceptable, and it was the only one rated as unacceptable (11 cases, corresponding to 14% of the total). The *ddxLast* strategy fares better than the alternatives in only five cases (6%) and it fares worse in 40 (50%).

The results of the experiment are shown below. For each pair of strategies under consideration, we present the percentage of answers that favours each alternative (%) excluding the cases in which both alternatives were equally rated, the average score obtained by each strategy, the sum of ranks (w), degrees of freedom (N), and significance (p) obtained from the Wilcoxon's signed-rank test. All results are statistically significant as indicated. All hypotheses were confirmed except for h.3, for which there was an effect in the opposite direction.

¹ For the actual documents used in the experiment, see [9].

Table 2. Summary of the experiment results

	h.1.a (p < .05)		h1.b (p < .005)	
	<i>ddxFirst</i>	<i>original</i>	<i>original</i>	<i>ddxFirst</i>
%	67.86%	32.14%	86.36%	13.64%
average	2.55	2.35	2.68	1.98
w / N	126.5 / 28		16.5 / 22	
	h.2 (p < .005)		h.3 (p < .05)	
	<i>ddxFirst</i>	<i>ddxLast</i>	<i>ddxFirst</i>	<i>original</i>
%	94.23%	5.77%	44.00%	56.00%
average	2.41	1.38	2.32	2.48
w / N	40.5 / 52		460 / 50	

Discussion

In contexts 1-3, in which the *regen* strategy predicts an instance of DDX (i.e., producing a description as in *ddxFirst*), *ddxFirst* was preferred to *original* in nearly 68% of the cases. This result confirms our hypothesis h.1a. Conversely, in contexts 4-5, in which the *regen* strategy does not predict *ddxFirst*, a description without document-related properties (as in the *original* strategy) is preferred to *ddxFirst* in 86% of the cases, hence confirming our hypothesis h.1b. Overall, the predictions of the *regen* strategy (triggering the generation of DDX when the description reaches a certain length) seem to be supported by these results.

The fact that the preference for *ddxFirst* in h.1a is smaller than the preference for *original* in h.1b may be related to the method used in the *regen* strategy to measure the length of the description. In the implemented algorithm, the length of the description is based on the number of existing (domain-related) properties. However, this method does not take into account the fact that the realisation of a document-related property may require a large linguistic expression. A single property of ‘being described by’ a certain document part may be realised for instance as “described in subsection B of section 2”, which means that the *regen* strategy may in some cases favour a document-deictic expression whose surface realisation is actually longer than its non-deictic counterpart. While using the number of semantic properties to limit the length of the description may be convenient from the computational point of view, this method is unlikely to correspond to the criteria adopted by the subjects of the experiment to rate the descriptions, and perhaps for that reason some of them chose fewer *ddxFirst* references in the situations addressed by h.1a (or, conversely, more *original* references in h.1b). Although we do not attempt to prove this claim, we believe that the implementation of a more psychologically realistic method for measuring the length of the description (e.g., based on the number of words in the surface realisation) would result in a more accurate *regen* strategy.

The comparison between *ddxFirst* and *ddxLast* strategies in h.2 is straightforward. In 94% of the cases in which there was a difference between the two scores, *ddxFirst* fares better than *ddxLast*. This result is not surprising given the complexity of the descriptions produced by *ddxLast* and, accordingly, the poor rates obtained by this strategy in the experiment.

Finally, the overall comparison between *ddxFirst* and *original* strategies in h.3 shows that the *original* option was preferred in 56% of the cases. This result not only disconfirms hypothesis h.3, but shows an effect in the opposite direction. A possible explanation for this is discussed below.

Although the predictions of the *regen* strategy in h.1a and h.1b are supported by the experiment data, and although some conflict was to be expected (given that h.1b potentially contradicts h.3) the rejection of h.3 seems to suggest a simple link between the choice for DDX and the length of description. More specifically, there seems to be a preference for the *shortest* description in all the situations investigated regardless of the use of document-related properties. This appears to be the case in h.1a (which favours *ddxFirst* over *original*, being *ddxFirst* usually the shortest option), in h.1b (which favours *original* over *ddxFirst*, being *original* usually the shortest option) and in h.2 (which favours *ddxFirst* over *ddxLast*, being *ddxLast* the shortest option). Thus, we decided to check whether this observation could also explain the preference for *original* over *ddxFirst* in h.3. Our hypothesis was the following:

- h.4: given the choice between *ddxFirst* with *original*, the shortest alternative is always preferred.

In order to test h.4, we first measured the length of all instances of *original* and *ddxFirst* descriptions in each of the five situations of reference, and determined the longest of the two. As an attempt to obtain a more accurate measurement, instead of taking the number of semantic properties into account (as in the *regen* strategy), we opted for measuring the length of the descriptions based on the number of words in their surface realisation. For this purpose, section labels and prepositions were also counted as words. For example, the length of the description “the insulin described in subsection A of section 2” according to this method is nine words. When two descriptions had the same length, we considered the *ddxFirst* alternative to be the shortest, the underlying assumption being that section labels (e.g., “A”) and prepositions (e.g., “of”) should count somehow less than e.g., nouns and adjectives.

Having determined the longest description of each situation of reference², we counted the number of cases in which these descriptions received the lowest score of the two options. Cases in which both options were rated as equal were eliminated from the analysis. Table 3 below summarises our findings obtained from the Wilcoxon’s signed-rank test.

Table 3. Summary of the results for the additional hypothesis (h.4)

	h.4 (p < .005)	
	short	long
%	76.00%	24.00%
average	2.61	2.19
w / N	210 / 50	

² In contexts 1 and 3, the longest description was the *original* option, and in contexts 2, 4 and 5 it was the *ddxFirst* option.

In 76% of the cases in which there was a difference between the two scores, the shortest alternative (regardless of being produced by *original* or *ddxFirst*) fares significantly better as indicated. This observation is consistent with all our research hypotheses (including the low score obtained for *ddxLast*, since this strategy always produces the longest description of the three, and it fares worst in almost all cases). This also confirms the principles underlying the *regen* algorithm, suggesting that DDX can be used as a legitimate means of reducing the length of the description³.

5 Final Remarks

We discussed the generation of document-deictic descriptions in systems in which these references are not originally planned as part of the input. In order to illustrate some of the issues under discussion, we implemented a simple Document Deixis generator, whose output was subsequently evaluated by subjects of an experiment.

The implementation work made use of an ‘ideal’ document structure in which the definition of ‘describe’ relations is straightforward. We did not address the question whether it is adequate to produce an instance of DDX in a particular place in the document. For example, a reference such as “See section B” is unlikely to be of much help if, say, it is overused throughout the text. What is considered overuse in one document genre may however be perfectly adequate in another. In our work these issues were large avoided by only allowing references to document parts under limited circumstances. The definition of well-justified constraints of this kind will however require further research.

We have discussed a number of triggers to DDX based on the length of the description under generation. We focused on the restrictive use of document-related properties and discussed how the Incremental algorithm could be adapted so as to use these properties alongside traditional, domain-related properties. Our experiment made use of a small number of referring expressions that were not based on naturally produced text. Despite these limitations, the results show a plausible link between description length and the use of document-related properties, suggesting that overly long description may in fact trigger the generation of Document Deixis in a way much similar to the generation of deictic pointing gestures.

Acknowledgements

This work was part of the PhD project of the first author and it has been supported by the CNPq – Brazilian research Council.

References

1. The Association of the British Pharmaceutical Industry: 1996-1997 ABPI Compendium of Patient Information Leaflets (1997)
2. Bouayad-Agha, N., et. al.: Integrating Content and Style in Documents: a Case Study of Patient Information Leaflets. *Information Design Journal*, 9 (2-3) (2000) 161-176

³ This is not to say that shorter references are *always* better: for a discussion on when longer, logically-redundant descriptions may be preferred, see [8,9].

3. Dale, R. and E. Reiter: Computational interpretations of the Gricean maxims in the generation of referring expressions. *Cognitive Science* (19) (1995)
4. Eckert, M. and M. Strube: Resolving Discourse Deictic Anaphora in Dialogues. In *Proceedings of EACL'99 Bergen* (1999)
5. Grice, H. P.: *Logic and Conversation*. In P. Cole and J. L. Morgan (eds.) *Syntax and Semantics, Vol. iii: Speech Acts*. New York, Academic Press (1975) 41-58
6. McCoy, K. F. and M. Strube: Generating Anaphoric Expressions: Pronoun or Definite Description? In *Proceedings of the ACL'99 Workshop "The Relation of Discourse/Dialogue Structure and Reference"* University of Maryland (1999)
7. McKeown, K. et al.: Generating Cross-References for Multimedia Explanation. In *Proceedings of AAAI-92* (1992) 9-16
8. Paraboni, I. and K. van Deemter: Generating Ease references: the case of document deixis. In *Proceedings of the 2nd International Conference on Natural Language Generation INLG-2002, New York* (2002) 113-119
9. Paraboni, I.: *Generating References in Hierarchical Domains: the Case of Document Deixis*. Information Technology Research Institute (ITRI), University of Brighton. Brighton, July (2003). A doctoral thesis. Also available as ITRI Technical Report ITRI-03-11 from <http://www.itri.bton.ac.uk/>.
10. van Deemter, K. : Generating Referring Expressions: Boolean Extensions of the Incremental Algorithm. In *Computational Linguistics* 28(1) (2002) 37-52
11. van der Sluis, I.: An Empirically Motivated Algorithm for Generation of Multimodal Referring Expressions. In *Proceedings of the Student Research Workshop and Tutorial Abstracts of ACL 39th meeting* (2001) Toulouse
12. Webber, B. L.: Structure and Ostension in the Interpretation of Discourse deixis. *Language and Cognitive Processes* 6(2) May (1991) 107-135

Appendix: Situations of Reference Considered in the Experiment

Context 1: forward local references in section 1 part A.

<i>Original</i>	<i>ddxFirst</i>	<i>ddxLast</i>	<i>Regen</i>
the white animal insulin containing folic acid	the insulin described in subsec C	the insulin containing folic acid described in subsec C	the insulin described in subsec C

Context 2: forward non-local references in section 1 part B.

<i>original</i>	<i>ddxFirst</i>	<i>ddxLast</i>	<i>regen</i>
the animal liquid insulin containing phenol	the insulin described in subsec A of section 2	the liquid insulin containing phenol described in subsec A of section 2	the insulin described in subsec A of section 2

Context 3: backward local references in section 1 part C.

<i>original</i>	<i>ddxFirst</i>	<i>ddxLast</i>	<i>regen</i>
the human liquid insulin containing phenol	the insulin described in subsec A	the liquid insulin containing phenol described in subsec A	the insulin described in subsec A

Context 4: backward non-local references in section 2 part A.			
<i>original</i>	<i>ddxFirst</i>	<i>ddxLast</i>	<i>regen</i>
the human insulin containing folic acid	the insulin described in subsec B of section 1	the insulin containing folic acid described in subsec B of section 1	the human insulin containing folic acid

Context 5: backward local references in section 2 part C.			
<i>original</i>	<i>ddxFirst</i>	<i>ddxLast</i>	<i>regen</i>
the soluble insulin containing phenol	the insulin described in subsec B	the soluble insulin containing phenol	the soluble insulin containing phenol

Generation of Natural Language Explanations of Rules in an Expert System*

María de los Ángeles Alonso-Lavernia¹,
Argelio Víctor De-la-Cruz-Rivera², and Grigori Sidorov¹

¹ Center for Computing Research (CIC), National Polytechnic Institute (IPN),
Av. Juan de Dios Bátiz, Zacatenco,
DF, 07738, Mexico
marial@uaeh.uaehred.mx, sidorov@cic.ipn.mx

² Center for Research on Technologies of Information and Systems (CITIS),
Autonomous University of Hidalgo State (UAEH), Mexico

Abstract. We present a domain-independent method for generation of natural language explanations of rules in expert systems. The method is based on explanatory rules written in a procedural formal language, which build the explanation from predefined natural language texts fragments. For better style, a specific text fragment is randomly selected from a group of synonymous expressions. We have implemented 16 groups of explanatory rules and 74 groups of explanatory texts containing about 200 text fragments.

1 Introduction

Expert systems are widely used to solve particular problems in a rather narrow area of expertise. They are based on knowledge obtained during interaction with human experts in the field, so they are also often referred to as *knowledge-based systems*.

One of important requirements for an expert system is the system's ability to explain its conclusions in a manner understandable to the user. The best form of presenting such an explanation is a text in natural language [5]. One approach to generation of explanations is to use as explanation the rules from the knowledge base that were fired during reasoning [6]. Another approach is writing special code that paraphrases the rules [8]. These approaches do not allow for description of the ideas behind the fired rules. An alternative is to use another knowledge database for generation of explanations [7]. This approach requires a double amount of work for constructing knowledge bases.

In this paper, we present a method that allows for representation of the ideas behind the rules, does not require any additional knowledge bases, and is domain-independent—i.e., it does not require reprogramming of an explanation system if the knowledge base is changed.

* This work was done under partial support of Mexican Government (CONACyT, SNI), Autonomous University of Hidalgo State, and National Polytechnic Institute, Mexico.

2 Generation of Explanations

Our system consists of:

- Explanatory rules written in a procedural formal language,
- Groups of predefined text fragments used by the rules to assemble sentences, and
- Textual representation of facts contained in the knowledge base (these facts are part of the knowledge-based production rules, thus, they are domain dependent).

Our target language was Spanish, though the same explanatory rules can be easily adapted for other languages. The main linguistic problem is agreement models that are specific to each language. Also, there are some other problems related to word order and restrictions on syntactic constructions [2]. All these language-dependent details can be easily taken into account when adapting our method to another language by modifying the code of the explanatory rules. We used the HArIES [1], [4] programming language to implement the explanatory rules.

Explanatory rules are functions that perform specific operations for generation of the corresponding part of explanation, like verification of the number of arguments or analysis of the type of antecedent or consequent, etc. We use 16 groups of explanatory rules that is enough for construction of the whole system of explanations.

Texts fragments are inserted in the text slots of explanatory rules. They are classified as initial, medium, and final. The fragments are grouped together into sets of totally synonymous ones. For example, the group 1 contains “*Then this contributes*”; group 2 has “*the following condition*” and “*the following fact*”, etc. The rules refer to such a group rather than to an individual fragment, and each time a random text is selected from the group to fill the slot, which makes the style of the explanations more vivid. We have 74 such groups of predefined texts with about 200 individual fragments. Some special text fragments are used for formatting in a way similar to HTML, representing text colors, line feeds, etc.

Textual representation of facts is taken from the knowledge base rules. Usually, knowledge-based systems contain knowledge in form of production rules. There are nine types of propositions that depend on the type of antecedent (left part of the rule) or consequent (right part of the rule)—whether it is a simple value or logical combination, and the presence of positive or negative intervals of weights associated with them. The system automatically generates the correct explanation depending on the type of rule.

The highest-level explanatory rule, which represents the whole explanatory system, is very simple (we omit changing of text attributes in the rule for simplicity, though we show them by underlining in the example below):

```

if (no antecedent in the rule) then
  print (text_1 + " " + text_2 + CONSEQUENT)
else
  print (ANTECEDENT + linefeed)
  if (a weight is given for antecedent) then
    print (text_3 + CONSEQUENT)
  else
    print (text_4)
  if (a weight is given for negation of antecedent) then
    print (linefeed + text_5 + linefeed + text_6 + CONSEQUENT_NEGATIVE)

```

The symbol + stands for concatenation. The expression *text_i* refers to a group of synonymous fragments; a specific expression is chosen randomly from the group. ANTECEDENT, CONSEQUENT, and CONSEQUENT_NEGATIVE are other explanatory rules, which in turn contain references to other such rules. Other explanatory rules, which are also rather short, are implemented in a similar manner.

We treat uncertainty by using corresponding natural language expressions for encoding of each uncertainty value or interval: for example, values from the interval 90% to 99% are expressed as *practically all, nearly in all cases*, etc.

Here is an example (in Spanish, see translation below) of an explanation generated using the knowledge base of oil production. Numbers in parenthesis stand for facts in the knowledge base.

Regla de producción generalizada “(R1) 10 ⇒ 12 (-20 34)”:
Si es establecido con peso absolutamente seguro el hecho siguiente:
 ({10} producción original ALTA)
Entonces esto contribuye con poca seguridad negativa (-20%)
al conocimiento sobre el hecho siguiente:
 ({12} Se pronostica el Efecto POSITIVO a la inyección de Caldo)
Pero si por el contrario, el antecedente se incumple
Entonces esto contribuye con ciertas razones positivas (34%)
al peso global sobre la siguiente condición:
 ({12} Se pronostica el Efecto POSITIVO a la inyección de Caldo)

Translation of the example is as follows:

Generalized production rule “(R1) 10 → 12 (-20 34)”:
If the following fact is established with complete certainty weight:
 ({10} original production is HIGH)
Then this contributes with some negative weight (-20%)
to the knowledge about the following fact:
 ({12} POSITIVE effect of the Soup injection is expected),
but if, on the contrary, the antecedent is false
Then this contributes with certain positive degree (34%)
to the global weight of the following condition:
 ({12} POSITIVE effect of the Soup injection is expected).

3 Conclusions

We presented a method for generation of explanations that allows for generation of natural language explanations of production rules in expert systems, does not depend on knowledge base domain, and is easy to implement. The method can generate varying explanations of the same rule because text fragments are randomly chosen from a group of synonymous expressions associated with a rule.

The system consists of explanatory rules written in procedural programming language, fragments of natural language texts that are inserted in text slots in the explanatory rules, and textual descriptions of facts taken from the knowledge base.

Use of explanation rules makes generation of explanations independent of the program code: to add a new rule one does not need to add special code for generation of its explanation for the user.

References

- [1] Alonso, M. A. y de la Cruz, A. V. Desarrollo de Sistemas Educativos Utilizando Técnicas de Computación Inteligente. *1er Coloquio de Investigación en Ciencias de la Computación*. Memoria Técnica, pp. 73–79. 2001.
- [2] Christian Boitet. Message Automata for Messages with Variants, and Methods for their Translation. In: A. Gelbukh (Ed.), *Proc. of CICLing-2005*, Springer, 2005, pp. 349–368.
- [3] Clancey, W. Neomycin: Reconfiguring a Rule-Based Expert System for Application to Teaching. In: *Proceedings of 7th IJCAI*, Vancouver, B.C., Canada, Morgan-Kaufmann, San Mateo, Calif., 1981.
- [4] De la Cruz, A. V., Valdés, J. J., Jock, E., Balsa, J. y Rodríguez, A. *Fundamentos y Práctica de la Construcción de Sistemas Expertos*. Editorial “Academia”, Cuba, 1993.
- [5] Giarratano, J., and Riley, G.. *Expert Systems: Principles and Programming*. Boston: PWS, 1998.
- [6] Luger, G. F. and Stubblefield, W. A. *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*. Third Edition, Addison-Wesley, 1998.
- [7] Wick, M. R. and Thompson. Reconstructive Explanation: Explanation as Complex Problem Solving. In: *Proceedings of 11th IJCAI*, Morgan-Kaufmann, 1989.
- [8] Winograd, T. A. Computer Program for Understanding Natural Language. *Technical Report TR-17*, MIT Artificial Intelligence Laboratory, Cambridge, Mass., 1971.

Generating a Set of Rules to Determine Honorific Expression Using Decision Tree Learning

Kanako Komiya, Yasuhiro Tajima, Nobuo Inui, and Yoshiyuki Kotani

Department of Computer, Information and Communication Sciences,
Tokyo University of Agriculture and Technology,
2-24-16, Nakacho, Koganei, Tokyo, Japan, 184-8588
komiya@fairry.ei.tuat.ac.jp

Abstract. In Japanese language, the speaker must choose suitable honorific expressions depending on many factors. The computer system should imitate this mechanism to make a natural Japanese sentence. We made a system to determine a suitable expression and named it honorific expression determining system (HEDS). It generates a set of rules to determine suitable honorific expression automatically, by decision tree learning. The system HEDS determines one out of the three classes for an input sentence: the respect expression, the modesty expression and the non-honorific expression and determines what expression the verb is. We calculated the accuracy of HEDS using the cross validation method and it was up to 74.88%.

1 Introduction

In Japanese language, one must choose suitable honorific expressions depending on the speaker, the addressees, the subject of the utterance, contents of the dialogue and situations in the conversation. The computer system should imitate this mechanism to make a natural Japanese sentence.

Japanese language has the two types of honorific expression: (1) respect or modesty expression and (2) polite expression. The respect expression is used to display respect to others, or their higher rank, and practically to show second person of the sentential implicit subject, in contrast that the modesty expression shows first person. The modesty expression is an expression that one display modesty to respecting persons. These two honorific expressions cannot be used in a single word at the same time, but the combination of (1) and (2) can be used for one word at the same time. We focus on the type (1) in this paper.

2 Honorific Expression Determining System (HEDS)

The user of HEDS provides honorific expressions and its factors for determining suitable honorific expressions as data. Then HEDS generates a set of selection rules. It determines one out of the three classes for input factors of a sentence: the respect expression, the modesty expression and the non-honorific expression and determines what expression the verb is. (cf. Fig. 1).

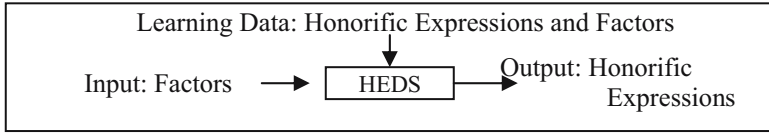


Fig. 1. The system HEDS

2.1 Input: The Factors to Determine the Honorific Expression

We extracted the two following types of the factors to determine the honorific expression and the honorific expressions that are used under the situation, gave them to HEDS and performed the decision tree learning.

First, we call the factors based on relationship between persons “relational factors”. We selected the 16 relational factors as follows.

- | | |
|--|---|
| (R1) The gender of the speaker, | (R10) The gender of the addressee |
| (R2) The age of the speaker | (R11) The standing point 1 of the addressee |
| (R3) Whether or not an subject is a human being | (R12) The standing point 2 of the addressee |
| (R4) The gender of the subject | (R13) The age of the addressee |
| (R5) The standing point 1 of the subject | (R14) Whether or not the subject is the same person as the addressee |
| (R6) The standing point 2 of the subject | (R15) The degree of familiarity between the speaker and the addressee |
| (R7) The age of the subject | (R16) The age difference between the speaker and the addressee |
| (R8) The degree of familiarity between the speaker and the subject | |
| (R9) The age difference between the speaker and the subject | |

Each relational factor has from 2 to 32 alternatives. We defined rules to select suitable alternative, and selected suitable alternatives manually according to the rules.

Second, we call the factors based on language structures or words in the utterance “linguistic factors”. We selected the 4 linguistic factors as follows.

- (L1) The linguistic form (e.g. imperative form, subjunctive form, etc.)
- (L2) Whether or not there are irregular expressions as the alternatives to a regular respect expression
- (L3) Whether or not there are irregular expressions as the alternatives to a regular modesty expression
- (L4) The ending of a word

Each linguistic factor has from 2 to 7 alternatives.

We used Chasen [3] to conduct morphological analysis and modified the inconvenient results manually. We refer to [1] about the alternatives to a regular honorific expressions and added some expressions like “□□(ORU)”. We defined rules to select suitable alternative, and selected suitable alternatives manually according to the rules.

2.2 Output: Honorific Expressions

We defined respect expression as 24 kinds of expressions like “お * * だ(O**DA)” including irregular expressions as the alternatives to a regular respect expression and their compound types. “おっしやる(OSSHARU)” is one of the examples of the irregular expressions. In addition we defined modesty expression as 26 kinds of expressions like “お * * いただく(O**ITADAKU)” including irregular expressions as the alternatives to a regular modesty expression and their compound types. “申す(MOUSU)” is one of the examples of the irregular expressions.

We refer to [1] about the honorific expressions and modified some examples partly. For example, we defined “お願いだ(ONEGAIDA)” as modesty expression. For verbs that include both the respect expression and the modesty expression, we made some rules to judge which expression they are and judged them.

2.3 Decision Tree Learning Based on Honorific Expressions

The decision tree is a way to describe some classification of data, and consists of query nodes. (cf. Fig. 2). Each node in the tree classifies the input into a few classes according to the attribute of the dataset. The decision tree learning is a method for constructing a decision tree. It generates a tree automatically from leaning data (many examples), and we follow the tree from the root node to the leaf top down and we get the suitable result. We used relational factors and linguistic factors as the datasets, used its alternatives as the attributes of the datasets and get the most honorific expression as the output. We used ID3 as the learning algorithm in HEDS, and generated binary decision tree.

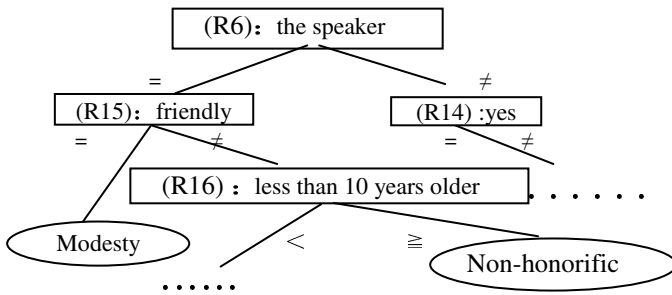


Fig. 2. A part of the decision tree made by the experiment

3 Experiment and Results

We gathered 1201 conversation sentences from 11 novels and selected factors to determine suitable honorific expressions, its alternatives and honorific expressions that are used under the situations.

We used following termination condition: (1) Whether the information gain is zero or not and (2) threshold values. We tried two kinds of threshold value: 1) A value of

entropy of a node and 2) A value that multiply a value of entropy of a node and numbers of data in the node together. We calculated the accuracy using the cross validation method and it was up to 74.88% when the 2) was 20.

We show the two examples.

1) “私もそう思うわ。” (Watashi mo sou *omou* wa. I *think* so too.)

In this case, a speaker is a female high school student, a subject is the same person as the speaker, and the addressee is a female high school student, a very friendly friend of the speaker and the same age as the speaker. The linguistic form is end-form, and there is no irregular expression as the alternatives to a regular respect expression but there is an irregular expression as the alternatives to a regular modesty expression, and the ending of a word is the other. HEDS answered it is a non-honorific expression and it is correct.

2) “私は聞かれてね。” (Watashi wa *kikarete* ne. I *was asked*.)

In this case, a speaker is an adult female, and a subject is the same person as the speaker (cf. Fig. 2). The addressee is an adult male and a costumer, and the speaker doesn't know much about him. The addressee is 10 years or more older than the speaker. The linguistic form is continuous form, and there is no irregular expression as the alternatives to a regular respect expression or a regular modesty expression, and the ending of a word is passive. It is not a correct example because HEDS answered it is a modesty expression although in fact it is a non-honorific expression.

4 Discussion

The question in the root node of generated tree is whether or not the standing point 2 of each subject is “the speaker”, that is, whether or not the subject is the same person as the speaker. If it is true, the verb in the sentence tends to be a modesty expression and if not, it tends to be a respect expression. It reflects that in Japanese language, one tends to use modesty expressions on him/herself and use respect expressions on other people.

5 Conclusion

We made honorific expression determining system (HEDS). It generates a set of rules to determine suitable honorific expression automatically, by decision tree learning. We calculated the accuracy using the cross validation method and it was up to 74.88%.

References

1. Masuro, O. (ed.): Dictionary of Situation-by-Situation Honorific Expression Usage, Tokyodo-Shuppan, Tokyo (1999)
2. Koichi, F.:Data Mining: Concepts and Techniques <http://web.sfc.keio.ac.jp/~soh/dm03/dm03-8.ppt>.
3. ChaSen's Wiki, <http://chasen.aist-nara.ac.jp/hiki/ChaSen/>

NLP (Natural Language Processing) for NLP (Natural Language Programming)

Rada Mihalcea¹, Hugo Liu², and Henry Lieberman²

¹ Computer Science Department, University of North Texas
rada@cs.unt.edu

² Media Arts and Sciences, Massachusetts Institute of Technology
{hugo, henry}@media.mit.edu

Abstract. Natural Language Processing holds great promise for making computer interfaces that are easier to use for people, since people will (hopefully) be able to talk to the computer in their own language, rather than learn a specialized language of computer commands. For programming, however, the necessity of a formal programming language for communicating with a computer has always been taken for granted. We would like to challenge this assumption. We believe that modern Natural Language Processing techniques can make possible the use of natural language to (at least partially) express programming ideas, thus drastically increasing the accessibility of programming to non-expert users. To demonstrate the feasibility of Natural Language Programming, this paper tackles what are perceived to be some of the hardest cases: steps and loops. We look at a corpus of English descriptions used as programming assignments, and develop some techniques for mapping linguistic constructs onto program structures, which we refer to as programmatic semantics.

1 Introduction

Natural Language Processing and Programming Languages are both established areas in the field of Computer Science, each of them with a long research tradition. Although they are both centered around a common theme – “languages” – over the years, there has been only little interaction (if any) between them¹. This paper tries to address this gap by proposing a system that attempts to convert natural language text into computer programs. While we overview the features of a natural language programming system that attempts to tackle both the descriptive and procedural programming paradigms, in this paper we focus on the aspects related to procedural programming. Starting with an English text, we show how a natural language programming system can automatically identify steps, loops, and comments, and convert them into a program *skeleton* that can be used as a starting point for writing a computer program, expected to be particularly useful for those who begin learning how to program.

We start by overviewing the main features of a descriptive natural language programming system METAFOR introduced in recent related work [6]. We then describe in

¹ Here, the obvious use of programming languages for coding natural language processing systems is not considered as a “meaningful” interaction.

detail the main components of a procedural programming system as introduced in this paper. We show how some of the most difficult aspects of procedural programming, namely steps and loops, can be handled effectively using techniques that map natural language onto program structures. We demonstrate the applicability of this approach on a set of programming assignments automatically mined from the Web.

2 Background

Early work in natural language programming was rather ambitious, targeting the generation of complete computer programs that would compile and run. For instance, the “NLC” prototype [1] aimed at creating a natural language interface for processing data stored in arrays and matrices, with the ability of handling low level operations such as the transformation of numbers into type declarations as e.g. `float-constant(2.0)`, or turning natural language statements like *add y1 to y2* into the programmatic expression `y1 + y2`. These first attempts triggered the criticism of the community [3], and eventually discouraged subsequent research on this topic.

More recently, however, researchers have started to look again at the problem of natural language programming, but this time with more realistic expectations, and with a different, much larger pool of resources (e.g. broad spectrum commonsense knowledge [9], the Web) and a suite of significantly advanced publicly available natural language processing tools.

For instance, Pane & Myers [8] conducted a series of studies with non-programming fifth grade users, and identified some of the programming models implied by the users’ natural language descriptions. In a similar vein, Lieberman & Liu [5] have conducted a feasibility study and showed how a partial understanding of a text, coupled with a dialogue with the user, can help non-expert users make their intentions more precise when designing a computer program. Their study resulted in a system called METAFOR [6], [7], able to translate natural language statements into class descriptions with the associated objects and methods.

Another closely related area that received a fair bit of attention in recent years is the construction of natural language interfaces to databases, which allows users to query structured data using natural language questions. For instance, the system described in [4], or previous versions of it as described in [10], implements rules for mapping natural to “formal” languages using syntactic and semantic parsing of the input text. The system was successfully applied to the automatic translation of natural language text into RoboCup coach language [4], or into queries that can be posed against a database of U.S. geography or job announcements [10].

3 Descriptive Natural Language Programming

When storytellers speak fairy tales, they first describe the fantasy world – its characters, places, and situations – and then relate how events unfold in this world. Programming, resembling storytelling, can likewise be distinguished into the complementary tasks of *description* and *proceduralization*. While this paper tackles primarily the basics of

building procedures out of steps and loops, it would be fruitful to also contextualize procedural rendition by discussing the architecture of the descriptive world that procedures animate.

Among the various paradigms for computer programming – such as logical, declarative, procedural, functional, object-oriented, and agent-oriented – the object-oriented and agent-oriented formats most closely embody human storytelling intuition. Consider the task of programming a MUD² world by natural language description, and the sentence *There is a bar with a bartender who makes drinks* [6]. Here, `bar` is an instance of the object class `bar`, and `bartender` is an instance of the agent (a class with methods) class `bartender`, with the capability `makeDrink(drink)`. Generalizing from this example, characters are reified as agent classes, things and places become object classes, and character capabilities become class methods.

A theory of programmatic semantics for descriptive natural language programming is presented in [7]; here, we overview its major features, and highlight some of the differences between descriptive and procedural rendition. These features are at the core of the Metafor [6] natural language programming system that can render code following the descriptive paradigm, starting with a natural language text.

3.1 Syntactic Correspondences

There are numerous syntactic correspondences between natural language and descriptive structures. Most of today’s natural languages distinguish between various parts of speech that taggers such as Brill’s [2] can parse – noun chunks are things, verbs are actions, adjectives are properties of things, adverbs are parameters of actions. Almost all natural languages are built atop the basic construction called *independent clause*, which at its heart has a *who-does-what* structure, or subject-verb-directObject-indirectObject (SVO) construction. Although the ordering of subject, verb, and objects differ across verb-initial (VSO and VOS, e.g. Tagalog), verb-medial (SVO, e.g. Thai and English), and verb-final languages (SOV, e.g., Japanese), these basic three ingredients are rather invariant across languages, corresponding to an encoding of agent-method and method-argument relationships. This kind of syntactic relationships can be easily recovered from the output of a syntactic parser, either supervised, if a treebank is available, or unsupervised for those languages for which manually parsed data does not exist. Note that the syntactic parser can also resolve other structural ambiguity problems such as prepositional attachment. Moreover, other ambiguity phenomena that are typically encountered in language, e.g. pronoun resolution, noun-modifier relationships, named entities, can be also tackled using current state-of-the-art natural language processing techniques, such as coreference tools, named entity annotators, and others.

Starting with an SVO structure, we can derive agent-method and method-argument constructions that form the basis of descriptive programming. Particular attention needs to be paid to the ISA type of constructions that indicate inheritance. For instance, the statement *Pacman is a character who ...* indicates a super-class `character` for the more specific class `Pacman`.

² A MUD (multi-user dungeon, dimension, or dialogue) is a multi-player computer game that combines elements of role-playing games, hack and slash style computer games, and social instant messaging chat rooms (definition from wikipedia.org).

3.2 Scoping Descriptions

Scoping descriptions allow conditional `if/then` rules to be inferred from natural language. Conditional sentences are explicit declarations of `if/then` rules, e.g. *When the customer orders a drink, make it*, or *Pacman runs away if ghosts approach*. Conditionals are also implied when uncertain voice is used, achieved through modals as in e.g. *Pacman may eat ghosts*, or adverbials like *sometimes* – although in the latter case the antecedent to the `if/then` is underspecified or omitted, as in *Sometimes Pacman runs away*.

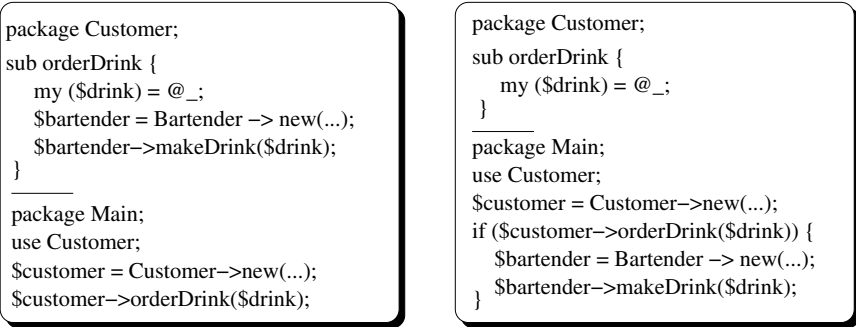


Fig. 1. The descriptive and procedural representations for the conditional statement *When customer orders a drink, the bartender makes it*

An interesting interpretative choice must be made in the case of conditionals, as they can be rendered either descriptively as functional specifications, or procedurally as `if/then` constructions. For example, consider the utterance *When customer orders a drink, the bartender makes it*. It could be rendered descriptively as shown on the left of Figure 1, or it could be proceduralized as shown on the right of the same figure. Depending upon the surrounding discourse context of the utterance, or the desired representational orientation, one mode of rendering might be preferred over the other. For example, if the storyteller is in a descriptive mood and the preceding utterance was *there is a customer who orders drinks*, then most likely the descriptive rendition is more appropriate.

3.3 Set-Based Dynamic Reference

Set-based dynamic reference suggests that one way to interpret the rich descriptive semantics of compound noun phrases is to map them into mathematical sets and set-based operations. For example, consider the compound noun phrase *a random sweet drink from the menu*. Here, the head noun *drink* is being successively modified by *from the menu*, *sweet*, and *random*. One strategy in unraveling the utterance’s programmatic implications is to view each modifier as a constraint filter over the set of all drink instances. Thus the object `aRandomSweetDrinkFromTheMenu` implies a procedure that creates a set of all drink instances, filters for just those listed in `theMenu`, filters for those having the property *sweet*, and then applies a random choice to the remaining drinks to select a single one. Set-based dynamic reference lends great conciseness and power to

natural language descriptions, but a caveat is that world semantic knowledge is often needed to fully exploit their semantic potential. Still, without such additional knowledge, several descriptive facts can be inferred from just the surface semantics of *a random sweet drink from the menu* – there are things called drinks, there are things called menus, drinks can be contained by menus, drinks can have the property *sweet*, drinks can have the property *random* or be selected randomly. Later in this paper, we harness the power of set-based dynamic reference to discover implied repetition and loops.

Occam's Razor would urge that code representation should be as simple as possible, and only complexified when necessary. In this spirit, we suggest that automatic programming systems should adopt the simplest code interpretation of a natural language description, and then complexify, or *dynamically refactor*, the code as necessary to accommodate further descriptions. For example, consider the following progression of descriptions and the *simplest common denominator* representation implied by all utterances *up to* that step.

- a) There is a bar. (atom)
- b) The bar contains two customers. (unimorphic list of type Customer)
- c) It also has a waiter. (unimorphic list of type Person)
- d) It has some stools. (polymorphic list)
- e) The bar opens and closes. (class / agent)
- f) The bar is a kind of store. (agent with inheritance)
- g) Some bars close at 6pm, others at 7pm. (forks into two subclasses)

Applying the semantic patterns of syntactic correspondence, representational equivalence, set-based dynamic reference, and scoping description to the interpretation of natural language description, object-oriented code skeletons can be produced. These description skeletons then serve as a code model which procedures can be built out of. Mixed-initiative dialog interaction between computer and storyteller can disambiguate difficult utterances, and the machine can also use dialog to help a storyteller describe particular objects or actions more thoroughly.

The Metafor natural language programming system [6] implementing the features highlighted in this section was evaluated in a user study, where 13 non-programmers and intermediate programmers estimated the usefulness of the system as a brainstorming tool. The non-programmers found that Metafor reduced their programming task time by 22%, while for intermediate programmers the figure was 11%. This result supports the initial intuition from [5] and [8] that natural language programming can be a useful tool, in particular for non-expert programmers.

It remains an open question whether Metafor will represent a stepping stone to real programming, or will lead to a new programming paradigm obviating the need for a formal programming language. Either way, we believe that Metafor can be useful as a tool in itself, even if it is yet to see which way it will lead.

4 Procedural Natural Language Programming

In procedural programming, a computer program is typically composed of sequences of action statements that indicate the operations to be performed on various data structures. Correspondingly, procedural natural language programming is targeting the generation

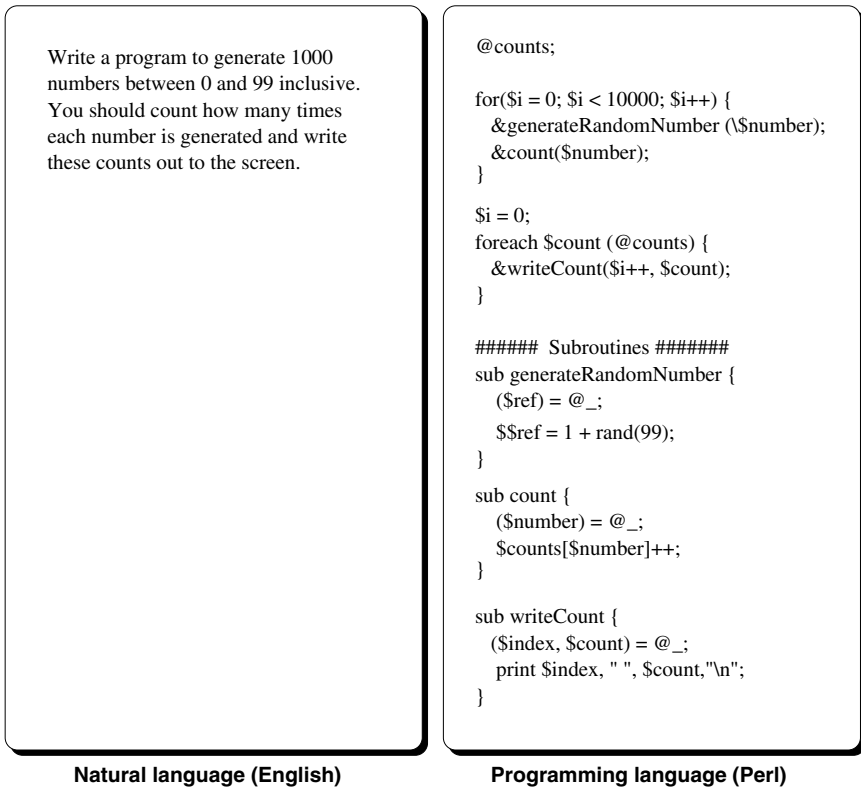


Fig. 2. Side by side: the natural language (English) and programming language (Perl) expressions for the same problem

of computer programs following the procedural paradigm, starting with a natural language text.

For example, starting with the natural language text on the left side of figure 2, we would ideally like to generate a computer program as the one shown on the right side of the figure³. While this is still a long term goal, in this section we show how we can automatically generate computer program skeletons that can be used as a starting point for creating procedural computer programs. Specifically, we focus on the description of three main components of a system for natural language procedural programming:

- The *step finder*, which has the role of identifying in a natural language text the action statements to be converted into programming language statements.
- The *loop finder*, which identifies the natural language structures that indicate repetition.
- Finally, the *comment identification* components, which identifies the descriptive statements that can be turned into program comments.

³ Although the programming examples shown throughout this section are implemented using Perl, other programming languages could be used equally well.

Starting with a natural language text, the system is first analyzing the text with the goal of breaking it down into steps that will represent action statements in the output program. Next, each step is run through the comment identification component, which will mark the statements according to their descriptive role. Finally, for those steps that are not marked as comments, the system is checking if a step consists of a repetitive statement, in which case a loop statement is produced using the corresponding loop variable. The following sections provide details on each of these components (step finder, loop finder, comment identification), as well as a walk-through example illustrating the process of converting natural language texts into computer program skeletons.

4.1 The Step Finder

The role of this component is to read an input natural language text and break it down into steps that can be turned into programming statements. For instance, starting with the natural language text *You should count how many times each number is generated and write these counts out to the screen.* (see figure 2), two main steps should be identified: (1) [`count` how many times each number is generated], and (2) [`write` these counts out to the screen].

First, the text is pre-processed, i.e. tokenized and part-of-speech tagged using Brill's tagger [2]. Some language patterns specific to program descriptions are also identified at this stage, including phrases such as *write a program*, *create an applet*, etc., which are not necessarily intended as action statements to be included in a program, but rather as general directives given to the programmer.

Next, steps are identified as statements containing one verb in the active voice. We are therefore identifying all verbs that could be potentially turned into program functions, such as e.g. `read`, `write`, `count`. We attempt to find the boundaries of these steps: a new step will start either at the beginning of a new sentence, or whenever a new verb in the active voice is found (typically in a subordinate clause).

Finally, the *object* of each action is identified, consisting of the direct object of the active voice verb previously found, if such a direct object exists. We use a shallow parser to find the noun phrase that plays the role of a direct object, and then identify the head of this noun phrase as the *object* of the corresponding action.

The output of the step finder process is therefore a series of natural language statements that are likely to correspond to programming statements, each of them with their corresponding action that can be turned into a program function (as represented by the active voice verb), and the corresponding action object that can be turned into a function parameter (as represented by the direct object). As a convention, we use both the verb and the direct object to generate a function name. For example, the verb *write* with the parameter *number* will generate the function call `writeNumber(number)`.

4.2 The Loop Finder

An important property of any program statement is the number of times the statement should be executed. For instance, the requirement to *generate 10000 random numbers* (see figure 2), implies that the resulting action statement of [`generate` random numbers] should be repeated 10000 times.

The role of the loop finder component is to identify such natural language structures that indicate repetitive statements. The input to this process consists of steps, fed one at a time, from the series of steps identified by the step finder process, together with their corresponding actions and parameters. The output is an indication of whether the current action should be repeated or not, together with information about the loop variable and/or the number of times the action should be repeated.

First, we seek explicit markers of repetition, such as *each X*, *every X*, *all X*. If such a noun phrase is found, then we look for the head of the phrase, which will be stored as the loop variable corresponding to the step that is currently processed. For example, starting with the statement *write all anagrams occurring in the list*, we identify *all anagrams* as a phrase indicating repetition, and *anagram* as the loop variable.

If an explicit indicator of repetition is not found, then we look for plural nouns as other potential indicators of repetition. Specifically, we seek plural nouns that are the head of their corresponding noun phrase. For instance, the statement *read the values* contains one plural noun (*values*) that is the head of its corresponding noun phrase, which is thus selected as an indicator of repetition, and it is also stored as the loop variable for this step. Note however that a statement such as *write the number of integers* will not be marked as repetitive, since the plural noun *integers* is not the head of a noun phrase, but a modifier.

In addition to the loop variable, we also seek an indication of how many times the loop should be repeated – if such information is available. This information is usually furnished as a number that modifies the loop variable, and we thus look for words labeled with a cardinal part-of-speech tag. For instance, in the example *generate 10000 random numbers*, we first identify *numbers* as an indicator of repetition (noun plural), and then find *10000* as the number of times this loop should be repeated. Both the loop variable and the loop count are stored together with the step information.

Finally, another important role of the loop finder component is the unification process, which seeks to combine several repetitive statements under a common loop structure, if they are linked by the same loop variable. For example, the actions [generate numbers] and [count numbers] will be both identified as repetitive statements with a common loop variable *number*, and thus they will be grouped together under the same loop structure.

4.3 Comment Identification

Although not playing a role in the execution of a program, comments are an important part of any computer program, as they provide detailed information on the various programming statements.

The comment identification step has the role of identifying those statements in the input natural language text that have a descriptive role, i.e. they provide additional specifications on the statements that will be executed by the program.

Starting with a step as identified in the step finding stage, we look for phrases that could indicate a descriptive role of the step. Specifically, we seek the following natural language constructs: (1) Sentences preceded by one of the expressions *for example*, *for instance*, *as an example*, which indicate that the sentence that follows provides an example of the expected behavior of the program. (2) Statements including a modal

verb in a conditional form, such as *should*, *would*, *might*, which are also indicators of expected behavior. (3) Statements with a verb in the passive voice, if this is the only verb in the statement⁴. (4) Finally, statements indicating assumptions, consisting of sentences that start with a verb like *assume*, *note*, etc. All the steps found to match one of these conditions are marked as comments, and thus no attempt will be made to turn them into programming statements.

An example of a step that will be turned into a comment is *For instance, 23 is an odd number*, which is a statement that has the role of illustrating the expected behavior of the program rather than asking for a specific action, and thus it is marked as a comment.

The output of the comment identification process is therefore a flag associated with each step, indicating whether the step can play the role of a comment. Note that although all steps, as identified by the step finding process, can play the role of informative comments in addition to the programming statements they generate, only those steps that are not explicitly marked as comments by the comment identification process can be turned into programming statements. In fact, the current system implementation will list all the steps in a comment section (see the sample output in Figure 2), but it will not attempt to turn any of the steps marked as “comments” into programming statements.

4.4 A Walk-Through Example

Consider again the example illustrated in figure 2. The generation of a computer program skeleton follows the three main steps highlighted earlier: step identification, comment identification, loop finder.

First, the step finder identifies the main steps that could be potentially turned into programming statements. Based on the heuristics described in section 4.1, the natural language text is broken down into the following steps: (1) [*generate 10000 random numbers between 0 and 99 inclusive*], (2) [*count how many of times each number is generated*], (3) [*write these counts out to the screen*], with the functions/parameters: `generateNumber(number)`, `count()`, and `writeCount(count)`.

Next, the comment finder does not identify any descriptive statements for this input text, and thus none of the steps found by the step finder are marked as comments. By default, all the steps are listed in the output program in a comment section.

Finally, the loop finder inspects the steps and tries to identify the presence of repetition. Here, we find a loop in the first step, with the loop variable `number` and loop count `10000`, a loop in the second step using the same loop variable `number`, and finally a loop in the third step with the loop variable `count`. Another operation performed by the loop finder component is unification, and in this case the first two steps are grouped under the same loop structure, since they have a common loop variable (`number`).

The output generated by the natural language programming system for the example in figure 2 is shown in figure 3.

⁴ Note that although modal and passive verbs could also introduce candidate actions, since for now we target program skeletons and not fully-fledged programs that would compile and run, we believe that it is important to separate the main actions from the lower level details. We therefore ignore the “suggested” actions as introduced by modal or passive verbs, and explicitly mark them as comments.


```

=====
# Write a program to generate 10000 random numbers between 0 and
# 99 inclusive. You should count how many of times each number
# is generated and write these counts out to the screen.
=====

for($i = 0; $i < 10000; $i++) {
    # to generate 10000 random numbers between 0 and 99 inclusive
    &generateNumber(number)

    # You should count how many of times each number is generated
    &count()
}

foreach $count (@counts) {
    # write these counts out to the screen
    &writeCount(count)
}

```

Fig. 3. Sample output produced by the natural language programming system, for the example shown in figure 2

4.5 Evaluation and Results

One of the potential applications of such a natural language programming system is to assist those who begin learning how to program, by providing them with a skeleton of computer programs as required in programming assignments. Inspired by these applications, we collect a corpus of homework assignments as given in introductory programming classes, and attempt to automatically generate computer program skeletons for these programming assignments.

The corpus is collected using a Web crawler that searches the Web for pages containing the keywords *programming* and *examples*, and one of the keyphrases *write a program*, *write an applet*, *create a program*, *create an applet*. The result of the search process is a set of Web pages likely to include programming assignments. Next, in a post-processing phase, the Web pages are cleaned-up of HTML tags, and paragraphs containing the search keyphrases are selected as potential descriptions of programming problems. Finally, the resulting set is manually verified and any remaining noisy entries are thusly removed. The final set consists of 120 examples of programming assignments, with three examples illustrated in Table 1.

For the evaluation, we randomly selected a subset of 25 programming assignments from the set of Web-mined examples, and used them to create a gold standard testbed. For each of the 25 program descriptions, we manually labeled the main steps (which should result into programming statements), and the repetitive structures (which should result into loops). Next, from the automatically generated program skeletons, we identified all those steps and loops that were correct according to the gold standard, and

Table 1. Sample examples of programming assignments

Write a program that reads a string of keyboard characters and writes the characters in reverse order.
Write a program to read 10 lines of text and then writes the number of words contained in those lines.
Write a program that reads a sequence of integers terminated by any negative value. The program should then write the largest and smallest values that were entered.

correspondingly evaluate the precision and the recall of the system. Specifically, precision is defined as the number of correct programmatic structures (steps or loops) out of the total number of structures automatically identified; the precision for the step identification process was measured at 86.0%, and for the loop identification at 80.6%. The recall is defined as the number of correct programmatic structures from the total number of structures available in the gold standard; it was measured at 75.4% for the step identification component, and at 71.4% for the loop finder.

5 The Future: NLP for NLP

Natural language processing for natural language programming or natural language programming for natural language processing? We would argue that the benefits could go both ways.

Despite the useful “universal” aspect of programming languages, these languages are still understood only by *very few* people, unlike the natural languages which are understood by *all*. The ability to turn natural into programming languages will eventually decrease the gap between *very few* and *all*, and open the benefits of computer programming to a larger number of users. In this paper, we showed how current state-of-the-art techniques in natural language processing can allow us to devise a system for natural language programming that addresses both the descriptive and procedural programming paradigms. The output of the system consists of automatically generated program skeletons, which were shown to help non-expert programmers in their task of describing algorithms in a programmatic way. As it turns out, advances in natural language processing helped the task of natural language programming.

But we believe that natural language processing could also benefit from natural language programming. The process of deriving computer programs starting with a natural language text implies a plethora of sophisticated language processing tools – such as syntactic parsers, clause detectors, argument structure identifiers, semantic analyzers, methods for coreference resolution, and so forth – which can be effectively put at work and evaluated within the framework of natural language programming. We thus see natural language programming as a potential large scale end-user (or rather, end-computer) application of text processing tools, which puts forward challenges for the natural language processing community and could eventually trigger advances in this field.

References

1. BALLARD, B., AND BIERMAN, A. Programming in natural language: "NLC" as a prototype. In *Proceedings of the 1979 annual conference of ACM/CSC-ER* (1979).
2. BRILL, E. Transformation-based error driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics* 21, 4 (December 1995), 543–566.
3. DIJKSTRA, E. On the foolishness of "Natural Language Programming". In *Program Construction, International Summer School* (1979).
4. KATE, R., WONG, Y., GE, R., AND MOONEY, R. Learning to transform natural to formal languages. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-05)* (Pittsburgh, 2005).
5. LIEBERMAN, H., AND LIU, H. *Feasibility studies for programming in natural language*. Kluwer Academic Publishers, 2005.
6. LIU, H., AND LIEBERMAN, H. Metafor: Visualizing stories as code. In *ACM Conference on Intelligent User Interfaces (IUI-2005)* (San Diego, 2005).
7. LIU, H., AND LIEBERMAN, H. Programmatic semantics for natural language interfaces. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI-2005)* (Portland, OR, 2005).
8. PANE, J., RATANAMAHATANA, C., AND MYERS, B. Studying the language and structure in non-programmers' solutions to programming problems. *International Journal of Human-Computer Studies* 54, 2 (2001).
9. SINGH, P. The Open Mind Common Sense project. *KurzweilAI.net* (January 2002). Available online from <http://www.kurzweilai.net/>.
10. TANG, L., AND MOONEY, R. Using multiple clause constructors in inductive logic programming for semantic parsing. In *Proceedings of the 12th European Conference on Machine Learning (ECML-2001)* (Freiburg, Germany, 2001).

Balancing Transactions in Practical Dialogues

Luis Albreto Pineda Cortés, Hayde Castellanos, Sergio Rafael Coria Olguin, Varinia Estrada, Fernanda López, Isabel López, Ivan Meza, Iván Moreno, Patricia Pérez, and Carlos Rodríguez

Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas (IIMAS),
Universidad Nacional Autónoma de México (UNAM),
Cto. Escolar S/N, Cd. Universitaria, Coyoacán, México, D.F.
luis@leibniz.iimas.unam.mx

Abstract. In this paper a theory of dialogue acts analysis in problem-solving tasks-oriented conversations is presented. The theory postulates that in practical dialogues every transaction has a component in the obligations and the common ground planes of expression, and contributions made by dialogue acts making a “charge” in the transaction should be “balanced” by contributions making the corresponding “credit”, and a complete transaction is balanced in both of these planes. In addition, transactions have a structure which constraints strongly the realization of dialogue acts. A dialogue act tagging methodology based on the theory is also presented. The theory and its related methodology have been applied to the analysis of a multimodal corpus in a design task, and the figures of the agreement reached in the preliminary experiments are presented.

1 Introduction

In this paper a theory for the analysis of dialog acts in practical dialogs is presented. In this theory dialogues acts are analyzed in relation to the obligations and common ground structures of task oriented conversations, and we provide an explicit analysis and tagging methodology for these two dialogue structures. According to Allen et al. [1], practical dialogues have the purpose to achieve a concrete goal, and the conversational competence required to engage in this kind of dialogs is significantly simpler than general human conversation (i.e. the practical dialogue hypothesis) and the main aspects of language interpretation and dialogue management are domain independent (i.e. domain independence hypothesis). Simple dialogues can be reduced to achieve a single goal and involve only one transaction, but often the dialogue involves a sequence of transactions. From the empirical study of a corpus in the kitchen design domain we suggest that transactions are also characterized in terms of an intention specification phase, followed by the intention satisfaction phase, and the structure of the dialogue is closely related to the structure of the problem-solving task, and in this regard, our approach loosely resembles Grosz and Sidner’s discourse theory [7]. We also postulate the hypothesis that transactions can be analyzed in terms of their conversational obligations and common ground structures, and that complete transactions are balanced in these two planes of expression; this is, for every “charge” in each of these planes there must be a “credit”; otherwise, the transaction cannot be completed successfully.

In addition to the principles of cooperative conversation, in which the conversational participants share beliefs and desires, and communicative intentions can be expressed and satisfied by a shared plan, there are strong social conventions involved in linguistic interaction that hold even if one of the conversational participants opts out of cooperative behavior [9]. A question by speaker *A*, for instance, imposes the obligation on *B* to provide an answer and a commitment made by *A* imposes the obligation on *A* himself to perform an action. Conversational participants should also try to establish the mutual belief that the addressee has understood what the speaker meant for each utterance; if this common ground is lost the conversation cannot proceed successfully. Different discourse theories make different assumptions about what knowledge constitutes the common ground, and most involve presuppositions and beliefs accumulated during the conversational interchange; however, in addition to this knowledge, successful grounding behavior requires that each utterance is understood as intended [5], and this basic level of linguistic communication is independent of the content of the utterance; for instance, all utterances must be acknowledged, either explicitly or implicitly, in a reasonable amount of time, or the flow of communication is interrupted and has to be repaired. At this level, the common ground involves agreement about the intended attitude towards the content of the utterances (e.g. whether a question is accepted or put on hold) and also communication factors, as when an utterance is acknowledged directly, through a back-channel or a repetition.

The notions of conversational obligations and grounding have been applied to the definition of dialogue managers [2,9]; however, these planes of expression are not reflected directly in annotation schemes, like DAMSL [3], which has been used for analysis of dialogues with the purpose of specifying performance goals for conversational systems [2]. This latter scheme distinguishes between the communicative status, the information level and the forward and backward looking functions of utterances, but discourse obligations and common ground acts are distributed implicitly in these four main dimensions. In particular, utterances expressing obligations are the prominent part of the forward looking functions, but these utterances have also a grounding import, and conversely, although most explicit tags of the backward looking functions are mainly concerned with grounding, there are also some backward functions that belong to the obligations structure. More generally, although the utterances of a dialogue perform both functions at the surface level, the obligations and grounding structures are different, as each is focused on a different linguistic function.

In this paper we report a study on the structure of obligations and common ground in task oriented dialogues; in this work we have extended DAMSL with a meta-layer of interpretation reflecting these structures. In this level, transactions are identified first, and utterances making a “charge” in the obligations and grounding structures must be eventually “credited” by another utterance, either implicitly or through an explicit utterance. The goal of a transaction is accomplished when the phases of intention specification and satisfaction are completed, and the transaction as a whole is balanced. This meta-level of tags is then used to identify the actual tags for dialogue acts, following closely the original DAMSL tagging scheme. A tagging exercise in the kitchen design domain is described, and the agreement among taggers is reported. Also, as the corpus is multimodal we have extended DAMSL with a dimension for tagging actions and visual interpretations that are not expressed linguistically. We refer to this tagging scheme as DIME-DAMSL.

2 The Obligations and Grounding Structures

To illustrate these structures we present the analysis of a basic transaction taken from the DIME Corpus [10], which was collected in the context of the DIME project [8]. This transaction is as follows (translated from the originally in Spanish):

Table 1. Basic transaction

Speaker	Utterance	Text
U	25	After that <sil> can you put <sil> the the air extractor on top of the <sil> of the stove
S	26	Okay
S	27	<Graphics interactive action> Is this okay?
U	28	Yes, it's okay

The transaction involves an action directive dialogue act made by the U (human-user), followed by a commitment performed by S (System); up to this point the main intention of the transaction has been expressed and its satisfaction agreed upon. Next, S performs the committed action, and asks for confirmation of whether the action was the one intended by U; finally U confirms this, and the transaction is concluded. The last two utterances, including the motor action, constitute the intention satisfaction phase of the transaction. Next we present the structure of this transaction in both of these planes, with the actual dialogue act tags in bold:

Obligations structure

- (1) Intention specification by U
 - Utt25: After that <sil> can you put <sil> the the air extractor on top of the <sil> of the stove (1st charge on S: **action-directive**)
- (2) Intention interpretation by S
 - Utt26: Okay (2nd charge on S: **commit**)
- (3) Intention satisfaction by S
 - Motor action on design space (credits 1st and 2nd charges on S: **move-object**)
 - Utt27: Is this okay? (1st charge on U: **inf-request**)
- (4) Action interpretation by U
 - Multimodal interpretation by a visual act
 - Utt28: Yes, it's okay (credits 1st charge on U: **response 27**)

The expression of an intention in (1) by U creates an obligation charge on S: to answer if the intention is an information request, or to perform an action if the intention is an action directive, as it is case in the present example; however, before U's intention can be satisfied it must be interpreted by S and this process may be complex and involve several utterances and turns, as very often happens in our corpus; when this has been accomplished, S has either to commit to satisfy the intention, or to reject it explicitly. The commitment creates an obligation charge on the speaker himself. The performance of the requested act in the intention satisfaction phase credits both U's requests and S's commitment, balancing the transaction up to this point. However, in

our multimodal setting involving a design space, in which intentions can be underspecified and references are vague, S's actions need to be accepted by U and very commonly S asks for confirmation explicitly, creating an obligation charge on U. The transaction is concluded when U interprets S's action and verifies that it was indeed what was expected; as the design actions are performed on the design space, this interpretation act is often visual. Finally, the transaction is concluded when U confirms that the intention was understood and satisfied correctly, crediting S's information request for confirmation. Next we consider the common ground structure for the transaction:

Common ground structure

- (1) Intention specification by U
 - Utt25: After that <sil> can you put <sil> the the air extractor on top of the <sil> of the stove (1st charge on S: **action-directive**)
- (2) Intention interpretation by S
 - Utt26: Okay (credits 1st charge on S: **accept 25**)
- (3) Intention satisfaction by S
 - Motor action (1st charge on U: **affirm**)
 - Utt27: Is this okay? (2nd charge on U: **inf-request**)
- (4) Action interpretation by U
 - Multimodal interpretation (visual act by U)
 - Utt28: Yes, it's okay (credits 1st and 2nd charge on U: **accept 27, affirm**)

The first difference between the obligations and the common ground planes surfaces in the interpretation of the intention in (2): while to commit to perform an action creates an obligation, the same elocution accepts the action directive that requested the action in the first place, and this elocution makes a credit at the common ground plane. The second difference appears in (3) when S performs the action, either linguistic or motor, that satisfies U's request. Through this act S provides a new piece of information to U that, although from the obligations point of view satisfies the action directive, this is also an affirm dialog act, and the new information must be accepted or rejected by U, and the confirmation that this knowledge is shared by both U and S belongs to the common ground plane of expression. If in addition to the motor action there is an explicit linguistic request for confirmation, such question should also be accepted. Finally, while the answer to the confirmation question takes place at the obligations plane, this answer is also an affirm by U that restores the common ground, letting know to S that U shares the same beliefs with S about the satisfaction of the intention.

The structure of the transaction shows that while some forward looking functions that influence the future actions of the conversational participants belong to the obligations plane (e.g. information requests, action directives, offers and commitments), other forward looking functions belong to the common ground plane (e.g. affirm and re-affirm); there are also forward looking functions that do not create an obligation either on the speaker or on the hearer, like the open-option. Similarly, although most backward functions belong to the common ground plane (e.g. all agreement acts), and also the understanding acts at the communication level, other backward functions, like responding, belong to the obligations plane. Also, in the case an intention is rejected, U and S have a conflict about the knowledge and presuppositions shared by them along the task, and a reject dialogue act belongs also to the common ground plane. Charges and credits in both planes of expression are made through the surface

utterances, and the same utterance may have one or more functions on both obligations and the common ground plane; for instance, an *okay* that functions as a commit in the obligations plane is an accept in the common ground.

3 Balancing Transactions

There are constraints between the kinds of acts that can participate in a charge/credit relation. These constraints can be stated as rules relating dialogue act tags that should be obeyed in balanced transactions. Next we illustrate the rules used in the DIME-DAMSL tagging scheme; in this specification we distinguish whether the charge has to be credited by the other conversational participant, or by the one who makes the charge on him or herself. Tables 2 and 3 summarize these relations for the obligations and common ground planes respectively.

Table 2. Balancing relations for the obligations plane

Charge	Time	Credit	On participant
Inf-request	I	Response	Other
Action-directive	I	Action	Other
Commit	I	Action	Same
Offer	P	Action	Same

Where:

- **Action** = {point-object | point-zone | point-path | point-coordinated-objects | place-new-object | move-object | remove-object | graph-plan | visual-interpretation}

In the scheme it is also considered whether a charge is made at the time the dialogue act is performed, or whether a dialogue act opens a conversational context in which a charge will be made, although the act itself does not make the charge directly; for instance, an action directive establishes an obligation at the time the act is made, but an offer does not create an obligation until the offer is accepted by the interlocutor; furthermore, if the offer is declined there is no charge at all. This property is specified for the each dialogue act type in the *Time* column of Table 2, where *I* stands for “immediately” and *P* for “postponed”.

Table 3. Balancing relations for common ground plane

Charge	Credit	On participant
Inf-request	Agr-action + Affirm	Other
Action-directive	Agr-action	Other
Offer	Agr-action	Other
Open-option	Agr-action	Other
Affirm	Agr-action	Other
Reaffirm	Agr-action	Other
Previous dialogue act	Understanding-Act	Other
Not-understanding-Signal (NUS)	Next utterance attending such signal	Other

The balancing relations of common ground dialogue acts are illustrated in Table 3. Most dialogue acts related to the agreement dimensions are expected to be “grounded” immediately by the hearer; however, dialogue acts related to the communication dimension signal that the common ground has been lost due to problems in the communication channel or because more information is required (e.g. when spatial referents are vague), and that it needs to be reestablished immediately.

Where:

- **Agr-action** = {accept | accept-part | hold | reject | reject-part}
- **Understanding-act** = {acknowledgment | back-channel | repetition | complementation | correction}

There are dialogue acts that do not make a charge or a credit in the common ground and only mark that the agreement act on a previous act has to be postponed. This is the case for the *hold* and *maybe* dialogue acts through which the speaker signals that the he or she is not sure or has reasons to believe that a previous act performed by the interlocutor has not been understood in the intended way and these acts open a context in which the common ground has to be reestablished before the conversation can proceed, or signal that more information is required to continue with the dialogue.

Tables 2 and 3 illustrate that the main dialogue acts follow simple rules at the obligations and the common ground planes: an information request is credited with a response at the obligations plane, and the interlocutor must adopt an agreement position towards such question, and in case it is accepted, the response is also an affirm act in the common ground plane. Action directives must be credited with an action in the obligations plane and an accept act in the common ground. The obligation acts that postponed its charge present a somehow more complex behavior; for instance, an offer charges the speaker, but only when the hearer has accepted the offer.

There are also dialogue acts that have an import on the common ground plane only; for instance, an open option creates no obligation but must be acknowledged either explicitly or implicitly by the flow of the conversation; also, affirming or reaffirming something with the purpose of introducing or highlighting new information does not impose an obligation on the other conversational partner, who only has to take notice and let know that he has done so to the information provider; making an unsolicited statement, or presenting new information through the visual modality, are also affirm acts that make a contribution to the common ground, and must be credited at this plane only.

The acts that postpone agreement have a more subtle behavior. A confirmation question has the purpose to support a belief that the agent that makes such a question has already; this act is an information request at the obligations plane, but it is also a hold act in the common ground; for this reason a confirmation question marks that a previous dialogue act is not accepted nor rejected immediately, but put on hold. The hold act opens a conversational segment or context whose purpose is to reestablish the common ground, and the dialogue act that closes this context accepts or rejects the postponed act.

A dialog act tagged as *maybe* is even more subtle. It arises in check questions, where the intent is, for instance, to confirm whether an agent really wants to do something that is not entirely justified for the task at hand. In a situation where U asks S to

do an action that does not seem to be relevant, or it is too expensive to accomplish or too restrictive in the utterance's context, and this is realized by S, but the evidence is not strong enough to reject U's request directly, a maybe dialogue act may arise. If at a point in the dialogue S says: *Okay, do you want me to move the stove to the left?* where *Okay* is pronounced with a dubitative tone, it may be a *maybe* dialogue act. The maybe act opens a context that postpones the acceptance or rejection of a previous act too. Also the dialogue act that closes the maybe context accepts or rejects the act that was questioned by the maybe act, and credits the corresponding charge.

Understanding acts make contributions to the common ground directly, and these are representative contributions to this plane of expression, as their only role is to strengthen the common ground or to restore it when it is lost; acknowledgments, back-channels, repetitions, complementations and corrections strengthen or repair the common ground and permit the flow of the conversation.

In the formulation of these rules, there is not a claim that there are necessary and sufficient conditions to classify an utterance as an specific dialogue act; rather, there may be several conditions that can be considered sufficient, and nevertheless not necessary for marking an act as a member of a given category, and the strength of one condition or combination of conditions may determine whether the act belongs or not

Table 4. Analysis of a transaction

U	T	Utterance	Obligations		Common ground				Dialogue Act Types	
					AGR		UND		Obligations	Common ground
			Ch	Cdt	Ch	Cdt	Ch	Cdt		
1	s	Do you want me to bring a piece of furniture to the kitchen?			1				offer	
2	u	Yes	1			1				accept
3		I need a stove	3		3				action-directive	
4	s	A second				3				accept
		These are the five models of stoves that we have, simple stoves and stoves with lateral cupboards			5				open-option	
6	u	mmmm <sil> I'm going to select that stove			6	5				affirm, accept
7	s	Okay				6				accept
	u	eh, please I need it in <sil> in the far wall			8					affirm
9	s	Which one is the far wall?	9						inf-req	hold,
10	u	Let's see, here		9	10		11		answer	affirm
11	s	There?	11					11	inf-req	hold, repetition
12	u	Yes		11					answer	
13	s	A second	13			10	8		commit	accept
14		Is there alright? <graphical action performed with the question>	14	13 3 1	14				graph-action inf-req	
15	u	Yes, for the moment, yes		14		14			answer	accept

to a given category. In particular, the intonation used in the production of a dialogue act is one of the main properties of dialogue act types. For these reasons we cannot expect complete categorical answers for tagging questions, and the theory should be supported by the definition of explicit tagging conventions, and also on empirical evidence, and the agreement between taggers. We conclude this section with the analysis of typical transaction of our corpus.

In this transaction the main intention is specified from utterance (1) to (13) and the satisfaction is produced by an immediate graphical action in (14) and a final confirmation question, also in (14), and its answer in (15). In Table 4 the entries in the charges and credits cells index the corresponding dialogue act. The charge made by the offer that the system imposes on itself in (1) is postponed until the offer is accepted by the user in (2), and this obligation act is credited in (14) when the act offered to is finally performed. The main intention is the action directive stated in (3) by the user, with the corresponding system's commit in (13), and both of these acts are credited when the action is performed too.

The structure of the common ground is a bit more subtle; the agreement level includes two hold acts, with the additional complication that the second is embedded within the first. The first hold in (9) postpones the acceptance of (8), and the second in (11) the acceptance of (10); when the reference of *the far wall* has been resolved through (12), the system can credit the charges of the two affirm acts performed by the user in (8) and (10) by accepting them in (13). The check question in (11) has a component in the understanding level too; the word *there* in (11) is considered a repetition of *here* in (10). This latter pattern is a common phenomenon observed in the corpus; the use of spatial language introduces vague references very often, and this causes the common ground to be broken down at the level of agreement, as some references cannot be resolved directly, and also at the communication level, as these charges depend on the vague nature of spatial references. The common ground is restored with a deictic act that fixes the spatial reference, crediting the agreement charge, and resolves the vagueness, crediting the communication charge.

4 The Tagging Methodology

The tagging methodology considers, in addition to a taxonomy of dialogue act types and the specification on the structural relations that constraint the realization of these acts, a well-defined set of tagging conventions. These conventions refer both to the tags associated to the content of dialogue acts, and also to the tagging format.

In relation to the conventions about content, some dialogue acts have a positive character and mark a behavior explicitly (e.g. most acts that charge the obligations plane) while others have a negative character, and are only marked when the linguistic behavior deviates from the expected one. Accordingly, a tagging convention used in the present methodology is that only dialogue acts that mark an intention have an explicit tag; also, if an utterance expresses several dialogue act types, only the most prominent dialogue act in relation of the utterance main intention is explicitly tagged. For instance, the utterance *these are the stoves* made at the time a catalogue is displayed is only marked as an open option, despite that this statement can also be considered an affirm dialogue act; the convention is that the main intention of the speaker

is to show the hearer what stoves are available in order that the hearer is able to select one for the current stage in the design task; in this situation, the hearer should know already that there are stoves available, as he or she is engaged in a kitchen design task, and for this reason the dialogue act is not an affirm, in spite that this utterance is making an statement and has a declarative form.

Conventions at this level of content determine and complement the theory of dialogue acts from a practical perspective, and our tagging exercise shows that this kind of specifications is indispensable for the speech act analysis of practical dialogues. Similarly, all utterances have to be acknowledged in the communication level, but dialogue acts at this level are marked only when the communication flow breaks down, or an explicit feedback is needed. We take the convention that if S makes a dialogue act at the communication level, it credits the previous message uttered by U, which is marked as a communication charge. However, a not-understanding signal is an explicit communication charge; in this case, the hearer is obliged to credit the speaker's signal: If U says something and S says *sorry, what did you say?* U is obliged to repeat or rephrase his last utterance in order to recover the common ground.

We have also observed that another source of confusion and low agreement is due to the tagging format and the tagging tools, and the way these are understood and used by taggers. The tagging format is a dynamic object that should be defined and refined by the tagger team during the training phase of the tagging exercise, and the productive phase should start only when taggers have mastered not only the theory and the conventions about both the interpretation of dialogue acts, but also the use of the tagging format and tools. The format should also consider an easy calculation of tagging agreement.

Theoretical guidelines, the definition of tagging conventions and the actual empirical work are three aspects of the tagging task that interact and evolve together during the tagging cycle. The present theory and its associated tagging rules and conventions were developed in conjunction with the tagging task, and we considered two phases for the work and a specific methodology for carrying on with the exercise. The first phase is a training one in which we had to make sure that the theory and tagging conventions were mature enough, that taggers were familiar enough with the theory, the conventions and the annotation scheme, and that they had enough tagging experience to carry out the task successfully. The second phase is a production one in which tagging tasks are assigned to individual taggers or taggers teams.

In terms of the methodology, we divide the tagging tasks on three levels. First, we identify the transactions in a dialogue; here, we rely on two main guidelines: transactions have a main intention, with its corresponding specification and satisfaction stages, and transactions are balanced at both planes of expression. Once transaction boundaries have been marked by several teams, we compute agreement factors through the kappa statistic [4] at this level; the kappa statistic is a measure of how well annotators agreement is better than random annotation, and figures above 0.8 are usually considered very good. If these values are not high enough, specific discrepancies are identified and discussed by the tagging team, and the task is repeated until the kappa statistic is satisfactory. The methodology involves refining and identifying new tagging conventions in every tagging round.

The second step consists of marking charges and credits of both the obligations and common ground planes, and balance each individual transaction. Here again, we

compute the kappa statistics that measure the agreement with regard to the balancing structure of the transactions. When the kappa values are acceptable, we proceed to the third stage in which the actual tags of the DIME-DAMSL scheme are transcribed. The tagging task is supported by an excel format that allows to input the tags for transactions boundaries, the obligations and common ground structures and the actual DIME-DAMSL tags directly, and the computation of kappa statistics for transaction boundaries is supported by the same excel format.

For the validation and training phase we selected two dialogues with 117 and 137 utterances from our corpus; these dialogues have been studied in several dimensions including their prosodic structure, and were tagged in preliminary experiments performed by a team of 15 people. In the training phase several tagging cycles were required to reach an acceptable level of agreement for the identification of the transaction boundaries and also for balancing the transactions. In the formal experiment, the first dialogue was tagged by 3 teams of three people each and the second by three experimented taggers. The overall figures for this process are shown in Table 5. In this table the kappa statistics for the agreement acts and understanding acts of the common ground plane are shown explicitly. The global figures for the four main dimensions of the tagging scheme are shown in Table 6. As can be seen, our current figures are very promising.

Table 5. Kappa values for transactions balance

Tagging cycle	Trans. Bounds	Charge/credits Structure		
		Obligations Plane Chrs/Cdts	Common Ground Plane	
			AGR Chrs/Cdts	UND Chrs/Cdts
kappas	0.87	0.94/0.92	0.83/0.85	0.82/0.83

Table 6. Kappa values for the tagging exercise

Dimension	Tagging categories	Kappa
Information level	Task, Task-managment, Communication-management	0.83
Forward looking functions	Declarative: Affirm, Reaffirm, Other	0.87
	Information request	0.93
	Influence future actions of listener: Action-directive and Open-option	0.89
	Influence future actions of speaker: Offer and commit	0.89
Backward looking functions	agreement	0.82
	understanding	0.89
	response	0.94
Modality	Graphical actions	0.80

5 Conclusion

A theory of dialogue acts involves not only the specification of a set of dialogue act types, but also the way these are related systematically in relation to the transaction's

structure. The present theory provides an ontology of dialogue act types for multimodal practical dialogues, and several structural constraints for the realization of specific dialogue acts. The main kind of constraint is defined in terms of the obligations and common ground planes of expression, that have to be balanced in complete transactions; other structural constraints impose contextual restrictions on the realization of dialogue acts. The theory contemplates, in addition, the definition of tagging conventions related to the content of the dialogue acts, and also to the tagging format. The tagging task involves the creation of a tagging team that evolves and matures throughout the tagging exercise, and the whole of the process must be developed in the context of a well-defined methodology; currently, this methodology is being applied to the transcription of the DIME Corpus, and we hope to have a large number of dialogues fully transcribed in the near future. A good value of the kappa statistic for this larger exercise would provide a strong support for our theory.

There are also interesting implications of the present theory and methodology for a more comprehensive theory of dialogue structure, reference resolution and vague reference in spatial discourse; we have also observed that discourse markers are important indicators for the main transaction boundaries, and also that the scope of referential terms is local to the transaction context. We have also noticed that spatial language introduces vague references very often and that this kind of reference disrupts the common ground, and prompts a conversation segment involving check questions and confirmations that is closed with a spatial deictic reference that resolves vagueness, reference, and reestablishes the common ground all at once. We let the investigation of these issues for further research.

Acknowledgments

We thank useful comments and suggestions from James Allen, at Rochester University, Joaquín Llisterrí and Monserrat Riera at Universidad Autónoma de Barcelona, and Javier Cuétara at Facultad de Filosofía y Letras, UNAM. We also thank Cesar Gamboa for technical support at IIMAS, UNAM. We also acknowledge useful comments by the anonymous reviewers of this paper. The theory and experiment reported in this paper were developed within the context of the DIME-II project, with partial support of NSF/CONACyT grant 39380-A.

References

1. Allen, J.F., Byron, D.K., Dzokovska, M., Ferguson, G., Galescu, L., Stent, A.: Toward Conversational Human-Computer Interaction. *AI Magazine*, 22(4):27–38, Winter, 2001.
2. Allen, J.F., Byron, D.K., Dzokovska, M., Ferguson, G., Galescu, L., Stent, A.: An Architecture for a generic dialogue shell. *Natural Language Engineering*, 6(3–4):213–228, 2000.
3. Allen, J.F., Core, M.G.: Draft of DAMSL: Dialog Act Markup in Several Layers Annotation Scheme. Department of Computer Science, Rochester University, October, 1997.
4. Carletta, J.: Assessing agreement on classification tasks: the kappa statistic. *Computational Linguistics*, 22(2):249–254, 1996.
5. Clark, H.H., Schaefer, E.F.: Contributing to Discourse. *Cognitive Science*, 13:259–294, 1989.

6. Core, M.G., Allen, J.F.: Coding Dialogs with the DAMSL Annotation Scheme. Department of Computer Science, Rochester University, 1997
7. Grosz, B.J., Sidner, C.L.: Attention, Intentions and the Structure of Discourse. *Computational Linguistics*, 12(3):175–204, 1986.
8. Luis A. Pineda, A. Massé, I. Meza, M. Salas, E. Schwarz, E. Uruga, L. Villaseñor. The Dime project. Proceedings of MICAI2002, Lectures Notes in Artificial Intelligence, Vol. 2313, Springer-Verlag, pp. 166–175, 2002.
9. David R. Traum and James F. Allen. Discourse Obligations in Dialogue Processing. In Proceedings of Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics (ACL-94), pages 1–8, June 1994.
10. Luis Villaseñor, Antonio Massé y Luis Pineda. The DIME Corpus. In Proceedings Corpus, in 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, September, 2001.

Finite State Grammar Transduction from Distributed Collected Knowledge

Rakesh Gupta¹ and Ken Hennacy²

¹ Honda Research Institute USA, Inc.,
800 California Street, Suite 300, Mountain View, CA 94041
rgupta@hira.com

² University of Maryland, Institute for Advanced Computer Studies,
College Park, MD 20742
khennacy@cs.umd.edu

Abstract. In this paper, we discuss the use of Open Mind Indoor Common Sense (OMICS) project for the purpose of speech recognition of user requests. As part of OMICS data collection, we asked users to enter different ways of asking a robot to perform specific tasks. This paraphrasing data is processed using Natural Language techniques and lexical resources like WordNet to generate a Finite State Grammar Transducer (FSGT). This transducer captures the variations in user requests and captures their structure.

We compare the task recognition performance of this FSGT model with an n-gram Statistical Language Model (SLM). The SLM model is trained with the same data that was used to generate the FSGT. The FSGT model and SLM are combined in a two-pass system to optimize full and partial recognition for both in-grammar and out-of-grammar user requests. Our work validates the use of a web based knowledge capture system to harvest phrases to build grammar models. Work was performed using Nuance Speech Recognition system.

1 Introduction

Humans often wish to communicate with robots about what they like done. It is awkward to be constrained to specific set of commands. Therefore, a free-form interface that supports natural human robot interaction is desirable.

A finite state transducer is a finite automaton whose state transitions are labeled with both input and output labels. A path through the transducer encodes a mapping from an input symbol sequence to an output symbol sequence [1]. Grammar is a structure that defines a set of phrases that a person is expected to say. In this work, our goal is to automate the process of creating a Finite State Grammar Transducer (FSGT) to map utterances to task labels from text data contributed by volunteers over the web.

It is a challenge to develop a grammar that will recognize a large variety of phrases and achieve a high recognition accuracy. Manual creation of a set of

grammar rules can be very tedious. In many cases, the out-of-grammar¹ rate obtained with hand crafted grammar rules is high because of poor coverage. In our system, multiple users on the web contribute knowledge. We are therefore likely to have better coverage than by one person exhaustively thinking of ways to ask the robot to perform a particular task.

In contrast to grammar, a Statistical Language Model (SLM) assigns probabilities to a sequence of words. SLM grammars are appropriate for recognizing free-style speech, especially when the out-of-grammar rate is high. They are trained from a set of examples, which are used to estimate the probabilities of combinations of words. Training sets for SLM grammars are often collected from users as they interact with the particular application. Over time, the SLM grammar is refined to recognize the statistically significant phrases.

In this paper we first describe OpenMind data collection and the paraphrasing data that we use in this work. We then discuss the use of Open Mind data to simultaneously generate the FSGT and train the SLM model. Using both techniques in a two pass system leverages on their advantages. This is followed by experiments and results where we compare the FSGT with SLM for both in-grammar and out-of-grammar user requests. We then describe how these two models may best be combined to optimize performance in a system with both in-grammar and out-of-grammar user requests. We then finish our discussion with sections on Conclusions and Future Work.

2 OpenMind Distributed Knowledge

Commonsense may be gathered from non-specialist netizens in the same fashion as the projects associated with the rather successful *OpenMind Initiative* [2, 3]. OpenMind Indoor Common Sense (OMICS) is an internet-based distributed knowledge capture framework that is used to capture knowledge from web users into a relational database schema [4]. This knowledge is acquired from user responses to questions in a fill-in-the-blank format. The purpose of the OMICS project is to capture knowledge about indoor home and office environments into a form useful for reasoning about common household tasks. The OMICS project has successfully captured thousands of text phrases of commonsense knowledge about home and office environments. A distinguishing characteristic of this approach is in the restriction of the domain to indoor home and office environments. This restriction makes the knowledge base dense for statistical processing.

Paraphrases are alternative ways of conveying the same information. In this work we have used the **Paraphrase** table entries from the OMICS database. Figure 1 shows an OpenMind prompt for the paraphrase activity with a possible answer. Paraphrase prompts ask questions such as: To ask someone to *make coffee*, one might say _____.

¹ We mean out-of-grammar with respect to the generated FSGT model. This has no relation to ungrammatical standard English.

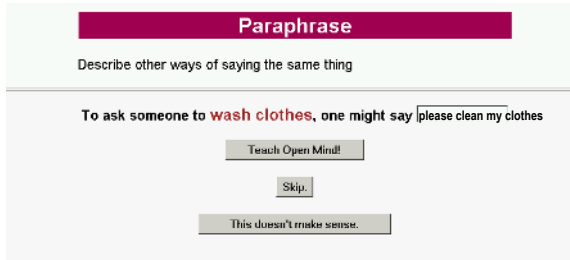


Fig. 1. OpenMind prompt for paraphrase entry showing a possible answer

Data collected from users in a distributed manner is inherently noisy. It contains spelling errors as well as completely out-of-context responses. More commonly, user data often relies upon implied associations, i.e. the commonsense understanding of what is being referred to within a sentence. An example of an implied association is: To ask someone to *cook pasta*, one might say *I am hungry*. Such a paraphrase is ambiguous as it could imply a wide range of tasks including heating food, getting food from a refrigerator, and cooking vegetables. These implied associations and other bad data are removed in OMICS by manual review which takes place before any knowledge is committed to the database.

3 Related Work

Commercial speech recognition systems rely on keyword extraction to recognize spoken utterances. Recent work by Punyakanok et. al. [5], Steedman et. al. [6], and Bangalore et. al. [7] uses semantic information to extract associations between the utterance and the knowledge representation. Such techniques can be used for both speech understanding and speech generation.

Using transcribed parent child speech, Solan et. al. [8] perform unsupervised learning of linguistic structures from the corpus. They extracted significant patterns of words and represented them in trees to generalize to variations in unseen text. Klein et. al. [9, 10] presented a generative probabilistic model for unsupervised learning of natural language syntactic structure. They do not learn a CFG but induce a distributional model based on constituent identity and linear context.

For grammar generation, Sinha et. al. [11] collected structured transcripts from Wizard of Oz based user tests. Participants spoke instructions and a wizard (real person through Microsoft NetMeeting tool) captured the spoken phrase in a transcript and performed the requested email management task. With simple text parsing, they generated the Context Free Grammar (CFG). This approach required labor intensive transcript collection and the scope was limited to only a few tasks. Martin [12] implemented an automated customer service agent to interact with users wanting to browse and order items from an online catalog. A small representative set of utterances (on the order of hundreds of sentences or phrases) was combined with an overly permissive grammar to generate a tighter

grammar for the domain. This approach required tagging of lexical entries and manual writing of rules to enforce semantic restriction among lexical entries. Among free-form recognition approaches using a grammar, Riccardi et. al. [13] built a system based on recognizing commonly used phrases instead of words to categorize people's responses to an open ended prompt *How may I help you?* They evaluated and selected phrases via perplexity minimization and clustered them using a similarity metric.

There has also been work in the area of semantic understanding of user utterances. SRI GEMINI project [14] implemented a natural language understanding system in the domain of air travel planning. Hundreds of formal syntactic and semantic rules were manually created to build up constituent phrases into fuller natural language utterances. They corrected recognizable grammar errors to increase the robustness of their system. Other work in combining CFG and SLM for improving recognition includes use of Expectation Maximization (EM) algorithm to estimate n-gram parameters and semantic labels [15], and modeling CFG fragments as words in a n-gram SLM [16], and use of grammar-based recognizer and SLM in a two pass system to provide feedback to users [17].

Our approach is to develop a two-pass recognition system where the FSGT model is built from the same corpus of utterances that are used to train the SLM models. Sentence structure is extracted from the recognized text and is used to determine the word relations for application of rules. There are several expected advantages to this approach. First, by utilizing the FSGT model, the SLM models do not have to be tagged for natural language interpretation. This simplifies the use of SLM models. Second, by setting a high acceptance threshold for the FSGT model, the system has an easier time identifying in-grammar and out-of-grammar utterances. Finally, the FSGT can be used for generating multiple ways of stating the user requested task for speech generation.

4 FSGT Grammar Generation

Figure 2 shows steps in the grammar generation process. We first connect to the OpenMind Indoor Common Sense MySQL database and read the paraphrasing data. We then run the data through a spelling checker to correct the spellings using WinterTree spelling checker software. Next the data is processed by a Part of Speech (POS) tagger to identify core and filler words in each phrase. We then categorize the sentence structures and construct FSGT models using these structures. Appropriate synonyms from WordNet and Thesaurus are then identified. Phrases that contain matches to synonyms are used to generate a hierarchical grammar consisting of filler and core parts. These steps are described in more detail in subsequent subsections.

4.1 Part of Speech Tagging

Standard techniques for extracting grammars from different corpora of text involves the use of parsers to generate the part of speech tags [18]. In our work, we use JMontyLingua [19] which is based on Brill Tagger [20], and developed by

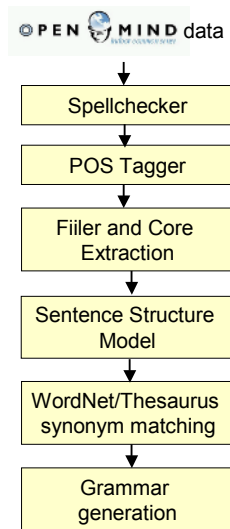


Fig. 2. Overview of automatic grammar generation

Hugo Liu at MIT with OpenMind commonsense enhancements. An example of a parsed user request is:

make/VB coffee/NN.

Here VB is the Penn Treebank notation for verb and NN is the notation for a noun. A grammar typically consists of the core portion that contains the most important information bearing words (the requested task) and a filler portion that contains additional expressions such as *i'd like you to* or *please*. We extract filler and core words using simple rules. In our system, filler words are combined across all tasks that the robot can perform, but core words are processed separately for each task.

We use a hierarchical grammar to combine filler and core words within the FSGT. The filler portion is reusable across all tasks, and is defined using a high-level grammar. Core words that pertain to the task are added in a sub grammar. This sub grammar is embedded in the high-level task grammar. Another example of a parsed user request is:

can/MD you/PRP prepare/VB coffee/NN.

In this example MD refers to a modal verb, PRP refers to a preposition. Here *can you* is the filler and *prepare coffee* is the core. For the short user request phrases that we are interested in for robot commands, we have empirically found the Brill tagging to be correct 85% of the time.

4.2 Modeling Sentence Structure

User requests can be broadly classified as imperative statements. Sentences with imperative structure often begin with a verb phrase and have no subject. They

are called imperative because they are almost always used for commands and suggestions.

We require that the phrase have at least one verb and one noun. We model the adverb modifiers in the command (*e.g. turn on television*) but we do not model the prepositional phrase (PP). For example *in the sink* in the command *put the dishes in the sink* is not modeled. The prepositional phrase provides additional information which is essential for correct command execution, but is not necessary for command recognition. We use rules to match among different sentence structures. For example, phrases with the structure:

VP \rightarrow Verb Adverb NP and VP \rightarrow Verb NP Adverb

are equivalent if the words with the same parts of speech are synonyms (or same) in the two candidate phrases. Example user requests that fits the above grammar are *pick up object* and *pick object up*.

These rules are themselves derived from the data by a frequency analysis of the POS trigrams. Since this processing is data driven, it can be easily automated.

4.3 Matching Synonyms

According to Deerwester et. al. [21] two people typically choose the same name for a single well-known object less than 20% of the time. Hence it is necessary to use synonyms to make the system vocabulary independent. We use lexical resources like WordNet and Thesaurus to provide synonyms. For sentence structure S \rightarrow VB NN, an example user request is:

make/VB coffee/NN

The verb in the request is *make* and noun is *coffee*. From all paraphrases we select the ones where both noun and the verb match the WordNet/Thesaurus synonym list.

For noun phrases, there are different ways of matching synonyms. Currently we accept general to specific matching between a noun in the NP (Noun Phrase) and the WordNet and thesaurus synonym list. For example, *coffee* matches *black coffee* and *cup of coffee*.

4.4 Grammar Generation

A sample of paraphrases for *make coffee* user request in the OMICS database are as follows:

1. please/VB make/VB coffee/NN
2. make/VB java/NN
3. prepare/VB coffee/NN
4. fix/VB me/PRP a/DT cup/NN of/IN coffee/NN
5. prepare/VB a/DT coffee/NN
6. need/NN coffee/NN
7. make/VB some/DT joe/NN
8. brew/VB coffee/NN

9. bring/VB coffee/NN

10. I/PRP 'd/MD like/RB to/TO have/VB coffee/NN

Prepare and fix are synonyms of make in WordNet. Similarly coffee and java are synonyms. So the first five phrases are matched and used to generate the FSGT. In the sixth phrase, need is incorrectly tagged as a noun, leaving no verb in the phrase. Joe is not a synonym of coffee. Brew, bring and have are not synonyms of make. Hence the latter five phrases are not captured in the current FSGT. The corresponding grammar rule that is generated is:

(?FILLER [fix prepare make] ?a ?(me a) [java coffee (cup of coffee)]) return "make coffee"

FILLER is the list of filler words found by a statistical analysis of all filler words found in the paraphrasing entries. It consists of phrases like *please, can you* etc. [] represents an OR clause while () represents an AND clause. ? indicates optional words that may or may not be present in the utterance. The slot value "make coffee" returned from the grammar indicates the user's requested task.

The code for FSGT generation is implemented in Java. It takes about 5 minutes to generate the grammar file by processing 2724 paraphrase entries in the current OMICS database. About 80% of this time is spent in diagnostic messages. Our system consists of 87 typical tasks that may be requested to a robot helper in indoor home and office environments. Of 2724 paraphrases collected so far for 172 tasks, 636 are used in generating the grammar for our tasks. This fraction of paraphrases used can be improved by using data in remaining tasks and allowing synonyms that do not exist in WordNet/Thesaurus (e.g. brew in the above grammar).

5 Statistical Language Model Construction

The other grammar model that is trained by the OMICS paraphrase statements is a probabilistic finite state grammar (PFSG) referred to as a n-gram Statistical Language Model (SLM). Our 3-gram SLM is trained directly from the corpus of paraphrasing data contained in the OMICS database. The SLM model captures the frequency of different words in the grammar.

6 Two-Pass Method for Speech Recognition

In our work, SLM recognized text is first processed using the FSGT to find the requested task via slot assignments. If the utterance is not recognized, a POS speech tagger is used for out-of-grammar recognition of keywords. With background noise and variations in voice and grammar, FSGT and SLM grammars individually may not match a spoken utterance. However the combination of the two approaches in a two pass system works well. There are several reasons for this.

The statistical n-gram model is expected to perform better than the FSGT model. While the FSGT model is very strict in terms of the possible combinations

of words that can be recognized, the SLM model is not as restrictive. With certain speakers, FSGT has difficulty because it is too restrictive. On the other hand, the statistical n-gram model has difficulty because too many phrases that sound similar to the utterance get chosen. Upon combining the two models, however, statistical n-gram model picks candidate phrases and FSGT discards the nonsensical and non grammatical phrases leading to a correct match.

The steps performed for the two-pass approach are as follows:

- 1: Recognition with FSGT is attempted. If a FSGT is matched, the corresponding task assignments and recognized text is returned.
- 2: If a FSGT does not match a spoken utterance exactly, then a SLM model is used in the second pass to generate and rank possible matching word phrases. The threshold for recognition is set low to include the capture of out-of-grammar utterances.
- 3: The ranked extracted word phrases are sent through the text recognition portion of the FSGT. The highest ranked match is selected.
- 4: If a match for all words is not found, the closest match is found based upon several criteria involving keywords (verbs and nouns), their confidence numbers, and their expression within the most probable phrase.

For the first pass, we set the threshold for the FSGT model to be high (60%). For 85% of the phrases, the correct task request is captured and the slot assignments have a high accuracy rate when recognition occurs. If no slots are returned, the recognition software looks at the N highest ranked text phrases returned by the SLM model. For this second pass of the speech recognition process, the confidence threshold is set low (20%) to generate numerous hypotheses. These low threshold hypotheses rarely match the FSGT, but are necessary for matching out-of-grammar speech.

The third step gives us a 96% recognition rate compared to 85% with FSGT alone (the 3.4% errors are in tagging). This step is critical in boosting the recognition performance of the SLM grammar for in-grammar phrases. High performance of the two-pass approach is due to the acceptance criteria introduced by passing candidate phrases from the SLM hypothesis generator through the FSGT model to eliminate nonsensical statements.

7 Experiments

In the OMICS database we started with 2724 paraphrase activity entries for 172 tasks. Of these, 636 entries corresponding to 87 tasks were used in generating the grammar, and rest were considered out-of-grammar. For the in-grammar test, we used 15 utterances from each of 20 subjects, leading to a total of 300 utterances. Subjects read the utterances from a paper. There were 18 male and 2 female subjects. These subjects spoke a random phrase out of 636 phrases that were represented in the FSGT model. For the out-of-grammar test, we used a random set of 400 phrases from the remaining 2088 phrases that were not used in generating the grammar for the same 87 tasks. These more accurately measure the situation in real world human robot interaction where the user utterances

are not guaranteed to match the utterances represented in the grammar model. For out-of-grammar studies we used 25 utterances each from 14 subjects leading to a total of 350 utterances. Only the SLM grammar model was used for the out-of-grammar analysis.

To compare performance of the FSGT and SLM models, we tested both in-grammar user requests and out-of-grammar user requests, using the commercially available Nuance Communications Speech Verifier 8.5 software [22]. Wave acoustic recordings of users were processed in a batch application.

8 Results

Table 1 shows the results for in-grammar tests. We focus on task recognition as a metric rather than word error rate since it more appropriately models the errors experienced by users in an interactive situation. The FSGT has low performance when the user utterance is partially recognized and the recognized part of the utterance matches multiple interpretations. The two-pass system is able to resolve most of these FSGT incorrect task recognitions and non-recognitions. As noted earlier, the two-pass system with N-best processing in SLM, followed by choosing the highest rank phrase recognized by the FSGT model does much better. It correctly recognizes the task in 96% of in-grammar cases.

Table 2 shows the results of out-of-grammar testing. We use noun and verb in the phrase as strong indicators of the task. In 43% of these cases we recognized

Table 1. Results for in-grammar testing

	FSGT Grammar	Statistical Language Model (SLM)	Two Pass FSGT + SLM
Correct Task Recognition	84.9%	71.8%	96%
Incorrect Task Recognition	11.7%	3.4%	3.4%
High weight Partial Recognition		12.4%	0.3%
Not Recognized	3.4%	12.4%	0.3%

Table 2. Results for out-of-grammar testing

	Statistical Language Model (SLM)
Context Match (noun and verb)	43%
Partial Context Match (only noun or verb)	28%
Not Recognized	29%

the correct verb and noun. In 28% of the cases only one of the noun and verb matched correctly. In remaining 29% of the cases, there were no noun or verb matches. The perplexity of the SLM model was 37, using a closed out-of-grammar set of 400 phrases. The baseline in-grammar SLM perplexity was 9.

We have used this two-pass system to get optimal performance from FSGT and SLM models for cases where we use a mixture of in-grammar and out-of-grammar utterances for user requests. Using these results, we can handle partial recognition in cases where the confidence is high so that the robot can ask intelligent questions (instead of reporting a non-recognition).

9 Conclusions

In this work, we implemented a FSGT generation from distributed knowledge. The same phrases used to generate the FSGT were used to train the SLM model. Our approach is scalable and can handle large number of phrases in generation of the FSGT. Our first contribution is in using a web collected knowledge base to build a FSGT model. The OpenMind distributed knowledge capture approach provides good data for constructing compact FSGT and SLM models for use by speech recognizers utilizing user-sampled acoustic models.

Our second contribution is the use of this FSGT in a two-pass grammar and statistical system to improve the recognition of in-grammar and out-of-grammar user requests. With the two-pass approach we get a 96% correct task recognition accuracy for in-grammar user requests. For out-of-grammar user requests we get a 43% correct match of both noun and verbs in the task, and correct noun or verb in 28% cases.

In future work, we want to test with more OpenMind data as it becomes available. Another issue is semi-automated synonym identification. Currently if synonyms dont exist in WordNet/Thesaurus, the paraphrase is not used in the grammar generation. But we can have the system output commonly used synonyms that were not found in WordNet/Thesaurus and have a person filter the ones that are appropriate to put in a personal version of the thesaurus. E.g. *turn* and *switch* are not synonyms but are commonly used for each other in the phrase *turn television on*.

Finally, we would like to tune weights and probabilities in the Statistical Language Model to improve performance of SLM recognition. The recognition engine can use these weights and probabilities while searching for matches in the space of allowable utterances. This tuning can be performed by making certain that the grammar structures of the FSGT and SLM are more in line with each other. We expect that with structural analysis provided by POS tagging, the statistical significance of word order can be represented in a EM framework.

Acknowledgements

This work was done while Ken Hennacy was supported by Honda Research Institute USA, Inc. The authors would like to thank Ryan Rifkin for his helpful

comments and suggestions. Thanks are also due to anonymous reviewers, volunteers of the Open Mind Indoor Common Sense website for their contributions, and Nuance Corporation for contributing software to the University of Maryland for this study.

References

1. Mohri, M., Pereira, F., Riley, M.: Weighted finite state transducers in speech recognition. *Computer Speech & Language* **16** (2002) 69–88
2. Stork, D.G.: The Open Mind Initiative. *IEEE Expert Systems and Their Applications* **14** (1999) 19–20
3. Stork, D.G.: Open data collection for training intelligent software in the open mind initiative. In: *Proceedings of the Engineering Intelligent Systems (EIS2000)*, Paisley, Scotland (2000)
4. Gupta, R., Kochenderfer, M.: Common sense data acquisition for indoor mobile robots. In: *Nineteenth National Conference on Artificial Intelligence (AAAI-04)*. (2004)
5. Punyakanok, V., Roth, D., Yih, W.: Natural language inference via dependency tree mapping: An application to question answering. *Computational Linguistics* (2005) submitted for review
6. Steedman, M.: Wide-coverage semantic representations from a ccg parser. In: *Proceedings of the 20th International Conference on Computational Linguistics*. (2004)
7. Bangalore, S., Joshi, A.K.: Supertagging: An approach to almost parsing. *Computational Linguistics* **25** (1999) 237–265
8. Solan, Z., Horn, D., Ruppin, E., Edelman, S.: Unsupervised context sensitive language acquisition from a large corpus. In: *NIPS*. (2003)
9. Klein, D., Manning, C.D.: Natural language grammar induction with a generative constituent-context model. *Pattern Recognition* **38** (2005) 1407–1419
10. Klein, D., Manning, C.D.: Distributional phrase structure induction. In: *Proceedings of the Fifth Conference on Natural Language Learning (CoNLL)*. (2001) 113–120
11. Sinha, A.K., Landay, J.A.: Towards automatic speech input grammar generation for natural language interfaces. In: *CHI 2000 Workshop on Natural Language Interfaces*, The Hague, The Netherlands (2000)
12. Martin, P.: The casual cashmere diaper bag: Constraining speech recognition using examples. In: *Proceedings of the Association of Computational Linguistics*. (1997)
13. Riccardi, G., Bangalore, S.: Automatic acquisition of phrase grammars for stochastic language modeling. In: *6th Workshop on Very Large Corpora*, Montreal (1998) 186–198
14. Dowding, J., Gawron, J.M., Appelt, D.E., Bear, J., Cherny, L., Moore, R., Moran, D.B.: GEMINI: A natural language system for spoken-language understanding. In: *Meeting of the Association for Computational Linguistics*. (1993) 54–61
15. Acero, A., Wang, Y., Wang, K.: A semantically structured language model. In: *Special Workshop in Maui (SWIM)*. (2004)
16. Wang, Y., Mahajan, M., Huang, X.: A unified context-free grammar and n-gram model for spoken language processing. In: *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, Istanbul, Turkey (2000)

17. Hockey, B.A., Lemon, O., Campana, E., Hiatt, L., Aist, G., Hieronymus, J., Gruenstein, A., Dowding, J.: Targeted help for spoken dialogue systems: intelligent feedback improves naive users' performance. In: Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics. Volume 1. (2003) 147–154
18. Klein, D., Manning, C.D.: Parsing with treebank grammars: Empirical bounds, theoretical models, and the structure of the penn treebank. In: Proceedings of the 39th Annual Meeting of the ACL. (2001)
19. Liu, H.: Montylingua: An end-to-end natural language processor with common sense. Available at: web.media.mit.edu/~hugo/montylingua (2004)
20. Brill, E.: A simple rule-based part-of-speech tagger. In: Proceedings of ANLP-92, 3rd Conference on Applied Natural Language Processing, Trento, IT (1992) 152–155
21. Deerwester, S.C., Dumais, S.T., Landauer, T.K., Furnas, G.W., Harshman, R.A.: Indexing by latent semantic analysis. *Journal of the American Society of Information Science* **41** (1990) 391–407
22. Nuance: Nuance Speech Recognition System 8.5: Grammar Developer's Guide. Nuance Corporation, USA (2004)

Predicting Dialogue Acts from Prosodic Information

Sergio Rafael Coria Olguin and Luis Albreto Pineda Cortés

Institute for Applied Mathematics and Systems (IIMAS),
Universidad Nacional Autónoma de México (UNAM),
Circuito Escolar S/N, Ciudad Universitaria, Del. Coyoacán, 04510 México, D.F. México
coria@turing.iimas.unam.mx, luis@leibniz.iimas.unam.mx

Abstract. In this paper, the influence of intonation to recognize dialogue acts from speech is assessed. Assessment is based on an empirical approach: manually tagged data from a spoken-dialogue and video corpus are used in a CART-style machine learning algorithm to produce a predictive model. Our approach involves two general stages: the tagging task, and the development of machine learning experiments. In the first stage, human annotators produce dialogue act taggings using a formal methodology, obtaining a highly enough tagging agreement, measured with Kappa statistics. In the second stage, tagging data are used to generate decision trees. Preliminary results show that intonation information is useful to recognize sentence mood, and sentence mood and utterance duration data contribute to recognize dialogue act. Precision, recall and Kappa values of the predictive model are promising. Our model can contribute to improve automatic speech recognition or dialogue management systems.

1 Introduction

A dialogue act tag characterizes the type of intention which a speaker intends to express in an utterance. A listener has to analyze the utterance, its intonation and its context to identify the correct dialogue act which his interlocutor wants to communicate. Two models to analyze dialogue acts are DAMSL (Dialogue Act Markup in Several Layers) [1] and DIME-DAMSL [2]; the latter is a multimodal adaptation of DAMSL to the DIME project [3]. The Verbmobil Project [4] developed another dialogue act model, which has been used in practical dialogue systems.

DAMSL assumes that dialogue acts occur on four dimensions: communicative status, information level, forward and backward looking function. The communicative status determines if an utterance was uninterpretable or abandoned or if it expressed a self-talk. The information level classifies utterances according to whether they refer to the task, the task management, or the communication management. The forward looking function identifies the effect which an utterance has on the future of the dialogue; this includes statements (assert, reassert), influencing an addressee future actions (open option, action directive), information requests, committing a speaker future actions (offer, commit), conventional (opening, closing), explicit performative, or exclamation. Backward looking function indicates the way an utterance relates to one or more previous utterances; this includes agreement (accept, accept part, maybe, reject part, reject, hold), understanding (signaling non-understanding; signaling understanding as acknowledgment, repeat or rephrase, completion; correct misspeaking), or answer.

DIME-DAMSL is a multimodal extension to DAMSL; this latter scheme introduces new tags to annotate graphical events occurring on a software interface where speakers interact. In addition, DIME-DAMSL incorporates the notions of transaction structure, expression planes and communicative charges and credits, all of which are distributed along the dimensions already defined by DAMSL. The transaction is a subset of utterances in a dialogue where the speakers interact in order to achieve a specific subgoal which is a part of a main goal of the dialogue. The expression planes are two: obligations and common-ground. The former includes communicative actions where the speaker creates an obligation (on his interlocutor or on himself) to execute an action or to give some piece of information; instances of this plane are information requests, action directives and commits. In the common-ground plane a speaker establishes his agreement or understanding regarding to the knowledge, the presuppositions or the believes of his interlocutor; this plane is subdivided into two subplanes: agreement and understanding; the first includes dialogue acts whose purpose is to establish mutual believes between the dialogue participants, for instance when a speaker asserts something which has not been asked before, or when the listener accepts or rejects something that was said by the speaker. The understanding subplane serves to express that an utterance was understood (or not) or that it was at least heard by the interlocutor; for instance, an acknowledgment, a complementation, or a rephrase.

Dialogue act recognition can contribute to improve the efficiency of spoken dialogue systems, specially of those defined by using dialogue models. In practical dialogues just a few types of dialogue acts can occur in a specific conversational situation, so their automatic recognition could be a relatively simple task. The information to distinguish a dialogue act might be in one or more sources: the lexical content of the utterance, its intonation, its duration, its intensity parameters, the presence and location of stressed syllables, the role (system or user) of the speaker who uttered it, etc. In the current research, we aim to develop a methodology to recognize (predict) dialogue acts by adopting a specific theory (DIME-DAMSL) and by analyzing empirical data organized on a series of tagging tiers.

The DIME project (Multimodal Intelligent Dialogues in Spanish) has among its goals to build a practical-dialogue management system, able to develop task-oriented dialogues with a human user by voice and graphical interfaces. One way to achieve this goal is by using both models of automatic speech recognition and automatic dialogue act identification. A way to create these models is analyzing empirical data, so empirical resources were created within the DIME project. These resources are the DIMEx100 [5] and the DIME corpora [6]; the former is being used to create acoustic models and pronunciation dictionaries for speech recognition, and the latter is being used to investigate and to create dialogue-act and speech-repair models and to evaluate intonation models. Both corpora were recorded in Mexican Spanish. In this paper, the DIME corpus is the source to assess the extent to which prosodic and speaker information can contribute to predict dialogue act types. The corpus is being tagged on several layers: orthographic transcription (already concluded), phonetic segments and suprasegments, dialogue act types, speech repairs, and tone and break indices.

Preliminary results of our machine-learning experiments, along the general lines of [7], are presented in this paper; our predictor data are prosodic data from utterances and speaker role, and the target data is the dialogue act type.

2 The Empirical Resource

The DIME corpus consists of a set of 26 task oriented dialogues in the kitchen design domain. The corpus was collected in a Wizard of Oz scenario (although the subjects knew that the Wizard was human). In the first phase of this project the corpus was segmented and transcribed orthographically. In the present phase a time aligned annotation in several layers is being developed; this includes the segmental (i.e. allophones) and suprasegmental (i.e. syllables, words and intonation patterns) layers; the corpus is also being tagged at the level of dialogue acts using the DIME-DAMSL annotation scheme. The most relevant tagging tiers for this experiment are: orthographic transcription, the intonation transcription with INTSINT scheme, utterance duration (in milliseconds); sentence mood (surface form), which was automatically predicted by a CART-style tree; speaker role (*system* or *user*), and dialogue acts tagging. The orthographic transcription of some instances of the corpus are as follows. In these transcriptions, *s* is the system (Wizard) and *u* is the human user.

utt1 : s: ¿Quieres que desplace o traiga algún objeto a la cocina? (*Do you want me to move or displace some object into the kitchen?*)

utt2 : u: <ruido> No (<noise> *No.*)

utt3 : u: ¿Puedes mover la estufa hacia la izquierda? (*Can you move the stove to the left?*)

utt4 : s: <ruido> ¿Hacia dónde? (<noise> *where to?*)

utt5 : u: <ruido> Hacia <sil> hacia la derecha (<noise> *to <sil> to the right.*)

3 Prosodic Tagging

Intonation patterns in the DIME Corpus are tagged with the INTSINT [8] annotation scheme; in this scheme, intonation is modeled through a sequence of tags associated to the inflection points of the F0 (fundamental frequency) contour. The tag assigned to each inflection point is relative to its predecessor and its successor along the contour. The tag set is: M (*medium*), T (*top*), B (*bottom*), H (*higher*), L (*lower*), U (*up-step*), D (*down-step*) and S (*same*). Tags are computed automatically by using the MOMEL algorithm [9] in the MES software tool [10]. MOMEL provides a default stylized F0 contour; then a perceptual verification task is performed by human annotators. In this latter process inflection points are modified, added or deleted, until the stylized intonation matches the original intonation of the utterance.

For instance, the original F0 of the utterance ¿Me puedes recorrer el el fregadero un poco hacia <sil> hacia el frigobar? (*Can you move the the sink a little bit to <sil> to the minibar?*) is shown in Figure 1.

The prosodic transcription is performed in four major stages using MES. The first is to extract the original F0 contour using AMDF (Average Magnitude Difference Function), autocorrelation or comb function algorithms; the second step is to produce the stylized contour using the MOMEL algorithm, which does not guarantee a perfect stylization and might produce a contour different from the original F0, as can be seen in Figure 2 (i.e. some regions of the stylized contour do not coincide with the original contour); in the third stage, a human annotator develops a perceptual verification task

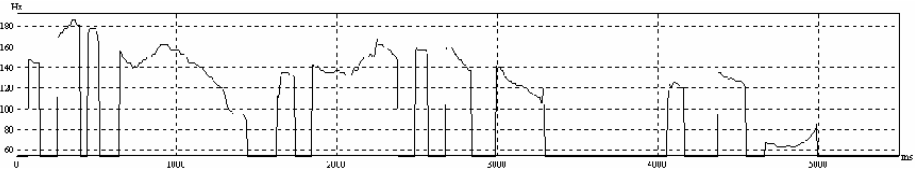


Fig. 1. Original F0

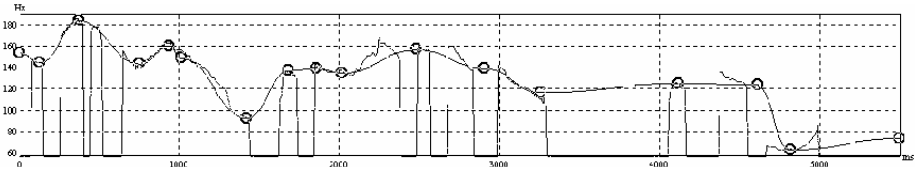


Fig. 2. Stylized F0 (dark contour) with its inflection point (circles)

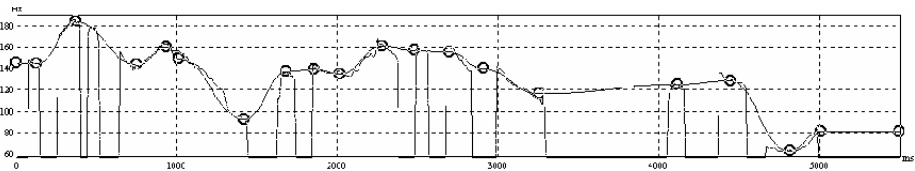


Fig. 3. Stylized F0 (dark contour) after perceptual verification

in which inflection points could be relocated, eliminated or inserted until the stylized contour is perceived as the original F0 curve as shown in Figure 3; finally, the fourth step consists in to produce INTSINT tags automatically, as can be seen in Figure 4; for our example these are MSTLHDLUHLHDSDLUHBUS. In addition to these four stages, and for the particular purpose of this experiment, INTSINT strings were cleansed by deleting S (*same*) tags because these are redundant. This transformation produces simpler strings without reducing the reliability of the representation. The final string for our example is MTLHDLUHLHDDLUHBU.

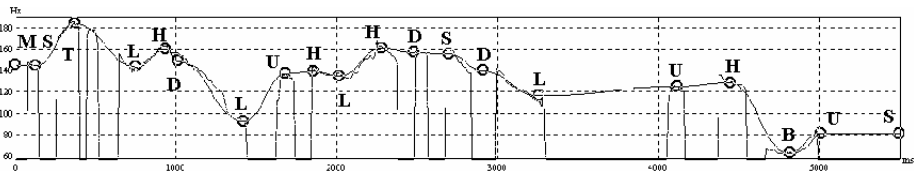


Fig. 4. INTSINT annotation of the inflection points

In addition to this prosodic transcription and utterance duration the duration of lower units including phonetic syllables, pauses, and break indices will be also available for future classification experiments.

4 Dialogue Act Tagging

The dialogue act tagging task was developed manually by three teams of three individuals each, analyzing the orthographic transcription, the audio and the video recordings of every utterance of one dialogue with 100 utterances (approx). Every team produced a DIME-DAMSL tagging data set, so three tagging data sets were obtained.

Since our proposal is basically a model to predict dialogue acts from prosodic information by using a machine-learning algorithm, the consistency of training data is highly critical in order to produce good levels of precision and recall. One of the most sensitive data to train our model are dialogue act taggings. Such production process has to be developed on a very consistent basis, so we supported it by a formal methodology, widely described in [2]; this also describes details about assessing of the tagging agreement. In our experiment the inter-teams agreement was measured with the Kappa statistics [11].

Nine Kappa values were calculated for the three teams, one Kappa for each of the following tagging categories: information level, declarative, information request, influence future actions of listener or of speaker, agreement (i.e. when a speaker agrees to the other in the dialogue), understanding, response (answer), and graphical actions. Every Kappa represents the agreement among the three teams regarding a tagging category, and each Kappa was greater than or equal to 0.8, the minimum suggested in [11] as a good consistency value; besides, our assessing criteria are compatible with the recommendations presented in [12].

We have selected a number of obligation dialogue acts for our preliminary experiments; these are action directive (*action-dir*), information request (*info-request*), and commitment (*commit*) which belong to the forward looking function of DAMSL. We contrast these three dialogue acts with the *other* label, which was used to tag any other dialogue act. Table 1 shows instances of utterances representing some of the dialogue acts considered.

The common-ground dialogue acts used in the experiments are *accept*, *hold* and *reject*, which belong to the agreement subplane; and *rep-rephr* (repeat or rephrase) and *ack* (acknowledgement), which belong to the understanding subplane; we contrast these five dialogue acts with the *other* label.

Table 1. Dialogue act taggings

UTTERANCE	DIALOGUE ACT TAG
utt3: u: Can you move the stove to the left?	<i>action-dir</i>
utt53: s: Where do you want me to put it?	<i>info-request</i>
utt26: s: okay	<i>commit</i>
utt82: u: that is all right.	<i>accept</i>
utt42 : s: this one close to the stove?	<i>hold</i>
utt12 : u: no.	<i>reject</i>
utt6: s: <no-vocal> to the right.	<i>rep-rephr</i>
utt52 : s: okay	<i>ack</i>
utt116: s: we have finished the task.	<i>assert</i>

Manual taggings for dialogue act and for other tiers are currently being developed for other dialogues of the DIME Corpus. Consistency in dialogue act annotation will be assessed with Kappas on a similar way it has already been done.

5 The Experiments

We have already run two sets of machine-learning experiments to define and to evaluate our methodology: one set of experiments to predict dialogue acts on the obligations plane [13], and other for the common-ground plane; we used J48, a CART-style decision tree algorithm [14] implemented in WEKA software [15]. Data from one tagged dialogue (100 utterances approx.) were used; for every experiment, several trees were created using different training and testing subsets in order to compare and validate results. Three modes were considered: 1) subsets which are statistically representative (manually stratified) of the whole data used, where 70% was for training and 30% for testing; 2) subsets which were randomly defined but not strictly representative in 10-fold, 5-fold, 3-fold and 2-fold cross validations; 3) finally, 50, 66, 70 and 75 percent of the whole data were splitted for training and the respective remainders were used for testing; these splits were randomly created and they were not strictly representative. The combination of different attributes and training/testing modes permitted the creation of forty-five decision trees. In some experiments data were used to predict obligation dialogue acts; in other experiment, the same dialogue was used to predict common-ground dialogue acts. Predictor data were intonation, sentence mood, utterance duration and speaker role. Preliminary results show that sentence mood (the surface form of the utterance: declarative, interrogative, imperative) is an important data to predict dialogue act; also, if sentence mood is known, then the dialogue act prediction does not need intonation data, and this was discarded by the decision tree algorithm. Since utterance mood itself would not be available in a real-world system, it would have to be predicted from other data, so specific models were developed to predict it, showing that the final region (the last tones) of the intonation contour are sufficient to this task. The predicted sentence mood was used as one of the predictor data in addition to speaker role and utterance duration for predicting dialogue act types. Experiments showed that using predicted sentence mood is better than using no sentence mood at all. Speaker role data (*user* or *wizard*, in the Wizard of Oz scenario) contributes to improve the dialogue act prediction.

Table 2 reproduces 3 out of the 19 rules from the tree presented in [16] to predict sentence mood in the same annotated dialogue, where the numbers in parentheses are the number of cases complying/non complying each rule. The decision tree algorithm discovered 19 rules, all of which use the data of the last 2 INTSINT labels of the INTSINT taggings. The tree accuracy is 85.1%, and Kappa (comparing against the manually tagged sentence mood) is 0.70390. Recalls, precisions and F-Measures of the tree to predict modalities are reproduced in Table 3. The same tree is used to predict sentence mood in the present experiment, using this as one of the predictor data. There were too few imperative utterances, so the prediction of this sentence mood is not reliable.

Table 2. Some rules to predict sentence mood, reproduced from [16]

RULES
If last_2 = UT, then <i>int</i> (20/1)
If last_2 = DB, then <i>dec</i> (20)
If last_2 = HB, then <i>imp</i> (3/1)

Table 3. Evaluation of the sentence mood prediction from [16]

SENTENCE MOOD	RECALL	PRECISION	F-MEASURE
dec	0.881	0.912	0.897
int	0.850	0.791	0.819

In the experiment to predict obligation dialogue acts, the predicted tags were *action-dir*, *info-request*, *commit*, and all these were contrasted with *other*. As a result, the general average accuracy to predict dialogue act was 66.1830%, with Kappa equal to 0.5153; the best results were obtained with the last 3 INTSINT labels datasets (68.7182% and 0.5538, averages); from the last 3 INTSINT labels datasets, the best tree had 74.1935% and 0.6265, obtained with 70% split-training mode. This could be considered the most useful tree of a 45-trees group and it is presented in Table 4.

Table 4. Tree for predicting obligation dialogue acts

Nr.	RULE
1	if (pred_sent_mood=int) and (sp_role=s), then info-request (29/5)
2	if (pred_sent_mood=int) and (sp_role = u) and (dur > 1568.6875) and (dur <= 4514.875), then info-request (9/3)
3	if (pred_sent_mood = imp), then info-request (3/1)
4	if (pred_sent_mood=int) and (sp_role = u) and (dur<= 1568.6875), then other (3)
5	if (pred_sent_mood = dec) and (sp_role = s) and (dur <= 1209.875) and (dur <= 652.75), then other (3)
6	if (pred_sent_mood = dec) and (sp_role = s) and (dur > 1209.875), then other (9)
7	if (pred_sent_mood = dec) and (sp_role = u) and (dur <= 1158.75), then other (12)
8	if (pred_sent_mood=int) and (sp_role = u) and (dur > 1568.6875) and (dur > 4514.875), then action-dir (4)
9	if (pred_sent_mood = dec) and (sp_role = u) and (dur > 1158.75), then action-dir(10/4)
10	if (pred_sent_mood = dec) and (sp_role = s) and (dur <= 1209.875) and (dur > 652.75), then commit (19/5)

Table 5. Evaluation of the obligation dialogue acts prediction

DIAL. ACT	RECALL	PRECISION	F-MEASURE
other	0.889	0.727	0.800
action-dir	0.200	0.500	0.286
info-request	0.917	0.786	0.846
commit	0.600	0.750	0.667

Predicted sentence mood, role of speaker and duration (on that order) were useful to predict dialogue act, while INTSINT tags were not necessary at this stage, although they were used for predicting sentence mood, which is consistent with the results in [13] and [16]. The precisions, recalls and F-Measures of the predicted dialogue act types are presented in Table 5, where *info-request* has the highest recall, then *other*, then *commit*, and finally *action-dir*. *Action-dir* instances were the least frequent in the data as could be seen in a statistical analysis; the dataset available was too small to assess the result for *action-dir*, so the prediction of this specific dialogue act is not reliable. Performance of *action-dir* label is much lower than the other classes; some possible causes for this can be the following: first, few instances of *action-dir* were available in this specific dialogue, so the machine learning algorithm did not have a sufficient number of examples to learn; and second, sentence mood of *action-dirs* in this particular dialogue is interrogative or declarative most times, so *action-dirs* could be confused with information requests or others.

In the experiment to predict common-ground dialogue acts, the specific tags to be predicted were *accept*, *hold*, and *reject*, which belong to the agreement level; also, *rep-rephr* (repeat or rephrase) and *ack* (acknowledgement), which belong to the understanding level; utterances tagged with *assert* were only used if the speaker was not answering an information request, and all these were contrasted with *other*. The resulting tree is presented in Table 6.

Utterance duration, predicted sentence mood and speaker role (on that order) were useful to predict common-ground dialogue acts, while INTSINT tags were not (although these tags were implicitly used when predicting sentence mood). This is evident by observing that no INTSINT attribute is in the tree in Table 6; that tree was generated using a dataset with the last 3 INTSINT labels, with 10-fold cross-validation. *Other* is the tag which had the highest recall (0.98), and then *accept* (0.774); the recall for the other tags was 0 (zero). Table 7 presents the recalls, precisions and F-measures of this tree. These results are consistent with the statistical description of dialogue acts in the tagged dialogue, where most of them were *others* and *accepts*. This involves that

Table 6. Tree for predicting common-ground dialogue acts

Nr.	RULES
1	if (dur <= 1209.875) and (pred_sent_mood=int) and (sp_role=s), then other (16/2)
2	if (dur <= 1209.875) and (pred_sent_mood=int) and (sp_role=u), then accept (3/1)
3	if (dur <= 1209.875) and (pred_sent_mood=dec) and (sp_role=s) and dura- tion_audio_mseg <= 652.75), then ack (3)
4	if (dur <= 1209.875) and (pred_sent_mood=dec) and (sp_role=s) and dura- tion_audio_mseg > 652.75), then accept (19/5)
5	if (dur <= 1209.875) and (pred_sent_mood=dec) and (sp_role= u), then accept (12/1)
6	if (dur <= 1209.875) and (pred_sent_mood=imp), then accept (0)
7	if (dur > 1209.875), then other (48/14)

Table 7. Evaluation of the common-ground dialogue acts prediction

DIAL. ACT	RECALL	PRECISION	F-MEASURE
other	0.980	0.706	0.821
reject	0	0	0
rep-rephr	0	0	0
ack	0	0	0
accept	0.774	0.727	0.750
hold	0	0	0
assert	0	0	0

more tagged data are necessary to assess the predictability of the other five common-ground dialogue acts which could not be predicted by this tree.

Although few data were available for experiments (from one dialogue only), we consider that these preliminary results seem to be promising. The selected set of dialogue act labels is small because these are preliminary experiments and the corpus annotation is still under process. More labels will be used in experiments when more tagging data are available. Results show that identifying sentence mood and using role of speaker data to identify dialogue act could be useful for a prototype dialogue management system. Other interesting setting to be evaluated in the experiments for the short term is using dialogue act tag of every previous utterance as an additional predictor data.

6 Discussion and Further Work

The methodology we propose to predict dialogue acts consists in using CART-style decision trees on a corpus data where predictor data are utterance duration and sentence mood, and the target data is the dialogue act type; first, sentence mood is predicted from INTSINT intonation taggings. The utility of predicting sentence mood was shown by comparing trees where tagged sentence mood, predicted sentence mood and no sentence mood at all were assessed. The resulting decision trees can be represented as if-then rule sets which can be programmed into a dialogue management system to identify the dialogue act type of an unknown utterance.

Our approach is different from other authors because we are abstracting the intonation representation on a higher level by using alphabetic strings (INTSINT sequences), which allow to analyze intonation patterns as categorical data. INTSINT scheme eliminates the necessity to normalize intonation data among speakers. We have found no references about works where INTSINT scheme is used to predict dialogue acts, so this could be a new approach to solve the problem of dialogue act prediction.

The present methodology promises a simple way to identify dialogue act types for the construction of dialogue managers for practical dialogues; at the present stage of this investigation we have few data available, so this work will be continued with more tagging experiments focusing on the identification of other obligation dialogue acts and also common ground dialogue acts, and then on the construction of a complete model including all dialogue act types contemplated in the DAMSL scheme. For

the completion of this experiment we plan to use, in addition, syllable and pause durations, stressed syllables location, break indices, and some lexical information.

Acknowledgments

We thank useful comments and suggestions from James Allen, at Rochester University, and from Joaquim Llisterri and Monserrat Riera at Universidad Autonoma de Barcelona. We also thank Hayde Castellanos, Varinia Estrada, Fernanda Lopez, Isabel Lopez, Ivan Meza, Ivan Moreno, Patricia Perez, Carlos Rodriguez, Issac Castillo, Javier Cuetara, Laura Irene Gonzalez, Ivonne Lopez, Laura Lorena Rosales and Arturo Wong who participated in the tagging task, and also for useful comments and suggestions. The theory and experiment reported in this paper are being developed within the context of the DIME-II project, with partial support of NSF/CONACyT grant 39380-A.

References

1. Allen, J.F., Core, M.: "Draft of DAMSL: Dialog Act Markup in Several Layers", Technical report, The Multiparty Discourse Group. University of Rochester, Rochester, USA, October, 1997
2. Pineda, L., Castellanos, H., Coria, S., Estrada, V., López, F., López, I., Meza, I., Moreno, I., Pérez, P., Rodríguez, C.: "Balancing Transactions in Practical Dialogues", in CICLING 2006, Centro de Investigacion en Computacion, Instituto Politecnico Nacional, Mexico City, February, 2006
3. Pineda, L., Massé, A., Meza, I., Salas, M., Schwarz, E., Uruga, E., Villaseñor, L.: "The DIME Project", Lecture Notes in Artificial Intelligence, Vol. 2313, Springer, pp. 166-175, 2002
4. Jekat, S., Klein, A., Maier, E., Maleck, I., Mast, M., Quantz, J.J.: "Dialogue Acts in VERBMOBIL", VM-Report 65, Universitat Hamburg, DFKI Saarbrucken, Universitat Erlangen, TU Berlin, April 1995
5. Cuetara-Priede, J., Villaseñor-Pineda, L., Pineda-Cortes, L.: "A New Phonetic and Speech Corpus for Mexican Spanish", Iberamia 2004, INAOE, Tonantzintla, Pue., Mexico, 2004
6. Villaseñor, L., Massé, A., Pineda, L.: "The DIME Corpus", ENC 01, 3er Encuentro Internacional de Ciencias de la Computación, SMCC-INEGI, Aguascalientes, Mexico, 2001
7. Shriberg, E., Bates, R., Stolcke, A., Taylor, P., Jurafsky, D., Ries, K., Coccaro, N., Martin, R., Meteer, M., Van EssDykema, C.: "Can Prosody Aid the Automatic Classification of Dialog Acts in Conversational Speech?", Language and Speech 41(3-4), Special Issue on Prosody and Conversation, 439-487, USA, 1998
8. Hirst, D., Di Cristo, A., Espesser, R.: "Levels of representation and levels of analysis for the description of intonation systems", In M. Horne (ed) Prosody: Theory and Experiment, Kluwer-Dordrecht, 2000
9. Hirst, D., Espesser, R.: "Automatic modeling of fundamental frequency using a quadratic spline function", Technical report, CNRS (URA 261), Institut de Phonétique d'Aix, Université de Provence, France, 1993
10. Espesser, R.: MES: Motif Environment for Speech software http://www.lpl.univ-aix.fr/ext/projects/mes_signaix.htm, 1999
11. Carletta, J.: "Assessing agreement on classification tasks: the kappa statistic", Computational Linguistics 22(2), Association for Computational Linguistics, USA, pp. 249-254, 1996

12. Craggs, R., McGee Wood, M.: "Evaluating Discourse and Dialogue Coding Schemes", in *Computational Linguistics* 31(3), Association for Computational Linguistics, USA, pp. 289-295, 2005
13. Coria, S., Pineda, L.: "Predicting obligation dialogue acts from prosodic and speaker information", in *Research on Computing Science (ISSN 1665-9899)*, Centro de Investigacion en Computacion, Instituto Politecnico Nacional, Mexico City, September, 2005
14. Witten, I., Frank, E.: *Data Mining. Practical Machine Learning Tools and Techniques with Java implementations*, Morgan-Kaufman Publishers, San Francisco, CA, USA, 2000
15. Frank, E., Hall, M., Trigg, L.: *WEKA: Waikato Environment for Knowledge Analysis software*, <http://www.cs.waikato.ac.nz/~ml/weka>, 2004
16. Coria, S., Pineda, L.: "Predicting obligation dialogue act types from prosodic information", 2nd Midwest Computational Linguistics Colloquium, Ohio State University, USA, 2005

Disambiguation Based on Wordnet for Transliteration of Arabic Numerals for Korean TTS

Youngim Jung¹, Aesun Yoon², and Hyuk-Chul Kwon¹

¹Pusan National University, Department of Computer Science and Engineering,
Jangjeon-dong Geumjeong-gu, 609-735 Busan, S. Korea
{acorn, hckwon}@pusan.ac.kr

²Pusan National University, Department of French,
Jangjeon-dong Geumjeong-gu, 609-735 Busan, S. Korea
asyoon@pusan.ac.kr

Abstract. Transliteration of Arabic numerals is not easily resolved. Arabic numerals occur frequently in scientific and informative texts and deliver significant meanings. Since readings of Arabic numerals depend largely on their context, generating accurate pronunciation of Arabic numerals is one of the critical criteria in evaluating TTS systems. In this paper, (1) contextual, pattern, and arithmetic features are extracted from a transliterated corpus; (2) ambiguities of homographic classifiers are resolved based on the semantic relations in KorLex1.0 (Korean Lexico-Semantic Network); (3) a classification model for accurate and efficient transliteration of Arabic numerals is proposed in order to improve Korean TTS systems. The proposed model yields 97.3% accuracy, which is 9.5% higher than that of a customized Korean TTS system.

1 Introduction

TTS technologies for naturalness have improved dramatically and have been applied to many unlimited systems in terms of domain. However, improvement on the technique for accurate transliteration of non-alphabetic symbols such as Arabic numerals and various text symbols¹ has been relatively static.

According to the accuracy test results of 19 TTS products by Voice Information Associates, the weakest area of TTS products is in number processing in the following ambiguity-generating areas, as shown in Table 1 [10].

In the modern Korean language, numerals have three different origins – Korean, Chinese and English – and they show a variety of variants. Since their distribution is largely dependent on context, automatic transliteration of Arabic numerals for Korean TTS is very complicated. For example, a single numeral, '4,' can be read in

¹ Since Arabic numerals and text symbols have graphic simplicity and deliver more precise information, the occurrence of Arabic numerals and text symbols is as high as 8.31% in Korean newspaper articles.

Table 1. TTS Accuracy Test Results Summary

Test area	Accuracy (%)
Number	55.6
Word of Foreign Origin	58.8
Acronym	74.1
Abbreviation	72.9
Name	70.7
Address	69.0
Homograph	83.4

five different ways depending on its following classifier² or its preceding morphemes, as shown in (E 1)³.

(E 1)	Input	Transliteration of ‘4’	Meaning
a.	4 <i>myeong</i>	<i>ne</i> / <i>neog</i> / <i>neol</i> / <i>sa</i> / <i>po</i>	four persons
b.	4 <i>il</i>	* <i>ne</i> / <i>neog</i> / <i>neo</i> / <i>sa</i> / <i>po</i>	four days
c.	<i>big</i> 4	* <i>ne</i> / <i>neog</i> / <i>neo</i> / <i>sa</i> / <i>po</i>	Big four
d.	<i>4</i> <i>mal</i>	* <i>ne</i> / <i>neog</i> / <i>neo</i> / <i>sa</i> / <i>po</i>	54ℓ
e.	<i>4</i> <i>dae</i>	<i>ne</i> / <i>neog</i> / <i>neo</i> / <i>sa</i> / <i>po</i>	four cars or the fourth, or the four biggest

In Example (E 1-e), however, the homographic classifier ‘*dae*’ does not give any clue for reading its preceding Arabic numeral. This numeral, ‘4,’ can be read as *ne* when ‘*dae*’ represents “unit of automobiles, machines,” whereas ‘4’ can be read as *sa* when ‘*dae*’ means “time of life or persons in the time of life” or “the biggest.” Thereby, disambiguation of homographic classifiers is prerequisite for selecting the correct reading of their preceding Arabic numerals.

In this paper, (1) contextual, pattern, and arithmetic features are extracted from the transliterated corpus; (2) ambiguities of homographic classifiers are resolved based on the semantic hierarchies in KorLex 1.0 (Korean Lexico-Semantic Network); and (3) a classification model for accurate and efficient transliteration of Arabic numerals is proposed. In Section 2, related studies on ambiguities in reading Arabic numerals and on word-sense disambiguation (WSD) are examined. In Section 3, a classification model constructed by learning contextual features, patterns, and arithmetic features extracted from our corpus is suggested. In Section 4, ambiguities in reading Arabic numerals caused by homographic classifiers are analyzed, the re-categorization of the semantic classes based on the lexical relations in

² In this paper, ‘classifiers’ refer to Korean function words representing units of countable or measurable objects, actions, time, and others. In the machine-learning field, a classifier is a learning model that classifies target answers. In order to avoid confusion, the term ‘classification model’ is used for indicating the latter in this paper.

³ In (E 1), word in italics stand for transliterated Korean alphabet, and text in bold font represents the correct pronunciation of target Arabic numeral. *Myeong* is “unit of persons” and *il* means “day”. *Mal* is a Korean unit of volume for measuring liquid or grain; one *mal* is about 18 ℓ. *Dae* is a homographic classifier representing (1) “unit of automobiles, machines,” (2) “time of life or persons in the time of life” or (3) “the biggest.”

KorLex1.0 is illustrated, and the performance of the proposed model is evaluated. Conclusions and future work follow.

2 Related Studies

Depending on the context, readings of Arabic numerals differ in various forms. In Section 2, work on the ambiguities of reading Arabic numerals in Korean texts, and two different approaches for WSD, is studied.

2.1 Ambiguities in Reading Arabic Numerals

[Yoon *et al.*, 2003] presented reading formulae of Arabic numerals (RFA) for the Korean numeric system. The criteria for subcategorizing the reading formulae are the origins, part of speech (POS) or senses of Arabic numerals, as well as the distribution of allo-morphemes in reading numerals. Six types of components of numerical expressions are required to select a correct reading of numerals. The components are (1) pre-numeral (e.g. *je*: prefix representing an order), (2) decimal scale marker (DSM)⁴, (3) post-numeral (e.g. *yeo*: suffix representing approximate amount), (4) classifiers (e.g. *myeong*, *il*, *dae* and others), and (5) post-classifier (e.g. *ssig*: suffix representing ‘by’). In addition, reading rules of numerical expressions are suggested by combining those components. An automatic transliteration system of Arabic numerals based on hand-craft rules by linguists for Korean TTS, was derived [3]. This system achieved 95.6~97.7% accuracies in transliterating Arabic numerals, which accuracies were 3.9~18.3% higher than those of two customized Korean TTS systems. However, this rule-based system has limitations in building and handling complicated rules when the components of numerical expressions are ambiguous. Furthermore, a learning algorithm of features extracted from new data is necessary in order to build a classification model for the transliteration of Arabic numerals that does not conflict with established models. Therefore, a learning model for new data is suggested in this paper.

2.2 Word-Sense Disambiguation

There are two approaches to WSD: one is WSD based on language knowledge or rules built by language analysis, and the other is WSD based on learning features extracted from language resources. Depending on what kind of language resource is available for training a WSD system, three methodologies have been suggested: ① supervised learning based on sense-tagged corpora, ② dictionary-based disambiguation based on dictionaries, thesauri or wordnets, and ③ unsupervised disambiguation, in which untagged text corpora are available. Since using mere untagged corpora does not improve the performance of classification, the third methodology has seldom been adopted for classification tasks. Thus, the first two methodologies are studied in this section.

⁴ Korean DSMs are *sib* (“10”), *baeg* (“100”), *cheon* (“1000”), *man* (“10,000”), *eog* (“100,000,000”), *jo* (“1,000,000,000,000”), and others.

2.2.1 WSD Based on Tagged Corpus

Since sense tags vary depending on applications, available corpora can be different. For example, the Brown corpus or Penn tree bank and the *Sejong* corpus are useful for English POS tagging and for Korean POS tagging, respectively. In order to disambiguate the strict senses of word, Semcor, an English corpus labeled with semantic tags, is available. For speech synthesis, correct pronunciation of a target ambiguous word in its context is tagged. If there is no established corpus fitting one's research purpose, corpora with tags are constructed by researchers according to the applications and the purposes [12], [15].

Since WSD is ultimately a problem of classification, features extracted from sense-tagged corpora are used to correctly classify instances of ambiguous word in new data [Yarowsky, 1997]. The Naive Bayes classification model, proposed in [Gale *et al.*, 1992], has an advantage in combining a large number of parameters efficiently for learning. The model assumes, however, that every parameter is independent of the others [6]. Because of the strong dependencies of contextual features, a decision tree algorithm has been adopted [9], [15], which algorithm is an efficient classification model for handling complex conditional dependencies and non-dependencies. Efficiency deteriorates when the classification model handles very large parameter spaces, such as highly lexicalized feature sets. However, in many NLP tasks, similar individual features (word) can be grouped as categories (POS or semantic categories), and the number of parameters is limited. In addition, the distinctive power of each feature is explicitly represented through a constructed tree. Consequently, decision tree has been adopted in numerous studies on NLP.

However, a tagged corpus is expensive because established sense-tagged corpora are rare and the cost of constructing corpora is a time- and labor-consuming job [5]. Currently, no established Korean corpus offers information on the sense and the pronunciation of Korean word for speech synthesis at the same time. Thus, appropriate pronunciations and senses of word or non-alphabetic symbols have to be assigned by hand. Otherwise, automatic (or semi-automatic) assignment of semantic categories of word from established thesauri or wordnets is necessary.

2.2.2 WSD Based on Thesauri and Wordnets

Definitions in machine-readable dictionaries and semantic categories of word in thesauri have been commonly used for WSD. When multiple semantic categories of ambiguous word are obtained from dictionaries or thesauri, algorithms for scoring candidates and selecting one semantic category in one context are used [6].

WordNet, an English electronic lexical database, has been developed and constructed by G. Miller and his colleagues [7], and many studies of WSD based on WordNet have been conducted. Twenty-five semantic categories in WordNet have been applied in [Agirre, 1996]. In respect of sense granularity, semantic categories in WordNet are finer than those of Roget thesaurus.

Word-sense disambiguation by combining local context and WordNet similarity measures has been explored. In order to identify the specific sense in the specific context among the four senses of 'serve', three approaches to WSD have been

examined: (1) using local context classification model, (2) using word similarity in WordNet and (3) combining local context and WordNet similarity measures. The method for combining syntactic information with semantic information from WordNet is proved to produce a modest improvement in performance [5]. Using a hierarchical structure containing nouns with 12 levels of depth and verbs with 4 levels of depth, conceptual distance or information content for application to WSD have been studied. Conceptual distance measures the semantic relatedness between two words, or two concepts by counting edges between them or by considering link direction or density [1]. According to information content based approach, semantic relatedness between a pair of concepts lexicalized in WordNet can be measured by the information content of their lowest-super-ordinate (most specific common subsumer) [7].

In this paper, the combined method using local context and lexical information from WordNet is adopted; however, any complicated calculation such as measuring conceptual distance or information content is avoided.

3 Classification Model for Transliteration of Arabic Numerals

As shown in Section 2, readings of Arabic numerals vary and can be determined using components of Arabic numeral expressions. In this section, contextual features, pattern features, and arithmetic features that affect the reading of Arabic numeral expressions are analyzed and extracted from a corpus to build a classification model. The corpus was randomly compiled from news articles of 10 major newspapers in Korea issued from January 1st, 2000 to December 31st, 2001. The size of the corpus is 100,000 words. Ten percent of the data was allocated to evaluating, in Section 4, our classification model in comparison with other customized Korean TTS systems. The rest of the data was trained using the C4.5 algorithm.

All instances of numeral expressions from the corpus are collected and then the correct RFA tags are labeled. The process of labeling RFA tags is semi-automated, using the rule-based transliteration system of Arabic numerals expressions [3].

3.1 Classification of Readings of Arabic Numerals

The sub-categorizations proposed by [Yoon *et al.*, 2003] are adopted in this paper depending on the origins, part of speech (POS) or senses of Arabic numerals, as well as the distribution of allo-morphemes in reading numerals, as shown in [Table 2]. Korean numeric systems are used only in the range of one and one hundred. Chinese numeric systems are used in a wider range, from zero to infinite number. Variants of Chinese numeric system with DSM are adopted only the numeral combines with *wol* (“a unit of month”). Chinese numeric system without DSM is widely used in addressing telephone numbers, zip codes, account numbers, and others, because of their phonetic simplicity. English numeric systems are borrowed recently, and their usage is rather limited. They are used only under ten.

Table 2. Classification of Readings of Arabic Numerals

Readings of Arabic numerals	RFA	Examples Input	Output
Korean cardinal in base form	Kca_b	4 <i>myeong</i> “four persons”	[<i>ne</i>]
Korean cardinal variants	Kca_v	4 <i>mal</i> “72 l”	[<i>neo</i>]
Korean ordinal	Kor_b	4 <i>jjae</i> “the fourth”	[<i>nes</i>]
Chinese in base form with DSM	C_b [+D]	4 <i>il</i> “four days”	[<i>sa</i>]
Chinese in base form without DSM	C_b [-D]	SM520 “a model of car”	[<i>o-yi-gong</i>]
Chinese variants with DSM	C_v	6 <i>wol</i> “June”	[<i>yu</i>]
English with DSM	Brn	big 4 “Big four”	[<i>po</i>]

3.2 Extracting and Training Features

Word around Arabic numerals can be used as distinctive features to predict the correct RFA. According to [Yarowsky, 1997], ± 20 words are considered to be a practical context width for the disambiguation of English homographs. However, the result of experimentations with the *Sejong* corpus composed of 150 million words has illustrated that 96.4% of words contributing contextual features are distributed within ± 3 words of the target ambiguous word in Korean [4]. Based on the result from the Korean corpus, ± 3 words from the target ANEs are extracted from our corpus. In addition to contextual features, pattern features and arithmetic features characterize the kinds of simpler forms or identifiers that an ANE represents. In this section, the steps of the extraction of contextual, pattern, and arithmetic features are shown with sample sentences.

Step 1: Morphological Analysis

In Korean, content word and function morphemes such as case markers, postpositions, or endings come in one word⁵. Content word should be separated from function morphemes and be lemmatized through morphological analysis. For example, lemmatized content words in (E 2) were marked in bold fonts as follows:

(E 2)	Input	Meaning
a.	sagwa chong 4-5 <i>gae</i>	total four or five apples
b.	gyeonggi-leul 4-5 <i>lo ji-n</i>	lost the game by four to five
c.	munui 02-5459-3333	Inquiry 02-5459-3333 (telephone number)

Step 2: Semantic Categorization of Context features

Among lemmatized content words, nouns analyzed to have distinctive power for disambiguation of the target ANE are extracted and listed as follows:

(E 2')	Nouns having distinctive powers
a.	sagwa (“apple”), chong (“total”), gae (“unit of things”)
b.	gyeonggi (“game”)
c.	munui (“inquiry”)

⁵ In this paper, the term ‘word’ is used to refer to “a cluster of continuous alphanumeric morphemes and text symbols, with a space on either side,” according to the definition of Francis & Kučera (1982).

Since too many parameters lower the learning efficiency in decision tree, individual word preceding or following ANEs should be clustered into semantic categories. The semantic categories are established by authors as shown in [Table 3].

Table 3. Semantic Categories of Contextual Features

Semantic category	Excerpted list of contextual features
Quantity (QT)	<i>sagwa</i> “apple”, <i>chong</i> “total”
Sport (SP)	<i>gyeonggi</i> (“game”)
Number (NB)	<i>mun-ui</i> (“inquiry”)
Time (TM)	<i>ojeon</i> (“morning”), <i>si</i> (“hour”)
Date (DT)	<i>nyeon</i> (“year”), <i>wol</i> (“month”), <i>il</i> (“day”)
Order (OR)	<i>je</i> (prefix meaning order), <i>wi</i> (“rank”),
Formula (FM)	<i>sig</i> (“formula”)
Index (IX)	<i>jisu</i> (“index”), <i>gagyeog</i> (“price”)
Location (LT)	names of states, cities, streets, <i>jangso</i> (“place”), <i>jiyeog</i> (“region”)
Name (NE)	names of entities

Step 3: Extraction of Learning Features

Pattern features and arithmetic features characterize the types of simpler forms or identifiers that an ANE represent. The attributes and the values of the pattern and arithmetic features are summarized in [Table 4].

Table 4. Pattern and Arithmetic Features of ANEs

Features	Attributes	Value
Pattern features	Number of numerals in an ANE	1~9
	Number of text symbols in an ANE	0~9
Arithmetic features	Types of text symbols	T0: none, T1 : ‘-’, T2 : ‘~’, T3 : ‘.’ T4 : ‘,’, T5 : ‘:’ T6 : ‘/’, T7: ‘+’
	Size of an Arabic numeral	S1: 1900<x<2100, <0<y<12, 0<z<32 (x, y, z are integers appeared in one ANE) S2: the rest
	Difference between two numerals	B1: (y-x)10 ⁿ =1*10 ⁿ (n≥0, x, y, n: integers), B2: the rest
	1 st place of an ANE	FP0: ‘0’, FP1: the rest (not ‘0’)
	Places of an Arabic numeral	P1: 1places, P2: 2places, P3: 3places, P4: more than 4places

Input ANEs are converted to patterns in advance. For example, ‘4-5’ in (E 2-a, b) are converted to ‘N-N’, and ‘Number of numerals in an ANE=2’, ‘Number of text symbols in an ANE=1’ and ‘Types of text symbols=T1’ are obtained, whereas ‘02-5459-3333’ in (E 2-c) are converted to ‘N-N-N’ obtaining ‘Number of numerals in an ANE=3’, ‘Number of text symbols in an ANE=2’ and ‘Types of text symbols=T1’ for its pattern features. Once the pattern features are obtained, the corresponding arithmetic features are extracted to distinguish the same patterns having different meanings. For example, to the pattern ‘N-N’, ‘Difference between two numerals’ is tested and the

value ‘B1’ (E 2-a, b) is given. For ‘N-N-N’ in (E 2-c), ‘Size of Arabic numeral = S2’ and ‘1st place of Arabic numerals = FP0’ are extracted. ‘*sagwa, chong, gae*’ in (E 2-a), ‘*gyeonggi*’ in (E 2-b) and ‘*munui*’ in (E 2-c) are attributed as ‘QT’, ‘SP’ and ‘NB’, respectively.

Step 4: Training Learning Features and Testing Classification Model

Since contextual features affect each other and arithmetic features largely depend on patterns, a decision tree was adopted as the learning algorithm. In order to construct the decision tree, the information gain of each feature is calculated, and then the best feature is selected step by step. Information gain tends to prefer attributes with large numbers of possible values. To compensate for this strong bias, a modification of the measure, called the gain ratio, is widely used [8].

$$gain(S, A) = E(S) - \sum_{k \in Values(A)} \frac{|S_k|}{|S|} E(S_k) \quad (1)$$

$$splitinfo(S, A) = - \sum_{k \in Values(A)} \frac{|S_k|}{|S|} \times \lg\left(\frac{|S_k|}{|S|}\right) \quad (2)$$

$$gainratio(X) = \frac{gain(X)}{splitinfo(X)} \quad (3)$$

S: Example set of ANEs; A: Attributes; S_k : Class to which S belongs (e.g. C_b[+D]).

Equation (2) represents the potential information generated by dividing S into k subsets. Then we can obtain the proportion of information generated by the split, as in Equation (3) [11]. After the decision tree is constructed, the performance of the model is tested using a 10-cross validation checking method. Baseline accuracy is measured by adopting this one rule: if the number of groups in the target ANEs is ‘1’, then the RFA is ‘C_b[+D]’, which is the most frequent class. The accuracy of the baseline is 75.8%. The proposed model yields 97.3% accuracy, which is 21.5% higher than that of the baseline. Though the result is good, two problems still remain to be resolved: (1) arbitrary categorization of contextual features and, (2) the WSD of homographic or polysemic word used as contextual features.

Therefore, in Section 4, disambiguation based on KorLex 1.0, in which lexical information is structured hierarchically, will be described.

4 Disambiguation of Homographic Classifiers Based on Wordnet

In this section, resolutions of the two problems underlying WSD using a tagged corpus are suggested based on the lexical hierarchy and semantic relations contained in KorLex 1.0. Two hypotheses are presupposed for the application of a lexical hierarchy in KorLex 1.0:

(H-1) semantic ambiguities caused by homographs or polysemic word can be reduced or removed by mapping the word to a lexical hierarchy;

(H-2) hyponyms inherit semantic characteristics from their hypernyms in a lexical hierarchy.

4.1 Ambiguities Caused by Homographic Classifiers

As seen in Example (E 1-a~d), classifiers following Arabic numerals play an important role in determining the reading of Arabic numerals. As we can see in (E 1-e), however, Arabic numerals combined with homographic classifiers do not select a unique RFA.

Since many Chinese homographic classifiers⁶ are combined with Arabic numerals, precedent analysis of the senses of the classifiers is required in order to select the correct RFA. [Table 5] shows each sense of homographic classifier and the RFA.

Table 5. Homographic Classifiers (excerpted)

Homographic classifiers		RFA	
Pronunciation	Sense		
<i>Dae</i>	1	Unit of automobiles, machines	Kca_b
	2	The time of life or persons in the time of life	C_b [+D]
	3	The biggest (item)	C_b [+D]
<i>Pyeon</i>	1	Flight	C_b [-D]
	2	Unit of volumes	Kca_b

Other homographic classifiers such as ‘*gi1* (unit of heavy machineries, rockets, tombs), *gi2* (unit of a stage, a session)’, ‘*gu1* (Unit of a dead body), *gu2* (Unit of a borough), *gu3* (Pitch)’, ‘*dan1* (unit of bundled vegetables), *dan2* (level)’, ‘*dong1* (unit of container for liquid), *dong2* (unit of village)’, ‘*byeong1* (a bottle), *byeong2* (rank of a soldier)’, ‘*chug1* (unit of ships), *chug2* (Korean measurement of height)’, ‘*bag1* (musical time), *bag2* (unit of stay)’, ‘*bun1* (honorific form for persons), *bun2* (a minute)’, ‘*su1* (a move in Go game), *su2* (a work of poetry), *su3* (sou)’, ‘*guan1* (Korean measurement of weight), *guan2* (unit of halls)’, ‘*jib1* (a series), *jib2* (a house)’, ‘*sedae 1* (unit of households), *sedae 2* (unit of generation)’ were analyzed.

4.2 Reclassification of Semantic Categories of Contextual Features

In Section 3.2, semantic categories of contextual features were classified depending on the senses of ANEs. They work well for disambiguation of ANEs and selecting correct RFA, as shown in the same section. However, the categories have limitations in that (1) they do not classify the semantic categories of homographic or polysemic word appropriately, and (2) they do not represent the semantic characteristics of word used as contextual features. Re-categorization of contextual features is required, and is performed based on lexical hierarchy in KorLex 1.0 under the hypothesis (H-1). The process is described taking the following sentences containing ‘*dae*’ as examples.

⁶ Many words in Korean have been borrowed from Chinese, and more homographs have been distributed more widely than in English [4].

(E 3) Input	Meaning
a. <i>taegsi-neun eobs-go beoseu 2dae-wa sugbag-eobso-ui seunghabcha-man unhaengdoe-n-da</i>	There is not taxi but two buses, and passenger vans run by hotels
b. <i>mihonnam-ui boheomlyo-neun 40dae gyeolhonhan namja-wa yeoja-boda</i>	premium of bachelors is higher than that of married men and women in their forties
c. <i>yadang-eun yeongsuhoedam-ui 3dae jeonje jogeon-eul jujangha-go</i>	The opposition required three main pre-conditions before key leaders conference

Step 1: Clustering lemmatized word used as contextual features extracted from the tagged corpus. {*taegsi* (“taxi”), *beoseu* (“bus”), *seunghabcha* (“passenger van”)} in (E 3-a), { *mihonnam* (“bachelor”), *namja* (“man”), *yeoja* (“woman”) in (E 3-b), and {*jeonje* (“premise”), *jogeon* (“condition”)} in (E 3-c) are clustered.

Step 2: Mapping words by cluster to the KorLex hierarchy.

Step 3: Listing all common hypernyms of synset nodes mapped from contextual features.

Step 4: Finding the Least Upper Bound (LUB) of synset nodes in a cluster mapped from contextual features. Here, *susonggigwan* (“transport”), *saram* (“person”) and *jeonje* (“premise”) are selected as LUBs, as shown in [Figure 1].

Step 5: Selecting the LUB as a semantic category for the cluster of contextual features.

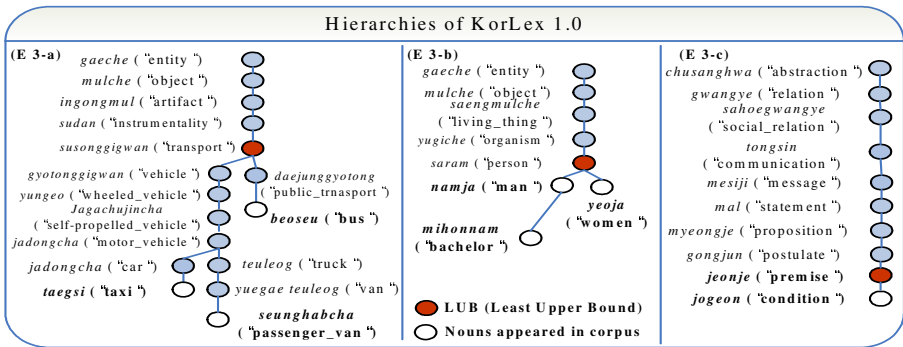


Fig. 1. Automatic Selection of Least Upper Bound

By application of the procedure to the training corpus, 46 semantic categories were obtained. Learning by application of the 46 semantic categories was performed in the same manner as that described in Section 3.2. The disambiguating ambiguous contextual features for transliteration of ANEs were improved in accuracy compared to the previous results, as shown in [Table 6].

Table 6. Comparison of WSD Accuracies

	Baseline	WSD based on local context	Application of KorLex 1.0
Accuracy	75.8%	97.3%	97.9%

The results show that the hybrid method based on the RFA-tagged corpus and KorLex1.0 performs better than the method based on the tagged corpus only or the application of a synonymy dictionary. The hybrid model yields 97.9% accuracy, which is 22.1% higher than that of the baseline determined by MFC.

In order to produce more reliable evidence on the performance of the proposed model, experimentation was performed with our model and a customized TTS system. In the case of the TTS system, the accuracy of the pronunciation of generated ANEs was measured, whereas in the case of our classification model, the accuracy of RFA selection was measured. The hold-out data was used as test data. The result, listed in [Table 7], shows that our proposed model outperforms the VoiceWare system.

Table 7. Comparison of Performance Between VoiceWare TTs System and Proposed Model

	VoiceWare system	Proposed model
Accuracy (%)	87.8	97.29

However, several problems remain. First, disambiguation of more than three homographic or polysemic contextual features based on the Least Upper Bound algorithm does not work sufficiently well. For example, many synsets having multi-parents make it difficult for the word to be assigned to a single semantic category in KorLex. In order to resolve this problem, a complementary scoring algorithm to select the correct sense among other senses should be included.

Second, automatic and appropriate positioning of the LUB in the appropriate level of the KorLex hierarchy when word in a cluster do not have a common hypernym, should be resolved.

Third, since KorLex 1.0 based on Prince WordNet has not been completely constructed, numerous Korean word or concepts that do not exist in WordNet are missing in KorLex1.0 as well.

5 Conclusions and Future Work

In this paper, the ambiguities of Arabic Numeral Expressions were analyzed, and the resolutions for their sense disambiguation based on an RFA-tagged corpus and KorLex 1.0 (Korean Lexical database) were proposed. For the purpose of analyzing and extracting learning features, the corpus was compiled of news articles from 10 major newspapers in Korea. By learning the three phases of learning elements, the system yielded 97.3% accuracy in the transliteration of Arabic numerals.

Nouns preceding or following ANEs were re-categorized into 46 semantic classes based on the lexical hierarchy in KorLex. Nouns labeled with semantic class(es) were trained to determine the meaning and the reading of the ANEs using the C4.5 algorithm. The application of KorLex for WSD improved the performance: it yielded a 97.9% accuracy, 22.1% higher than that of the baseline. The experimentation results show that the proposed classification model outperforms the current Korean TTS system. As future work, using WSD for ambiguous contextual features by adopting a scoring algorithm should be continued. Since KorLex1.0 has not yet been

completely constructed, continuous studies on WSD for other applications with the refined KorLex are promising.

Acknowledgement

This work was supported by Korea Science and Engineering Foundation's support program: R05-2004-000-12584-0.

References

1. Agirre, E. et al, (1996), "Word Sense Disambiguation using Conceptual Density", *Proceedings of the 16th International Conference on Computational Linguistics (COLING'96)*, pp.16~22.
2. Castillo, M. et al. (2004), "Automatic Assignment of Domain Labels to WordNet", *Proceeding of the 2nd International WordNet Conference*, pp.75~82
3. Jung, Y. I. (2004), *Implementation of an Automatic Transliteration System of Arabic Numerals for Korean TTS*, Master's thesis, Pusan National University.
4. Kim, J. S. et al.(2003), "Disambiguation model of Homographs based on Statistic using Weight", *Korean Information Science: Softwares and Applications*, Vol.30, No.11, pp.1112~1123.
5. Leacock, C. et al (1998), "Combining Local Context and WordNet Similarity for Word Sense Identification", *WordNet - An electronic lexical database*, MIT Press: Cambridge, pp. 265-283.
6. Manning, C. D. et al (2001), *Foundations of Statistical Natural Language Processing*, MIT Press, Cambridge, Massachusetts.
7. Fellbaum, C. ed. (1998), *WordNet - An electronic lexical database*, MIT Press: Cambridge
8. Quinlan, J. R. (1993), *C4.5: programs for machine learning*, Morgan Kaufmann Publishers, San Mateo, Calif.
9. Sproat, R. et al (2001), "Normalization of Non-Standard Word," *Computer Speech and Language*, Vol. 15(3), pp.287-333.
10. Tetschner, W. "Text-to-Speech - Naturalness and Accuracy", *ASR News*, July, 2003 <http://www.asrnews.com/ttsap/ttsap11.htm> (referred to on June 7, 2004).
11. Witten, I. H., et al. (1999) *Data Mining*, Morgan Kaufmann Publishers: San Diego
12. Yarowsky, D. (1997), "Homograph Disambiguation in Text-to-speech Synthesis", *Progress in Speech Synthesis*, pp.159~174, New York: Springer
13. Yoon, A. S. et al. (2003) "An Automatic Transcription System for Arabic Numerals in Korean", *Proceedings of 2003 International Conference on Natural Language Processing and Knowledge Engineering*, pp. 221~226.
14. Yoon, A. S. et al.(2004) "Automatic Transcription of Three Ambiguous Symbols Used with Arabic Numerals: Period, Colon and Slash ", *Language and Information*, Vol. 8, Jun. 2004, pp. 117-136.
15. Yu, M. S. et al.(2003), "Disambiguating the senses of non-text symbols for Mandarin TTS systems with a three-layer classifier", *Speech communication*, Vol.39 no.3/4, pp.191-229
Learning Tool: Weka 3 : <http://www.cs.waikato.ac.nz/ml/weka/>

MFCRank: A Web Ranking Algorithm Based on Correlation of Multiple Features

Yunming Ye¹, Yan Li¹, Xiaofei Xu¹, Joshua Huang², and Xiaojun Chen¹

¹ Shenzhen Graduate School, Harbin Institute of Technology,
Shenzhen 518055, China
yym_sjtu@yahoo.com.cn

² E-Business Technology Institute, The University of Hong Kong, Hong Kong
jhuang@eti.hku.hk

Abstract. This paper presents a new ranking algorithm *MFCRank* for topic-specific Web search systems. The basic idea is to correlate two types of similarity information into a unified link analysis model so that the rich content and link features in Web collections can be exploited efficiently to improve the ranking performance. First, a new surfer model *JBC* is proposed, under which the topic similarity information among neighborhood pages is used to weigh the jumping probability of the surfer and to direct the surfing activities. Secondly, as *JBC* surfer model is still query-independent, a correlation between the query and *JBC* is essential. This is implemented by the definition of *MFCRank* score, which is the linear combination of *JBC* score and the similarity value between the query and the matched pages. Through the two correlation steps, the features contained in the plain text, link structure, anchor text and user query can be smoothly correlated in one single ranking model. Ranking experiments have been carried out on a set of topic-specific Web page collections. Experimental results showed that our algorithm gained great improvement with regard to the ranking precision.

Keywords: Ranking, Search Engine, Link Analysis, *PageRank*, Web.

1 Introduction

The enormous volume of the Web presents a big challenge to Web search, as there are always too many results returned for specific queries, and going through the entire results to find the desired information is very time-consuming for the user. To improve the information retrieval efficiency, Web search engines need to employ a suitable page ranking strategy to correctly rank the search results so that the most relevant (or important) pages will be included in the top list of the search results.

In traditional information retrieval, ranking measures, such as $TF*IDF$ [1], usually rely on the text features alone to rate plain text documents. This strategy can give poor results on the Web, due to the fact that the indexed Web document collection is so enormous and diverse that the text alone is not selective enough to limit the number of search results to a manageable size. An

important characteristic that differentiates Web ranking from traditional ranking is that the former offers more features to be exploited. Besides plain text features, HTML tags, anchor text, hyperlinks among pages and meta data, all provide rich information for Web ranking. Effectively exploiting these features is critical for the success of any ranking strategy. In recent years, various link-based ranking methods have been developed to exploit hyperlink information for improving the search results. Among them, *PageRank* [2, 3] and *HITS* [4] are the two best-known algorithms. It has been testified that proper utilization of link information is very helpful for Web search, where the success of *PageRank* in *Google*'s search engine is one well-known example.

However, the initial *PageRank*-like algorithms purely depend on the link structure information to rank the search results, and can't effectively integrate the multiple features of the Web pages. Thus, they are not robust enough and suffer from various topic drift problems [5]. Recently, integrating the text features with link structure features for Web ranking has been a very active research topic. Several algorithms have been proposed, including Richardson's query-dependent *PageRank* [6], Haveliwala's topic-sensitive *PageRank* [7], the personalized *PageRank* [8], and similarity ranking method for queries of 'related pages' [9, 10]. When combining the content features with link information, these previous approaches mainly focused on utilizing the similarity relationship between the user query and the retrieved pages (text features), while the topic similarity information among *neighborhood pages*¹ has not been used in computing the rank scores of indexed pages. We argue that to improve the accuracy of ranking algorithms, the topical similarity information among neighborhood pages should be consolidated into the link analysis model, because this similarity information can be a good measurement in computing the rank score for a given page. This is similar to a real-world scenario: when evaluating a man, the opinions from the people with more similar background will be more valuable than those from irrelevant communities. Based on this intuition, we develop a new Web ranking algorithm, *MFCRank*², which can effectively combine both the similarity information among neighborhood pages and the query similarity information into one ranking model.

MFCRank is based on the correlation of multiple features in a Web document collection. The ranking algorithm consists of two correlation steps: (1) First, similar to the random surfer model in *PageRank*, we propose a new surfer model, i.e. *JBC* surfer model, which uses the similarity information of neighborhood pages to weigh the jumping probability of the surfer. The surfer in the new model is not 'random' any more, but directed by the neighborhood similarity information, and therefore the first step is the correlation between the text features of pages with the link structure features. (2) However, *JBC* surfer model is still query-independent as *PageRank*, therefore a correlation between the query and the *JBC* surfer model is essential. This is implemented by the definition of *MFCRank* score, which is the linear combination of *JBC* score and the similarity

¹ Two pages are neighbors to each other if they are connected by at least one hyperlink.

² *MFCRank* stands for 'Multiple Features Correlation Ranking'.

value between the query and the matched pages. Through the two correlation steps, the features contained in plain text, link structure, anchor text and the query are combined in a single ranking model.

We have implemented *MFCRank* in a topic-specific Web search platform. Ranking experiments were carried out on a set of topic-specific Web page collections. Experimental results show that the *MFCRank* algorithm gains great improvement w.r.t. the ranking precision.

The rest of this paper is organized as follows. In Section 2, we discuss the random surfer model in *PageRank*. Section 3 describes the *MFCRank* algorithm, including the *JBC* surfer model and the definition of *MFCRank* score. Experimental results and analysis are presented in Section 4. In Section 5 we draw the conclusions and points out some avenues for future work.

2 Random Surfer Model in *PageRank*

The Web is logically a directed graph $G=(\mathbf{V}, \mathbf{E})$, where \mathbf{V} is a set of nodes representing pages and \mathbf{E} is a set of directed edges representing hyperlinks. Assume that the Web graph is strongly connected, that is, from any node u there is a directed path to another node v . Imagine a Web surfer starting from a random page, clicking the hyperlinks on pages forever, and picking a link on a page at random to move on to the next page. Occasionally, the surfer will not follow the hyperlinks on the page (or when a page has no out-links), but jump to a random page with some small probability ε . In this random surfer model, the probability that the surfer visit some page (node) d_i at one point of time can be defined as:

$$P(d_i) = \frac{\varepsilon}{|\mathbf{V}|} + (1 - \varepsilon) * \sum_{d_j \in B(d_i)} \frac{P(d_j)}{|F(d_j)|} \quad (1)$$

where $|\mathbf{V}|$ is the total number of the nodes in \mathbf{V} , $B(d_i)$ is the set of nodes linking to node d_i , that is, $B(d_i)$ and d_i are neighborhood pages. $|F(d_j)|$ denotes the total number of the nodes d_j links to. The probability $P(d_i)$ is the *PageRank* score for page d_i , and formula (1) defines the page ranking strategy in *PageRank*. Pages with greater *PageRank* score will get higher ranks in search results.

In fact, *PageRank* is query-independent. The *PageRank* score is assigned to each page independent of a specific user query. At query time, this score is used with or without some query-dependent ranking criteria to rank all pages matching the query. The *PageRank* score is a measure for distinguishing important (high-quality) pages from unimportant (low-quality) pages, and its computation is completely based on the link structure information without considering the content of pages. However, important pages may not be relevant. An elegant ranking algorithm should give high rank scores to pages with both high relevance and great importance. This presents two requirements for more effective ranking strategies:

- (1) First, the content information in pages should be combined with link information in the definition of rank score in order to improve the scoring accuracy and robustness;
- (2) Secondly, the rank score should also be smoothly correlated with user query, so that it is query-dependent.

The development of *MFCRank* algorithm follows the two requirements.

3 Web Ranking Based on Multi-feature Correlation

There are two correlation steps in *MFCRank* algorithm. In the first step, through a new surfer model, *JBC*, the content similarity information among neighborhood pages is correlated with link information to define the query-independent rank score, i.e. *JBC* score. The second step is the definition of query-dependent *MFCRank* score, which combines the *JBC* score with the similarity value between user query and the matched pages.

3.1 *JBC* Surfer Model

Similar to the PageRank algorithm, *MFCRank* defines a query-independent rank score function based on a surfer model, i.e. *JBC* (**J**umping-**B**ased on **C**ontent) model. The basic idea of *JBC* can be described as follow. Similar to the random surfer, the *JBC* surfer starts from a random page and clicks the hyperlinks on the visited pages constantly, however, unlike the random surfer, when picking a link on a page to follow, the *JBC* surfer is not at random, but tend to choose preferentially the links of which the corresponding pages (the child pages) have higher similarity to the page being visited (the parent pages). That is, the jump probabilities from one page to a linked page are weighted based on the similarities between the parent page and the neighborhood child pages. The intuition captured by this idea is the following: when surfing the Web, it is more likely that the surfer will focus on some topic and tend to follow similar pages over a period of time, but after some time he may jump to another topic with some probability. This idea is encoded in the definition of *JBC* rank score as follows:

$$F_{JBC}(d_i) = \frac{\varepsilon}{|V|} + (1 - \varepsilon) * \sum_{d_j \in B(d_i)} \lambda_{ji} \cdot F_{JBC}(d_j) \quad (2)$$

the definition is similar to formula (1), where $F_{JBC}(d_i)$ is the *JBC* rank score for page d_i , λ_{ji} represents the jumping probability from the page d_j to the page d_i (that is d_i and d_j are neighbors to each other), which is computed according to the similarity scores between neighborhood pages as:

$$\lambda_{ji} = \frac{Sim(d_j, d_i) + \sigma}{\sum_{d_k \in B(d_k)} (Sim(d_j, d_k) + \sigma)} \quad (3)$$

where $Sim(d_j, d_i)$ is the similarity of the page d_j to the page d_i . σ is a small positive value acting as a normalization factor. The value $Sim(d_j, d_i)$ can be

computed as the similarity of the original text features in the two pages, or by concatenating their anchor texts as the virtual pages to calculate the degree of similarity. In our experiments, we use the traditional $tf * idf$ scheme [1] as the term weighting measure to compute the similarity value.

The *JBC* rank score makes a smooth tradeoff between the relevance measure and importance measure through the correlation of content features with link features. It will be more robust to tackle the topic drift problem. The following gives an illustration.

3.1.1 An Illustration of *JBC* Ranking

Fig.1 shows a sample Web graph for computing rank scores, and the corresponding similarity matrix for connected nodes. In this graph, nodes *A, B, C* are in the same topic, i.e., topic 1, while topic 2 includes the nodes *E, F, G*. The node *D* is a popular page with many in-links, such as the Yahoo homepage. Although page *D* doesn't focus on specific topic, it may have some keywords which appear in some topics (such as the topic 1). Therefore pages like *D* will often be included in the result lists for many topic-specific user queries.

Now assume that the user query is on topic 1, and the matched pages include page *A, B, C* and *D*. The matched list is ranked according to their rank scores. Table 1 shows resulting rank scores for all nodes in the given Web graph, which are computed according to formula (1) and (2) (During computation, the damping factor ϵ is set to 0.5). In *PageRank* scoring, the rank score of page *D* is the highest, as *PageRank* score is defined to bias for the strongly connected pages (namely important pages). Therefore, page *D* will be ranked as the No.1 in the top list, although it is not relevant to topic 1, which is a typical topic drift problem. The problem is solved in the *JBC* scheme, as shown in Table

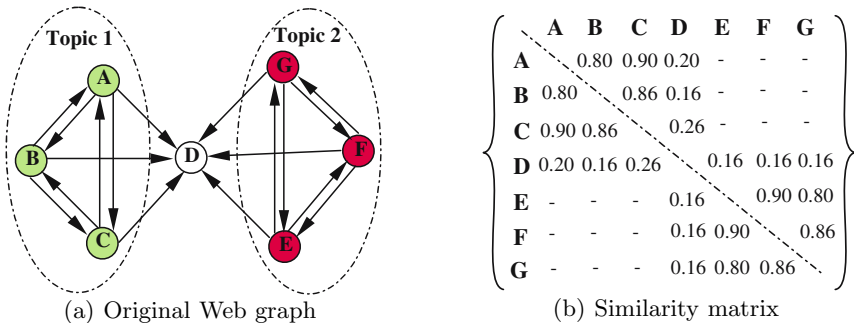


Fig. 1. A Web graph and its similarity matrix for computing rank scores

Table 1. Ranking scores of the nodes in the sample Web graph

Algorithm	A	B	C	D	E	F	G
PageRank	0.12901	0.12901	0.12901	0.22597	0.12901	0.12901	0.12901
JBC Model	0.14438	0.14196	0.14814	0.12119	0.14776	0.15076	0.14580

1, the rank score of page D becomes lower than those of pages A, B, C which are relevant to the user query. This is due to the correlation of topic (content) locality [11] information with link information in the JBC surfer model.

This example implies that to determine the rank score for a given page, more authoritative pages (experts), that is, those with greater similarity to the given page (such as on the same topic), should play more important roles (i.e. contributing more weights according to formula (2)). It is clear that JBC surfer model considers both the link structure information and the topical relevance information, which makes it more robust in dealing with topic drift problems.

3.2 The Definition of MFCRank Score

JBC ranking scheme is still query-independent as PageRank, which may bring up another topic drift problem. For Fig.1, assume the user query is still about topic 1, and unfortunately the pages in topic 2 are also included in the matched result list. According to JBC ranking, the irrelevant page F will get higher rank score than the relevant pages A, B, C . This problem is because that the definition of JBC rank score is query-independent and biased for the pages of the topics with stronger topical locality [11], which we call 'topical winner-take-all effect'. Since the simple keyword matching is not selective enough to filter out the irrelevant pages w.r.t the user query, a more effective query analysis technique should be developed to aid the JBC ranking scheme. The definition of $MFCRank$ score is under this motivation.

The $MFCRank$ score is the rank score defined in $MFCRank$ algorithm. It is the linear combination of JBC score and the similarity value between the query and the matched pages, defined as follows:

$$F_{MFC}(d_i^Q) = (1 - \mu)F_{JBC}(d_i) + \mu \cdot Sim(d_i, Q) \cdot F_{JBC}(d_i) \quad (4)$$

where $F_{MFC}(d_i^Q)$ is the $MFCRank$ score for page d_i w.r.t the user query Q , $Sim(d_i, Q)$ is the similarity of the page d_i to the user query calculated through $tf*idf$ scheme, μ is a bias factor between the query-independent JBC rank score and the query similarity score. Previous works have shown that the anchor texts of Web page are very informative and descriptive for the original page [12, 13]. To accelerate the ranking process, when computing the value $Sim(d_i, Q)$, we didn't use the original content of the pages, but concatenated the anchor texts of pages to construct the virtual pages, and $Sim(d_i, Q)$ was computed as the similarity between the virtual page of d_i and the user query Q .

Through the definition of $MFCRank$ score, the $MFCRank$ ranking strategy becomes query-dependent. The online query analysis results is closely correlated with the offline link analysis results, which will make the ranking strategy more accurate and robust.

3.3 Computation Scalability

The computation cost in $MFCRank$ includes two parts. First is the offline pre-computation of the JBC score vector for the pages in the indexed document

collection. This is similar to the iterative computation of *PageRank* score vector in *PageRank* algorithm, which has been proved to be scalable in practical use [2]. An overhead in *JBC* is that the similarity matrix of the indexed pages should be pre-computed before computing the *JBC* score vector. Assume the number of the indexed pages is n , if there is a hyperlink between any two pages, i.e. any two pages are neighbors, it will need $n^2/2$ times to calculate the similarity between two pages. This will take enormous computation when n is a large value because computing similarity of pages is very costly. Fortunately, in practice each page always has a very limited neighbor pages, and the average number of neighbors for each page is a small constant k (such as 11). Therefore, computing the similarity matrix will cost $k \cdot n$ times of the similarity computation of two pages, which is scalable to very large page collections.

The second part of computation is to calculate the *MFCRank* scores for each matched page w.r.t. the user query, which is performed online. The main time cost of this part is the computation of the similarity between the user query and the virtual documents of the matched pages (anchor text concatenation), which is similar to the computation in traditional *tf * idf* ranking. We have developed a fast anchor text index to speed up the retrieval of anchor text for computing the query similarity. Although there is some overhead for online computation, it is affordable for practical applications.

4 Experiments

4.1 Experiment Setup

MFCRank has been implemented in a topic-specific Web search platform-*TopSearch* [14], which is a scalable and configurable platform for building topic-specific search systems. Experimental study was performed on this platform. In *TopSearch*, a focused crawling system *iSurfer* [15] was employed to collect Web pages of specific topics from the Web. The crawled pages were used to build the topic-specific page collections for the search experiments. Each collection has its own topic (such as '*Chinese history*'), therefore it can be regarded as a search engine on a certain topic (such as a topic-specific search engine on '*Chinese history*').

For each topic-specific collection, we built inverted full-text index and other auxiliary indexes, such as anchor text index, before doing search experiments. A fast link graph data structure was constructed on each page collection as well, upon which the rank score vectors for evaluated ranking algorithms (*PageRank* and *MFCRank*) were computed. As the design of *TopSearch* has carefully considered the encoding and language problems, it can process both English and Chinese Web pages efficiently. This feature greatly facilitates our multi-language experiments.

In traditional information retrieval research, the precision and recall are the main evaluation metrics. However, it is difficult to get the recall for Web search, as measuring the size of relevance set from a large Web collection is almost impossible. We employ the precision in the top K list of the search results as the

main performance evaluation metric. Typical values for K include 10, 30, and 50, which represent the search results user may pay attention to. The precision will be simply calculated as the percentage of relevant pages for all the experiments.

4.2 Experimental Results on Topic-Specific Collections

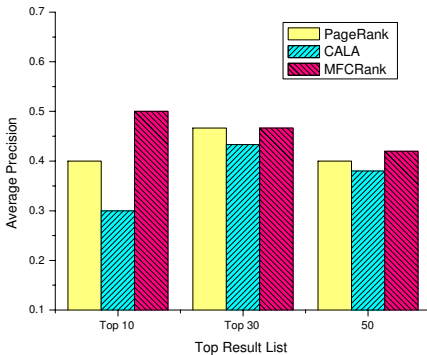
We built a set of topic-specific collections, each of which consists of tens of thousands pages crawled by *iSurfer*. For each collection, we used a list of user queries (about 60 queries) to search the corresponding collection. Table 2 shows the size of the collections and the number of corresponding user queries performed.

Three ranking algorithms were implemented for performance comparison: (1) the original *PageRank* algorithm, (2) ranking algorithm *CALA* [16], which defines the rank score based on the linear combination between *PageRank* and online anchor text analysis, and (3) the *MFCRank* algorithm. In previous work, **CALA** has been testified experimentally to have higher precisions than *PageRank* in general Web search [16].

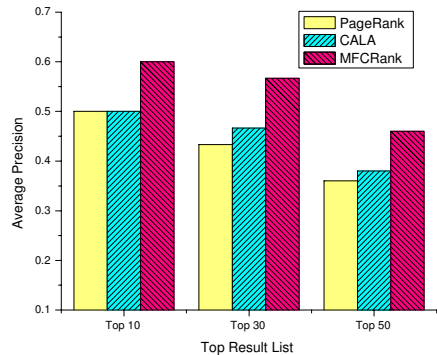
Fig.2 shows the query results on the English collections 'American History' and 'American African History'. Fig.3 presents the results on the Chinese collections 'Travel in China' and 'Travel in Beijing' (the user queries are in Chinese). For each collection, we compute the average precision of the top 10, top 30, and top 50 result lists for all user queries.

Table 2. A statistics of the user queries performed

Topic of The Collection	Number of Pages	Number of Queries
American History	251,820	62
American African History	82,218	60
Travel in China	321,336	65
Travel in Beijing	182,215	60



(a) Collection: 'American History'



(b) Collection: 'American African History'

Fig. 2. The search results on two English collections

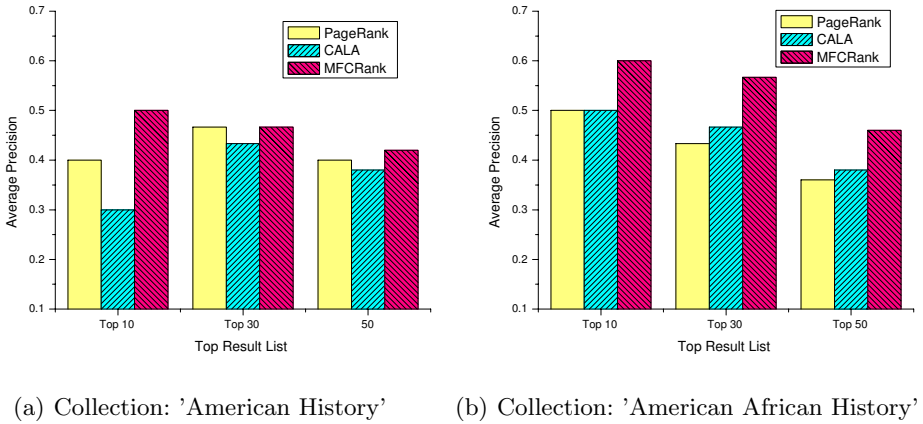


Fig. 3. The search results on two Chinese collections

In most of the queries, *CALA* and *MFCRank* outperformed *PageRank*. This testified that combining online query analysis information into link analysis model is very useful for improving the ranking performance. The more important observation is that *MFCRank* got precisions higher than *CALA* persistently, which demonstrate the effectiveness of the *JBC* surfer model w.r.t the random surfer model also used in *CALA*. As relevance is a more important evaluation metric than the importance metric in topic-specific Web search, we believe that integrating content analysis techniques into the ranking strategy is very essential.

4.3 Tradeoff Between Online Content Analysis and Offline Link Analysis

When implementing the *MFCRank* in the search system, an interesting issue is to determine the value of the bias factor in formula (4). The factor μ means the tradeoff between online content analysis and offline link analysis in deciding the final rank scores for matched pages. Our assumption was that the optimal value depended on the 'topical broadness' of the topic-specific page collection searched, which should be set higher for narrow topics and lower for broad topics. To test this assumption, we have carried out some elementary experiments.

Fig.4 shows the results on two page collections with different topical broadness, where the collection 'Travel in China' has greater topical broadness than that of the collection 'Travel in Beijing'. We set μ to different values and prepared about 100 queries to search the collections. The precision of the search results (the top 50 in our experiments) was calculated w.r.t the value of μ . As shown in Fig.4, to get optimal search performance, the factor should be set near 0.4 for the collection 'Travel in China', and 0.6 for the collection 'Travel in Beijing'. The elementary results demonstrated our initial assumption. This observation means that: for the collections with narrower topics, the online content analysis will

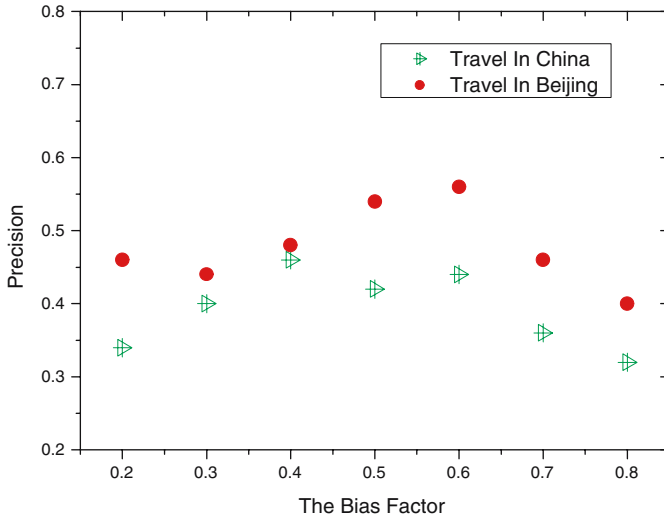


Fig. 4. The results of the experiments on the bias factor μ

play a more important role in page ranking as the link graphs of the collections are usually too dense and provide less discriminative information for ranking algorithm. On the contrary, link analysis is more important for the collections with greater topical broadness.

5 Conclusions

In this paper we have proposed a new ranking algorithm *MFCRank* for topic-specific Web search engines. Its intuition is to correlate two types of similarity information in a unified link analysis model so that the rich content and link features in Web collections can be exploited efficiently to improve the ranking performance. First, a new surfer model *JBC* is proposed, under which the topic similarity information between neighborhood pages is used to weigh the transition probability of the surfer. Secondly, a linear correlation between the query analysis and *JBC* model is designed to endow the ranking algorithm with query-dependent capability. We implemented *MFCRank* in a topic-specific Web search platform. Ranking experiments have been carried out on some topic-specific collections. Experimental results showed that the *MFCRank* algorithm gained better ranking precisions.

In the future, we will use more refined link context analysis methods to improve the stability of the online query analysis, such as by employing the word disambiguation technique. Moreover, the topical characteristics in topic-specific search engine should be studied further to clarify the relationships between the topical regularities and the ranking performance.

References

1. Baeza-Yates, R., Ribeiro-Neto: *Modern Information Retrieval*. ACM Press Series/Addison Wesley, New York (1999)
2. Page, L.: *Pagerank: Bring order to the web*. In: *Stanford Digital Libraries Working Paper*. (1997)
3. Brin, S., Page, L.: *The anatomy of a large-scale hypertextual web search engine*. *Computer Networks and ISDN Systems* **30** (1998) 107–117
4. Kleinberg, J.: *Authoritative sources in a hyperlinked environment*. In: *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*. (1998) 668–677
5. Chakrabarti, S., Dom, B., Raghavan, P.: *Automatic resource compilation by analyzing hyperlink structure and associated text*. In: *Proceedings of the 7th International WWW Conference*. (1998)
6. Richardson, M., Domingos, P.: *The intelligent surfer: Probabilistic combination of link and content information in pagerank*. In: *Advances in Neural Information Processing Systems 14*. (2002)
7. H., H.T.: *Topic-sensitive pagerank*. In: *Proceedings of the 11th International WWW Conference*. (2002)
8. Jeh, G., Widom, J.: *Scaling personalized web search*. In: *Proceedings of the 12th International WWW Conference*. (2003)
9. Jeh, G., Widom, J.: *Simrank: A measure of structural-context similarity*. In: *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. (2002)
10. Fogaras, D., Racz, B.: *Scaling link-based similarity search*. In: *Proceedings of the 14th International WWW Conference*. (2005)
11. Brian, D.D.: *Topical locality in the web*. In: *Proceedings of the 23rd Annual International Conference on Research and Development in Information Retrieval (SIGIR 2000)*. (2002)
12. Kraft, R., Zien, J.: *Mining anchor text for query refinement*. In: *Proceedings of the 13th International WWW Conference*. (2004)
13. Eiron, N., McCurley, K.S.: *Analysis of anchor text for web search*. In: *Proceedings of the 26th Annual International ACM SIGIR'03 Conference on Research and Development in Information Retrieval*. (2003)
14. Ye, Y., C.L., X., L.L., Z.: *The anatomy of a scalable topic-specific web search platform*. In: *Technical report in ICE research center*. (2005)
15. Ye, Y., Ma, F., Lu, Y., Chiu, M., Huang, J.: *isurfer: A focused web crawler based on incremental learning from positive samples*. In: *Proceedings of the 6th Asia-Pacific Web Conference*. (2004)
16. Zhang, L., F.Y., M., Ye, Y.: *Cala: A web analysis algorithm combined with content correlation analysis method*. *Journal of Computer Science and Technology* **18** (2003) 21–25

On Text Ranking for Information Retrieval Based on Degree of Preference

Bo-Yeong Kang¹ and Dae-Won Kim^{2,*}

¹ Center of Healthcare Ontology R&D, Seoul National University,
Yeoncheon-dong, Jongro-gu, Seoul, Korea

² School of Computer Science and Engineering, Chung-Ang University,
221 Heukseok-dong, Dongjak-gu, Seoul, Korea
dwkim@cau.ac.kr

Abstract. A great deal of research has been made to model the vagueness and uncertainty in information retrieval. One such research is fuzzy ranking models, which have been showing their superior performance in handling the uncertainty involved in the retrieval process. However, these conventional fuzzy ranking models are limited to incorporate the user preference when calculating the rank of documents. To address this issue, we develop a new fuzzy ranking model based on the user preference.

1 Introduction

In recent years a great deal of research in information retrieval has aimed at modelling the vagueness and uncertainty which invariably characterize the management of information. The application of fuzzy set theory to IR have concerned the representation of documents and the query [1], and many fuzzy ranking models such as MMM, PAICE, and P-NORM have been showing their superior performance in handling the uncertainty in the retrieval process [2, 3, 4]. The ranking is achieved by calculating a similarity between two fuzzy sets, a document D and a query Q . However, in spite that the user has an ability to reflect their preference for the information need in searching, these conventional models are limited to incorporate the user preference when calculating the rank of documents. Let us suppose that we are given a vector of query Q with a fuzzy set of the term and its membership degree:

$$Q = \{fuzzy(0.8), IR(0.7), korea(0.3), author(0.2)\}$$

A document collection consists of four documents (D_1, D_2, D_3, D_4) in which each document is represented as a fuzzy set of the index term and its weight.

$$\begin{aligned} D_1 &= \{fuzzy(0.8), IR(0.7)\} \\ D_2 &= \{fuzzy(0.2), IR(0.2), korea(0.3), author(0.2)\} \\ D_3 &= \{korea(0.7), IR(0.8)\} \\ D_4 &= \{fuzzy(0.8), IR(0.7), korea(0.3), author(0.2)\} \end{aligned}$$

* Corresponding author.

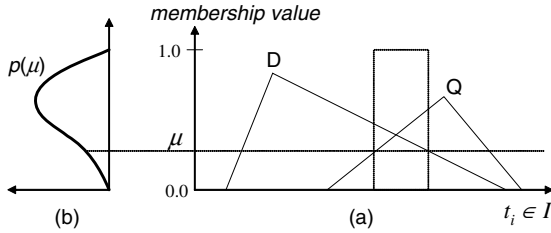


Fig. 1. Preference-based similarity computation: (a) overlap degree at μ_p between a document D and a query Q ; (b) a membership preference function

Given a query Q , we are wondering what is the best result of ranking? Intuitively, we know that D_4 is the most relevant document and D_3 is the least relevant. However, it is arguable to say which one of the two documents D_1 and D_2 has a higher rank. The rank of D_1 can be higher than that of D_2 because D_1 contains the highly-matched index terms (‘fuzzy’ and ‘system’). Conversely, D_2 can be more relevant than D_1 because the number of matched terms in D_2 is larger than those in D_1 . Such discrepancies arise because conventional ranking models are limited to resolve the uncertainty in a retrieval system. To solve the addressed problems, we develop a ranking model based on a similarity measure between fuzzy sets in which users assign their preference to the decision.

2 Ranking Texts with Preference Degree

Given index terms, a ranking model to calculate the similarity between a document and a query is required. In this study, each document is represented as a fuzzy set $D = \{(t_i, \mu_D(t_i))\}$ where t_i is an index term for $1 < i < n$ (=number of terms). $\mu_D(t_i)$ quantifies the degree to which D is characterized by each t_i . Firstly, we compute the degree of overlap between a document (D) and a query (Q). Given D and Q , we obtain the overlap between two fuzzy sets at each membership degree (μ) before computing the total overlap. The overlap function $f(\mu)$ at a membership degree μ between D and Q is defined as:

$$f(\mu : D, Q) = \sum_{i=1}^n \delta(t_i, \mu : D, Q) \tag{1}$$

where

$$\delta(t_i, \mu : D, Q) = \begin{cases} 1.0 & \text{if } \mu_D(t_i), \mu_Q(t_i) \geq \mu \\ 0.0 & \text{otherwise} \end{cases} \tag{2}$$

$\delta(\cdot)$ determines whether two sets are overlapped at the membership degree μ for t_i . It returns an overlap value of 1.0 when the membership degrees of the two sets are both greater than μ ; otherwise, it returns 0.0. Figure 1(a) depicts an overlap value $f(\mu)$ between two fuzzy sets. The index terms $t_i \in I$, satisfying

both $\mu_D(t_i) \geq \mu$ and $\mu_Q(t_i) \geq \mu$, are given a value 1.0 by Eq. 2. Based on $f(\cdot)$ and a preference function $p(\mu)$, a similarity between D and Q is defined as:

$$S(D, Q) = \sum_{\mu} f(\mu : D, Q)p(\mu) \quad (3)$$

$S(D, Q)$ is obtained by summing $f(\mu : D, Q)$ over the whole range of membership degrees. A larger value of $S(D, Q)$ means that D and Q are more similar to each other; D is more relevant to Q . Note that $p(\mu)$ is a preference function of membership, which is determined by users. When two ranking results that have different fuzzy sets yield the same degree of similarity, $p(\mu)$ discerns the two ranking results by focusing on the higher membership degrees. When users search the Web, they focus on the document with the terms of highest matching. Thus the relevance of the highest-matched document plays an important role in user satisfaction. In such cases, $p(\mu)$ is given a higher value when $\mu_D(t_i)$ is significant, i.e., $\mu_D(t_i) \geq 0.7$. Under this case, index terms with higher weights place greater emphasis on the calculation of $S(D, Q)$.

Consider the ranking example in the Introduction. For simplicity, let us suppose that $f(\mu : D, Q)$ is calculated at six μ values ($\mu = 0.0, 0.2, 0.4, 0.6, 0.8, 1.0$). Given that $p(\mu)$ is assigned a value of 1.0 if $\mu_D(t_i) \geq 0.6$ and 0.5 otherwise, the similarity $S(D_1, Q)$ is calculated as: $S(D_1, Q) = f(0.0)p(0.0) + f(0.2)p(0.2) + f(0.4)p(0.4) + \dots + f(1.0)p(1.0) = 2 \times 0.5 + 2 \times 0.5 + 2 \times 0.5 + 2 \times 1.0 + 1 \times 1.0 + 0 \times 1.0 = 6.0$. Similarly, we find that $S(D_2, Q) = 4.0$, $S(D_3, Q) = 2.0$, and $S(D_4, Q) = 8.0$. It is clear that D_4 is the most relevant to Q , and D_3 is the least relevant. Note that D_1 has a higher rank than D_2 even though the number of matched terms of D_1 is smaller than those of D_2 . The resolution of uncertainty is achieved by assigning greater preference on the terms of higher membership degrees.

3 Experiments and Conclusion

We conducted retrieval tests in which the proposed ranking method was compared with the PAICE, P-NORM, and vector model in the normalized TF \times IDF index. The retrieval results were assessed by the precision and recall. The data was the TREC-2 Wall Street Journal 21,705 texts; the built-in 40 queries were used for judgement. The preference $p(\mu)$ is given a value of 1.0 if $\mu_D(t_i) \geq \mu_p$ and 0.1 otherwise; $\mu_p \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$ is a preference threshold.

Firstly, we analyze the dependence of the search performance of the proposed method on the choice of $p(\mu)$, specifically the preference threshold μ_p . Table 1 lists the search results of the proposed ranking model, average precision and recall ranging from Top 1 to Top 10 documents for 40 queries varying the preference threshold μ_p . The best precision is obtained for $\mu_p = 0.5$. Overall, the precision and recall values at $\mu_p \geq 0.5$ are better than those at $\mu_p < 0.5$.

As a second experiment, the search result obtained by the proposed method using the preference function with $\mu_p = 0.5$ was compared with the search result obtained using the PAICE, P-NORM and vector model. Figure 2 shows the

Table 1. Precision and recall (%) of the proposed model for $\mu_p \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$

Top N	Precision					Recall				
	0.1	0.3	0.5	0.7	0.9	0.1	0.3	0.5	0.7	0.9
Top 1	22.5	25.0	35.0	28.0	27.5	1.4	1.7	3.2	1.9	1.9
Top 3	21.7	17.5	25.8	27.5	27.5	4.3	3.4	6.3	7.0	7.0
Top 5	19.5	18.5	25.0	25.5	25.5	6.2	5.9	10.8	12.1	12.1
Top 7	16.8	17.1	22.5	22.5	22.5	10.7	9.8	12.7	13.9	13.9
Top 9	15.3	15.3	21.4	20.6	20.8	12.5	10.4	16.7	15.3	15.7
Avg.	19.1	18.7	25.9	24.7	24.8	7.0	6.2	9.9	10.1	10.1

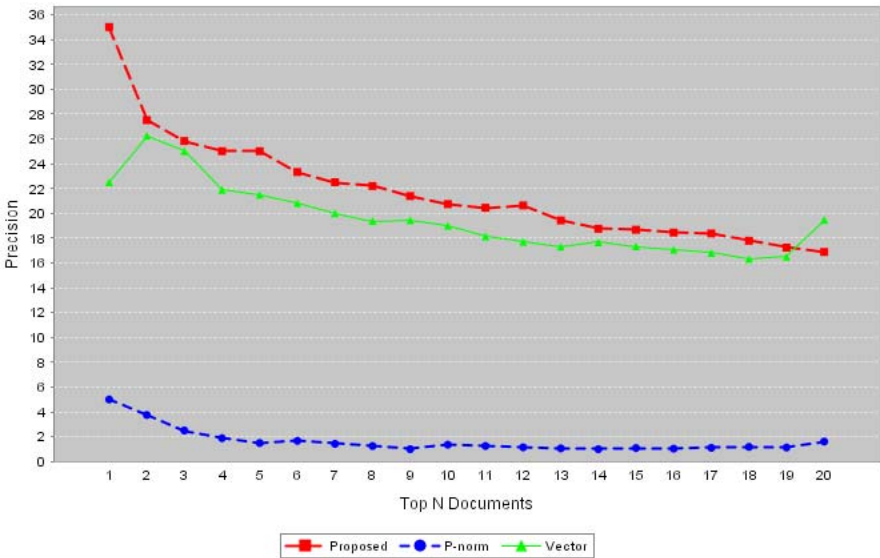


Fig. 2. Comparison of average precision of each model for 40 queries

search results of each ranking model, P-NORM, vector and the proposed. The P-NORM and vector models give average precisions of 1.62% and 19.48% respectively. The search performance of the PAICE model was much similar with that of the P-NORM. In contrast, the proposed model gives the higher average precision of 21.8%. Moreover, we see that the average precision of the proposed ranking model for the Top-ranked document (35.0%) is remarkably higher than those of the other two models. Similarly, the proposed model gave a higher average recall than the P-NORM and vector models. From these tests, we see that the proposed method provides more clear similarity calculation between a document and a query by allowing users to assign their preference or intention to the weights of terms.

References

1. M.J. Martín-Baustista, D.H.Kraft, M.A.Vila, J.Chen, J.Cruz, User profiles and fuzzy logic for web retrieval issues, *Soft Computing*, Vol. 6, 2002, 365–372.
2. J.H. Lee, On the evaluation of Boolean operators in the extended boolean retrieval framework, *Proceedings of the 17th SIGIR conference*, 1994, 182–190.
3. R. Baeza-Yates, et al., *Modern information retrieval*, Addison-Wesley, 1999.
4. J. Fan, W. Xie, Some notes on similarity measure and proximity measure, *Fuzzy Sets and Systems*, Vol.101, 1999, 403–412.

Lexical Normalization and Relationship Alternatives for a Term Dependence Model in Information Retrieval

Marco Gonzalez¹, Vera Lúcia Strube de Lima¹, and José Valdeni de Lima²

¹ PUCRS - Faculdade de Informática,
Av. Ipiranga, 6681 – Prédio 16 – PPGCC, 90619-900 Porto Alegre, Brazil
{gonzalez, vera}@inf.pucrs.br

² UFRGS – Instituto de Informática,
Av. Bento Gonçalves, 9500, 91501-970 Porto Alegre, Brazil
valdeni@inf.ufrgs.br

Abstract. We analyze alternative strategies for lexical normalization and term relationship identification for a dependence structured indexing system [14], in the probabilistic retrieval approach. This system uses a dependence parse tree and Chow expansion [5]. Stemming, lemmatizing, and nominalization processes are tested as lexical normalization, while head-modifier pairs and binary lexical relations are tested as term relationships. We demonstrate that our proposal, binary lexical relations with nominalized terms for Portuguese, contributes to the performance improvement in information retrieval.

1 Introduction

Many information retrieval (IR) systems are based on the assumption that each term is statistically independent of all other terms in the text. Those systems have been developed because this independence leads to a formal representation of the probabilistic approach more easily. But, the independence assumption is understood to be inconsistent [6] and there are regularities provided by term dependences that need to be considered [16].

Some models have been proposed to incorporate term dependence strategies (e.g., [19], [16]). However, the formal representation of the probabilistic approach cannot be easily maintained when there are no constraints for term relationships, i.e., when a higher order model of term dependence is applied. For reducing this problem, Rijsbergen [19] adopted the algorithm proposed by Chow and Liu [5] that uses a maximum spanning tree for incorporating term dependence into a probabilistic approach.

Adapting the Rijsbergen's strategy, Changki Lee and Gary Lee [14] presented a method for incorporating term dependence into the probabilistic retrieval approach using Chow expansion. They proposed a dependence structured indexing (DSI) system that avoids the problem of a high order model of term dependence. In a DSI system a term dependence model (TDM) is created using grammatical connections.

We have tested, in a DSI system, some alternative strategies (i) for identifying those connections (head-modifier pairs and our proposal, the binary lexical relations) and (ii) for generating terms that are nodes in those connections. Such terms may be

generated through some alternative processes for lexical normalization (stemming, lemmatizing, and our proposal, the nominalization process).

The rest of this paper is organized as follows. Section 2 presents works related to lexical normalization processes, term weighting, and TDMS, including DSI system. Section 3 introduces alternative strategies for lexical normalization processes and for identifying relationships between terms in the text. Section 4 describes data and methods adopted to evaluate the alternative strategies tested and their results, and Section 5 presents final considerations.

2 Related Work

The first three text operations in IR [2] are lexical analysis, elimination of stopwords, and stemming. This last operation performs morphological normalization for (i) reducing the number of terms and the index file size, and (ii) making retrieval independent from the specific word form used in the query [3]. Morphological normalization may be achieved through conflation. This process, at the lexical level, infers conceptual proximities from morphological similarities.

Stemming [7,18] is a process of lexical normalization which conflates words with morphological similarities into a common representation: the stem, i.e., the common part of those words. Stemming is the most usual term normalization procedure in IR. However, while simple stemmers are adequate for languages such as English, more sophisticated strategies are demanded for languages with complex inflectional morphology, like German, French, Finnish, Spanish, and Portuguese. Therefore, stemmers for such languages present higher computational cost [22].

Another usual term normalization procedure is lemmatizing [1,13]. The lemmatizing algorithm reduces the variant forms of a word to their canonical form (lemma), i.e., verbs to infinitive, and other words to their singular and (if it exists) masculine form.

A stem can be originated from words of different morphological classes. For instance, the noun “constructions” and the verb “constructed” have the same stem “construct”. An important difference between these two procedures (stemming and lemmatizing) is that lemmatized words maintain their original classes. For example, the noun “constructions” and the verb “constructed” have different lemmas: the noun “construction” and the verb “construct” respectively.

At the indexing phase, after the usual elimination of stopwords and the lexical normalization process, the next step is the term weighting. Given a document d , the eliteness [20,21] for a term t is usually inferred from the occurrence frequency of t in d . This inference is expressed through the weight of t in d .

This weight ($W_{t,d}$) may be computed by Okapi BM25 formula [20,21]:

$$W_{t,d} = \frac{f_{t,d}(k_1 + 1)}{k_1((1-b) + b \frac{DL_d}{AVDL}) + f_{t,d}} w_t \quad (1)$$

where:

$f_{t,d}$ is the occurrence frequency of t in d ,
 k_1 and b are parameters,

DL_d is the length of d ,
 $AVDL$ is the average document length in the collection, and
 w_t is the *IDF* (inverse document frequency) factor, which is given by:

$$w_t = \log \frac{N}{df_t} \quad (2)$$

where:

N is the total number of documents in the collection, and
 df_t is the number of documents containing term t .

Usual term weighting, mainly for the probabilistic approach, is based on the independence assumption. However, alternative models are proposed in the literature.

2.1 Term Dependence

In the probabilistic approach [21], applying the Bayes' Theorem, the probability that the document d is relevant to a query q is expressed by:

$$\Pr(rel \mid d) = \frac{\Pr(d \mid rel)\Pr(rel)}{\Pr(d)}$$

where rel is the event that d is relevant to q .

So, the probability $\Pr(d)$ must be estimated. Considering n index terms, when the statistical independence is assumed, the problem is for estimating only n probabilities. On the other hand, in a TDM, if we must estimate 2^n possibilities, an enormous task is need for processing such a higher order model.

For reducing this task Rijsbergen [19] adopted the algorithm proposed by Chow and Liu [5] – the Chow expansion. In this algorithm, a maximum spanning tree is based on the expected mutual information measure between the terms and considers the distribution of co-occurrences.

An alternative strategy is provided by finding an expansion for $\Pr(d)$ and approximating $\Pr(d)$ by partial sum [14], e.g., the Rademacher-Walsh expansion and the Bahadur-Lazarsfeld expansion (BLE). Losee [15] studied the performance of probabilistic IR systems where different statistical dependence assumptions are adopted. Using the BLE and examining the span of dependence in natural language text, Losee showed that the best performance was obtained when degree 3 and a window of ± 3 to ± 5 terms in width were used.

Cho, Lee, and Lee [4], through experiments in Korean and English, demonstrated that improvement of performance was obtained by incorporating the term dependence using the BLE with degree of 2 applied to Okapi BM25 formula.

On the other hand, Gao, Nie, Wu, and Cao [9] presented a method to identify term dependences using a grammar and an acyclic, planar, undirected graph. This structure limits the dependences to relationships identified by the grammar. This method was proposed in dependence language modeling approach to IR.

2.2 Adapting Chow Expansion Through Dependence Parse Tree

Changki Lee and Gary Lee [14] developed an adaptation of Chow expansion. They used a dependence parse tree with grammatical connections to include linguistic knowledge in a TDM.

In a dependence parse tree, a relationship between two nodes (parent and son) determines that the son node is dependent (or modifier) of the parent node. Such head-modifier pairs are presented in this work in the form

$$(t', t)$$

where t' is the head term, and t is the modifier term.

Figure 1 shows the dependence structure for “The train for Paris departed from Rome” where, discarding the stopwords, there are two pairs of dependence relationships depicted by arcs from heads to modifiers.

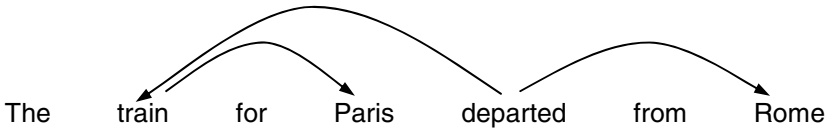


Fig. 1. A dependence structure of a sentence

Then, the head-modifier pairs for the phrase of Figure 1, used in a DSI system, are:

(train,paris), (depart,train), and (depart,rome).

If we take into account the dependence structure, considering that the modifier term t has the head term t' , w_t in equation (1) is substituted by $w_{t,d}$ for t in a document d . According to Changki Lee and Gary Lee [14], $w_{t,d}$ is given by:

$$w_{t,d} = x_t \log \frac{1}{q_t} + k_7 x_{t'} \log \frac{q_{t'} - q_{tt'}}{q_{t'}(1 - q_t)} + k_8 x_t x_{t'} \log \frac{q_t q_{t'}}{q_{tt'}} \tag{3}$$

where:

$x_i = 1$ if i (t' or t) appears in d , $x_i = 0$ otherwise,

k_7 and k_8 are parameters,

$$q_t = \frac{df_t}{N}, \quad q_{t'} = \frac{df_{t'}}{N}, \quad \text{and} \quad q_{tt'} = \frac{df_{tt'}}{N}, \quad \text{where:}$$

df_i is the number of documents in which i (t' or t) occurs,

$df_{tt'}$ is the number of documents in which t' and t occur as a head-modifier grammatical relation (i.e., t' is the parent of t), and

N is the total number of documents in the collection.

Changki Lee and Gary Lee [14] compared conventional Okapi BM25 formula to Chow expansion incorporating dependence parse tree in experiments for Korean and English. They demonstrated that their method improves the performance of the IR system.

3 Alternative Strategies

We present some alternative strategies for lexical normalization and term relationships identification for incorporating term dependence into a probabilistic retrieval approach using a DSI system for Portuguese.

As processes for lexical normalization, we have tested stemming, lemmatizing, and nominalization. The nominalization process is described in Section 3.1.

As term relationships, we have tested head-modifier pairs and binary lexical relations (BLRs). The BLRs are described in Section 3.2.

3.1 Nominalization Process

In broader context, nominalization is a word formation process in which a new noun is derived from an existent word, mainly verb or adjective. We propose that nominalization must be understood as the transformation of a word (adjective, verb, or adverb) in the text, into a semantically corresponding noun existent in the lexicon. So, the derivation “friendly → friend” is a valid nominalization, although in this case the adverb is the derived word in the language.

Nominalization operations [11,10] may derive abstract or concrete nouns. Abstract nouns refer to:

- events (e.g., to meet → meeting),
- qualities (e.g., good → goodness),
- states (e.g., free → freedom),

or other abstract entities, which can be derived from adjectives, verbs, or adverbs. Concrete nouns, on the other hand, refer to:

- agents mostly derived from verbs (e.g., to build → builder), or
- something that is involved or associated with an entity, mainly derived from adjectives (e.g., numerical → number).

Nominalization is an alternative for lexical normalization process based on the fact that nouns are usually the most representative words of the document content [24], and queries are usually formulated through noun phrases.

3.2 Binary Lexical Relations

BLRs [11,10] identify relationships between nominalized terms. These relationships capture phrasal cohesion mechanisms [17], like those that occur between subject and predicate, subject and object (direct or indirect), noun and adjective or verb and adverb. Such mechanisms reveal term dependences.

A BLR has the form

$$id(t', t)$$

where id is a relation identifier, and t' and t are arguments.

Considering the id , there are three kinds of BLRs: classification, restriction, and association.

- Classification BLR: the *id* is the equal sign, the argument *t'* is a subclass or an instance of the argument *t*, and *t* is a class.
- Restriction BLR: the *id* is a preposition and, in general, the argument *t'* is a head and the argument *t* is a complement. When a preposition is not identified as *id*, the default is the preposition “of”.
- Association BLR: the *id* is an event, the argument *t'* is a subject and the argument *t* is a direct or indirect object or an adjunct.

Considering the arguments, a BLR may be an original or a derived relationship.

- Original BLR: both the arguments *t'* and *t* are original nouns from the text. An original restriction BLR is a head-modifier pair with *id*.
- Derived BLR: at least one argument is a nominalized term.

Table 1 shows some examples of BLRs.

Table 1. BLR examples

original phrases	RLBs	types
dog Rex	=(rex,dog)	original classification
Rex is a dog	=(rex,dog)	original classification
the neighbor sang	=(neighbor,singer)	derived classification
quick team	of(quickness,team)	derived restriction
to decide quickly	of(quickness,decision)	derived restriction
lawyer with style	with(lawyer,style)	original restriction
the coach trains the athlete	training(coach,athlete)	derived association
the tourist traveled across Europe	travel.across(tourist,europe)	derived association

Table 2. Examples of arguments' roles in restriction BLRs

<i>id</i>	<i>t'</i>	<i>t</i>	examples
of, for, from, in, to, with	object or event	modifier or comple- ment	for(book,children) in(bird,picture) with(association,word)
of, by	predicate or event	theme or agent	of(honesty,police) of(fall,wall) by(decision,editor)
of	part	whole	of(month,year)

The mapping of syntactic dependencies onto semantic relations [8] defines the restriction BLR identification. The exact role of the arguments depends on the *id*. Table 2 presents some examples.

For more details about BLRs and BLR extraction rules see [10].

3.3 Differences Among the Alternative Strategies Tested

There are some crucial differences among the alternatives that we have tested in this work.

Consider the lexical normalization examples presented in Table 3 for stemming, lemmatizing, and nominalization processes. While a stemmer collapses together those different original words (see Table 3) with a common stem, the lemmatizing process

performs three different derivations. From those words, like the stemming, the nominalization result is also a unique term (except for the verb “commercializes”) but, unlike the stemming, a valid word in the lexicon (a noun) is derived.

Table 3. Lexical normalization examples

original words	stemming	lemmatizing	nominalization
commercializes	commerc	commercialize	commerce or merchant
commercial	commerc	commercial	commerce
commercially	commerc	commercially	commerce

As our nominalization strategy was designed for BLR extraction, the derivation from “commercializes” may also be “merchant”. For example, from the phrase “the artist commercializes the painting”, the BLRs extracted are:

=(artist,merchant),
by(commerce,artist),
of(commerce,painting), and
commerce(artist,painting).

The first BLR is a classification that uses the concrete noun “merchant”, the other BLRs (restrictions and association) use, in this case, the abstract noun “commerce”.

The BLRs with nominalized terms present the following features:

- Distinct concepts are represented through distinct BLRs, even when the text presents the same words.
- The same concept is represented through the same BLR, even when the text presents different syntactic forms to express that concept.
- Each BLR’s argument has discerning role and positioning in the relationship.
- The restriction BLR’s *ids* (relation identifiers) use prepositions, which are not considered stopwords merely.

There are some advantages associated with these referred features for BLRs and nominalized terms over head-modifier pairs and stems or lemmas in a DSI system. The examples presented in Table 4 are used for discussing those BLR features and advantages.

While, the head-modifier pairs for both first phrases (a) and (b) in Table 4 are the same ones, the BLRs between “train” and “paris” have *id* “for”, for the phrase (a), and *id* “from”, for the phrase (b). The inverse case occurs between “departure” and “rome”. Note that, although the phrases (a) and (b) have exactly the same words, BLRs present distinct representations for that distinct term dependences.

Comparing the phrases (b) and (c), two different head-modifier pairs where “train” is an argument are extracted (respectively “(depart,train)” and “(departure,train)”). On the other hand, concerning these cases, the BLR is the same one (“of(departure,train)”). So, using BLRs, we have the same representation for the same concept expressed through different syntactic forms.

For the phrases (d) and (e), the arguments “river” and “tranquillity” have not the same position in the respective BLRs. While “river”, as first argument, is a complement of “beach”, “tranquillity”, as second argument, is a predicate. We see that, in restriction BLRs, the arguments’ roles depend on the preposition used as *id*. Also, the distinction

between abstract and concrete nouns is crucial for indentifying the correct position (and the role) of the arguments in some derived restriction BLRs. Note that “beach of tranquillity” is not an appropriate interpretation of “tranquil beach” in the phrase (e), unlike “beach of river” for “fluvial beach” in the phrase (d).

Table 4. Term relationship examples

original phrases	head-modifier pairs	BLRs
(a) The train for Paris departed from Rome	(train,paris) (depart,train) (depart,rome)	for(train,paris) of(departure,train) from(departure,rome)
(b) The train from Paris departed for Rome	(train,paris) (depart,train) (depart,rome)	from(train,paris) of(departure,train) for(departure,rome)
(c) To wait for the train departure	(wait,departure) (departure,train)	for(wait,departure) of(departure,train)
(d) To swim in the fluvial beach	(swim,beach) (beach,fluvial)	in(swimming,beach) of(beach,river)
(e) To swim in the tranquil beach	(swim,beach) (beach,tranquil)	in(swimming,beach) of(tranquillity,beach)

Table 4 present head-modifier pairs with lemmas. When stems are used, the same concept is represented through the same head-modifier pair (“(depart,train)”), considering phrases (b) and (c), because the verb “depart” and the noun “departure” have the same stem: “depart”. In this case stemming and nominalization have the same effect. However, the comments about the different *ids*, in phrases (a) and (b), and about the roles and positioning of the arguments, in phrases (d) and (e), point out valid differences even between head-modifier pairs with stems and BLRs with nominalized terms.

4 Experimental Evaluation

4.1 Data and Methods

Figure 2 shows the scheme of a DSI system, adapted from [14].

In a DSI system, the lexical normalization process and the term relationship extraction of the text documents are performed at indexing time (see Figure 2). The same processing for the user query is performed at searching time.

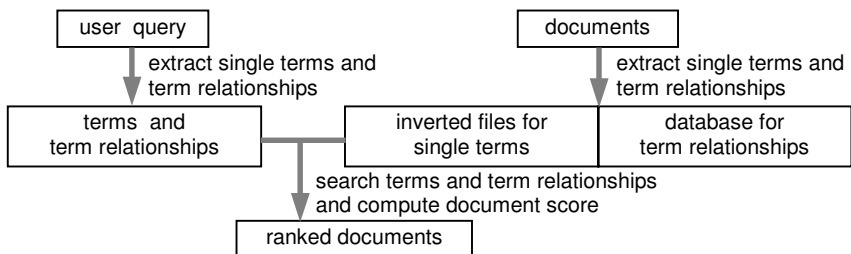


Fig. 2. The DSI system scheme

The dependence structure analysis of both the documents and query allows, in retrieving, the search of both terms and relationships to compute document score and to rank the documents.

We have examined four DSI versions. Table 5 shows a concise characterization of these evaluated versions (B through E) and the baseline A strategy.

Table 5. IR strategy characterization

models	strategies	lexical normalization	relationships
term independence model	A	stemming	no
DSI versions	B	stemming	head-modifier pairs
	C	lemmatizing	head-modifier pairs
	D	nominalization	head-modifier pairs
	E	nominalization	BLRs

The baseline A strategy does not use a DSI system. It uses the conventional Okapi BM25 model, according to equations (1) and (2), based on the term independence assumption. The other strategies (DSI versions) use the equations (1) and (3), and they are based on grammatical connections for including term dependences.

Table 6. Memory space, terms, relationships, and processing time

strategies	inverted files (Kb)	relationship database (Kb)	# of terms	# of relationships	indexing time (s)	searching time (s)
A	2,873	0	24,013	0	0.062	0.137
B	2,873	3,360	24,013	161,154	0.359	0.185
C	3,220	3,869	37,240	168,853	0.377	0.194
D	3,713	3,860	36,479	165,732	0.370	0.190
E	3,713	5,985	36,479	245,093	0.579	0.194

Table 6 shows the size of the inverted files (for terms) and relationship databases, the amount of terms and relationships, and the time (in a 866 MHz Pentium III machine) spent, in average, at indexing and searching by each strategy. The indexing time considers a document with 1,000 tokens (words and punctuations marks), and the searching time takes into account a query with 2 terms.

Considering the DSI versions, the largest economy (in memory space and in amount of terms and relationships) occurs for the B strategy with head-modifiers pairs and stems. There are two reasons for this: (i) stemming produces the largest reduction of the amount of terms and (ii) stems are usually smaller than lemmas and nominalized terms. On the other hand the quantities of terms in the strategy with lemmas (C) and in the strategies with nominalized terms (D and E) are similar.

E has more relationships than the other strategies with head-modifier pairs for the following reasons: (i) two BLRs with the same arguments are considered different relationships when they have not the same relation identifier (this distinction does not occur for head-modifier pairs); and (ii) an association BLR has not a corresponding head-modifier pair.

The indexing time for E, which uses BLRs, is longer than the time spent by the other strategies because it needs to construct a more complex indexing structure. The E strategy uses one inverted file for terms and three different files, one for each BLR type (classification, restriction, and association) in the relationship database. However, such structure allows the same searching time, compared to C, although E uses more relationships. The main reason for this is that E searches each BLR type in the corresponding relationship database file.

We have used the document collection called Folha94 (subset of Mac-Morpho [12]) constituted by articles extracted from 229 editions from Folha de São Paulo newspaper of the year 1994. Folha94 has 4,156 documents.

In order to comparatively evaluate the examined approaches, we followed the strategy in use by the Text Retrieval Conferences [23]. We took the title of 50 topics for generating test queries. Description and narrative of these topics and pooling method were adopted to perform the relevance judgments with the top 100 documents from each approach.

4.2 Results

Figure 3 presents the recall-precision curves for the five strategies examined here.

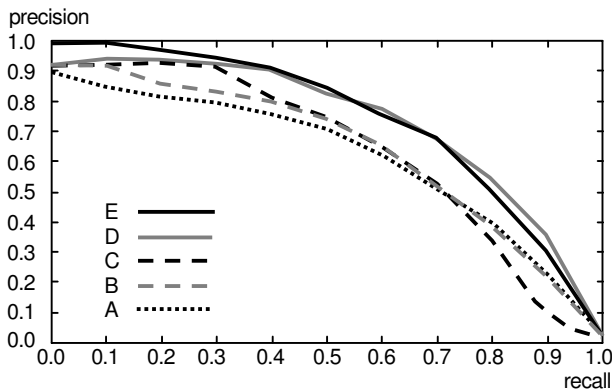


Fig. 3. Recall-precision curves

The E strategy has the higher precision values, with recall of 0.0 through 0.4. With recall of 0.4 through 1.0 there are alternations between E and D concerning the precision values. Note that E and D use nominalized terms. On the other hand, B and C (both with head-modifier pairs) have the worst results for the strategies with DSI.

Table 7 presents precision (P) and mean uninterpolated average precision (MAP) measures for each strategy. The Wilcoxon signed rank test was employed with significance threshold 0.05. In Table 7, the values that include statistically significant performance improvements of B, C, D, and E relative to the baseline A strategy are boldfaced. See that only E (with nominalization and BLRs) has significant improvements relative to A, considering all measures.

Table 7. Values of precision (P) and MAP measures

	A	B	C	D	E
P at 1 doc	0.900	0.920	0.920	0.920	1.000
P at 10 docs	0.626	0.660	0.668	0.726	0.736
MAP	0.741	0.767	0.761	0.833	0.847

E also presents statistically significant performance improvements relative to the following strategies: (i) B, C, and D considering P at 1 document; (ii) B and C considering P at 10 documents; and (iii) C considering MAP measure.

5 Conclusion

We present some alternative approaches for lexical normalization and identification of relationships between terms for a DSI system. As lexical normalization process, in alternative to stemming and lemmatizing, we propose the nominalization process. As term relationships, in alternative to head-modifier pairs, we propose BLRs.

Linguistic knowledge in IR systems has a cost. However, our experiments show that, at searching phase, the time to process a query with 2 terms differs, on average, only in 0.057 seconds between the baseline strategy (unigram model with stems) and the best-tested strategy (term dependence model with BLRs and nominalized terms).

Using head-modifier pairs, the strategy with nominalization achieved statistically significant performance improvements relative to both strategies with stemming or lemmatizing. On the other hand, the strategy with BLRs and nominalized terms improves on the best performance compared to the other ones. Using a DSI system, our experiments demonstrate that the alternative approaches proposed here for lexical normalization and term relationship identification contribute to the improvement of performance in IR for Portuguese.

References

1. Arampatzis, A. T.; Weide, T. P.; Koster, C. H. A.; Bommel, P. Linguistically-motivated Information Retrieval. *Encyclopedia of Library and Inf. Science*, (2000) 69:201-222
2. Baeza-Yates, R.; Ribeiro-Neto, B. *Modern Information Retrieval*. New York: ACM Press (1999)
3. Braschler, M.; Ripplinger, B. How Effective is Stemming and Decompounding for German Text Retrieval?. *Information Retrieval Journal*, (2004) 7:291-316
4. Cho, B.-H.; Lee, C.; Lee, G. G. Exploring term dependences in probabilistic information retrieval model. *Information Processing and Management*, (2003) 39:505-519
5. Chow, C.; Liu, C. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, (1968) IT-14(3): 462-467
6. Cooper, W. S. Some inconsistencies and misnomers in probabilistic information retrieval. *14th Annual International ACM SIGIR conference on research and development in IR. Proceedings*, (1991) 57-61
7. Frakes, W. B., Baeza-Yates, R. *Information Retrieval: Data Structures and Algorithms*. Prentice-Hall, New York (1992)

8. Gamallo, P.; Gonzalez, M.; Agustini, A.; Lopes, G.; Lima, V. L. S. de. Mapping Syntactic Dependencies onto Semantic Relations. ECAI'02, Workshop on NLP and Machine Learning for Ontology Engineering, Lyon, France. Proceedings, (2002) 15-22
9. Gao, J.; Nie, J.; Wu, G.; Cao, G. Dependence language model for information retrieval. 27th Annual International ACM SIGIR conference on research and development in IR. Proceedings, (2004) 170-177
10. Gonzalez, M.; Lima, V. L. S. de; Lima, J. V. de. Binary Lexical Relations for Text Representation in Information Retrieval. 10th Int. Conf on Applications of NL to Inf. Systems, NLDB. LNCS, 3513. Springer-Verlag, (2005) 21-31
11. Gonzalez, M. Termos e Relacionamentos em Evidência na Recuperação de Informação. PhD thesis, Instituto de Informática, UFRGS (2005)
12. <http://www.nilc.icmc.usp.br/lacioweb>
13. Korenius, T.; Laurikkala, J.; Järvelin, K.; Juhola, M. Stemming and Lemmatization in the Clustering of Finnish Text Documents. 13th Conference on Information and Knowledge Management, CIKM. Proceedings, (2004) 625-634
14. Lee, C.; Lee, G. G. Probabilistic information retrieval model for a dependency structured indexing system. Information Processing and Management, (2005) 41:161-175
15. Losee, R. M. Term Dependence: Truncating the Bahadur Lazarsfeld Expansion. Information Processing and Management, (1994) 30(2):293-303
16. Losee, R. M. Term Dependence: a basis for Luhn and Zipf Models. Journal of the American Society for Information Science, (2001) 52(12):1019-1025
17. Mira Mateus, M. H.; Brito, A. M.; Duarte, I.; Faria, I. H. Gramática da Língua Portuguesa. Lisboa: Ed. Caminho (2003)
18. Orenge, V. M., and C. Huyck. A Stemming Algorithm for the Portuguese Language. 8th Symp. on String Processing and IR, SPIRE. Proceedings, (2001) 186-193
19. Rijsbergen, C. van. Information Retrieval. London: Bitterworths (1979)
20. Robertson, S. E.; Walker, S. Some Simple Effective Approximations to the 2-Poisson Model for Probabilistic Weighted Retrieval. 17th Annual International ACM SIGIR conference on research and development in IR. Proceedings, (1994) 232-241
21. Spark-Jones, K.; Walker, S.; Robertson, S. E. A Probabilistic Model of Information Retrieval: Development and Comparative Experiments - Part 1 and 2. Information Processing and Management, (2000) 36(6):779-840
22. Vilares, J.; Barcala, F. M.; Alonso, M. A. Using Syntactic dependency-pairs conflation to improve retrieval performance in Spanish. Computational Linguistics and Intelligent Text Processing. LNCS, 2276. Springer-Verlag, (2002) 381-390.
23. Voorhees, E. M. Overview of TREC 2003. NIST Special Publication - SP500-255. 12th Text Retrieval Conference, Gaithersburg (2003)
24. Ziviani, N. Text Operations. In: Baeza-Yates, R.; Ribeiro-Neto, B. Modern Information Retrieval. New York : ACM Press (1999)

Web Search Model for Dynamic and Fuzzy Directory Search

Bumghi Choi, Ju-Hong Lee, Sun Park, and Tae-Su Park

School of Computer Science and Engineering, Inha University, Incheon, Korea
{neural, juhong}@inha.ac.kr
{sunpark, taesu}@datamining.inha.ac.kr

Abstract. In web search engines, index search used to be evaluated at a high recall rate. However, the pitfall is that users have to work hard to select relevant documents from too many search results. Skillful surfers tend to prefer the index searching method, while on the other hand, those who are not accustomed to web searching generally use the directory search method. Therefore, the directory searching method is needed as a complementary way of web searching. However, in the case that target documents for searching are obscurely categorized or users have no exact knowledge about the appropriate categories of target documents, occasionally directory search will fail to come up with satisfactory results. That is, the directory search method has a high precision and low recall rate. With this motive, we propose a novel model in which a category hierarchy is dynamically constructed. To do this, a category is regarded as a fuzzy set which includes keywords. Similarly extensible subcategories of a category can be found using fuzzy relational products. The merit of this method is to enhance the recall rate of directory search by reconstructing subcategories on the basis of similarity.

1 Introduction

The index searching method has an advantage in that it quickly searches the documents indexed by an input keyword. However, it may exhibit a critical defect by generating too many results or failing to search even a single one of the targeted documents. It is because that given keywords can't be satisfactorily matched with the subjects of the target documents, or they happen to be heteronyms or homonyms, or the target documents may not be properly indexed by the keywords inside them.

In spite of many advantages of the index searching method and many efforts to improve its efficiency, we absolutely need the directory search as a complementary method of the index search. Especially for beginners, the directory searching method is preferred because it can zoom in more detailed subjects by reconstructing the subcategories of a category in a fast manner if they are familiar with the exact information of the categorization of the search subjects. However, if users don't know the categories regarding the subjects of the target documents, or if documents are not exactly categorized, it can't provide users with satisfactory results, and occasionally it causes inconvenience by navigating too many categories before reaching the targets[4, 5, 6].

The above mentioned problems in the directory search method may be due to a fixed category hierarchy that is constructed by manually making relations of subcategories to a category at first, and never changing after that. Therefore, we need an automatic construction method for a flexible category hierarchy.

In this paper, from the above motivation, we propose a novel model in which we construct the relationship between keywords and categories and the reciprocal relationship between categories, thus automatically reconstructing a dynamic category hierarchy to enhance search efficiency[7]. The relationship between category and keyword can be constructed based on the frequency of keywords in the corresponding directory page. This relationship enables a category to be regarded as a fuzzy set comprising keywords and their membership degrees as members. The relationship of two categories can be defined using the similarity of two categories, and their similarity can be calculated in the inclusion degree of two fuzzy sets. Therefore, a similar relation of two different categories can be created so as to automatically construct a dynamic category hierarchy.

2 Fuzzy Relational Products

Definition 1. The fuzzy implication operators vary in the environments of given problems. The afterset aR for $a \in U_1$ is a fuzzy subset of U_2 such that y is related to a , for $y \in U_2$. Its membership function is denoted by $\mu_{aR}(y) = \mu_R(a,y)$. The foreset Sc for $c \in U_3$ is a fuzzy subset of U_2 such that y is related to c , for $y \in U_2$. Its membership function is denoted by $\mu_{Sc}(y) = \mu_S(y,c)$ for $y \in U_2$. So it is defined as follows:

$$(R \triangleleft S)_{ac} = \pi_m(aR \subseteq Sc) = \frac{1}{N_{U_2}} \sum_{y \in U_2} (\mu_{aR}(y) \rightarrow \mu_{Sc}(y)) \tag{1}$$

Here, \rightarrow is a fuzzy implication operator. And π_m is a function to calculate the mean degree. The above mean degree denoted by $(R \triangleleft S)_{ac}$ can be regarded as the mean degree of relation from a to c [3].

3 Dynamic Category Hierarchy Construction

In this paper, the relationship between keywords and categories can be decided by inducing relevant information from the relationship between keywords and documents and the relationship between documents and categories. The related relevancy of documents for keywords has been referred to by many existing researches [1]. A category also can be regarded as a fuzzy set consisting keywords in the documents in that category. The relationship of two categories can be decided by calculating the average degree of inclusion of a fuzzy set to another one using the *fuzzy relational products*[3]. An average degree of fuzzy set inclusion can be used for making a similarity relationship between two categories. By using this, similarity relations of categories can be obtained dynamically.

The fuzzy implication operator has to be presented differently according to the needs of each application. In this paper, Kleen-Diense fuzzy implication operator is used[2]. By applying the Kleen-Diense fuzzy implication operator to the fuzzy rela-

tional products of the formula (1), we can get the average degree of fuzzy sets inclusion for categories, $\pi_m(C_i \subseteq C_j)$. We interpret this degree as the *similarity relationship degree* of C_i to C_j , it is the degree to which C_i is similar to C_j , or we interpret it as the *fuzzy hierarchical degree* of C_i to C_j , it is the degree to which C_j can be a subcategory of C_i . However, $\pi_m(C_i \subseteq C_j)$ have some problems representing the similarity relationship degree between C_i and C_j . That is, if C_i had many element x 's which membership degrees are small, $\mu_{C_i}(x)$, we could have a problem in which the fuzzy relational products tend to be converged to 1 regardless of the average degree of fuzzy sets inclusion of $C_i \subseteq C_j$. Thus, The mean degree of inclusion of a category C_i for a category C_j , $C_i \subseteq C_j$ is defined by *the Modified Fuzzy Relational Products* (\triangleleft') as follows:

$$C_i \Rightarrow C_j = (R^T \triangleleft' R)_{ij} = \frac{1}{|U|} \sum_{K_k \in U} \mu_{C_i}(K_k) \cdot w_k \cdot (R_{ik}^T \rightarrow R_{kj}) \quad (2)$$

Here, K_k is the k 'th keyword, C_i, C_j are the i 'th and the j 'th categories, U is a set of keywords. $|U|$ is the number of U 's elements. R is $m \times n$ matrix such that R_{ij} is $\mu_{C_j}(K_i)$, that is, the membership degree of $K_i \in C_j$, m is the number of keywords and n is the number of categories. R^T is the transposed matrix of R such that $R_{ij} = R_{ji}^T$. \rightarrow is a fuzzy implication operator such that $[0,1] \times [0,1] \rightarrow [0,1]$ expanding a crisp implication operator by multi-valued logic. The following fuzzy implication operator is used in this paper[2]:

$$a \rightarrow b = (1 - a) \vee b = \max(1 - a, b), \quad a = 0 \sim 1, \quad b = 0 \sim 1$$

w_k is a weighted value for each keyword, purposed to weight the importance of the keyword high in case of the search model (2) of section 3. But in (1), it is meaningless and neglected, being all the same value.

4 Experimental Results

We examined the level number of subcategories navigated down until a student found the results that he/she wanted. We assume that the smaller the level number of subcategories navigated is, the faster it will be. 367 university freshmen took part in this experiment. They chose five arbitrary subjects and selected the most relevant categories to the subject and began their search there. In Yahoo Korea, the level number of the last category averaged 6.22, while it was 4.3 in the new search model. In the worst case, Yahoo Korea found the relevant results at 15 levels, while the new search model found them at 7 levels. In search performance, about 31% improvement was achieved. This means that the proposed model implemented in the new search model is more efficient in directory search than conventional method like Yahoo Korea, since the new search model provides similar subcategories of a category by using the category hierarchy, which is constructed dynamically.

5 Conclusions

In this paper, we have proposed a novel search model using *the Modified Fuzzy Relational Products*. The proposed model is purported to enhance the recall rate of direc-

tory search, producing subcategories of a category by using *the Modified Fuzzy Relational Products* in dynamic category reconstruction. An actual system has been implemented, and broad experiments have been executed through this system. We can summarize the following advantages from the results.

- (1) The model makes web searching easy for vague keywords by providing a dynamic reconstruction of a category to similar subcategories.
- (2) The model manages a dynamic category hierarchy in the manner of multiple inheritances of categories and flexibility of category levels.
- (3) The reciprocal compatibility is provided as a singular characteristic, in which a category is the parent and the child of some category at the same time.

Acknowledgments. This research was supported by the Ministry of Information and Communication, Korea, under the Information Technology Research Center support program supervised by the Institute of Information Technology Assessment.

References

1. R. Baeza-Yates, B. Ribeiro-Neto. Modern Information Retrieval. Addison Wesley, 1999.
2. W. Bandler and L. Kohout. Fuzzy Power Sets and Fuzzy Implication Operations. *Fuzzy Set and Systems*, Vol.4, No.1, pp. 13-30, 1980.
3. W. Bandler and L. Kohout. Semantics of Implication Operators and Fuzzy Relational Products. *International Journal of Man-Machine Studies*. Vol. 12, pp.89-116, 1980.
4. L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, and E. Ruppin. Placing Search in Context: The Concept Revisited. In *Proceedings of the 10th International Conference on World Wide Web*, pp.406-414, HongKong, Chine, May 2001.
5. M. NikRavesh. Fuzzy Conceptual-Based Search Engine using Conceptual Semantic Indexing. *NAFIPS-FLINT 2002*, pp.146-151, New Orleans, LA, June 2002.
6. J. Wen, J. Nie, and H. Zhang. Clustering User Queries of a Search Engine. In *Proceedings of the 10th International Conference on World Wide Web*, pp. 162-168, Hong Kong, China, 2001.
7. <http://webbase.inha.ac.kr/paper/websearch.pdf>, Web Search Model for Dynamic and Fuzzy Directory Search, full paper version of this paper

Information Retrieval from Spoken Documents^{*}

Michal Fapšo, Pavel Smrž, Petr Schwarz, Igor Szöke, Milan Schwarz,
Jan Černocký, Martin Karafiát, and Lukáš Burget

Faculty of Information Technology, Brno University of Technology,
Božetěchova 2, 612 66 Brno, Czech Republic
speech@fit.vutbr.cz
<http://www.fit.vutbr.cz/speech/>

Abstract. This paper describes a designed and implemented system for efficient storage, indexing and search in collections of spoken documents that takes advantage of automatic speech recognition. As the quality of current speech recognizers is not sufficient for a great deal of applications, it is necessary to index the ambiguous output of the recognition, i. e. the acyclic graphs of word hypotheses — recognition lattices. Then, it is not possible to directly apply the standard methods known from text-based systems. The paper discusses an optimized indexing system for efficient search in the complex and large data structure that has been developed by our group. The search engine works as a server. The meeting browser JFerret, developed withing the European AMI project, is used as a client to browse search results.

1 Introduction

The most straightforward way to use a large vocabulary continuous speech recognizer (LVCSR) to search in audio data is to use existing search engines on the textual (“1-best”) output from the recognizer. For such data, it is possible to use common text indexing techniques. However, these systems have satisfactory results only for high quality speech data with correct pronunciation. In the case of low quality speech data (noisy TV and radio broadcast, meetings, teleconferences) it is highly probable that the recognizer scores a word which is really in the speech worse than another word.

We can however use a richer output of the recognizer – most recognition engines are able to produce an oriented graph of hypotheses (called *lattice*). On contrary to 1-best output, the lattices tend to be complex and large. For efficient searching in such a complex and large data structure, the creation of an optimized indexing system which is the core of each fast search engine is necessary. The proposed system is based on principles used in Google [1]. It consists of **indexer**, **sorter** and **searcher**.

^{*} This work was partly supported by European project AMI (Augmented Multi-party Interaction, FP6-506811) and Grant Agency of Czech Republic under project No. 102/05/0278. Pavel Smrž was supported by MŠMT Research Plan MSM 6383917201. The hardware used in this work was partially provided by CESNET under project No. 119/2004.

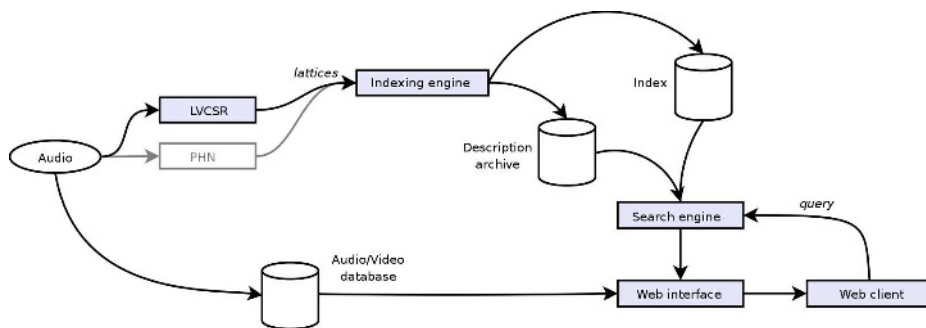


Fig. 1. The overall proposed design of the audio/speech search engine. Till now, only the LVCSR search module is implemented.

2 Indexer

Word lattices generated by LVCSR are input to the indexing and search engine. The lattices (see example in Fig. 2) are stored in standard lattice format (SLF) [4]:

A lattice stored in SLF format consists of optional header information followed by a sequence of node definitions and a sequence of links (arc) definitions. Nodes and links are numbered and the first definition line must give the total number of each.

Each link represents a word instance occurring between two nodes, however for more compact storage the nodes often hold the word labels since these are frequently common to all words entering a node (the node effectively represents the end of several word instances). This is also used in lattices representing word-level networks where each node is a word end, and each arc is a word transition.

Each node may optionally be labelled with a word hypothesis and with a time. Each link has a start and end node number and may optionally be labelled with a word hypothesis (including the pronunciation variant, acoustic score and segmentation of the word hypothesis) and a language model score.

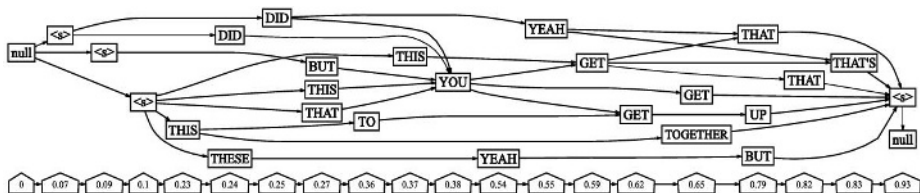


Fig. 2. Example of a word lattice

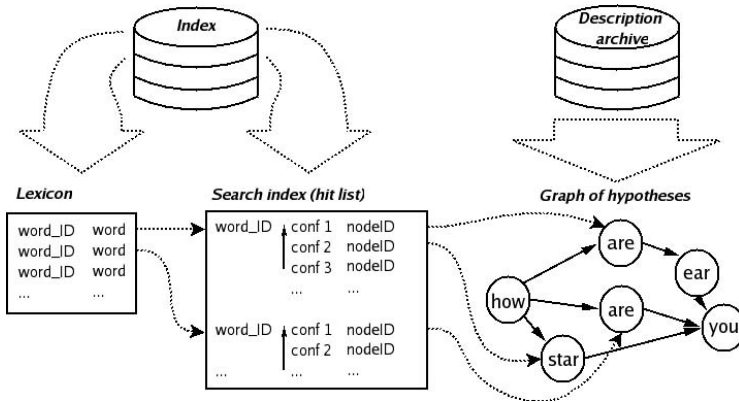


Fig. 3. Simplified index structure

The indexing mechanism (Fig. 3) consists of three main phases:

- creating the lexicon
- storing and indexing lattices
- creating the reverse index

The lexicon provides a transformation from word to a unique number (ID) and vice versa. It saves the used disk space and also the time of comparing strings (numbers need less space than words).

Lattices are stored in a structure which differs from the SLF structure. For each search result, not only the found word, but also it’s context has to be extracted. It means that we need to traverse the lattice from the found word in both directions (forward and backward) to gather those words lying on the best path which traverses through the found word. As SLF structure keeps the nodes separated from links, lattices are converted to another structure which stores all forward and backward links for each particular node at one place. We also need to assign a confidence to each hypothesis. This is given by the likelihood ratio:

$$C^{lucsr}(KW) = L_{alpha}^{lucsr}(KW)L_t^{lucsr}(KW)L_{beta}^{lucsr}(KW)/L_{best}^{lucsr}, \quad (1)$$

where the $L^{lucsr}(KW) = L_a^{lucsr}(KW)L_t^{lucsr}(KW)$.

The acoustic likelihood is computed using Hidden Markov Model and measures similarity between model and parameters of speech signal. The language model likelihood is computed by trigram language model incorporating probabilities of word tripples.

The forward likelihood $L_{alpha}^{lucsr}(KW)$ is the likelihood of the best path through lattice from the beginning of lattice to the keyword and the backward likelihood $L_{beta}^{lucsr}(KW)$ is the likelihood of the best path from the keyword to the end of lattice. For node N, these two likelihoods are computed by the standard Viterbi formulae:

$$L_{\alpha}^{lucsr}(N) = L_a^{lucsr}(N)L_l^{lucsr}(N)\max_{N_P}L_{\alpha}^{lucsr}(N_P) \quad (2)$$

$$L_{\beta}^{lucsr}(N) = L_a^{lucsr}(N)L_l^{lucsr}(N)\max_{N_F}L_{\beta}^{lucsr}(N_F) \quad (3)$$

where N_F is a set of nodes directly following node N (nodes N and N_F are connected by an arc), N_P is a set of nodes directly preceding node N . $L_a^{lucsr}(N)$ and $L_l^{lucsr}(N)$ are acoustic and language-model likelihoods respectively. The algorithm is initialized by setting $L_{\alpha}^{lucsr}(first) = 1$ and $L_{\beta}^{lucsr}(last) = 1$. The last likelihood we need in Eq. 1: $L_{best}^{lucsr} = L_{\alpha}^{lucsr} = L_{\beta}^{lucsr}$ is the likelihood of the most probable path through the lattice.

3 Sorting and Searching Lattices

During the phase of **indexing** lattices, the forward index is created. It stores each hypothesis (the word, it's confidence, time and nodeID in the lattice file) in a hit list. Records in the forward index are sorted according to the document ID (number which represents the lattice's file name) and time. The forward index itself is however not very useful for searching for a particular word, because it would be necessary to go through the hit list sequentially and select only matching words. Therefore the reverse index is created (like in Google) which has the same structure as the forward index, but is sorted by words and by confidence of hypotheses. It means that all occurrences of a particular word are stored at one place. There is also a table which transforms any word from lexicon into the start position of corresponding list in reverse index.

Each speech record (ie. meeting) is represented by several huge lattices (depending on number of recorded channels or speakers). The reverse index tells us, in which lattice the keyword appears and what is it's nodeID in this particular lattice. As we need to get the context of the found keyword, it is needed to load the corresponding part of lattice into memory. For this purpose time index is used, which splits the lattice into smaller parts by time.

Searching for one word then consists only from jumping right to the beginning of it's list in reverse index, selecting first few occurrences and getting their context from corresponding lattice. The advantage of splitting the indexing mechanism into three phases is that the second phase (storing and indexing lattices), which is the most demanding, can be run in parallel on several computers. Each parallel process creates it's own forward index. These indices are then merged together and sorted to create the reverse index.

The **searcher** uses the reverse index to find occurrences of words from query and then it discovers whether they match the whole query or not. For all matching occurrences, it loads into the memory only a small part of lattice within which the found word occurs. Then the searcher traverses this part of lattice in forward and backward directions selecting only the best hypotheses; in this way it creates the most probable string which traverses through the found word.

4 Experiment

The recognition lattices were generated using the AMI-LVCSR system incorporating state-of-the-art acoustic and language modeling techniques [2].

Our keyword spotting systems were tested on a large database of informal continuous speech of ICSI meetings [6] (sampled at 16 kHz). Attention was paid to the definition of fair division of data into training/development/test parts with non-overlapping speakers. It was actually necessary to work on speaker turns rather than whole meetings, as they contain many overlapping speakers. We have balanced the ratio of native/nonnative speakers, balanced the ratio of European/Asiatic speakers and moved speakers with small portion of speech or keywords to the training set. The training/development/test parts division is 41.3, 18.7 and 17.2 hours of speech respectively. Development part is used for system tuning (phoneme insertion penalty, etc.).

In the definition of keyword set, we have selected the most frequently occurring words (each of them has more than 95 occurrences in each of the sets) but checked, that the phonetic form of a keyword is not a subset of another word nor of word transition. The percentage of such cases was evaluated for all candidates and words with high number of such cases were removed. The final list consists of 17 keywords: actually, different, doing, first, interesting, little, meeting, people, probably, problem, question, something, stuff, system, talking, those, using.

Our experiments are evaluated using *Figure-of-Merit* (FOM) [7], which is the average of correct detections per 1, 2, . . . 10 false alarms per hour. We can approximately interpret it as the accuracy of KWS provided that there are 5 false alarms per hour. The FOM for this data is 66.95

Then the time duration necessary to retrieve the best hypothesis (or few best ones, ie. 10) of keyword was measured. The indexing and search engine was tested on meeting data of 1.9 hour, the lattices consist of 3,607,089 hypotheses and 36,036,967 arcs. The system is able to find the 10 best hypotheses in these 1.9 hours of records in 0.8197 s (average real time), while the processor time without waiting for disk I/O is only 0.0421 s. The average memory usage is 4.5 MB. This experiment was evaluated on a Pentium-P4 2G processor.

5 Integration into JFerret Meeting Browser and Client/Server Architecture

JFerret [5] is a new multi-media browser for the AMI project¹ written by Mike Flynn from IDIAP Research Institute. The browser is extremely flexible, enabling almost any user interface to be composed, using a combination of plug-in modules. An XML configuration specifies which plug-in components to use, how to arrange them visually, and how they will communicate with each other. The JFerret plug-in for the search engine was implemented at our University.

On the main screen (Fig. 4), the user can see (just as in other searchers on the Web) a text field for inserting query word(s) and buttons to choose between

¹ Augmented Multi-party Interaction, <http://www.amiproject.org>

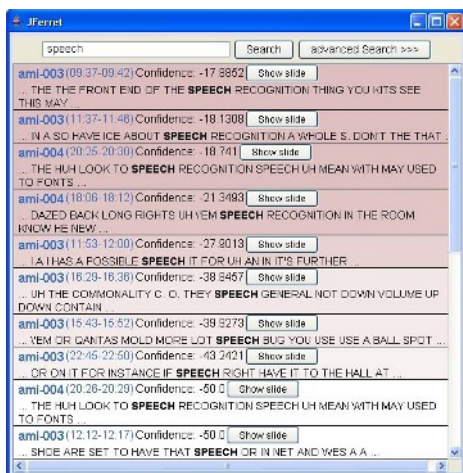


Fig. 4. Search window with found hypothesis

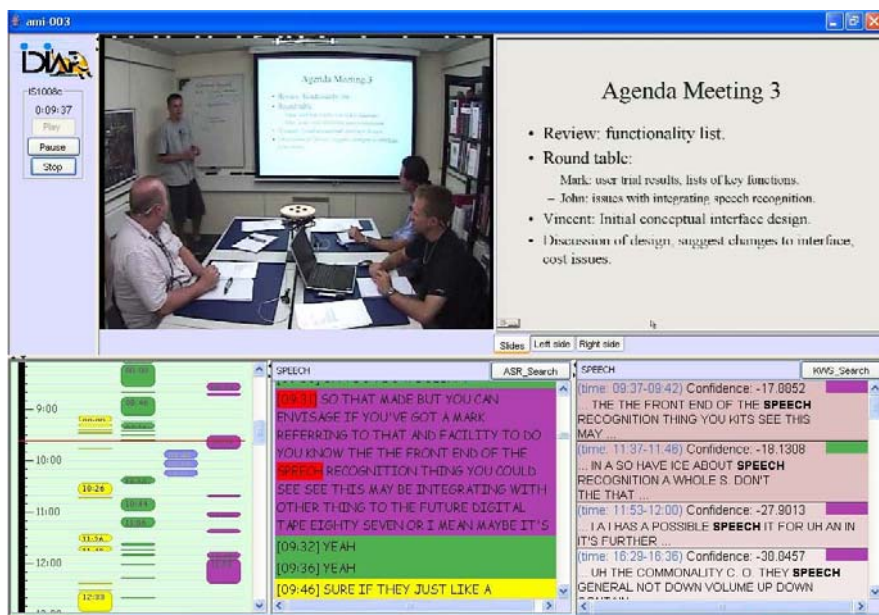


Fig. 5. JFerret replaying search result

simple and advanced search. In the advanced mode, the user can narrow the search by entering additional parameters such as name of the meeting, time interval for search etc.

The results are presented as a sorted list of hypothesis. When a user clicks on a hypothesis, a window with the particular meeting including all the available

information (audio, video, slides) is opened and the particular segment is played. A list of hypothesis relevant to this particular meeting is shown as well (lower-right panel in Fig. 5) so that the user can directly browse other occurrences of the keyword. JFerret ensures the necessary synchronization of all information streams.

Although the search engine can run as a standalone application on Linux or Windows, it is more useful to run the search engine as the server. The communication is based on a simple TCP text-based protocol, so even a simple telnet client is able to send a query to the search server. One of the advantages of using JFerret as the client is that it can play audio and video data from a remote server and also synchronize several audio/video streams. All the data including audio/video files and indices can therefore be stored on the server.

6 Conclusion

We have presented a system for fast search in speech recognition lattices making extensive use of indexing. The results obtained with this system are promising and the software already serves as basis for several LVCSR-keyword spotting demonstrations. The system was extended by the possibility to enter multi-word queries, and options to narrow search space (restriction for particular meetings, speakers, time intervals). It was integrated with the powerful and flexible meeting browser JFerret from IDIAP. The future work will address phoneme-lattice based keyword spotting which eliminates the main drawback of LVCSR — the dependency on recognition vocabulary [3], and methods for indexing of its results.

References

1. Sergey Brin, Lawrence Page: *The Anatomy of a Large-Scale Hypertextual Web Search Engine*, Computer Science Department, Stanford University
2. Thomas Hain et al.: *Transcription of Conference Room Meetings: an Investigation*, In Proc. Eurospeech 2005, Lisbon, Portugal, September 2005.
3. Igor Szöke et al.: *Comparison of Keyword Spotting Approaches for Informal Continuous Speech*, In Proc. Eurospeech 2005, Lisbon, Portugal, September 2005.
4. Steve Young et al.: *The HTK Book (for HTK Version 3.3)*, Cambridge University Engineering Department, 2005, <http://htk.eng.cam.ac.uk/>.
5. Bram van der Wal et al.: *D6.3 Preliminary demonstrator of Browser Components and Wireless Presentation System*, AMI deliverable, August 2005.
6. A. Janin and D. Baron and J. Edwards and D. Ellis and D. Gelbart and N. Morgan and B. Peskin and T. Pfau and E. Shriberg and A. Stolcke and C. Wooters: "The ICSI Meeting Corpus", In "International Conference on Acoustics, Speech, and Signal Processing, 2003. ICASSP-03", Hong Kong, april 2003
7. J. R. Rohlicek and W. Russell and S. Roukos and H. Gish: "Continuous hidden Markov modeling for speaker-independent word spotting", In "International Conference on Acoustics, Speech, and Signal Processing, 1989. ICASSP-89", Glasgow, UK, may 1989, vol. 1

Automatic Image Annotation Based on WordNet and Hierarchical Ensembles

Wei Li and Maosong Sun

State Key Lab of Intelligent Technology and Systems,
Department of Computer Science and Technology, Tsinghua University,
Beijing 100084, China
wei.lee04@gmail.com, sms@mail.tsinghua.edu.cn

Abstract. Automatic image annotation concerns a process of automatically labeling image contents with a pre-defined set of keywords, which are regarded as descriptors of image high-level semantics, so as to enable semantic image retrieval via keywords. A serious problem in this task is the unsatisfactory annotation performance due to the semantic gap between the visual content and keywords. Targeting at this problem, we present a new approach that tries to incorporate lexical semantics into the image annotation process. In the phase of training, given a training set of images labeled with keywords, a basic visual vocabulary consisting of visual terms, extracted from the image to represent its content, and the associated keywords is generated at first, using K-means clustering combined with semantic constraints obtained from WordNet, then the statistical correlation between visual terms and keywords is modeled by a two-level hierarchical ensemble model composed of probabilistic SVM classifiers and a co-occurrence language model. In the phase of annotation, given an unlabeled image, the most likely associated keywords are predicted by the posterior probability of each keyword given each visual term at the first-level classifier ensemble, then the second-level language model is used to refine the annotation quality by word co-occurrence statistics derived from the annotated keywords in the training set of images. We carried out experiments on a medium-sized image collection from Corel Stock Photo CDs. The experimental results demonstrated that the annotation performance of this method outperforms some traditional annotation methods by about 7% in average precision, showing the feasibility and effectiveness of the proposed approach.

1 Introduction

With the exponential growth of multimedia information, efficient access to a large image/video databases is highly desired. To address this problem, Content-Based Visual Information Retrieval, has become a hot research topic in the domain of both computer vision and information retrieval in the last decade.

Traditionally, most of the content-based image retrieval techniques is based on the query-by-example (QBE) architecture, in which user should provide an image example firstly, the visual similarity of low-level visual features such as color, texture and shape descriptors is then computed to find the visually similar images compared to the user-provided image. However, some critical problems still remain with this retrieval

architecture. First, image semantics are not necessarily captured by the low-level features. Second, due to the semantic gap between low-level visual features and high-level semantic concepts, visual similarity is not semantic similarity. Third, it is non-intuitive and difficult for common users to specify the query concept using the low-level perceptual features. Typically, users would prefer using textual queries rather than visual features to conduct image retrieval. Not only because users' query concepts can be represented more precisely by using keywords than by using low-level visual features, but also this method can resort to many powerful text-based retrieval techniques to support semantic image retrieval via keywords at different semantic levels. The key to image retrieval using textual queries is image annotation. But most images have not annotations and manually annotating image is a time-consuming, error-prone and subjective process. So, automatic or semi-automatic image annotation is the subject of much ongoing research. Its main goal is to automatically annotate images using a pre-defined lexicon to describe the image semantics, which has been recognized as a promising technique for bridging the semantic gap between low-level image features and high-level semantic concepts. Discovering complex relationship between low-level image features and associated text is thus considered to be an effective solution to derive the image semantics. Then, images can be retrieved by using natural languages, which is also called query-by-keyword (QBK) architecture.

Given a training set of images labeled with text (e.g. keywords, captions) that describe the image semantics, many state-of-the-art models have been proposed by researchers to discover the hidden correlation between keywords and image features. However, two common problem shared by most approaches also exist: first, most approaches perform the clustering merely based on the visual features, therefore regions with different semantics but similar visual appearance may be easily grouped into a region cluster, leading to a poor clustering performance. Second, in the process of annotating images, each annotated word is predicted independently from other annotated words, word-to-word correlation is not taken into consideration, which degrades the quality of automatic annotations. In this paper, a new method is proposed using semantics-constrained K-means clustering in combination with hierarchical ensembles. The semantics-constrained K-means can enhance the quality of clustering to some degree and the hierarchical ensemble annotation architecture also provides more accurate and consistent annotations compared to the state-of-the-art models.

This paper is organized as follows: Section 2 presents related work. Section 3 describes the image content representation and discusses the visual vocabulary construction using semantics-constraint K-means. Finally, we provide details of how to use the hierarchical ensemble to automatically annotate unlabeled images. Section 4 demonstrates our experimental results. Section 5 presents conclusions and a comment for future work.

2 Related Work

Recently, many models using machine learning techniques have been proposed for automatic and semi-automatic image annotation and retrieval, Including co-occurrence model [4], statistical machine translation model [2], hierarchical aspect model [1][15], relevance models [3][8] and Correlation LDA [18]. Mori et al [4] collects statistical co-occurrence information between keywords and image grids and

uses it to predict annotated keywords to unseen images. Dyugulu et al [2] views annotating images as similar to a process of translation from “visual information” to “textual information” by the estimation of the alignment probability between visual blob-tokens and textual keywords. Barnard et al [1][15] proposed a hierarchical aspect model to capture the joint distribution of words and image regions using EM algorithm. Jeon et al [3] presented a cross-media relevance model similar to the cross-lingual retrieval techniques to perform the image annotation and ranked image. Lavrenko et al [8] extended the cross-media relevance model using actual continuous-valued features extracted from image regions. This method avoids the clustering and constructing the discrete visual vocabulary stage. Blei et al [18] proposed a correlation LDA and assumes that a Dirichlet distribution can be used to generate a mixture of latent factors that can further relates words and image regions. In general, these approaches can be considered as unsupervised semantic annotation which aims to model the joint probability of image and words based on a weakly labeled image data without the explicit correspondence between labels and particular image regions. Yet, the other approaches formulate image annotation as a semantic classification task which requires the training image dataset to be precisely labeled with explicit correspondence. Wang and Li [7] introduced a 2-D multi- resolution HMM model to automate linguistic indexing of images. Clusters of fixed-size blocks at multiple resolution and the relationships between these clusters is summarized both across and within the resolutions. Chang et al [5] proposed content-based soft annotation (CBSA) for providing images with semantic labels using (BPM) Bayesian Point Machine. Cusano C et al [11] proposed using Multi-class SVM to classify each square image region into one of seven pre-defined concepts of interest and then combine the partial decision of each classifier to produce the overall description for the unseen image. In addition to the above methods, recently, R.jin et al [14] proposed to estimate a language model for each image, which can not only relax the estimation of $p(\{w\} | I)$, but also take the word correlations into consideration to improve the annotation performance. Fan et al uses a two-level model to associate salient objects and associated keywords. At the first level, salient objects are automatically extracted and classified by Support Vector Machines. At the second level, Gaussian Mixture Models learned by an adaptive EM algorithm is used to annotate images. Motivated by R.jin and Fan’s methods, we also propose a simple yet effective approach to use the combination of constrained clustering and language models to improve the quality of image annotation.

3 The Implementation of Automatic Annotation Model

3.1 The Hierarchical Framework of Automatic Annotation

Fig. 3.1 shows the framework for automatic image annotation and. Given a training set of annotated images, first, we perform the segmentation of each image into a collection of image regions, followed by clustering all the regions across the image dataset using semantics-constraint K-means to create a discrete visual vocabulary consisting of visual terms. Then, a hierarchical ensemble is learned using the visual terms and textual terms and then we can use the learned model to automatically generate annotation for unseen images.

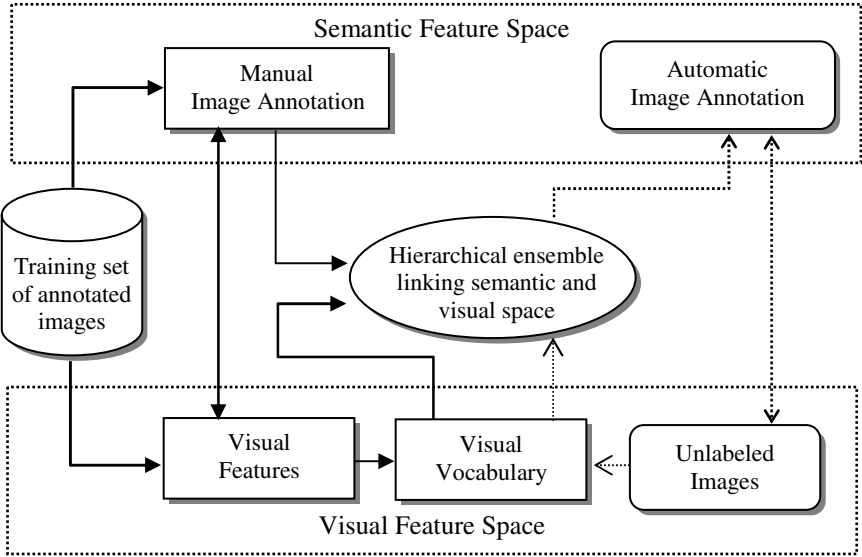


Fig. 3.1. Hierarchical Framework of Automatic Annotation. —> Learning the hierarchical ensemble based on the training set of annotated images. - - -> Applying the hierarchical ensemble to automatically generate image annotations.

3.2 Image Content Representation

A central issue in content-based image annotation and retrieval is how to describe the visual information in a way compatible with human visual perception. But until now, no general framework is proposed. For different tasks and goals, different image scale and corresponding low-level features are used to describe and analyze the visual content. Image-based approaches are more desirable to enable low-cost framework for feature extraction, however due to its rough representation for the image content using global visual properties, this method does not work well, especially for images contain many individual objects. Fixed-size blocks are effective to consider the local features and contextual information, but in some cases, it is unable to model the objects contained in the image well, leading to a poor annotation result. In our method, we carried out these following two steps: First, segment images into image regions using a general purpose image segmentation algorithm, second, extract appropriate features for each region, cluster visually similar regions in an assumed semantic subspace by semantics-constrained K-means which will be discussed in following sections and then use the centroid in each cluster as a visual term.

Specifically, we segment images into a number of meaningful regions using Normalized cuts [6] against using JSEG. Because the JSEG is only focusing on local features and their consistencies, but Normalized cuts aims at extracting the global impression of an image data. So Ncuts may get a better segmentation result than JSEG. Fig. 3.2 shows segmentation result using Ncuts and JSEG respectively, the left is the original image, the mid and the right are the segmentation result using Ncuts and JSEG, respectively. After segmentation, each image region is described by a

feature vector consisting of color, texture, shape, position and area. Similar regions in a semantic subspace will be grouped together based on semantics-constrained K-means to form the visual vocabulary of visual terms. Then each of the labeled and unlabeled images can be described by a number of visual terms in a discrete feature space, instead of the real-valued feature vectors in a continuous feature space. So we can avoid the image representation in a high-dimensional, sparse and complex continuous visual space.



Fig. 3.2. Segmentation Results using Normalized cuts and JSEG

3.3 Constructing Visual Vocabulary with Semantic Constraints

Given a training set of annotated images, each image can be represented by both textual modality and visual modality in a multi-modal feature space. Most of the current approaches perform clustering merely based on low-level visual features in the visual space to construct the visual vocabulary. Therefore, some regions with different semantics but the same or identical visual properties may easily cross the semantic boundary to be wrongly grouped into the same region cluster, leading to poor clustering performance. For example, consider the region “snow” and “cloud” which are irrelevant in terms of semantics, but they share the “white” color distribution in visual feature space. Thus two semantically irrelevant regions may be clustered into one region cluster in visual space without considering the semantic relationship between the two words. A natural solution to this problem is to impose semantic constraints to the process of clustering [17]. We exploit a thesaurus-aided approach to improve the quality of clustering. First, we stipulate that the semantic relevance of two regions can be deduced by the relevance of all annotations between two corresponding images. If two regions are relevant in semantics, then they belong to the same semantic subspace, otherwise, not. WordNet [12] is a powerful electronic lexical database and provides the semantic relationship between words through either hierarchical or non-hierarchical relationships. In our experiment, we tried the WordNet::Similarity function [13] to measure and quantify the relevance/irrelevance between two image annotations and further induce the semantic relationship between two regions. Then, we can form a series of semantic subspaces to ensure that regions belonging to different semantic subspace may not cross the semantic boundary to be clustered into the same visual term. Under this framework, an indicator function is defined to determine whether two regions reside in the same semantic subspace or not. We assert that each annotated image I_i can be represented by both the low level visual features and associated annotations $I_i = \{r_{i1}, r_{i2}, \dots, r_{im}; w_{i1}, w_{i2}, \dots, w_{in}\}$, where r_{ij} denote the feature vector for each segmented region and w_{ij} denote the associated labels, m, n are the number of the segmented regions and the keywords respectively. Given every

image pair I_p and I_q , the indicator function for each pair of regions in the corresponding two images can be defined as

$$S(r_{pi}, r_{qj}) = \begin{cases} 0 & \frac{1}{N_{W_p} \times N_{W_q}} \sum_{u \in W_p} \sum_{v \in W_q} sim(w_{pu}, w_{qv}) > \theta \\ 1 & \text{Otherwise} \end{cases} \quad (1)$$

where $sim(w_{pu}, w_{qv})$ denote the semantic relationship between each pair of annotations in either two images which is measured by WordNet, W_p and W_q are the set of labels for the two images, N_{W_p} and N_{W_q} represent the number of labels of two images respectively. If the semantic relevance of the two image annotations is smaller than a given threshold θ , then the two image regions are regarded as semantic irrelevant or belong to different semantic subspaces, the value of indicator function is 1. We use K-means as our base clustering method. Since K-means can't handle the constraints directly, a variant of K-means with semantic constraint based on WordNet is formulated with the goal of minimizing the combined objective function:

$$J = \sum_{k=1}^N \sum_{i \in S_k} |r_i - \mu_k|^2 + \sum_{\{(r_i, r_j)\}_{r_i \in S_k \cap r_j \in S_k}} S(r_i, r_j) * P \quad (2)$$

where, P is the cost of violating the semantic constraints, r_i denote the segmented regions, μ_k represent the centroid of clusters and S_k is the label of a region cluster.

3.4 The Annotation Strategy Based on Hierarchical Ensembles

For the annotation procedure, given a training set of images labeled with multiple pre-defined semantic labels, image segmentation is first performed, followed by constructing the visual vocabulary constrained by the semantic relationships, then the complex visual content of images can be represented by a number of visual terms from the visual vocabulary rather than the continuous feature vector. To annotate an unlabeled image, a two-level hierarchical ensemble is trained for propagating the semantic labels to unlabeled images. For the first level, an ensemble of binary classifiers is trained for predicting label membership for segmented images. In this paper, we use SVM as the base classifier and each binary classifier assumes the task of determining the confidence score for a semantic label. The trained ensemble is applied to each individual unlabeled segmented region to generate multiple soft labels, and each label is associated with a confidence factor indicating the association probability between the semantic label and the image region. After the annotation of the first-level ensemble, each segmented region in an unlabeled image is assigned a probabilistic label-vector with length equal to the size of lexicon, and each label is associated with a confident value. For example, a region probabilistic label-vector (*sun*: 0.7, *city*: 0.2, *sky*: 0.1, ...) means that region is expected to contain semantics of “*sun*”, “*city*” and “*sky*” with 70%, 20% and 10% confidence, respectively. Since the probabilistic

label vector for each region is predicted independently from that of other regions at the first-level ensemble, the word-to-word correlation is not taken into consideration which may result in a poor annotation quality. For instance, consider word “*sky*” and “*ocean*”, since both words are related to regions with blue color, it is difficult to determine which word is the correct annotation. However, since certain words such as “*trees*” and “*grass*” are more likely to co-occur with the word “*sky*” than “*ocean*”, if “*trees*” or “*grass*” has been selected as an annotation word, “*sky*” will be preferred over “*ocean*” because of the word-to-word correlation. Thus, at the second-level ensemble, we consider the use of a language model to improve annotation quality. A word co-occurrence probability matrix is computed from the pre-defined lexicon and then used to modify or re-weight the association probability between visual terms and textual terms derived from the first-level ensemble. We use the following re-weighting scheme to update the association probability P_i^k in a label-vector for k -th visual term:

$$P_i^k = P_i^k \times \prod_j P_{co(i,j)} \quad (3)$$

$$P_{co(i,j)} = IF(w_i \wedge w_j) / IF(w_i \vee w_j) \quad (4)$$

where, $P_{co(i,j)}$ represent the co-occurrence probability of label i and label j , and $IF(w_i \wedge / \vee w_j)$ denote image frequency, i.e., the number of images containing label w_i and (or) w_j . Finally, label i with the highest probability P_i^k is selected as the annotation for the corresponding image region k and the image-level annotation can be obtained by integrating all the region-level annotations as shown in Fig. 3.3.

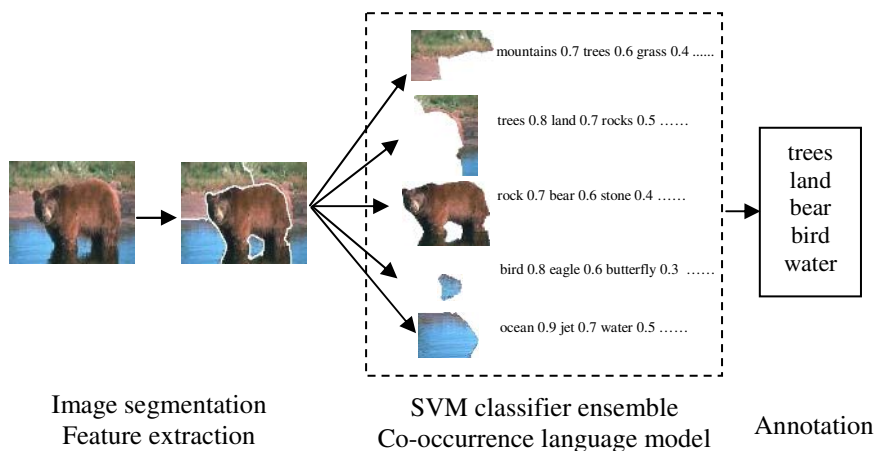






Fig. 3.3. Hierarchical Ensemble Model for Auto Image Annotation

4 Experiments and Analysis

We carried out experiments using a mid-sized image collection comprising about 5000 images from Corel Stock Photo CDs in which each image is annotated by 3-5 keywords. In our experiment, 300 images are selected, 200 images for training and the remaining 100 images for testing. Images are segmented using Normalized cuts. Only regions larger than a threshold are used, each image segment is represented by a 33-dimensional feature vector including color, region average orientation energy, region area and position, etc. The resulting visual vocabulary and the pre-defined lexical set contain 180 visual terms and best 24 keywords, respectively. Here, we choose probabilistic SVM as our base classifier for the first-level ensemble [9][10], *one-against-one* rule is used for the ensemble scheme and each binary classifier is responsible for predicting one semantic label. The kernel function is radial basis function (RBF). We tried 5-fold cross-validation and find the optimal parameters of RBF for binary SVM classifier is: $C = 25, \lambda = 0.1$. Due to the data sparseness problem in our small size of lexicon, we use a simple smoothing technique to address the zero frequency of word co-occurrence by adding one to the count of word co-occurrence whose value is zero.

Fig. 4.2 shows the some of the retrieval results using the keyword “trees” as a single word query.

Table 4.1. Automatic image annotation results

Images	True Annotation	Automatic Annotation based on Constrained Clustering	Automatic Annotation based on Unconstrained Clustering
	sand grass water sky	sky water trees grass	sky water stress grass
	airplane sky	sky airplane cloud	sky airplane waves
	grass trees house mountains sky	trees grass mountains rock	trees grass mountains coat
	sky trees lion rock	trees sky grass lion	cars trees grass lion

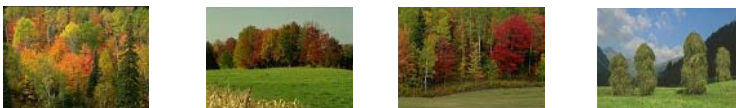


Fig. 4.2. Some of retrieved images using “tree” as a query

Fig. 4.3 and Fig. 4.4 show the precision and recall of using a set of example keywords as single word queries. We compared our proposed method with the start-of-the-art cross-media relevance model (CMRM) and the statistical machine translation model.

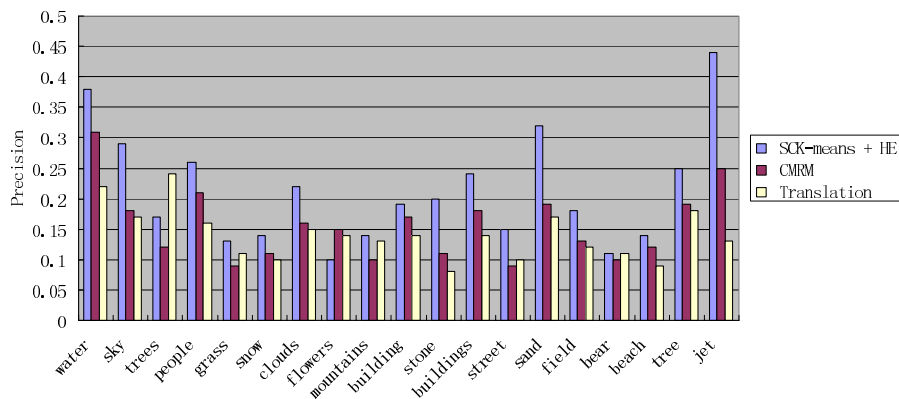


Fig. 4.3. Precision of retrieval using some keywords

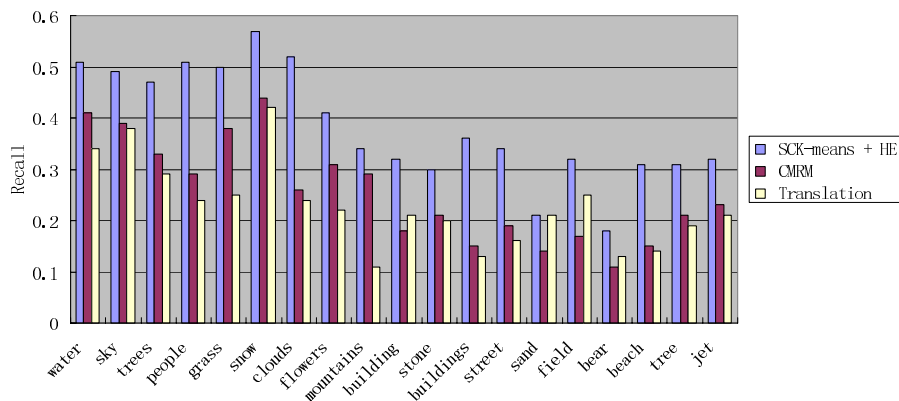


Fig. 4.4. Recall of retrieval using some keywords

The annotation accuracy is evaluated by using precision and recall indirectly. For each single word query, precision and recall are computed using the retrieved list based on the true annotation and automatic annotation. Let I_j be a test image, t_j be the true annotations, and a_j be the auto annotations. For a single query keyword w , precision and recall are calculated respectively as:

$$precision(w) = \frac{|\{I_j | w \in t_j \wedge w \in a_j\}|}{|\{I_j | w \in a_j\}|} \quad recall(w) = \frac{|\{I_j | w \in t_j \wedge w \in a_j\}|}{|\{I_j | w \in t_j\}|} \quad (5)$$

The above experimental results in table 4.5 show that the semantics-constraint K-means combined with hierarchical ensemble outperforms the cross-media relevance model [3] and the machine translation model [2] in both average precision and recall. Our approach uses the visual terms to represent the contents of the image regions in a discrete visual space which can avoid the large variance and data sparseness in continuous visual feature space. Moreover, we impose additional semantics-constraint to the process of vector quantizing the visual feature space. Thus, the visual vocabulary is created in a multimodal feature space, which can perform the clustering of visual class distribution in a series of corresponding semantic subspaces. For the annotation strategy, hierarchical ensembles are adopted and the task of automatic image annotation is formulated as a procedure of classifying visual terms into semantic categories (keywords) through two levels. The ensemble of classifiers at the base level can provide a probabilistic label vector to each image segment, and then the second-level model re-weights the confident value of each possible label by considering a word co-occurrence language model underlying the annotation lexicon. Through the two-level annotation technique, we can get a more accurate and consistent annotations for unlabeled images.

Table 4.5. Experimental results with average precision and recall

Annotation Method	Average precision	Average recall
Translation Model	0.12	0.21
CMRM	0.14	0.24
SCK-means + HE	0.21	0.36

5 Conclusion and Future Work

In this paper, we propose a new approach for automatic image annotation and retrieval using semantics-constrained K-means together with hierarchical ensembles. Compared to other more classical methods, the proposed model gets better annotation and retrieval results, suggesting the importance of language models in semantic image retrieval. But some major challenges still remain:

- 1) Semantically meaningful segmentation algorithms are still not available, so the segmented region may not correspond to a semantic object and region features are insufficient to describe the image semantics.
- 2) The basic visual vocabulary construction using semantic constraints enhance the quality of clustering to some degree. But in some cases, it may still cross semantic boundaries, because the degree of difference in the visual space is higher than that of similarity on semantics between two images and the unique cost P can't balance the semantic relevance and visual similarity efficiently.
- 3) Our annotation task mainly depends on the trained model linking image features and keywords, the spatial context information of image regions and the word correlations are not fully taken into consideration.

In the future, more work should be done on image segmentation techniques, clustering algorithms, appropriate feature extraction and contextual information between regions and semantic relationships between keywords to improve the annotation accuracy and retrieval performance.

Acknowledgements

We would like to express our deepest gratitude to Michael Ortega-Binderberger, Kobus Barnard and J.Wang for making their image datasets available. The research is supported in part by the National Natural Science Foundation of China under grant number 60321002, and the Tsinghua-ALVIS Project co-sponsored by the National Natural Science Foundation of China under grant number 60520130299 and EU FP6.

References

1. K. Barnard, P. Dyugulu, N. de Freitas, D. Forsyth, D. Blei, and M. I. Jordan. Matching words and pictures. *Journal of Machine Learning Research*, 3: 1107-1135, 2003.
2. P. Duygulu, K. Barnard, N. de Freitas, and D. Forsyth. Object recognition as machine translation: Learning a lexicon from a fixed image vocabulary. In *Proceedings of Seventh European Conf. on Computer Vision*, 97-112, 2002.
3. J. Jeon, V. Lavrenko and R. Manmatha. Automatic image annotation and retrieval using cross-media relevance models. In *Proceedings of the 26th intl. SIGIR Conf*, 119-126, 2003.
4. Y. Mori, H. Takahashi, and R. Oka. Image-to-word transformation based on dividing and vector quantizing images with words. In *Proceedings of the First International Workshop on Multimedia Intelligent Storage and Retrieval Management*, 1999.
5. Edward Chang, Kingshy Goh, Gerard Sychay and Gang Wu. CBSA: Content-based soft annotation for multimodal image retrieval using Bayes point machines. *IEEE Transactions on Circuits and Systems for Video Technology Special Issue on Conceptual and Dynamical Aspects of Multimedia Content Descriptions*, 13(1): 26-38, 2003.
6. J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions On Pattern Analysis and Machine Intelligence*, 22(8): 888-905, 2000.
7. J. Li and J. A. Wang. Automatic linguistic indexing of pictures by a statistical modeling approach. *IEEE Transactions on PAMI*, 25(10): 175-1088, 2003.
8. V. Lavrenko, R. Manmatha and J. Jeon. A model for learning the semantics of pictures. In *Proceedings of the 16th Annual Conference on Neural Information Processing Systems*, 2004.
9. Chih-Chung Chang and Chih-Jen Lin. LIBSVM: a library for support vector machines, 2001. Software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
10. Chih-Wei Hsu and Chih-Jen Lin. A comparison of methods for multi-class support vector machines. *IEEE Transactions on Neural Networks*, 415-425 (2002).
11. Cusano C, Ciocca G, Schettini R. Image Annotation using SVM. In *Proceedings of SPIE-IS&T Electronic Imaging*, 330-338, SPIE Vol. 5304 (2004).
12. C. Fellbaum. *WordNet: An electronic lexical database*, MIT Press, 1998.
13. Pedersen T., Patwardhan S., and Michelizzi J. WordNet::Similarity - measuring the relatedness of concepts. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI-04)*, 2004.

14. R. Jin, J. Y. Chai, and L. Si. Effective automatic image annotation via a coherent language model and active learning. In Proceedings of ACM International Conference on Multimedia. ACM, New York, 892-899, 2004.
15. K. Barnard and D. A. Forsyth. Learning the semantics of words and pictures. In Proceedings of International Conference on Computer Vision, 408-415, 2001.
16. Fan, J., Gao, Y., and Luo, H. Multi-level annotation of natural scenes using dominant image components and semantic concepts. In Proceedings of the ACM International Conference on Multimedia. ACM, New York, 540 -547, 2004.
17. K. Wagstaff, C. Cardie, S. Rogers, and S. Schroedl. Constrained k-means clustering with background knowledge. In Proceeding of 18th ICML, 557-584, 2001.
18. D. Blei and M. I. Jordan. Modeling annotated data. In Proceedings of the 26th intl. SIGIR Conf, 127-134, 2003.

Creating a Testbed for the Evaluation of Automatically Generated Back-of-the-Book Indexes

Andras Csomai and Rada F. Mihalcea

University of North Texas, Computer Science Department
csomaia@unt.edu, rada@cs.unt.edu

Abstract. The automatic generation of back-of-the book indexes seems to be out of sight of the Information Retrieval and Natural Language Processing communities, although the increasingly large number of books available in electronic format, as well as recent advances in keyphrase extraction, should motivate an increased interest in this topic. In this paper, we describe the background relevant to the process of creating back-of-the-book indexes, namely (1) a short overview of the origin and structure of back-of-the-book indexes, and (2) the correspondence that can be established between techniques for automatic index construction and keyphrase extraction. Since the development of any automatic system requires in the first place an evaluation testbed, we describe our work in building a gold standard collection of books and indexes, and we present several metrics that can be used for the evaluation of automatically generated indexes against the gold standard. Finally, we investigate the properties of the gold standard index, such as index size, length of index entries, and upper bounds on coverage as indicated by the presence of index entries in the document.

1 Introduction

"Knowledge is of two kinds. We know a subject ourselves, or we know where we can find information on it." (Samuel Johnson)

The automatic construction of back-of-the-book indexes is one of the few tasks related to publishing that still requires extensive human labor. While there is a certain degree of computer assistance, mainly consisting of tools that help the professional indexer organize and edit the index, there are however no methods or tools that would allow for a complete or nearly-complete automation. Despite the lack of automation in this task, there is however another closely related natural language processing task – keyphrase extraction – where in recent years we have witnessed considerable improvements.

In this paper, we argue that the task of automatic index construction should be reconsidered in the light of the progress made in the task of keyphrase extraction. We show how, following methodologies used for the evaluation of keyphrase extraction systems, we can devise an evaluation methodology for back-of-the-book indexes, including a gold standard dataset and a set of evaluation metrics.

We hope that this evaluation testbed will boost the research in the field of automatic index construction, similar to the progress made in other NLP areas following the deployment of an evaluation framework¹.

Specifically, in this paper: (1) We shortly overview the origin and typical structure of back-of-the-book indexes; (2) We show that a close correspondence can be established between techniques for automatic index construction and keyphrase extraction, and consequently we briefly review the state-of-the-art for the latter problem; and finally (3) We describe our work in creating a testbed for the evaluation of automatic indexing systems, including a dataset of books and indexes, and a proposed set of evaluation metrics. We also discuss the properties of the gold standard, such as index size, length of index entries, and upper bounds on coverage as indicated by the presence of index entries in the document.

2 Definition, Origins, and Structure

The history of indexing dates back to ancient times. Several authors link the name *index* to the little papyrus slips (also called index) attached to papyrus scrolls in Roman libraries, containing the name of the author and the title of the document, and often also a small extract, which would allow the identification of the scroll without opening it. Several examples of index uses can be found throughout the following ages, but the real boost came in the nineteenth century, when the basic structure of the back-of-the-book indexes was defined.

While there are several definitions of what a back-of-the-book index is, the most complete and recent definition is perhaps the one provided in [7]. According to this definition, the index should enumerate all the words or phrases that refer to information that will most probably be sought by a reader. Specifically:

1. an index is a guide to names, places, items, concepts in a document or collection of documents;
2. the items or concepts in the index are arranged systematically, generally in alphabetical order; and
3. there are references to where each of these items are located in the document or documents.

The style of a back-of-the-book index has undergone several changes during its long history, arriving to the current more or less standard appearance, where each index entry contains the following two components [5]: (1) a *heading* including the indexing word or phrase, and (2) one or more *page reference numbers* and/or *cross references*. The page reference number shows the page or pages where the information relevant to the entry is located, while the cross-reference points to related entries, generally consisting of a synonym (marked by “*see ...*”), or other topically related terms (marked by “*see also...*”). When there are several entries referring to the same concept, they are usually ordered hierarchically under the heading that describes the shared concept.

¹ See for instance the progress in machine translation following the release of the Bleu evaluation metric, or the large number of publications on the problem of semantic entailment following the availability of the Pascal entailment dataset.

illustrations, indexing of, 108 in newspaper indexes, 147 in periodical indexes, 137 indexes, 399–430 author title, 429, 444 column width in, 423, 444 editing, 51 first page, number of, 81 indexes vs. indices, 129 justified lines in, 424	Jackson–Harmsworth Expedition, 216 Jeannette, the, xxix Johansen, Lieut., xxx, 132 Jones, Cape, 557 Kayaks, Nansen’s use of, xxxi Keltie Glacier, 358 Killer whale. See Whale, killer King Edward VII.’s Land, xxxiv, xlviii Kinsey, Mr. J. J., 48 Knight, E. F., 12, 18
--	---

Fig. 1. Examples of entries in a back-of-the-book index

Figure 1 shows two sample snapshots from a back-of-the-book index, illustrating the structure of index entries (headings, references, cross-references), the various types of index entries (names of persons, locations, terminology, important concepts in the text), and the hierarchical organization.

Index entries are often composed of more than one word, which results in *compound headings*. Typically, for such compound headings, indexing guidelines indicate that the head word has to be listed first, mainly for the purpose of an alphabetical ordering, which leads to the so-called *inversion*. As an example, consider the *indexing of illustrations* entry shown in Figure 1, which was changed to *illustrations, indexing of* through the process of inversion. The inversion can sometimes lead to hard-to-read headings like *children, disabled, hospitals, for* for the phrase *hospitals for disabled children*, and consequently recent work on manual indexing has discouraged the overuse of inversion.

Another important aspect of the index is the length. The literature usually defines the length of an index as a ratio between the number of pages of the index and the number of pages of the text. The length is typically affected by several factors, including the topic and specificity of the text. Less domain-specific texts such as children books or elementary school textbooks require indexes with a length accounting for about 1–3% of the length of the book, while highly specialized monographs on scientific topics may require indexes with a length of up to 15% of the text. History, biography and undergraduate textbook indexes are usually within the 5–8% range.

Finally, the content of the index, just like the length, also depends on the topic of the text. For instance, biographies tend to contain a larger number of names of persons and locations, while scientific books contain more entries referring to technical concepts and terminology.

3 Back-of-the-Book Indexing and Keyphrase Extraction

As mentioned before, an index is typically composed of names of persons or locations, terminology, and important concepts. Some of these index entries can be easily identified with a name entity recognizer as for instance the one described in [6], which automatically labels all entities that represent persons, locations, or organizations. The most difficult part of the index is however represented by the

so-called *important concepts*, which consist of words or phrases that are neither person names, nor locations, and yet represent important elements in the text. This is the part that is typically handled by the keyphrase extraction methods which target the automatic identification of important concepts in a text.

The task of automatic generation of back-of-the-book indexes can be therefore defined as a compound task consisting of (1) identifying named entities and (2) extracting keyphrases, followed by a post-processing stage that combines the output of the two tasks in a way that follows the traditional indexing guidelines. Consequently, the indexing task can be accomplished by using a named-entity recognizer coupled with a keyphrase extraction system. Since in recent years the named-entity recognition task has achieved relatively high levels of performance², for the remainder of this section we concentrate on the state-of-the-art in keyphrase extraction, as this represents the most difficult aspect of index construction. Note that we focus on keyphrase *extraction*, as opposed to keyphrase *generation*, since the former is a more feasible goal for current automatic systems.

The main approaches to keyphrase extraction can be divided into supervised methods that require the availability of a (sometimes large) training corpus, and unsupervised approaches that require only unlabeled data and eventually a very small set of annotated seeds.

Supervised Keyword Extraction. All the supervised keyword extraction methods that were developed so far appear to share a common framework: they start with a preprocessing step that handles the extraction and filtering of candidate phrases, followed by the actual ranking of the keywords using a set of contextual features and a standard machine learning algorithm.

In some cases the preprocessing stage also performs several transformations on the input data set, such as stemming or lemmatisation, changing the capital letters into lower case, etc. Next, candidate phrases are extracted, typically using one of the following three methods:

1. *n-grams*: all n-grams extracted from the document, usually covering uni-grams, bigrams, and trigrams [1], since they account for approximately 90% of the keyphrases.
2. *np-chunks*: a syntactic parser is employed to find np chunks in the document; this usually leads to increased precision at the cost of lower recall.
3. *syntactic patterns*: a part-of-speech tagger is used to label all the words in the document, and candidate phrases are extracted according to a predefined set of part-of-speech patterns.

Perhaps the most important step in supervised keyword extraction is the ranking of candidate phrases, usually performed using a machine learning algorithm, which can range from Naive Bayes [1,2,10], to rule induction [3] and genetic algorithms [9]. In terms of features, several have been proposed so far, including:

1. *tf.idf*: A weighting factor based on term frequency inverse document frequency feature, as defined in information retrieval.

² See for instance the state-of-the-art systems from the recent CoNLL 2002 and CoNLL 2003 shared tasks.

2. *tf* and *idf*: Sometimes term frequency and inverse document frequency are not combined, thus allowing the learning algorithm to eventually improve on the *tf.idf* combination of the two features.
3. *distance*: The distance of the phrase from the beginning of the document, usually measured by the number of individual words preceding the candidate phrase.
4. *POS pattern*: The part-of-speech pattern of the candidate phrase
5. *length*: The length of the candidate phrase. The distribution of the length of human expert assigned keywords, as reported by [3], shows that 13.7% of the human assigned keyphrases contain a single term, 51.8% contain two terms, and 25.4% contain three terms.
6. *stemmed forms*: The frequency of the stemmed word forms
7. *syntactic elements*: Binary features showing the presence of an adjective at the end of the phrase, or the presence of a common verb anywhere in the phrase [9]
8. *domain specific features*: Using a domain-specific hierarchical thesaurus and features indicating the presence of semantically related terms, an almost spectacular jump in recall was reported in [4], from 64% to 94%.
9. *coherence feature*: A new feature based on the hypothesis that candidates that are semantically related to one another tend to be better keyphrases is introduced in [10]. The semantic relatedness of the candidate terms is estimated by a measure of mutual information (pointwise mutual information), with the help of a search engine.

In terms of performance, supervised keyword extraction systems usually exceed by a large margin the simple frequency-based baselines. The best system was reported in [3] with an F-measure of 41.4%, comparing the automatically generated keyphrase set against human expert assigned keywords on a corpus containing scientific article abstracts. [2] reports an F-measure of 23%, also calculated based on author assigned keywords, but on a collection of full length computer science technical reports, which is a more difficult task than extracting keywords from abstracts. Finally, [9] reports a precision of around 25% over a large and varied collection. They also performed a manual evaluation of acceptability, and reported an 80% acceptability rate.

Unsupervised Methods. Unsupervised methods generally rely on variations of *tf.idf* or other similar measures, in order to score the candidate phrases. The method proposed in [11] extracts a set of candidate terms (only nouns), and ranks them according to their *relative frequency ratio*, which is in fact similar to *tf.idf*. First, only the terms with scores higher than a given threshold are kept, and all these terms are associated with their WordNet synsets. A pairwise semantic similarity score is calculated between all the terms, and a single link clustering is performed on the candidate set, using the similarity scores. Next, a cluster score and concept score are calculated, reflecting the "coherence" of the cluster (the sum of all pairwise similarities) and the overall "importance" of the cluster. The ranking of the candidate phrases is then performed with respect to the cluster and concept scores. The results of the method show clear improvement with respect to a baseline method that performs only *tf.idf* score ranking.

Another method is presented in [8], where keyword extraction is performed using language models. The method is intended to extract keyphrases not from a single document, but from a collection of documents. They make use of two document collections, called "background" and "foreground", with the later being the target set. They build n-gram language models for both the foreground and background corpora, with the goal of measuring the *informativeness* and *phraseness* of the phrases of the foreground corpus. The phraseness is defined as the Kullback-Liebler divergence of the foreground n-gram language model (see article), which represents the "information loss" by assuming the independence of the component terms. The "informativeness" is calculated by applying the same statistical measure to the foreground and background models. Once the informativeness and phraseness of the candidate phrases is defined, they can be combined into an *unified score* that can be used to order the candidate phrases.

4 Building an Evaluation TestBed for Back-of-the-Book Indexing

The construction of a gold standard benchmark that can be used for the evaluation of automatically generated back-of-the-book indexes requires a collection of books in electronic format, each of them with their corresponding index. We had therefore to: (1) identify a collection of books in electronic format, and (2) devise a method to extract their index entries in a format that can be used for automatic evaluations.

4.1 Collecting Books in Electronic Format

Currently one of the largest available on-line collection of electronic books is the Gutenberg project³, built as a result of volunteer contributors, and containing the electronic version of books that are in the public domain.

Project Gutenberg contains approximately 16,000 titles, however only very few of them include a back-of-the-book index, either because they never had one, or because the person who contributed the book decided not to include it. In order to find the books that contained their back-of-the-book index we used a search engine to identify those books in the Gutenberg collection that contained keywords such as *index of content*. Using an external search engine ensured a certain degree of topical randomness as well.

A problem that we noticed with the results obtained in this way was that many documents covered topics in the humanities, while very few books were from the scientific/technical domain. To ensure the presence of technical documents, we used the Gutenberg Project search engine to identify all the documents classified as *science* or *technology* according to the Library Of Congress Classification (LOCC) system, and manually extracted only those books that contained an index. As a result, we retrieved a total of 56 documents, out of which 26 have an LOCC classification. Table 1 shows the distribution of the books across different topics.

³ <http://www.gutenberg.org>

Table 1. Distribution of books across topics

Category	# books
Humanities	
History & Art	7
Literature & Linguistics	7
Psychology & Philosophy	7
Science	
Agriculture	1
Botany	4
Geography	2
Geology	2
Natural history	9
Zoology	6
Technology	
Electrical and nuclear	2
Manufacturing	1
Ocean engineering	1
Misc	7
TOTAL	56

4.2 Extracting Index Entries

Once we obtain a collection of books in electronic format, the next step is to extract the index in a format that can be used for the evaluation of automatically constructed back-of-the-book indexes.

First, we separate the index from the main body of the document. Next, since our long term goal is to devise methods for automatic discovery of index entries, and not referencing, all page numbers and cross references are removed, as well as special marks used by the transcriber, such as e.g. the symbol “-” used to emphasize a text as in *Institution name*.

Once we have a candidate list of index entries, the next step is to clean them up and convert them into a format suitable for automatic evaluation. The first problem that we faced in this process was the *inversion* applied to compound headings. As mentioned before, indexing guidelines suggest that the head word of an index phrase has to be listed first, to facilitate the search by readers within the alphabetically ordered index. However, in order to measure the performance of an automatic system for index generation, the index entries have to be reconstructed in the form they are most likely to appear in the document, if they appear at all. Starting with an index entry whose structure follows the standard indexing guidelines, we therefore try to create an English phrase that is likely to be found in the document. This reconstruction is sometimes very difficult, since the human indexers do not strive to create grammatically correct phrases. In some cases, even if we manage to correctly reorder the index entry (e.g. list of modifiers followed by head word), the resulting phrase may not be always proper English, and therefore it is very likely that it will not be identified by any indexing algorithm.

Table 2. Examples of index entries and their reconstructions

1	Acetate, of Ammonium Solution, Uses of	uses of Acetate of Ammonium Solution
2	Goldfinch, American	American goldfinch
3	Goose, Domestic	domestic goose
4	Cainozoic, term defined	cainozoic term defined
5	France, history of the use of subsidies in, the navigation laws of, commercial treaty between England and, the Merchant Marine Act of,	history of the subsidies in France the navigation laws of France commercial treaty between England and France the Merchant Marine Act of France

The reconstruction algorithm is based on the presence of prepositions. As shown in table 2, the sequences of the original scrambled index entry are delimited by commas into smaller units. We devised several heuristics that can be used to recover the original order of these components. In the case of prepositional phrase units, the preposition is a strong clue about the placement of the phrase relative to the head-phrase, e.g. the preposition *to*, *in*, *for* at the beginning of the phrase suggests that it should follow the head-phrase, whereas the preposition *as*, *from*, *of*, *among* at the end of the phrase suggests that it should precede the head; see for instance example 1 in table 2. Similarly, the position of the conjunction *and* determines the placement of the phrase that contains it.

When there are no prepositions or conjunctions, the reconstruction becomes more complicated. We were able however to identify several patterns that occur fairly often, and use these patterns in the reconstruction process: (1) If the second component is a modifier of the head-phrase (adjective or adverb, or corresponding phrase) then it should be placed before the head; see for instance examples 2 and 3 in table 2. (2) If the second phrase contains an explanation referring to the head, or some additional information, then it should be placed after the phrase head. Note that the structures corresponding to the second pattern can sometime lead to ungrammatical phrases, as for example the phrase 4 in table 2. In such cases, the phrase will be post-processed using the filtering step described below. Reconstructing the index entries based on the mentioned patterns is only a back-off solution for the cases where no prepositions were found in the entry. We attempt to determine which is the most frequent pattern at the document level (based on the number of the index entries reconstructed using the selected pattern and found in the document), and use it throughout the index. This pattern selection is individually carried out for every document.

The hierarchic structures (see example 5 in table 2) are reconstructed by attaching the head-phrase of the entry from the higher level to all its descendants. This results in a set of compound entries, usually inverted, which can be reconstructed using the heuristics described before.

4.3 Index Granularities

Following the example of other NLP tasks that allow for different levels of granularity in their evaluation⁴, we decided to build gold standard indexes of different

⁴ For instance, word sense disambiguation gold standards allow for the evaluation of systems that can perform either coarse-grained or fine-grained disambiguation.

granularities. This decision was also supported by the fact that compound index terms are sometimes hard to reconstruct, and thus different reconstruction strategies pose different levels of difficulty.

We decided to extract two different indexes for every text: (1) a short index, consisting only of head phrases, which allows us to evaluate a system’s ability to extract a coarse-grained set of index entries; and (2) a long index, containing the full reconstructed index entries, which corresponds to a more fine-grained indexing strategy.

As pointed out earlier, the reconstruction of the inverted compound entries is fairly difficult, therefore the fine grained index will sometimes contain ungrammatical phrases that could never be found by any extraction algorithm. Consequently, we decided to also create a third, filtered index, that excludes these ungrammatical phrases and allows us to measure a system performance with a higher upper bound, meaning that a larger number of index entries are present in the text and could be potentially found by an automatic indexing system. We use a simple filtering method that measures the frequency of each index entry on the Web, as measured using the AltaVista search engine. If the number of occurrences is higher than a given threshold n , we consider the phrase plausible. If the frequency is below the threshold, the entry is discarded. Finding a good value for the threshold n may be a difficult issue, since a large value will allow for the inclusion of longer phrases with small occurrence probability, while a small value may let many incorrect phrases slip through. In our experiment we use a value of $n = 2$, which was empirically determined, and resulted in the elimination of roughly 50% of the fine grained entries.

4.4 Properties of the Gold-Standard Collection

Starting with the gold standard collection described in the previous section, we measured several properties of back-of-the-book indexes, such as length of the index entries, index size, and upper bounds on coverage as indicated by the presence of index entries in the document.

The length of the index entries can influence the accuracy of the index extraction algorithm and the choice of methods used for candidate phrase extraction. For instance, in the case of coarse-grained indexes, most of the index entries consist of four words or less, and therefore a four-gram model would probably be sufficient. On the other side, when fine grained entries are used, larger phrases are also possible, and thus longer n -grams should be considered. Table 3 shows the distribution by length of the gold-standard index entries.

Another important aspect of the gold-standard index is whether it includes entries that can be found in the text, which impacts the value of the recall

Table 3. Distribution of index entries by length (defined as number of tokens)

Index type	Length of index entry										
	1	2	3	4	5	6	7	8	9	10	>10
Coarse grained	31312	9068	2232	1268	642	311	162	56	36	13	8
Fine grained	8250	9352	7627	7057	5787	4500	3188	1906	1241	727	862
Filtered	7500	8112	4590	2191	1151	657	437	303	186	138	171

Table 4. Presence of index entries in the original text

Length of index entry	index style		
	coarse grained	fine grained	filtered index
1	92.95%	92.20%	93.22%
2	72.89%	48.80%	52.59%
3	48.68%	23.75%	36.75%
4	28.36%	10.16%	27.06%
5	16.73%	4.15%	16.95%
6	9.93%	1.99%	8.53%
7	8.66%	0.85%	3.45%
8	12.77%	0.79%	3.97%
9	3.33%	0.32%	1.08%
10	8.33%	0.55%	0.74%
Total	81.29%	30.34%	54.78%

upper bound that can be achieved on the given index. This aspect is of particular interest for methods that create indexes by extracting candidate phrases from the text, rather than generating them. To determine the average value for this upper bound, we determined the number of index entries that appeared in the text, for each of the three index types (fine-grained, coarse-grained, filtered index). The results of this evaluation are shown in table 4. Not surprisingly, the smallest coverage is observed in the case of the fine-grained indexes, followed by the filtered indexes and the coarse-grained indexes. It is also worth noting that the Web-based filtering process increases the quality of the index significantly, from a coverage of 30.34% to 54.78%.

Finally, another important property of the index is its size relative to the length of the document. We measured the ratio of the number of entries in the index and the number of tokens in the text. On average, the coarse-grained indexes contain about 0.44% of the text tokens, which corresponds roughly to one coarse grained keyphrase for every 227 words of text. The fine-grained indexes have a ratio of 0.7%, which represents one index phrase for every 140 words in the document.

4.5 Evaluation Metrics

Finally, another important aspect that needs to be addressed in an evaluation framework is the choice of metrics to be used. Provided a gold standard collection of back-of-the-book indexes, the evaluation of an automatic indexing system will consist of a comparison of the automatically extracted set of index entries against the correct entries in the gold standard. We propose to use the traditional information retrieval metrics, *precision* and *recall*. Precision measures the accuracy of the set automatically extracted, as indicated by the ratio of the number of correctly identified entries and the total number of proposed entries. Recall is defined as the ratio of the number of correctly identified entries and the total number of correct entries in the gold standard.

$$\textit{precision} = \frac{\textit{extracted and correct}}{\textit{extracted}}$$

$$\textit{recall} = \frac{\textit{extracted and correct}}{\textit{correct}}$$

In addition, the F-measure combines the precision and recall metrics into a single formula:

$$F - \textit{measure} = \frac{2 * \textit{precision} * \textit{recall}}{\textit{precision} + \textit{recall}}$$

Moreover, we also suggest the use of a “relative recall”, which represents the ratio between the traditional recall as defined earlier, and the maximum recall that can be achieved on a given gold standard index using only entries that literally appear in the text. The relative recall is therefore defined as the fraction of correctly identified index entries and the total number of entries from the gold standard that appear in the text.

$$\textit{recall}_r = \frac{\textit{extracted and correct}}{\textit{correct and in text}}$$

This measure targets the evaluation of systems that aim to extract indexes from the books, rather than generating them. Correspondingly, we can also define an F-measure that takes into account the precision and the relative recall.

5 Conclusion and Future Work

In this paper, we described our work in creating an evaluation testbed for automatic back-of-the-book indexing systems. We also overviewed the background of back-of-the-book indexing and current trends in keyphrase extraction that are relevant to this problem. The long term goal of this work is to devise an automatic method for building back-of-the-book indexes. Since no evaluation framework is currently available for this task, we had to start our work in this project by creating a testbed that will allow for the comparative evaluation of a variety of indexing methods.

We plan to extend our collection by splitting the index entries into named entities and important concepts, which will allow for a separate evaluation of the named entity recognition and the keyphrase extraction components. We also plan to include a larger number of contemporary books, in order to eliminate the discrepancies arising from the stylistic variety due to the age of the books in our collection.

The data set described in this paper is publicly available for download from <http://www.textrank.org/data>.

Acknowledgments

This work was partially supported by an award from Google Inc.

References

1. FRANK, E., PAYNTER, G. W., WITTEN, I. H., GUTWIN, C., AND NEVILL-MANNING, C. G. Domain-specific keyphrase extraction. In *Proceedings of the International Joint Conference on Artificial Intelligence* (1999).
2. GUTWIN, C., PAYNTER, G., WITTEN, I., NEVILLMANNING, C., AND FRANK, E. Improving browsing in digital libraries with keyphrase indexes, 1998.
3. HULTH, A. *Combining Machine Learning and Natural Language Processing for Automatic Keyword Extraction*. PhD thesis, Stockholm University, 2004.
4. HULTH, A., KARLGREN, J., JONSSON, A., AND ASKER, H. B. L. Automatic keyword extraction using domain knowledge. In *Proceedings of the Conference on Computational Linguistics and Intelligent Text Processing* (2001).
5. KNIGHT, N. *Indexing, the art of*. Allen & Unwin, 1979.
6. MIKHEEV, A., MOENS, M., AND GROVER, C. Named entity recognition without Gazetteers. In *Proceedings of the European Association for Computational Linguistics* (Bergen, Norway, 1999).
7. PRASHER, R. *Index and Indexing Systems*. Medallion Press, 1989.
8. TOMOKIYO, T., AND HURST, M. A language model approach to keyphrase extraction. In *Proceedings of ACL Workshop on Multiword Expressions* (2003).
9. TURNEY, P. Learning algorithms for keyphrase extraction. *Information Retrieval* 2, 4 (2000).
10. TURNEY, P. Coherent keyphrase extraction via web mining. In *Proceedings of the International Joint Conference on Artificial Intelligence* (Acapulco, Mexico, 2002).
11. VAN DER PLAS, L., PALLOTTA, V., RAJMAN, M., AND GHORBEL, H. Automatic keyword extraction from spoken text. a comparison of two lexical resources: the EDR and WordNet. In *Proceedings of the Language Resources and Evaluations Conference* (Lisbon, Portugal, 2004).

Automatic Acquisition of Semantic-Based Question Reformulations for Question Answering

Jamileh Yousefi and Leila Kosseim

CLaC laboratory,
Department of Computer Science and Software Engineering,
1400 de Maisonneuve Blvd. West,
Montreal, Quebec, Canada H3G 1M8
j_yousefi@cs.concordia.ca, kosseim@cs.concordia.ca

Abstract. In this paper, we present a method for the automatic acquisition of semantic-based reformulations from natural language questions. Our goal is to find useful and generic reformulation patterns, which can be used in our question answering system to find better candidate answers. We used 1343 examples of different types of questions and their corresponding answers from the TREC-8, TREC-9 and TREC-10 collection as training set. The system automatically extracts patterns from sentences retrieved from the Web based on syntactic tags and the semantic relations holding between the main arguments of the question and answer as defined in WordNet. Each extracted pattern is then assigned a weight according to its length, the distance between keywords, the answer sub-phrase score, and the level of semantic similarity between the extracted sentence and the question. The system differs from most other reformulation learning systems in its emphasis on semantic features. To evaluate the generated patterns, we used our own Web QA system and compared its results with manually created patterns and automatically generated ones. The evaluation on about 500 questions from TREC-11 shows comparable results in precision and MRR scores. Hence, no loss of quality was experienced, but no manual work is now necessary.

1 Introduction

Question reformulation deals with identifying possible forms of expressing answers given a natural language question. These reformulations can be used in a QA system to retrieve answers in a large document collection. For example given the question *What is another name for the North Star?*, a reformulation-based QA system will search for formulations like $\langle NP \rangle$, *another name for the North Star* or $\langle NP \rangle$ *is another name for the North Star* in the document collection and will instantiate $\langle NP \rangle$ with the matching noun phrase. The ideal reformulation should not retrieve incorrect answers but should also identify many candidate answers.

Writing reformulations by hand is a tedious tasks that must be repeated for each type of question and for each language in the case of a multilingual QA

system. This is why there have been many attempts at acquiring reformulations automatically (ex. [1,2,3,4,5]). However, most work have analyzed string-based or syntactic paraphrases and few have worked on generating semantically equivalent reformulations such as *<NP>*, also known as *the North Star* or *the North Star is also called <NP>*.

Our goal is to learn semantically equivalent reformulation patterns automatically from natural language questions. We hope to find useful reformulation patterns, which are general enough to be mapped to potential answer contexts but specific enough not to retrieve wrong answers.

2 Related Work

Soubbotin et al. [6] along with [7] were among the first to use surface patterns as the core of their QA system. This approach searches in the document collection for predefined patterns or exact sentences that could be the formulation of the potential answer. In [6], the patterns were hand-crafted, while in [7] simple word permutations were performed to produce paraphrases of the question. More recently, [1] also uses simple word permutations and verb movements to generate paraphrases for their multilingual QA system.

In the work of [5,4,8], answer formulations are produced for query expansion to improve information retrieval. While in [8] reformulation rules to transform a question of the form *What is X?* into *X is* or *X refers to* are built by hand, [4,5] learns to transform natural language questions into sets of effective search engine queries, optimized specifically for each search engine.

[9] use a machine learning technique and a few hand-crafted examples of question-answer pairs to automatically learn patterns along with a confidence score. However, the patterns do not contain semantic information. They include specific string of words such as *was born on*, *was born in*, . . . with no generalisation of the *is-born* relation. [2] does use semantic paraphrases, called *phrasal synonyms*, to enhance their TextMap QA system. However, many of these patterns are manual generalisations of patterns derived automatically by [9].

[10] use transformational grammar to perform syntactic modifications such as Subject-Aux and Subject-Verb movements. [11] learn the best query reformulations (or paraphrases) for their probabilistic QA system. Here again, the paraphrases are syntactic variations of the original question.

[12], however, do try to learn semantically equivalent reformulations by using the web as a linguistic resource. They start with one single prototypical argument tuple of a given semantic relation and search for potential alternative formulations of the relation, then find new potential argument tuples and iterate this process to progressively validate the candidate formulations.

In these systems and most similar approaches, automatic paraphrases are constructed based on lexical or syntactic similarity. When searching a huge document collection such as the Web, having only syntactic reformulations is acceptable because the collection exhibits a lot of redundancy. However, in a smaller collection, semantic reformulations are necessary.

3 Learning Reformulation Patterns

3.1 Question and Answer Patterns

Our work is based on our current reformulation-based QA system [13,14], where reformulations were hand-crafted. Given a question, the system needs to identify which answer pattern to look for. It therefore uses:

A question pattern: that defines what the question must look like. For example *Who Vsf PERSON?* is a question pattern that matches *Who is George Bush?*

An answer pattern: Once a question pattern is activated (is found to match the input question) a set of answer patterns will be looked for in the document collection. An answer pattern specifies the form of sentences that may contain a possible candidate answer. For example, for the question *Who is George Bush?*, the system tries to find sentences that match any one of these answer patterns:

```
<QT> <Vsf> <ANSWER>
<ANSWER> <Vsf> by <QT>
```

Where *<ANSWER>* is the candidate answer, *<QT>* is the question term (i.e. *George Bush*), and *<Vsf>* is the verb in simple form.

In the current implementation, both sets of patterns are hand-crafted using the following types of tags:

- Named-entity tags (e.g. PERSON) – found using the GateNE named entity tagger [15].
- Part-of-speech tags (e.g. Vsf) – found using [16]
- Tags on strings (e.g. QT, ANY-SEQUENCE-WORDS)
- Specific keywords (e.g. Who, by)

In this paper, we will discuss how answer patterns can be discovered automatically.

3.2 The Training Corpus

Our learning algorithm starts with a training corpus of 1343 question-answer pairs taken from the TREC-8, TREC-9, and TREC-10 collection data [17,18,19]. Each question-answer pair is composed of one question and its corresponding answer. The following are some examples:

```
Where is the actress,Marion Davies,buried? Hollywood Memorial Park
When did Nixon die? April 22, 1994
Who is the prime minister of Australia? Paul Keating
```

Table 1. Training corpus files according to the type of question. The size is the number of question-answer pairs.

Corpus	Size
Who Corpus	208
Where Corpus	119
When Corpus	88
What Corpus	747
Why Corpus	8
Which Corpus	32
How Corpus	111
Other Corpus	30
Total	1343

We divided the training corpus according to the question type. Questions are classified depending on the kind of information sought. In fact, one major factor to guessing the answer type is to know the question type. We used the classification used in [20] to categorize questions into 7 main classes (what, who, how, where, when, which, why) and 20 subclasses (ex. what-who, who-person, how-many, how-long, ...). Table 1 shows our training files along with their sizes (number of question-answer pairs).

3.3 Overview of the Algorithm

Each question-answer pair is analyzed to extract the arguments and the semantic relation holding between the question arguments and the answer. A query is then formulated using the arguments extracted from the question-answer pair. The formulated query is sent to a Web search engine which returns the N most relevant documents. The sentences that contain all the query terms are then filtered to keep only these that contain the same semantic relation. These are then passed to a sentence splitter, a part-of-speech tagger, and a noun phrase chunker, to select be generalized into an answer pattern using syntactic and semantic tags. Finally, a confidence weight is assigned to each generated pattern according to its semantic distance from the question and the frequency of the pattern. Let us now describe each of these steps in detail.

4 Generating Answer Patterns

The goal here is to find many sentences from the document collection (here, the Web) that contain the answer and see if we can generalize them into syntactico-semantic pattern.

4.1 Extracting Arguments

For each question-answer pair, we define an argument set as the set of terms which we believe a relevant document should contain. To give an example, consider the question-answer pair:

Q: *Who provides telephone service in Orange County, California?*

A: *Pacific Bell*

Any relevant document to this question-answer pair must contain the terms “*telephone service*”, “*Orange County, California*”, and “*Pacific Bell*”. Therefore to search documents on the Web, we formulate a query made up of all the arguments found in the question-answer pair. To obtain the argument sets, the question is chunked (with the BaseNP chunker [16]) to identify its base noun phrases. All the base noun phrases detected in the question are grouped in a set called Q-ARGUMENTS.

$$\text{Q-ARGUMENTS} = \{ \text{‘‘telephone service’’}, \text{‘‘Orange County, California’’} \}$$

In the TREC 8-11 collections, the candidate answer is typically a noun phrase that can contain one or more words. Some supporting documents may only contain part of this noun phrase. To increase the recall of document retrieval, we search for a combination of question arguments and each sub-phrase of the answer. We restrict each sub-phrase to contain less than four¹ words and to contain no stop word. Finally, we assign a score to each sub-phrase according to proportion of the words in the sub-phrase compared to the total number of words in the candidate answer. For example, the sub-phrases and the score assigned for the previous question-answer pair are:

Pacific Bell	1	Pacific	$\frac{1}{2}$	Bell	$\frac{1}{2}$
--------------	---	---------	---------------	------	---------------

The sub-phrase score will be used later to rank the patterns (and ultimately, the extracted answers) from the retrieved sentences (see section 4.6). Finally, we group the original candidate answer and all its sub-phrases in the set ANS-ARGUMENTS. For example,

$$\text{ANS-ARGUMENTS} = \{ (\text{Pacific Bell}, 1), (\text{Pacific}, \frac{1}{2}), (\text{Bell}, \frac{1}{2}) \}$$

4.2 Document Retrieval

At this stage, we construct a query in the format accepted by the Google search engine. The query is formed using all the arguments extracted from the question (Q-ARGUMENTS), and the original candidate answer or one of its sub-phrases (ANS-ARGUMENTS) are conjugated with arithmetic operators. For example,

‘‘telephone service’’ + ‘‘Orange County, California’’ + ‘‘Pacific Bell’’

We post the structured query to the Google search engine and then we scan the first 500 retrieved documents to identify the sentences that are likely to contain the answer. From the above documents, only those sentences that contain all of the question arguments and at least one answer argument are retained.

4.3 Extracting Semantic Relations

The key aspect of this research is to find reformulations that are semantically equivalent. To do this, we need to find sentences that contain equivalent semantic

¹ This limit was set arbitrary.

relations holding between question arguments and the answer. In fact, semantic relations are used in measuring the relevance of sentences with the question-answer pair. We assume that the semantic relation generally appears as the main verb of the question. For example, the verb ‘*provide*’ is considered as the semantic relation in the following question-answer pair:

Q: *Who provides telephone service in Orange County, California?*
 A: *Pacific Bell*

The representation of the above concepts is done in the following constructs:

Relation schema: ARGUMENT-1: *telephone service*
 RELATION: *provide*
 ARGUMENT-2: *Orange County, California*

If the main verb of the question is an auxiliary verb, then we ignore the semantic relation. For example, in:

Q: *Who is the president of Stanford University?*
 A: *Donald Kennedy*

The semantic relation is ignored and the semantic validation of the relevant sentence (see section 4.4) is based on the frequency of the answer context alone.

Once the semantic relation is determined, we generate a semantic representation vector composed of the relation word, its synonyms, one-level hyponyms and all hypernyms obtained from WordNet. This vector will serve later to verify the semantic validity of the sentences retrieved. To weight each answer pattern according to our confidence level, we also assign a weight to each term in the vector based on the semantic distance of the term to the original verb in WordNet. We want to estimate the likelihood that the sentence and the question actually refer to the same fact or event. We assign the similarity measure using the following weights:

- 1: for the original verb in the question.
- $\frac{1}{2}$: for strict synonyms of the question verb, i.e. a verb in the same synset.
- $\frac{1}{8}$: for hyponyms and hypernyms of the question verb.

Since the relation word can be polysemous, we consider all its possible senses. The representation of the above concepts is done in the following construction:

Relation	provide
Synonyms(provide)	{supply, render, offer, furnish, ... }
Hyponyms(provide)	{charge, date, feed, calk, fund, stint, ... }
Hypernyms(provide)	{give, transfer stipulate, qualify, ... }
Semantic representation vector	{(provide,1), (supply, $\frac{1}{2}$), (render, $\frac{1}{2}$), (offer, $\frac{1}{2}$), (furnish, $\frac{1}{2}$), ..., (charge, $\frac{1}{8}$), (date, $\frac{1}{8}$), (feed, $\frac{1}{8}$), ..., (give, $\frac{1}{8}$), (transfer, $\frac{1}{8}$), ... }

4.4 Semantic Filtering of Sentences

We then filter the set of sentences retrieved by Google, according to the validity of the semantic relation that they contain. We only choose sentences that have

the same relation as the relation extracted from the question-answer pair. To do so, we examine all verbs in the selected sentences for a possible semantic relation. We check if the main verb of the sentence is a synonym, hypernym, or hyponym of the original verb in the question. The verb is valid if it occurs in the semantic representation vector. For example, with our running example, both these sentences will be retained:

Sentence 1 *California's Baby Bell, SBC Pacific Bell, still provides nearly all of the local phone service in Orange County, California, California.*

Sentence 2 *Pacific Bell Telephone Services today offers the best long distance rate in Orange County, California.*

Because both sentences contain a verb (“provide” and “offer”) that is included in the semantic representation vector of the question verb (“provide”).

At first, we only attempt to validate verbs but if the semantic relation is not found through the verbs, then we also validate nouns and adjectives because the semantic relation may occur as a nominalisation or other syntactic and morpho-syntactic variations. In such a case, we use the Porter stemmer [21] to find the stem of the adjectives and nouns and then we check if it has the same stem as the original verb or another verb from its semantic representation vector. For example, for the phrase “provider of” we check if the stem of the original verb “provide” or one of its synonyms, hyponym or hypernym is the same as “provide” (the stem of “provider”). For example, the following sentence is also selected:

Sentence 3 *Pacific Bell, major provider of telephone service in Orange County, California...*

4.5 Generating the Answer Pattern

Once we have identified a set of sentences containing the answer, the arguments and the same semantic relation, we try to generalize them into a pattern using both syntactic and semantic features. Each sentence is tagged and syntactically chunked (with [16]) to identify POS tags and base noun phrases. To construct a general form for answer patterns, we replace the noun phrase corresponding to ANS-ARGUMENT by the tag <ANSWER> and the noun phrases corresponding to Q-ARGUMENTS by the tag <QARGx> where x is the argument counter. We replace the other noun phrases that are neither question arguments nor answer arguments with <NPx>, where x is the noun phrase counter. To achieve a more general form of the answer pattern, all other words except prepositions are removed. For example, the following sentence chunked with NPs:

[California's/NNP Baby/NNP Bell,/NNP SBC/NNP Pacific/NNP Bell,/NNP]/NP still/RB provides/VBZ nearly/RB all/DT of/IN [the/DT local/JJ phone/NN service/NN]/NP/ NP in/IN [Orange/NNP County,/NNP California./NNP]/NP

will generate the following pattern:

<ANSWER> <VERB> <QARG1> in <QARG2> | senseOf(provide)

The constraint `senseOf(provide)` indicates the semantic relation to be found in the candidate sentences through a verb, a noun or an adjective.

Finally, we replace the <ANSWER> tag with the corresponding named-entity tag. To do so, we tag the answer in the question-answer pair of the training set with the GateNE named entity tagger [15]. Since the question calls for an organization, the following is produced:

<ORGANIZATION> <VERB> <QARG1> in <QARG2> | senseOf(provide)

The answer patterns generated for our example question-answer pairs thus becomes:

<ORGANIZATION> <VERB> <QARG1> <QARG2> | senseOf(provide)
 <ORGANIZATION> <QARG1> <QARG2> | senseOf(provide)
 <ORGANIZATION> <QARG1> <VERB> <QARG2> | senseOf(provide)

4.6 Assigning Confidence Weights

As one pattern may be more reliable than another, the last challenge is to assign a weight to each candidate pattern. This helps us to better rank the pattern list, and ultimately the answer extracted from them, by their quality and precision. From our experiments, we found that the frequency of a pattern, its length, the answer sub-phrase score, and the level of semantic similarity between the main verbs of the pattern and the question are the most indicative factors in the quality of each pattern. We set up a function to produce a weight for each pattern over the above major factors; these weights are defined to have values between 0 and 1. More formally, let P_i be the i th pattern of the pattern set P extracted for a question-answer pair; we compute each factor as the following:

$count(P_i)$ is the number of times pattern P_i was extracted for a given question pattern. The most frequent the pattern, the more confidence we have in it and the better we rank it.

$distance$ measures the distance (number of words) between the answer and the closest term from Q-ARGUMENTS in the pattern. The smallest the distance, the more confidence we have in the pattern.

$length(P_i)$ is the length of the pattern P_i measured in words. A shorter pattern will be given a better rank.

sub_phrase_score is the score of the candidate answer sub-phrase. The score of each answer sub-phrase depends on its similarity to the full candidate answer.

Here we have used the simple heuristic method to score a sub-phrase by its length as $\frac{\text{number of words that are present in both } p_i \text{ and candidate answer}}{\text{total number of words in the candidate answer}}$.

$semantic_sim(V_Q, S_{P_i})$ measures the similarity between the sense expressed in the candidate pattern (S_{P_i}) (through a verb, a noun or an adjective) and the original verb in the question (V_Q). We want to estimate the likelihood that the two words actually refer to the same fact or event. Here, we use the weights given to terms in the semantic representation vector of (V_Q). As

described in section 4.3, this weight is based on the type of semantic relation between the terms and V_Q as specified in WordNet: 1 pt for the original verb in the question; $\frac{1}{2}$ pt for strict synonyms of the question verb and $\frac{1}{8}$ pt for hyponyms and hypernyms of the question verb.

The final weight for a pattern is based on the combined score of the previous four factors computed as:

$$\text{Weight}(P_i) = \frac{\text{count}(P_i)}{\text{count}(P)} \times \frac{1}{\text{length}(P_i)} \times \frac{1}{\text{distance}} \times \text{sub_phrase_score}(used) \\ \times \text{semantic_sim}(V_Q, S_{P_i})$$

Note that this function is not necessarily the optimal way of combining these contributing factors, nor are the factors complete by any means. However, as long as long as it produces an acceptable ranking, we can apply it to the patterns. The error produced by just a simple acceptable ranking function is negligible compared to the error present in other modules, such as the named entity recognizer.

Figure 1 shows an example of a question pattern along with its ranked answer patterns for the question *Who was the first man to fly across the Pacific Ocean?*

Weight	Answer Pattern
1.00	<QARG1> on <QARG2> <PERSON> senseOf(fly)
0.59	<QARG1> to <VERB> on <QARG2> <PERSON> senseOf(fly)
0.43	<PERSON> QARG1 on QARG2 senseOf(fly)
0.24	<QARG2> to <VERB> on <QARG2> <PERSON> senseOf(fly)

Fig. 1. Example of ranked answer patterns for the question *Who was the first man to fly across the Pacific Ocean?*

5 Evaluation

We tested our newly created patterns using the 493 questions-answers from the TREC-11 collection data [22]. We submitted these questions to our Web-QA

Table 2. Results for each question category with the original hand-crafted patterns

Question type	Nb of questions	Nb of questions with a correct answer in the top 5 candidates	Precision of candidate list
who	52	20	0.571
what	266	42	0.500
where	39	8	0.533
when	71	11	0.687
how + adj/adv	53	5	0.277
which	12	0	0
Total	493	92	0.538

Table 3. Results for each question category with the generated patterns

Question type	Nb of questions with a correct answer	Nb of questions in the top 5 candidates	Precision of candidate list
who	52	24	0.648
what	266	42	0.552
where	39	11	0.578
when	71	18	0.720
how + adj/adv	53	6	0.462
which	12	0	0
Total	493	113	0.646

Table 4. The results based on question categories

Question Type	Frequency	Hand-crafted patterns		Automatic patterns	
		MRR	Precision	MRR	Precision
who	52 (10.4%)	0.301	0.571	0.396	0.648
what	266 (53.2%)	0.229	0.500	0.317	0.552
where	39 (7.8%)	0.500	0.533	0.348	0.578
when	71 (14.2%)	0.688	0.687	0.643	0.720
how + adj/adv	53 (10.6%)	0.194	0.277	0.310	0.462
which	12 (2.4%)	0	0	0	0

system [13,14]. The system was evaluated with the original hand-crafted reformulation patterns and with learned ones. Then the answers from both runs were compared. Tables 2, 3 and 4. Tables 2 and 3 show the result of this comparison based on precision and the number of questions with at least one candidate answer. Table 4 shows the mean reciprocal rank (MRR) for each type of question. The evaluation shows comparable results in precision and MRR scores with a slight increase with the generated patterns. Hence, no loss of quality was experienced, but no manual work is now necessary.

Although the results show an increase in precision and MRR; we actually expected a greater improvement. However, we believe that the potential improvement is actually greater than what is shown. For now, the results are limited to the syntax of the patterns currently implemented in the QA system: the tags in the patterns that are recognized are very limited, preventing a greater granularity of patterns. For example, currently there is no differentiation between past, past participle, or third person verbs. This makes most of the new reformulated patterns we extracted not recognizable by the QA component.

6 Conclusion and Future Work

We presented a method for acquiring reformulations patterns automatically based on both syntactic and semantic features.

The experimental evaluation of our reformulations shows that using new generated patterns does not increase the precision and MRR significantly compared to hand-crafted rules, but do eliminate the need for human intervention.

As opposed to several other approaches that reinforce their candidate answers by looking on the Web; our approach is less strict as it looks for reinforcement of the semantic relation between the arguments, rather than looking only for lexically similar evidence. In this respect, our approach is much more tolerant and allows us to find more evidence. On the other hand, as we look for evidence that fit a certain pattern with many possible words fitting the pattern, rather than a strict string match, we are more sensitive to mistakes and wrong interpretations. Indeed, we are only interested in finding a word that carries a similar sense without doing a full semantic parse of the sentence. Negations and other modal words may completely change the sense of the sentence, and we will not catch it. When looking in a very large corpus such as the Web, this may lead to more noise than a strict lexical string match approach. However, if we perform the QA task on a much smaller corpus, such as in closed-domain QA, looking for semantic equivalences may be more fruitful.

The current implementation only looks at semantic relations holding between two arguments. However, it can easily be extended to consider variable-size relations. However, as more constraints are taken into account, the precision of the candidate list is expected to increase, but recall is expected to decrease. A careful evaluation would be necessary to ensure that the approach does not introduce too many constraints and consequently filters out too many candidates.

Our approach to checking syntactic and morpho-syntactic variations (as described in section 4.4) is very crude: we only check for the presence or absence of a similar stem with no respect to syntax. A more precise approach should be used; see the work of [23,24,25] for example.

Finally, to improve the quality of the patterns, we suggest a systematic evaluation and adjustment of the parameters that take part in weighting the patterns; for example, the size of the windows and the sub-phrase scoring.

Acknowledgments

This project was financially supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) and Bell University Laboratories (BUL). The authors would also like to thank the anonymous referees for their valuable comments.

References

1. Aceves-Pérez, R., nor Pineda, L.V., Montes-y-Gòmez, M.: Towards a Multilingual QA System based on the Web Data Redundancy. In: Proceedings of the 3rd Atlantic Web Intelligence Conference, AWIC 2005. Lecture Notes in Artificial Intelligence, No. 3528, Lodz, Poland, Springer (2005)
2. Hermjakob, U., Echihab, A., Marcu, D.: Natural language based reformulation resource and wide exploitation for question answering. [22]

3. Ravichandran, D., Hovy, E.: Learning surface text patterns for a question answering system. In: Proceedings of ACL-2002, Philadelphia (2002)
4. Agichtein, E., Gravano, L.: Snowball: Extracting Relations from Large Plain-Text Collections. In: Proceedings of the 5th ACM International Conference on Digital Libraries. (2000)
5. Agichtein, E., Lawrence, S., Gravano, L.: Learning search engine specific query transformations for question answering. In: Proceedings of WWW10, Hong Kong (2001) 169–178
6. Soubbotin, M., Soubbotin, S.: Patterns of potential answer expressions as clues to the right answers. In: Proceedings of The Tenth Text Retrieval Conference (TREC-X), Gaithersburg, Maryland (2001) 175–182
7. Brill, E., Lin, J., Banko, M., Dumais, S., Ng, A.: Data-Intensive Question Answering. [19] 393–400
8. Lawrence, S., Giles, C.L.: Context and Page Analysis for Improved Web Search. *IEEE Internet Computing* **2** (1998) 38–46
9. Rivachandram, D., Hovy, E.: Learning Surface Text Patterns for a Question Answering System. In: Proceeding of ACL Conference, Philadelphia (2002) 41–47
10. Kwok, C.C.T., Etzioni, O., Weld, D.S.: Scaling question answering to the web. In: *World Wide Web*. (2001) 150–161
11. Radev, D.R., Qi, H., Zheng, Z., Blair-Goldensohn, S., Zhang, Z., Fan, W., Prager, J.M.: Mining the web for answers to natural language questions. In: *CIKM*. (2001) 143–150
12. Duclaye, F., Yvon, F., Collin, O.: Using the Web as a Linguistic Resource for Learning Reformulations Automatically. In: *LREC'02*, Las Palmas, Spain (2002) 390–396
13. Kosseim, L., Plamondon, L., Guillemette, L.: Answer formulation for question-answering. In: Proceedings of The Sixteenth Canadian Conference on Artificial Intelligence (AI'2003), Halifax, Canada, AI-2003 (2003)
14. Plamondon, L., Lapalme, G., Kosseim, L.: The QUANTUM Question-Answering System at TREC-11. [22]
15. Cunningham, H.: GATE, a General Architecture for Text Engineering. *Computers and the Humanities* **36** (2002) 223–254
16. Ramshaw, L., Marcus, M.: Text chunking using transformation-based learning. In: Proceedings of the Third ACL Workshop on Very Large Corpora, MIT (1995) 82–94
17. NIST: Proceedings of TREC-8, Gaithersburg, Maryland, NIST (1999)
18. NIST: Proceedings of TREC-9, Gaithersburg, Maryland, NIST (2000)
19. NIST: Proceedings of TREC-10, Gaithersburg, Maryland, NIST (2001)
20. Plamondon, L., Lapalme, G., Kosseim, L.: The QUANTUM Question Answering System. [22]
21. Porter, M.: An algorithm for suffix stripping. *Program* **14** (1980) 130–137
22. NIST: Proceedings of TREC-11, Gaithersburg, Maryland, NIST (2002)
23. Jacquemin, C.: Spotting and discovering terms through natural language processing. MIT Press, Cambridge, Mass. (2001)
24. Ferro, J.V., Barcala, F.M., Alonso, M.A.: Using syntactic dependency-pairs conflation to improve retrieval performance in spanish. In: *CICLing*. (2002) 381–390
25. Ribadas, F.J., Vilares, M., Vilares, J.: Semantic similarity between sentences through approximate tree matching. In Jorge S. Marques, N.P.d.l.B., Pina, P., eds.: *Pattern Recognition and Image Analysis*. Volume 3523 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin-Heidelberg-New York (2005) 638–646

Using N-Gram Models to Combine Query Translations in Cross-Language Question Answering*

Rita M. Aceves-Pérez, Luis Villaseñor-Pineda, and Manuel Montes-y-Gómez

Language Technologies Group, Computer Science Department,
National Institute of Astrophysics, Optics and Electronics (INAOE), Mexico
{rmaceves, mmontesg, villasen}@inaoep.mx

Abstract. This paper presents a method for cross-language question answering. The method combines multiple query translations in order to improve the answering precision. The combination of translations is based on their pertinence to the target document collection rather than on their grammatical correctness. The pertinence is measured by the translation perplexity with respect to the collection language model. Experimental evaluation on question answering demonstrates that the proposed approach outperforms the results obtained by the best translation machine.

1 Introduction

A question answering (QA) system is a particular kind of search engine that allows users to ask questions using natural language instead of an artificial query language. In a cross-lingual scenario the questions are formulated in a language different from the document collection. In this case, the efficiency of the QA system greatly depends on the way it confronts the idiomatic barrier. Traditional approaches for cross-lingual information access involve translating either the documents into the expected query language or the questions into the document language. The first approach is not always practical, in particular when the document collection is very large. The second approach is more common. However, because of the small size of questions in QA, the machine translation methods do not have enough context information, and tend to produce unsatisfactory question translations.

A bad question translation generates a cascade error through all phases of the QA process. This effect is evident in the results of cross-lingual QA reported on the last edition of CLEF [4]. For instance, the results from the best cross-lingual system (that uses the French as target language) were 64% of precision for the monolingual task, and 39.5% when using English as question language. In this case, the errors in the translation of the question cause a drop in precision of 61.7%.

Recent methods for cross-lingual information access attempt to minimize the error introduced by the translation machines. In particular, the idea of combining the

* This work was partially financed by the CONACYT (grants 43990 and 184663). We also like to thanks to the CLEF for the provided resources.

capacities of several translation machines has been successfully used in cross-lingual information retrieval [2]. In this field, most works focus on the selection of the best translation from a set of candidates [1]. In opposition, in this paper we propose a method that considers a weighted combination of the passages recovered from each translation in order to enhance the final precision of a cross-lingual QA system. In this way, all translations are treated as –possible– relevant reformulations of the original question.

2 Proposed Method

The proposed method assumes that machine translation is not a solved task, and tries to face it by combining the capacities of different translators. Figure 1 shows the general scheme of the method. It considers the following procedures. First, the user question is translated to the target language by several different translators. Then, each translation is used to retrieve a set of relevant passages. After that, the retrieved passages are combined in order to form one single set of relevant passages. Finally, the selected passages are analyzed and a final question answer is extracted.

The main step of this method is the combination of the passages. This combination is based on the pertinence of the translations to the target document collection. The pertinence of a translation indicates its probability of being generated from the document collection. In other words, the pertinence of a translation expresses how it fits in the n -gram model calculated on the target document collection. The idea is to combine the passages favoring those retrieved by the more pertinent translations.

The following subsections describe the measuring of the pertinence of a translation to a target document collection, and the combination of the relevant passages in one single set.

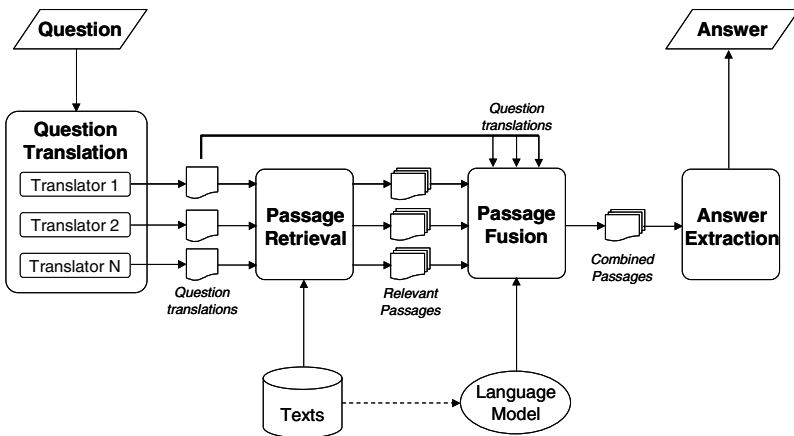


Fig. 1. General scheme of the method

2.1 Translation Evaluation

As we mentioned, the pertinence of a translation to the target document collection is based on how much it fits in the collection n -gram model. In order to quantify this attribute we apply a general n -gram test on the translation. An n -gram test computes the entropy (or perplexity) of some test data –the question translation– given an n -gram model. Basically, it is an assessment on how probable is to generate the test data from the n -gram model. The entropy is calculated as:

$$H = -\frac{1}{Q} \sum_{i=1}^Q \log P(w_i | w_{i-1}, w_{i-2}, \dots, w_{i-N+1})$$

where w_i is a word in the n -gram sequence, $P(w_i)$ indicates the probability of w_i , Q is the number of words of the test data, and N is the order of the n -gram model.

The final score for a translation is expressed by its perplexity, defined as $B = 2^H$. In this case, a low perplexity value indicates a more predictable language, and therefore, that the translation is pertinent to the target collection.

2.2 Passage Fusion

This module combines the retrieved passages from each translation in one single set. Its purpose is to favor passages recovered by the more pertinent translations. The following formula is used to calculate the number of passages from a given translation that will be included in the combined passage set.

$$E_x = \frac{k}{\sum_{i=1}^n \frac{1}{B_i}} \times B_x$$

In this formula E_x indicates number of selected passages from the translator x , that is, the extension of x in the combined set. B_x is the perplexity of the translator x , n is the number of translation machines used in the experiment, and k indicates the number of passages retrieved by each translator as well as the total extension of the combined set. In the experiments we set $k = 20$, which corresponds to the best performance rate of our QA system [3].

3 Experiments

For the experimental evaluation of the method we considered a set of 141 factual questions extracted from the Multi-Eight Corpus of the CLEF¹. We used the passage retrieval and answer extraction components of the TOVA question answering system [3], which was the second best in the Spanish QA task at the last edition of the CLEF.

The evaluation consisted of three bilingual experiments: English-Spanish, French-Spanish and Italian-Spanish. For the translation from English and French to Spanish

¹ Cross-Language Evaluation Forum (www.clef-campaign.org).

we use four different translation machines²: Systran, Webtranslation, Reverso and Ya. For the translation from Italian to Spanish we used³: Systran, IdiomaX, Worldlingo, Zikitrake.

For the three experiments we measured the lost of precision in the answer extraction caused by the question translation in relation to the Spanish monolingual task. Table 1 shows the lost of precision, indicated as an error rate, for the three bilingual experiments. The first four columns indicate the error rates generated by each machine translation when they were used alone. The last column shows the error rates that were obtained when using the combined passages. In all cases, except for French, the proposed combination of the passages obtained lower error rates than the best translation machine. In addition, our method outperforms two other naïve approaches. One based on the selection of the translation with the lowest perplexity [1] (see column 5), and other one based on a uniform combination of the recovered passages (see column 6).

Table 1. Error rates in relation to the Spanish monolingual task

	MT1	MT2	MT3	MT4	Lowest perplexity	Uniform Combination	Proposed method
English-Spanish	17%	24%	17%	27%	14%	27%	<u>7%</u>
French-Spanish	<u>17%</u>	38%	27%	31%	31%	34%	27%
Italian-Spanish	52%	45%	41%	34%	41%	34%	<u>24%</u>

4 Conclusions

In this paper we presented a method for cross-lingual QA that tackles the problem of question translation by combining the capacities of different translators. The experiments demonstrated that the combination of passages retrieved by several translation machines tend to reduce the error rates introduced by the question translation process.

In the French-Spanish experiment, our method produced error rates higher than those from the best translation machine. This situation was caused by, on the one hand, the incorrect translation of several named entities from French to Spanish, and on the other hand, by the inadequate treatment of unknown words by our n -gram model.

As future work we plan to improve the n -gram test in order to handle unknown words, and to apply the method on different target languages.

References

1. Callison-Burch C., and Flounoy R. A Program for Automatically Selecting the Best Output from Multiple Machine Translation Engines. In Proceedings of the Machine Translation Summit VIII, Santiago de Compostela, Spain, 2001.

² www.systranbox.com, www.imtranslator_webtranslation.paralink.com, elmundo.reverso.net, traductor.ya.com.

³ www.systranbox.com, www.idiomax.com, www.worldlingo.com, www.zikitrake.com.

2. Di Nunzio G. M., Nicola Ferro, Gareth J.F. Jones, Carol Peters. CLEF 2005: Ad Hoc Track Overview. CLEF 2005, Vienna, Austria, 2005.
3. Montes-y-Gómez, M., Villaseñor-Pineda, L., Pérez-Coutiño, M., Gómez-Soriano, J. M., Sanchis-Arnal, E. & Rosso, P. INAOE-UPV Joint Participation in CLEF 2005: Experiments in Monolingual Question Answering. CLEF 2005, Vienna, Austria, 2005.
4. Vallin, A., Giampiccolo, D., Aunimo, L., Ayache, C., Osenova, P., Peñas, A., de Rijke, M., Sacaleanu, B., Santos, D. & Sutcliffe, R. Overview of the CLEF 2005 Multilingual Question Answering Track. CLEF 2005, Vienna, Austria, 2005.

A Question Answering System on Special Domain and the Implementation of Speech Interface

Haiqing Hu^{1,2}, Fuji Ren¹, Shingo Kuroiwa¹, and Shuwu Zhang³

¹ Faculty of Engineering, The University of Tokushima,
Tokushimashi 770-8506, Japan

{huhq, ren, kuroiwa}@is.tokushima-u.ac.jp

² Xian University of Technology, Xian, China

³ The Institute of Automation Chinese Academy of Sciences, Beijing, China
swzhang@hitic.ia.ac.cn

Abstract. In this paper, we propose a construction of Question Answering(QA) system, which synthesizes the answers retrieval from the frequent asked questions database and documents database, based on special domain about sightseeing information. A speech interface for the special domain was implemented along with the text interface, using an acoustic model HMM, a pronunciation lexicon, and a language model FSN on the basis of the feature of Chinese sentence patterns. We consider the synthetic model based on statistic VSM and shallow language analysis for sightseeing information. Experimental results showed high accuracy can be achieved for the special domain and the speech interface is available for frequently asked questions about sightseeing information.

Keywords: Question Answering System, Similarity Computing, Special Domain, FSN, Speech Recognition, Chinese.

1 Introduction

Question Answering (QA) is a technology that aims at retrieving the answer of a question written in natural language in large collections of documents. QA systems are presented with natural language questions and the expected output is either the exact answer identified in a text or small text fragments containing the answer. A lot of research has been done on the QA technology, and the technology relates to a lot of fields of NLP (Natural Language Processing), such as Information Retrieval(IR), Information Extraction(IE), Conversation Interface, etc. Recently, systems based on statistical retrieval techniques and shallow language analysis are much used techniques in answer retrieval using natural language. In the Question Answering task of TREC(Text REtrieval Conference) QA track, the target has become the open domain (the search object domain to the questions is not limited) in recent years. But the treatment of special domains and the construction of a practical QA system as a specialist are very difficult. On the other hand, it is easier to use special domain knowledge by

limiting an object field, and it is feasible to improve the efficiency and precision of the reply.

In this research, a Chinese QA system was proposed which restricted the question domain within sightseeing information. The technique proposed is that of integrating a statistical technique and an analytical base. The proposed technique integrates the answer retrieval of frequency asked questions (we call it “Question&Answer database” in this paper), and the document retrieval of sightseeing information. Furthermore, in this paper, a speech-driven QA system is designed, that the input and output of speech for a special domain was implemented in the interface using a language model FSN(Finite State Network) on the basis of the feature of Chinese sentence patterns.

This paper is organized as follows. Section 2 introduces the related research on QA systems. In section 3, we describe the basic architecture of the QA system. Section 4 shows the implementation of the speech interface of the system. Section 5 explains the questions analysis and the narrowing-down process of candidate answers in the system. Section 6 shows experimented results for the system based on the proposed method. Section 7 contains conclusions of this paper and future work.

2 Related Work on QA System

2.1 Related Work on Open Domain QA System

The current trend in Question Answering(QA) is oriented towards the processing of open domain texts. Under the promotion of evaluation exercises such as TREC, CLEF (Cross-Language Evaluation Forum), and NTCIR (NII Test Collection for Information Retrieval), there has been a large interest in developing QA systems in the last years, Such as LCCmain [1], Qanda [2], DLT [3]. Furthermore, the research of Chinese QA systems using natural language has been especially paid attention to in recent years, though it was developed later than that of western countries. Some Chinese QA systems have been constructed up until now, for example, the “NKI question answering system” (<http://www.nki.net.cn>) based on a large-scale database in open domain is a representative among them. QACAS [4] is constituted using about 73 rules of reply types made manually for Web documents. Moreover, the Marsha Chinese QA system [5] was made by Xiaoyan Li et al, as a crossing retrieval QA system, and they have proposed some experiments that apply the technology of QA system for English text to the Chinese QA system.

2.2 Related Work on Special Domain QA System

QA system of open domain is lacking to treat the special domains for all question types, because no restriction is imposed either on the question type or on the user’s special vocabulary. Recently, restricted-domain QA regains attention, as shown by a dedicated ACL workshop to be held in 2004(In Proceedings ACL 2004 Workshop on Question Answering in Restricted Domains) and

the AAAI-05 Workshop to be held in 2005(The AAAI-05 Workshop on Question Answering in Restricted Domains), etc. Many researchers also begin to focus their attention on restricted domain QA and have built some advanced restricted domain QA systems. O. Tsur, et al [6] presented a biographical QA system(BioGrapher) that addresses the problem by machine learning algorithms for biography classification. Farah [7] showed an experiment to design a logic based QA system(WEBCOOP) for the tourism domain. Niu, et al [8] provided an QA research in clinical-evidence texts, which identify occurrences of the semantic classes and the relations between them.

As for Chinese QA systems of special Domain, there was “XiaoLingTong QA System” (<http://159.226.40.18/ask/pub/>) for travelers of FAQ. FAQAS [9] is an applied system for the financial domain, by constructing a finance ontology database. The QAS was proposed by Shuxi Wang [10] which is about human relationships in “Dream of the Red Chamber”. In the year 2000, the Tsinghua University Campus Guide EasyNav System [11], started actual application.

2.3 Related Work on Speech-Driven QA System

About the speech interface to QA system, some reseachs appear to be at the forefront of this field. Edward and Zhiping Zheng [12] describe a multimodal interface to a open domain QA system designed for rapid input of questions using a commercial dictation engine, and indicate that speech can be used for automatic QA system by their evaluation. Akiba, et al [13] proposed a method for producing statistical language models for speech-driven question answering, which enable recognizing spoken questions with high accuracy by magnifying N-gram counts corresponding to the frozen patterns in the original N-gram. Later Akiba, et al [14] also proposed a speech-driven QA system for WH-questions, which focused mainly on the effects of language modeling. On speech-driven QA system of Chinese, Zhang, et al [15] described a Chinese spoken dialogue system about real-time stock market quotations inquiry, which used a model of situation semantic frame-key technology.

For Chinese QA systems, the construction of a practical system is considerably difficult, because the manual cost to make the large-scale database is high such as NKI. On the other hand, in the application to a more extensive domain it is more difficult to use the syntactic-analysis information and the shallow structure mode inference as “FAQAS” of financial domain and QAS of person kinship (“Dream of the Red Chamber”). In this paper, as a part of QA technical research, the object domain is restricted to a middle-sized domain of sightseeing information. A speech interface for the special domain was implemented along with the text interface, using an acoustic model HMM(Hidden Markov Model) and a language model FSN(Finite State Network) on the basis of the feature of Chinese sentence patterns. We propose the construction of a QA system that combines a retrieval mechanism for a Question&Answer(FAQ) database as well as a mechanism for web documents.

3 Proposed Method and System Architecture

3.1 Proposed Method

In this paper, a QA system that integrates a statistical base and a shallow analytical base is proposed. The bases were created from answers retrieved from a Question&Answer database and a document database, that uses VSM(Vector Space Model) according to the Chinese language feature information based on morphological analysis. Because of that the questions for sightseeing asked by the user contain a lot of similar questions through our investigation, the questions of the Question&Answer database can be used effectively. Concretely, answer retrieval from a Question&Answer database of frequently asked questions and the answer retrieved from documents concerning sightseeing information are integrated.

Moreover, a long document(one web page) is often assumed to be a retrieval result in past IR techniques. However, it is a large encumbrance to the user to search for the desired information from the document. On the other hand, it is technically difficult to provide a concise answer in a single phrase to the user, although the final target of the QA system is that, but the resulting accuracy tends not to be good. This research adopts a method to request the sentences including the answer with high similarity to the user from similar documents.

3.2 System Architecture

This system can automatically answer questions about travel information that are asked by a tourist using natural language of Chinese. The fundamental architecture of this system is shown in Figure 1, and consists of the user interface, the speech synthesis and recognition, the question analysis, the Question&Answer database retrieval, the document retrieval processing and the preprocessing, and some databases.

The user interface is a dialog to the system, that the user's natural language can be entered into and the answers outputted to the user. In speech recognition and speech synthesis processing, the speech I/O (input/output) for a special domain is examined by using the feature of Chinese sentence patterns except the I/O of the text. In the question analysis module, some processing of the sentence is carried out which include morphological analysis, question classification, stop-word processing, extraction of keywords and expansion of keywords. The answer retrieval module consists of the processing of Question&Answer database retrieval and document retrieval, and will be introduced later.

In the preprocessing module extraction processing of terms (includes index words and appearance frequency) is executed for retrieving the answers by the vector space method (VSM). When new Question&Answer data and document data are obtained, the index words and documents (term-document matrix) can be renewed. In the extraction of terms, 39 kinds of part of speech tags given by morphological analysis are analyzed, then we removed the functional word that doesn't express semantic content directly, such as Particles, Adjectives, and

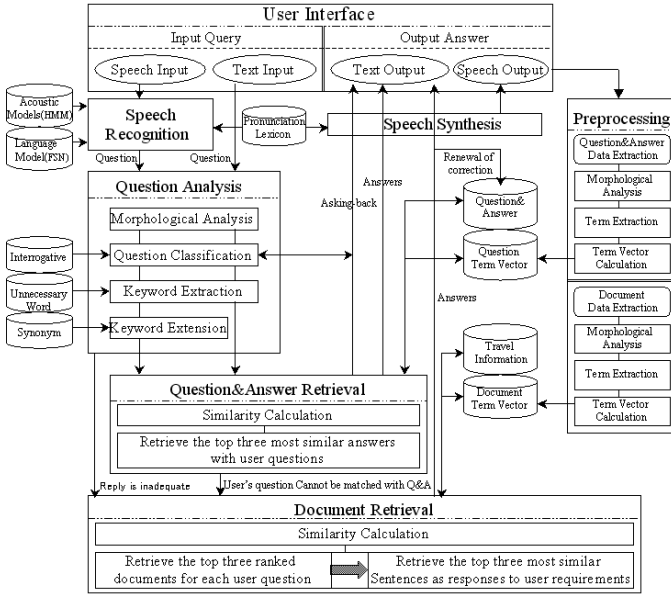


Fig. 1. Configuration of the proposed system

General Adverbs, etc. The remaining part of speeches include General Noun (n), Person Name (nr), Place Name(ns), Organization Name (nt), Other Proper Nouns (nz), General Verb (v), Noun-verbs (vn), Time Words (t), and Azimuth Words (f) and Place Words (s) , etc. Next, the stop words are removed and the terms are extracted from the remaining words. In the system, the retrieval processing consists of three retrieval processes that include similar question sentence retrieval, similar document retrieval, and similar answer retrieval from the document.

Moreover, the knowledge base of this system consists of an interrogative knowledge base, a stop words list, a synonym knowledge base, a Question&Answer database, a question term(index words) vector matrix database, a travel information knowledge database, and a document term(index words) vector matrix database.

4 Speech Recognition and Speech Synthesis Processing

Many of the present Question Answering systems search for the answer to a query using the text entered by a keyboard. Automatic speech recognition (ASR) technologies have progressed in recent years, and have been enhanced so that past information retrievals may correspond to the voice input. After I/O of the text from the user, we examine the practicality of the speech I/O for the special domain.

Generally the speech recognition part is constructed with the modules as follows: a pronunciation dictionary, a language model (word N-gram), and an acoustic model (HMM: Hidden Markov Model) (Figure 2). Since the system is a QA system for a specific domain, utterances by voice have the feature of “Fixed form questions are often used”. Therefore, we pay attention to the formulation part of speech recognition. If the language model of a specific task is manually made for these highly frequent fixed form expressions using the descriptive grammar (namely, the word string shown in a regular language, is a grammar expressed on the word network where only the partial word concatenation was permitted by the grammar), we think it possible to obtain a good accuracy. The language model is mainly classified into a determination grammar and a probability statistical model, usually the former is manually described, and the latter is learned from the text corpus automatically.

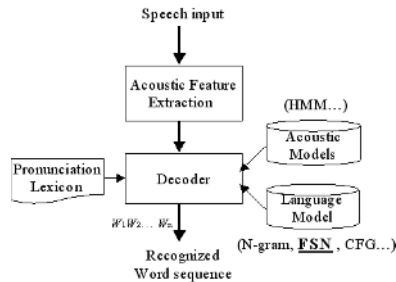


Fig. 2. Configuration of speech recognition mechanism

Among the grammars, a Finite State Network (FSN) is an effective method in the conversation system when the task is limited to a specific domain, etc. Therefore, in the sightseeing system, the construction is based on the feature of Chinese sentence patterns, using a pronunciation dictionary, an acoustic model (HMM), and some grammars for the special domain defined by a FSN beforehand. They are utilized as the language model instead of a word N-gram language model.

In the paper, we composed a finite state grammar for the sightseeing domain based on analysis of Chinese sentence patterns dealing with sightseeing information and extracting fixed form expressions with high frequency manually. We defined the grammar of a specific domain related to the sightseeing field beforehand. About the grammatical definition language, we will explain it by an example of the typical grammatical definition that is applied to ASR from the HTK toolkit [16].

We defined the grammar related to the sightseeing field based of Question&Answer database collected by using this grammatical definition language. An example is shown in Figure 3. In this system, it is necessary to remove a lot of unknown characters for the speech synthesis of the answer, since the answer that is retrieved comes from a web file. Therefore, the system executes


```

Variable:
SPolite = 请问 [我 想 知道] (告诉 我);
SPlace = 景点 | 地方 | 旅游地点 | 风景区 (特色 景点) (好玩 的);
SWhere = 哪里 | 哪儿 | (什么 位置) (什么 地方) (什么 方位);
SWhen = 何时 | (什么 时候) (什么 时间) (什么 季节) 多久;
SAuxiliary = 可以 | 能;
SModifier = 高 | 便宜 | 贵;
SWhich = 哪些 | 哪个;
STraffic = 飞机 | 电车 | 汽车 | 巴士 | 公交车 | 地铁 | 打的;
STravelSite = 日本 | 东京 | 首都 | 名古屋 | 北海道 | 富士山 | 箱根;
STravelSiteEvent = 温泉 | 温泉之乡 | 有马温泉 | 拉面 | 樱花 ;
STravelSiteMerit = 看 | 观 | 看 | 体验 | 钓鱼 | 吃 | 赏 | 赏 | 赏 | 赏;
STravelSiteCommon = 海拔 | 高度 | 人口 | 人 | 电器;
Grammar:
SGRAM=(
( [SPolite] [去] [到] [从] $TravelSite [旅游]的 $TravelSite)
( 在 $Where )
( [ 到 $TravelSite ] [坐乘 $Traffic] 如何 (前往去) ) ( 到 $TravelSite ) [呀] )
( 离 $TravelSite [很] 近 (么吗) )
( [那儿] [那些] 有 ( 什么 ) [哪] [SWhich] ) [ $TravelSite Merit ] [有名] [著名] [的] [ $Place
$TravelSiteEvent ] )
( 好玩 [好] [吗] 不好玩 )
( 的特色 $TravelSiteEvent (是 什么) (有 SWhich) )
( [有名] [著名] 的 $Place $TravelSiteEvent (是 什么) [有 SWhich] (在 $Where) )
( $When [去] [玩] 有意思 | 合适 | 最好 )
( [ $TravelSiteCommon ] 有 多 少 [ $Modifier ] )
( [比 $TravelSite] 怎么样 )
( 有没有 [有] [什么] [ $Auxiliary ] [ $TravelSite Merit ] [ $TravelSiteEvent ] [的] [有名] [著名]
好] $Place ] 注意 事项 )
( 哪些 $Place 值得 去 [玩] )
( ($Where) 可以 $TravelSite Merit $TravelSiteEvent )
);

```

Fig. 3. An example of the FSN grammar defined

speech synthesis on the answer by result of morphological analysis allowing the quality of speech synthesis to be advanced.

5 Question Analysis and Extraction of Answer Candidate Processing

5.1 Question Analysis Processing

Morphological Analysis: The system does morphological analysis by using the morphological analysis system ICTCLAS (Institute of Computing Technology Chinese Lexical Analysis System) [17]. 39 kinds of part of speech tags (such as, General Noun (n), Noun-verb (vn), Noun-adjective(an), and General Verb (v), Sub-verb (vd) and Particle (u), etc.) were given according to Chinese language features.

Question Classification: The 55 interrogatives have been extracted from the HowNet Knowledge Database [18]. The question type was subdivided into 11 kinds, categorized by these interrogatives. They include, CAUSE, LOCATION, PERSON, TIME&DATE, METHOD, QUANTITY&AMOUNT, DEGREE, SUBSTANCE&DEFINITION, EVENT, REQUEST TONE and OTHERS.

Keyword Extraction: In the system, the keywords were extracted by morphological analysis and stop word processing. The detail of the processing is the same as index keyword extraction used for retrieval processing later. Such as particle, adjective, interrogative and general adverb, or verb, and the words except for salutatory language. For example, the following words are considered to be unnecessary by retrieval and are removed beforehand, “Qing3Wen4(please tell me)”, “De(of)”, “Xie4Xie4(thanks)”, etc.

Keyword Expansion: Since it is rare that a keyword appears in the database for retrieval as it is, extension of the keyword for retrieval is necessary. The keyword is extended by a synonym word knowledge base in the system.

5.2 Retrieval Processing

The system first calculates the similarity between the user question and each question sentence in the Question&Answer database. It is assumed that the answer to the user’s question will correspond to the answer of the question with high similarity. When the retrieval result is not obtained or the user is not satisfied with the answer, the document retrieval processing is executed using the travel information. Then, the similarity of the user’s question and the sentence in the documents is calculated, and a suitable sentence with high similarity is returned as the answer. In the system, it is important to calculate the similarity of the user’s question sentence and the retrieval sentence in the Question&Answer database and the documents database, and we use the vector space model which is at present is a standard method in information retrieval. The basic idea of the system is a Question Answering System that integrates the retrieval of two kinds of databases.

(1) The retrieval of the Question&Answer database: we correspond the similarity between the questions in the Question&Answer database and the user’s question. The typical question and the important past question were registered into the Question&Answer database beforehand, because of the characteristic that similar questions are comparatively concentrative(similar questions are often seen) in travel questions by the user. Then the similarity of the retrieval question and the user’s question was calculated. The top three most similar questions are retrieved and presented to the user. The judgment whether to retrieve the answers from the documents is left to the user, though it can be examined using a threshold in the future. Concretely, the system is designed so that the user only needs to click the sentence retrieval button when the user is not satisfied with the answer by the Question&Answer database.

(2) Retrieval from the travel information documents database: When the answer is not obtained by the Question&Answer database or the user thinks the answer to be insufficient, retrieval from a large travel document collection is done. At first, we retrieve similar documents to the user’s question from the travel information documents database. The top three high-ranking documents are obtained. Next, the top three most similar sentences in the retrieved document are chosen. Generally, the possibility that the answer appears near a sentence with high similarity is also high, in the system. We return the retrieved

sentence with high-ranking and the sentence before and after concatenated as a final answer.

6 Experiment and Discussion

Two kinds of data were collected for the evaluation experiment. One is data for making the Question&Answer database, and other is data for the travel information documents database. We have been collected 730 pair of questions and answers from the Chinese website about Japan travel. The Chinese documents concerning Japan travel information were collected from websites such as “Japanese National Tourist Organization”. The Hypertext (HTML) files of 1300(about 15000 sentences) were been collected up to now. These data were considered to be the source database for the evaluation experiment. As for the collection of experimental questions, we questioned ten Chinese international students about Japan sightseeing by a questionnaire. 15 questions per person were written randomly for us. Then, we removed questions with similar meaning, and experimented using remaining 115 questions. The GUI of executing answer retrieval is shown in Figure 4. At first, we performed an

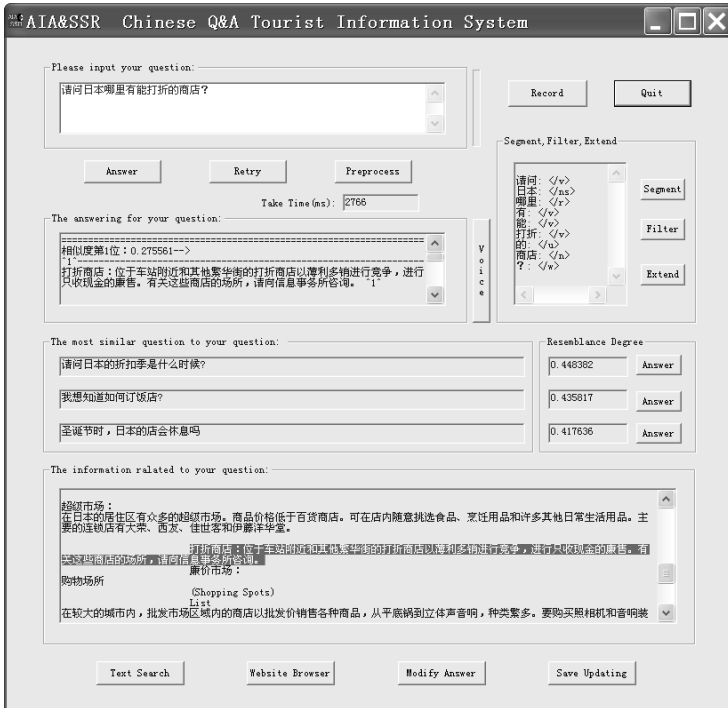


Fig. 4. The GUI of executing answer retrieval

experiment to evaluate of speech recognition. For the specific domain based on the Question&Answer database, we obtained a word recognition rate of 94% by the speech recognition engine in real-time. As a result, it is possible for the speech formula to use frequent prior information from the Question&Answer database. Moreover, we also examined a test for the 115 questions tested, approximately 70% questions could be recognized correctly cause of limited grammars, for improving the system's robustness the user can correct and re-input the question to retrieve the answer in text when the speech recognition fails. As a comparison, in the similar research of speech-driven QA system by Zhang, et al [15], although their accuracy achieved 72.7%, they dealt with a narrow domain which are stock market quotations and Shanghai Traffic Route. In our system, the domain is restricted to a middle-sized domain of sightseeing information.

As a result of the experiment, the accuracy for the 115 questions at obtaining the correct answer from the Question&Answer database using the top three most similar sentences was 26.8%. This showed the insufficiency of the retrieval only from the Question&Answer database, though the answers could be obtained to the question about 1/4 of the time from Question&Answer database.

Table 1. Result of the Experiment

Question Type	Questions	MRR
CAUSE	2	0.50
TIME&DATE	12	0.77
LOCATION	15	0.82
PERSON	2	1.00
METHOD	16	0.69
QUANTITY&AMOUNT	13	0.79
DEGREE	8	0.60
SUBSTANCE&DEFINITION	7	0.52
EVENT	32	0.83
REQUEST TONE	3	0.67
OTHER	5	0.60
Total	115	0.76

To determine if the response is actually an answer to the question, in TREC QA tracks, a question answering system is required to return top five ranked answers for each questions, the mean reciprocal rank (MRR) was used as an evaluation metric (equation 1). We performed an experiment to evaluate the answer retrieval using the MRR metric up to the top three responses, defined as follows. The result is shown in Table 1.

$$MRR = \frac{\sum_{i=1}^N 1/Rank_i}{N} \quad (1)$$

Where, $Rank_i$ is the rank of the first correct occurrence in the top three answers for question i ; N is the number of test questions asked; If for a question i , the correct answer is not in the top three responses then is taken to be zero. As the result, the method proposed achieved 0.76 in the MRR up to the top three answers

for the special domain of sightseeing information. The MRR score of question type correlating named entity were higher than others, such as TIME&DATE, LOCATION, PERSON, QUANTITY&AMOUNT, the average MRR of them achieved about 0.84. However, the question type of PERSON can be considered to be especial cause of the number of this type is not enough in our test questions collected, even if its MRR score is highest 1.

We found 12 questions that could not be answered, as shown in Table 1. We considered the cause is as follows. (1) Five sentences among the 12 did not have the information of answers in the database. This point needs to be improved by expansion of the database in the future. (2) In seven sentences, the question keywords were not in the terms. There are two causes for this: One is that the keywords for such as verbs “He2Zhao4” (Photograph together) and adjective “Pian2Yi4” (Cheap) were leaked from the terms. Since the entire verb and adjective were not used as the keywords, the mechanism of the verb that deeply relates to the domain assumed to be the keywords is necessary. Another is the proper nouns like “Fu4Liang2Ye3” are not registered in the dictionary and mistakes occurred during morphological analysis (actually, that has been divided into three Chinese characters “Fu4”, “Liang2”, and “Ye3”). The efficient recognition of unknown words is necessary in this case. But it is a deep-rooted problem that a single Chinese character may exist respectively as a word for Chinese, so more examinations are necessary in the future.

7 Conclusion

In this paper, the aim was to improve accuracy for retrieving answers to question in a special domain. We proposed a QA system that integrated answer retrieval from a Question&Answer database and the documents about travel information. We made use of term extraction, the VSM, and shallow language analysis. The MRR achieved 0.76 using the combination of the document retrieval and sentence retrieval for the special domain. Moreover, we implemented the speech interface and showed it is available for the sightseeing domain.

In the future, we want to examine the method of automatically retrieving the document by using a threshold even if the user doesn't judge the similarity by oneself for the retrieved similar question sentence from the Question&Answer database. Moreover, we will verify the effectiveness of the system using a larger-scale database and consider more effective term extraction rules using location information of the word.

Acknowledgment

This research has been partially supported by the Ministry of Education, Culture, Sports, Science and Technology of Japan under Grant-in-Aid for Scientific Research (B), 1438-0166, 17300065, Exploratory Research, 17656128, 2005 and the Outstanding Overseas Chinese Scholars Fund of the Chinese Academy of Sciences (No.2003-1-1).

References

1. Harabagiu, S., Moldovan, D., Pasca, M., Surdeanu, M., Mihalcea, R., Girju, R., Rus, V., Lacatusu, F., Morarescu, P. and Bunescu, R.: Answering Complex, List and Context Questions with LCC's Question-Answering Server. In Proceedings of the TREC-10 Conference. Gaithersburg, MD. (2001)
2. Burger, J. D.: MITRE's Qanda at TREC-12. The Twelvth Text REtrieval Conference. NIST Special Publication SP. (2004)500-255
3. Richard, F. E., Gabbay, I., Mulcahy M. and White, K.: Question Answering using the DLT System at TREC 2003. In Proceedings of the TREC-2003 Conference. (2003)
4. Guan, Y., Wang, X., Zhao, Y. and Zhao, J.: The Research on Professional Website Oriented Chinese Question Answering System. ICCPOL2003. Shenyang. China. August4-6. (2003)490-497
5. Li, X. and Croft, W. B.: Evaluating Question Answering Techniques in Chinese. In Proceeding of HLT 2001. San Diego. CA. March. (2001)18-21
6. Tsur, O., Rijke, M. and Sima'an, K.: Biographer: Biography questions as a restricted domain question answering task. In Proceedings ACL 2004 Workshop on Question Answering in Restricted Domains. (2004)
7. Benamara, F.: Cooperative question answering in restricted domain : the WEB-COOP experiment. In Proceedings ACL 2004 Workshop on Question Answering in Restricted Domains. (2004)
8. Niu, Y. and Hirst, G.: Analysis of Semantic Classes in Medical Text for Question Answering. In Proceedings ACL 2004 Workshop on Question Answering in Restricted Domains. (2004).
9. Li, H., Fan, X. Li, L. and Zhang, F.: The Study and Implementation of Finance-domain Chinese Automatic Question- Answering System: FAQAS. ICCPOL2003. Shenyang. China. August4-6. (2003)483-489
10. Wang, S. Liu, Q. and Bai, S.: A Survey on Question Answering System. IC-CPOL2003. Shenyang. China. August4-6. (2003)498-506
11. Huang, Y., Zheng, F., Yan, P., Xu, M. and Wu, W.: The Design and Implementation of Campus Navigation SystemEasyNav. Journal of Chinese Information Processing. Vol.15. No.4 (2001)
12. James, E. and Zheng, Z.: A Speech Interface for Open-Domain Question-Answering. The 41st Annual Meeting of the Association for Computational Linguistics (ACL2003). Sapporo. Japan. July7-12. (2003)
13. Akiba, T., Itou, K., Fujii, A. and Ishikawa, T.: Towards Speech-Driven Question Answering: Experiments Using the NTCIR-3 Question Answering Collection. In Proceedings of the Third NTCIR Workshop on Research in Information Retrieval, Automatic Text Summarization and Question Answering. (2002)
14. Akiba, T., Fujii, A. and Itou, K.: Effects of Language Modeling on Speech-driven Question Answering. Proceedings of the 8th International Conference on Spoken Language Processing (ICSLP 2004). Oct. (2004)1053-1056
15. Zhang, L., Gao, F., Guo,R., Mao J. and Lu, R.: A Chinese Spoken Dialogue System about Real-time Stock Information. Journal of Computer Applications. Vol.24. No.7. (2004)61-63
16. Young, S., Kershaw, D. Odell, J. et. al.: The HTK Book (for HTK version 3.0). July. (2000)
17. Zhang, H. Yu, H. Xiong, D. and Liu, Q.: HHMM-based Chinese Lexical Analyzer ICTCLAS. proceedings of 2nd SigHan Workshop. (2003)184-187
18. Dong, Z. and Dong, Q.: Hownet. <http://www.keenage.com/>

Multi-document Summarization Based on BE-Vector Clustering

Dexi Liu^{1,2,3}, Yanxiang He^{1,3}, Donghong Ji^{3,4}, and Hua Yang^{1,3}

¹ School of Computer, Wuhan University, Wuhan 430079, P.R. China

² School of Physics, Xiangfan University, Xiangfan 441053, P.R. China

³ Center for Study of Language and Information, Wuhan University, Wuhan 430079, P.R. China

⁴ Institute for Infocomm Research, Heng Mui Keng Terrace 119613, Singapore
dexiliu@gmail.com, yxhe@whu.edu.cn,
dhji@i2r.a-star.edu.sg, yh@eis.whu.edu.cn

Abstract. In this paper, we propose a novel multi-document summarization strategy based on Basic Element (BE) vector clustering. In this strategy, sentences are represented by BE vectors instead of word or term vectors before clustering. BE is a head-modifier-relation triple representation of sentence content, and it is more precise to use BE as semantic unit than to use word. The BE-vector clustering is realized by adopting the k-means clustering method, and a novel clustering analysis method is employed to automatically detect the number of clusters, K. The experimental results indicate a superiority of the proposed strategy over the traditional summarization strategy based on word vector clustering. The summaries generated by the proposed strategy achieve a ROUGE-1 score of 0.37291 that is better than those generated by traditional strategy (at 0.36936) on DUC04 task-2.

1 Introduction

With the rapid growth of online information, it becomes more and more important to find and describe textual information effectively. Typical information retrieval (IR) systems have two steps: the first is to find documents based on the user's query, and the second is to rank relevant documents and present them to users based on their relevance to the query. Then the users have to read all of these documents. The problem is that these docs are much relevant and reading them all is time-consuming and unnecessary. Multi-document summarization aims at extracting major information from multiple documents and has become a hot topic in NLP. Multi-document summarization can be classified into three categories according to the way that summaries are created: sentence extraction, sentence compression and information fusion.

The sentence extraction strategy ranks and extracts representative sentences from the multiple documents. Radev [1] described an extractive multi-document summarizer which extracts a summary from multiple documents based on the document cluster centroids. To enhance the coherence of summaries, Hardy Hilda [2] and Mitra [3] extracted paragraphs instead of individual sentences.

Knight and Marcu [4] introduced two algorithms for sentence compression based on the noisy-channel model and the decision-tree approach. The input to each algorithm is the parse tree of a long sentence, and the output is expected to be a reduced sentence keeping the major semantic information. However, it is hard to control the compression ratio using this strategy.

Barzilay [5] described an algorithm for information fusion, which tries to combine similar sentences across documents to create new sentences based on language generation technologies. Although this strategy can simulate, to some degree, the human's action in summarization process, it heavily relies on some external resources, e.g. dependency parsers, interpretation or generation rules, etc, which inevitably limit its portability.

In the sentence extraction strategy, clustering is frequently used to eliminate the redundant information resulted from the multiplicity of the original documents [6]. There are two levels of clustering granularity: sentence and paragraph. Generally, word is employed as the minimal element of a document [1]. However, word may be not precise enough for clustering. So the researchers have turned to terms as the semantic unit [7]. The trouble is that most term extraction methods are based on statistical strategy, thus, a term is not a real syntactic or semantic unit.

In this paper, we apply Basic Elements (BE)[8] as the minimal semantic unit. BE is a head-modifier-relation triple representation of document contents developed for summarization evaluation system at ISI, and is intended to represent the high-informative unigrams, bigrams, and longer units of a text, which can be built up compositionally. BEs can be generated automatically without the support of large corpus that terms based on.

This multi-document summarization approach (MSBEC for abbreviation) consists of four main stages: 1) Preprocessing: break down sentences into BEs and calculate the score of each BE and each sentence. 2) BE clustering: represent each sentence with a BE-vector and apply the k-means clustering method on these BE-vectors. 3) Sentence selection: from each cluster, select a sentence with highest score as the representation of this cluster. 4) Summary generation: output the selected sentences to form the final summary according to their positions in the original documents.

We also propose a novel clustering analysis method, which is based on evaluating the cohesion of within-clusters and the scatter of between-clusters, to automatically determine K , the number clusters.

The rest of this paper is organized as follows. In the next section, we give a short overview of Basic Elements. Section 3 describes the strategy of multi-document summarization based on BE-vector clustering. Section 4 shows the performance comparison of BE-vector clustering and word-vector clustering. Finally, we conclude this paper and discuss future directions in Section 5.

2 Basic Element

Basic Element [8] is a relation between a head-BE and a single dependent, expressed as a triple (head | modifier | relation), where "head" denotes the head of a major syntactic constituent (noun, verb, adjective or adverbial phrases).

Figure 1 presents BE examples for "The United Nations imposed sanctions on Libya in 1992 because of their refusal to surrender the suspects". BEs can be extracted automatically in several ways. Most of them use a syntactic parser to

produce a parse tree and then apply a set of ‘cutting rules’ to extract valid BEs from the tree. In this paper, we use the BE package 1.0 [8] distributed by ISI.

With the triple BE, one can quite easily decide whether any two units match (express the same meaning), and word in BEs is more meaningful. For instance, “United Nations”, “UN”, and “UNO” can be matched at this level (but require work to isolate within a longer unit or a sentence), allowing any larger unit encompassing this to accept any of the three variants. Moreover, the pronoun “their” in the example sentence designates “United Nations” clearly.

head	modifier	relation	
imposed	united nations	subj	(BE-F)
imposed	sanctions	obj	(BE-F)
sanctions	libya	on	(BE-F)
libya	1992	in	(BE-F)
refusal	their	gen	(BE-F)
libya	refusal	because of	(BE-F)
refusal	surrende	comp1	(BE-F)
surrender	united nations	subj	(BE-F)
surrender	suspects	obj	(BE-F)

Fig. 1. Example of BEs in a sentence: “The United Nations imposed sanctions on Libya in 1992 because of their refusal to surrender the suspects.”

3 Multi-document Summarization Based on Basic Elements

In this section, we will introduce the Basic Element-based summarization strategy in details.

3.1 Preprocessing

The preprocessing stage consists of 3 sub-steps.

3.1.1 BE Generation

To break down sentences in a document set into BEs, we employ the BE breaker module in the BE Package distributed by USC/ISI. This module first uses the Minipar [9] parser to create the syntactic tree and then prune it. Once relations between its nodes are resolved, it can result in a list of BEs illustrated in figure 1.

3.1.2 BE Score Calculation

To distinguish which BE is indeed important and uniquely indicative in the document set, we calculate for each BE its informativeness. Every BE has three parts: head-BE, modifier and the relation between head and its modifier, where the head-BE is more

representative for the meaning of a BE than the other parts. So the calculation of BE score is replaced by the calculation of head-BE score. We adopt the typical word weight calculating method TF*IDF [10] to calculate BE score. Let D be the source document set for summarization, d_k denotes the k^{th} document in D , s_{jk} be the j^{th} sentence in document d_k , BE_{ijk} be the i^{th} BE in sentence s_{jk} , H_{ijk} be the head-BE of BE_{ijk} . The score of BE_{ijk} is defined as follows:

$$S'_{BE}(BE_{ijk}) = -\log(1 + \text{TF}(H_{ijk})) * \log(\text{IDF}(H_{ijk})) . \quad (1)$$

Where $\text{TF}(H_{ijk})$ denotes the number of occurrence of H_{ijk} in document d_k , $\text{IDF}(H_{ijk}) = \log \frac{\#\text{documents-contains-}H_{ijk}}{\#\text{documents}}$ is also known as ‘‘Inverted Document Frequency’’ which is computed over the documents in a large corpus (we use BNC corpus in this work).

Finally, the score is normalized among the documents:

$$S_{BE}(BE_{ijk}) = S'_{BE}(BE_{ijk}) / \max_j (S'_{BE}(BE_{ijk})) . \quad (2)$$

3.1.3 Sentence Score Calculation

The score of a sentence is the summation of two weighted scores: the average score of its BEs and the score of the sentence position. Because the sentence occurs in the beginning of the document is more important, sentence position feature should be taken into account when calculating the sentence score. Suppose sentence s_{jk} contains l_{jk} BEs, document d_k contains n_k sentences. The score of sentence s_{jk} is calculated by formula (3):

$$S_S(s_{jk}) = \frac{\alpha}{l_{jk}} \sum_{i=1}^{l_{jk}} S_{BE}(BE_{ijk}) + (1 - \alpha) \frac{n_k - j + 1}{n_k} . \quad (3)$$

Where α is the weight of BE score, $(1 - \alpha)$ is the weight of position score. We let $\alpha = 0.8$ in this work.

3.2 BE Clustering

To process sentences in different documents as a whole, we create a sentence list SL that contains all of sentences in the document set D .

3.2.1 Sentence Representation

Vector space model (VSM) [11] handle massive real documents by adopting the existing mathematical instruments. In this paper, the BEs extracted from all the documents are used to represent the feature vector in VSM. In order to reduce the influence from BEs of little importance, those BEs with score less than half of average BEs score are removed. According to this, we set up the sentence VSM, where each sentence s_i in SL is represented as the weights of BEs, $VS_i = (WB_{i1}, WB_{i2}, \dots, WB_{iN})$, $i = 1, 2, \dots, M$. where $M = \sum_{d_k \in D} n_k$ is the number of sentences in SL , N is the total number

of remained BEs in document set D , WB_{ij} denotes the weight of the j^{th} BE in the i^{th} sentence. In this paper, we adopt TF*IDF to calculate WB_{ij} :

$$WB'_{ij} = -\log(1+TF(BE_{ij})) * \log(\frac{M_j}{M}) . \tag{4}$$

Where $TF(BE_{ij})$ denotes the number of occurrence of the j^{th} BE in the i^{th} sentence, M_j/M denotes the inverted sentence frequency of BE_{ij} , and M_j denotes the number of sentence in which BE_{ij} occurs.

Finally, WB_{ij} is normalized as follows:

$$WB_{ij} = WB'_{ij} / \max_{i,j} (WB'_{ij}) . \tag{5}$$

3.2.2 K-Means Clustering

The k-means clustering method [12] is a fine choice in many circumstances due to its effectiveness with the complexity of $O(nkt)$, where n is the number of sample points, k is the number of clusters and t is the number of iteration. We regard each sentence as a sample point in the N -dimensional sample space, and the sample space contains M sample points, where N is the number of all BEs in the document set D and M is the number of sentences.

To use the k-means method, the distance between two sentences must be defined. The calculation of sentence distance can be achieved by calculating the BE-vector distance. Generally, the cosine method is employed to calculate the similarity between two BE-vectors.

$$\begin{aligned} SIM(VS_i, VS_j) &= \cos(VS_i, VS_j) \\ &= \frac{\sum_{t=1}^N WB_{it} \cdot WB_{jt}}{\sqrt{\sum_{t=1}^N WB_{it}^2} \sqrt{\sum_{t=1}^N WB_{jt}^2}} . \end{aligned} \tag{6}$$

Correspondingly, the distance between two BE-vectors can be calculated by the following formula:

$$DIS(VS_i, VS_j) = 1 - SIM(VS_i, VS_j) . \tag{7}$$

Figure 2 presents the formal description of the BE-vector clustering process based on the k-means method.

Input: the BE-vectors and the cluster number K (2 to $M-1$).
Output: K clusters

- 1) randomly select K BE-vectors as the initial centres of the clusters;
- 2) repeat:
 - assign each BE-vector to the nearest cluster according to its distance to the cluster centres;
 - recalculate the new centre for each cluster;
- 3) until the change of centres is very little.

Fig. 2. BE-vector clustering process using k-mean method

3.2.3 Automatic Determination of K

A classical problem with the k-means clustering method and many other clustering methods is the determination of K , the number of clusters. In the traditional k-means method, K must be decided by the user in advance. In many cases, it's impractical. As for BE clustering, user can't predict the latent cluster number, so it's impossible to offer K correctly.

In this paper, two kinds of methods are proposed to detect K automatically.

The first method is simple and inspired by the limited summary length fixed by the user. On the one hand, summary length is usually fixed by user, so the number of extracted sentences is approximatively fixed at the same time. On the other hand, to generate an anti-redundant summary, summarizer usually extracts only one sentence from each cluster. So, the number of sentences in fixed-length-summary is an acceptable value for the number of clusters. The most probable number of sentences in a fixed-length-summary is the length of summary divided by the average length of sentences in document set. Thus, we determine the approximate number of clusters as:

$$K' = L_{SM} / \text{avg}(L_s) . \quad (8)$$

Where L_{SM} denotes the summary length fixed by the user, $\text{avg}(L_s)$ denotes the average length of sentences in the document set D .

The basic idea of the second strategy is that if the cluster number K is correct, the within-cluster-similarity of vectors should be higher whereas the between-cluster-similarity of vectors should be lower.

We define the cohesion of a cluster and scatter between two clusters as formula (9) and (10) respectively.

$$\text{CHN}(c_i) = \frac{2}{|c_i|(|c_i| - 1)} \sum_{\substack{VS_p, VS_q \in c_i \\ VS_p \neq VS_q}} \text{SIM}(VS_p, VS_q) . \quad (9)$$

$$\text{SCT}(c_i, c_j) = \frac{1}{|c_i||c_j|} \sum_{VS_q \in c_i} \sum_{VS_p \in c_j} \text{DIS}(VS_p, VS_q) . \quad (10)$$

Where c_i is the i^{th} cluster generated by the k-means clustering method. $|c_i|$ is the number of elements (members) in cluster c_i .

The evaluation function of the clustering result is defined as follows:

$$F(C) = \frac{1}{K} \sum_{c_i \in C} \text{CHN}(c_i) + \frac{2}{K(K-1)} \sum_{\substack{c_i, c_j \in C \\ c_i \neq c_j}} \text{DSP}(c_i, c_j) . \quad (11)$$

Where C is the cluster set of clusters, which is the result of the k-means method.

The number of clusters is determined by maximizing the evaluation function $F(C)$:

$$K^* = \operatorname{argmax}_{K \in \{2, \dots, M-1\}} F(C) . \tag{12}$$

3.3 Sentence Selection

The easiest way to select sentences from the sentence list is to output the topmost sentence from each cluster until the required summary length limitation is reached. However, this simple approach does not consider the relation between length of summary and number of clusters. Suppose we get K clusters after the k -means algorithm presented above, the total length of K topmost sentences from each cluster may be longer or shorter than the required summary length limitation. In this paper, we re-sort the sentence list in descendant order according to the sentence score at first, and then select sentences from the clusters repeatedly according to the sentence order in sentence list. Figure 3 presents the detail process of this method.

Input: sentence list (attributes of element: sentence no., sentence score, sentence length and cluster no. this sentence is assigned to), the required summary length L_{SM} .

Output: a set of the selected sentences.

(Let M be the number of elements in sentence list, s_i be the i^{th} element in sentence list, $SN(s_i)$ be the sentence no. of s_i , $CN(s_i)$ be the cluster no. that s_i is assigned to, c_i be the i^{th} cluster in cluster set C , $HBS(c_i)$ be the number of sentences have been selected from cluster c_i , $LEN(s_i)$ be the length of s_i , L_{SM} be the required summary length, SLC be the set of selected sentences.)

- 1) Resort the sentence list SL in descendant order according to the sentence score.
- 2) For i from 1 to M
- 3) If s_i satisfies the following two conditions
 - a. $HBS(c_{CN(s_i)}) \leq \min_{c_k \in C} (HBS(c_k))$;
 - b. $LEN(s_i) + \sum_{s_j \in SLC} LEN(s_j) \leq L_{SM}$;

then

 - add s_i in SLC ;
 - recalculate $HBS(c_{CN(s_i)})$
- 4) output SLC

Fig. 3. The sentence selection method

3.4 Summary Generation

Finally, the selected sentences are output according to their positions in the original document to form the final summary. To improve the consistency of the final summary, the original document set should be sorted by the temporal order.

4 Experimentation

4.1 Experimental Setting

The data used in this work is the document set for task 1&2 in DUC04 [13]. There are 50 sets of English TDT documents. Each set contains 10 documents. Task 2 of DUC04 requires participants produce a short summary no more than 665 bytes for each document cluster. Four human model summaries are provided for each cluster for evaluation.

ROUGE [14] stands for recall-oriented understudy for gisting evaluation. It includes measures to automatically determine the quality of a summary by comparing it with ideal summaries created by humans. The measures count the number of overlapping units such as n-gram, word sequences, and word pairs between the computer-generated summary to be evaluated and the ideal summaries created by humans. DUC04 use ROUGE-1,2,3,4, ROUGE-L, and ROUGE-W to measure summaries generated by participants. We follow the same requirement of DUC04 task 2. All ROUGE evaluation configurations also follow the configurations used in DUC04 by using the same command and options: stop words included, porter stemmed and use only the first 665 bytes.

4.2 Evaluation

Figure 4 illustrates the results of two methods for K detection on 50 document sets. K' and K^* are the numbers of clusters detected using formula (8) and (12) respectively. We can shrink the search space of formula (12) from $[2, M-1]$ to $[2, 2K']$ on two reasons: one is that the K^* detected by formula (12) has not much great discrepancy compared with K' , the other is that the required summary length limits the number of clusters.

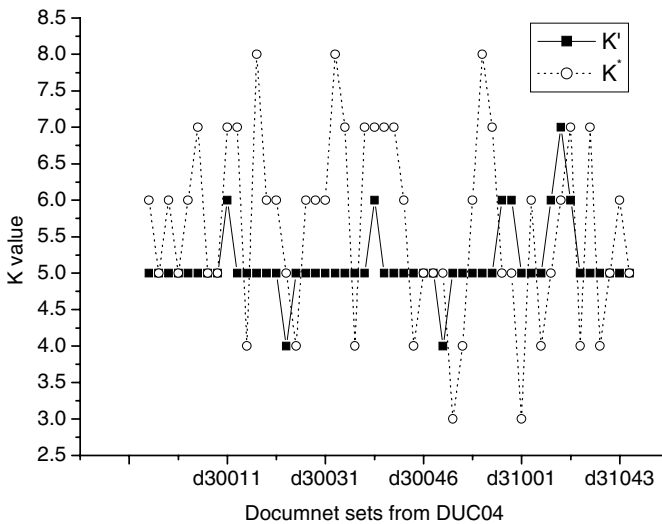


Fig. 4. Results of K detection method (K' using formula (8); K^* using formula (12))

Table 1 lists the ROUGE scores of summaries using MSBEC and summaries using the word-vector clustering strategy (MSWC for abbreviation). To compare our strategy with DUC04 participants, this work re-evaluated all summaries generated by participants using ROUGE1.5.5 package (note that the results are of neglectable difference between ROUGE1.5.5 package and ROUGE package in DUC04). Table 1 lists the average scores of human summaries and the scores of best peers generated by participants as well (unfortunately, there is no paper submission for the best system in DUC 04). Evaluation results show that the BE-vector clustering strategy (MSBEC) is superior to the word-vector clustering strategy (MSWC) for multi-document summarization. The comparison between MSBEC and the best system on DUC04 demonstrates that our strategy is effective.

Table 1. Rouge score comparison

N-gram (F-measure)	Average Human Peers	Best System	MSBEC	MSWC	MSBEC VS. MSWC	MSBEC VS. Best system
Rouge 1	0.40441	0.37917	0.37291	0.36936	+0.96%	-1.65%
Rouge 2	0.09665	0.09152	0.08951	0.08570	+4.44%	-2.20%
Rouge 3	0.03021	0.03332	0.03214	0.03017	+6.53%	-3.54%
Rouge 4	0.01094	0.01533	0.01433	0.01353	+5.86%	-6.56%
Rouge L	0.36193	0.32757	0.32371	0.32194	+0.548%	-1.18%
Rouge w1.2	0.15897	0.14691	0.14499	0.14408	+0.63%	-1.31%

5 Conclusions

In this paper, we have proposed a new multi-document summarization strategy based on BE-vector clustering. Because BEs can represent high-informative unigrams, bigrams, and longer units of a text, the performance of multi-document summarizer can be improved by using BE as the minimal semantic unit. Experiments on DUC04 data set proved the efficiency of our strategy. Moreover, we adopted a novel clustering analysis method to automatically detect the number of clusters in the k-means clustering method. For the future work, we will explore more features and apply the BE-vector clustering strategy in query-based multi-document summarization system.

References

1. Radev Dragomir, Jing Hongyan, Budzikowska Malgorzata: Centroid-Based Summarization of Multiple Documents: Sentence Extraction, Utility-Based Evaluation and User Studies. *Information Processing and Management*, Vol. 40. (2004) 919–938
2. Hardy Hilda: Cross-Document Summarization by Concept Classification. In *Proceedings of the 25th annual international ACM SIGIR Conference on Research and Development in Information Retrieval*. New York: ACM Press (2002) 121–128
3. Mitra M., Singhal Amit, Buckley Chris: Automatic Text Summarization by Paragraph Extraction. In *ACL/EACL-97 Workshop on Intelligent Scalable Text Summarization*, Madrid, Spain (1997) 31–36

4. Knight Kevin, Marcu Daniel: Summarization Beyond Sentence Extraction: a Probabilistic Approach to Sentence Compression. *Artificial Intelligence*, Vol. 139. (2002) 91–107
5. Barzilay Regina, McKeown Kathleen R., Michael Elhadad: Information Fusion in the Context of Multi-Document Summarization. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, New Jersey: Association for Computational Linguistics (1999) 550–557
6. Manuel J, MAN'A-LO'PEZ: Multi-document Summarization: An Added Value to Clustering in Interactive Retrieval, New York: *ACM Transactions on Information Systems*, Vol. 22. (2004) 215–241
7. Po Hu, Tingting He, Donghong Ji, Meng Wang: A Study of Chinese Text Summarization Using Adaptive Clustering of Paragraphs. In *Proceeding of the Fourth International Conference on Computer and Information Technology (CIT'04)*, Wuhan, (2004) 1159–1164.
8. Eduard Hovy, Chin-Yew Lin, Liang Zhou, Junichi Fukumoto: Basic Elements. Technical Report, <http://www.isi.edu/~cyl/BE/index.html> (2005)
9. Dekang Lin: Minipar. <http://www.cs.ualberta.ca/~lindek/minipar.htm> (1998)
10. Baeza Yates R., Ribeiro Neto B.: *Modern Information Retrieval*. New York: Addison Wesley (1999) 27–30
11. Patrick Pantel, Dekang Lin: Document Clustering with Committees. In *Proceedings of ACM, SIGIR'02*, New York: ACM (2002) 199–206
12. Andrew R. Webb: *Statistical Pattern Recognition*, 2nd edn. John Wiley & Sons (2002) 376–379
13. Over Paul, Yen James: An Introduction to DUC-2004. In *Proceedings of the 4th Document Understanding Conference (DUC 2004)* (2004)
14. Chin-Yew Lin, Eduard Hovy: Automatic Evaluation of Summaries Using N-gram Co-Occurrence Statistics. In *Proceedings of the Human Technology Conference (HLTNAACL-2003)*, Edmonton, Canada (2003)

Deriving Event Relevance from the Ontology Constructed with Formal Concept Analysis

Wei Xu^{1,2}, Wenjie Li¹, Mingli Wu¹, Wei Li¹, and Chunfa Yuan²

¹ Department of Computing, The Hong Kong Polytechnic University, Hong Kong
{cswxu, cswli, csmlwu, cswli}@comp.polyu.edu.hk

² Department of Computer Science and Technology, Tsinghua University, China
{vivian00, cfyuan}@mails.tsinghua.edu.cn

Abstract. In this paper, we present a novel approach to derive event relevance from event ontology constructed with Formal Concept Analysis (FCA), a mathematical approach to data analysis and knowledge representation. The ontology is built from a set of relevant documents and according to the named entities associated to the events. Various relevance measures are explored, from binary to scaled, and from symmetrical to asymmetrical associations. We then apply the derived event relevance to the task of multi-document summarization. The experiments on DUC 2004 data set show that the relevant-event-based approaches outperform the independent-event-based approach.

1 Introduction

Extractive summarization is to select the sentences which contain salient concepts in documents. An important issue with it is what criteria should be used to extract the sentences. Event-based summarization attempts to select and organize the sentences in a summary with respect to the events or the sub-events that the sentences describe [1, 2]. As the relevance of events reveals the significance of events, it helps singling out the sentences with the most core events. However, the event-based summarization techniques reported so far explored the events independently.

In the realm of information retrieval, term relations were commonly derived either from a thesaurus like WordNet or from the corpus where the contexts of the terms were investigated. Likewise, mining event relevance requires taking contexts of event happenings into account. The event contexts in our definition are event arguments, such as participants, locations and occurrence times, etc. They are important in defining events and distinguishing them from one another. By this observation, we make use of the named entities associated with the events as event contexts and characterize the events with the verbs and action-denoting nouns prescribed by the named entities.

In this paper, we present a novel approach to learn event relevance with the event ontology constructed from a set of relevant documents and according to the named entities associated to the events. Formal Concept Analysis (FCA) is employed as an effective learning technique to support the building of the event ontology. Based on the ontology, various relevance measures are explored, from binary to scaled, and from symmetrical to asymmetrical associations. The events are then evaluated with

their relevance and in turn the sentences are ranked according to the events they describe. Finally, the top-ranked sentences are selected into the summary. The experiments on DUC 2004 data set suggest that the event-relevance-based approaches outperform the independent-event-based approach.

The remainder of this paper is organized as follows. Section 2 discusses the related work. Section 3 explains how event ontology is constructed and introduces different relevance measures. Section 4 introduces event-relevance-based summarization. Section 5 then presents the experiments and evaluations. Finally, Section 6 concludes the paper.

2 Related Work

Event-based summarization has been investigated in previous researches. Daniel Radev and Allison recognized a news topic in multi-document summarization as a series of sub-events according to human understanding of the topic [1]. They determined the degree of the sentence relevance to each sub-event by human judgment and evaluated six extractive approaches. It was concluded in their paper that recognizing the sub-events that comprise a single news event is essential to produce better summaries. However, it is an obstacle to automatically break a news topic into sub-events. Later, in Filatova and Hatzivassiloglou's work [2], they defined atomic events as the links of major constituent parts of the actions (such as participants, locations, and times) through the verbs or action-denoting nouns. They evaluated the sentences based on the co-occurrence statistics of the events and the named entities involved. As a matter of fact, events in the documents are related in some ways. Judging whether the sentences are salient or not and organizing them in a coherent summary can take advantage from event relevance. Unfortunately, it was neglected in their work and most other previous work. On the other hand, Barzilay and Lapata exploited the use of distributional and referential information of discourse entities to improve summary coherence [3]. While they captured text relatedness with entity transition sequences, i.e. entity-based summarization, we will introduce the relevance between events into event-based summarization.

Ontology is described as a hierarchy of concepts related by subsumption relations [4]. It can be seen as a system containing the concepts and their relations, which can be utilized to analyze the relevance between concepts. In addition to its application in machine translation [5], ontology was also used as the domain knowledge to guide information extraction and summarization. For instance, Artequakt [6] was a system to generate biographies of artists based on the extracted relations between the entities of interest, by following ontology relation declaration and WordNet. Formal Concept Analysis (FCA) had been used as an effective learning technique for ontology construction. While, Haav constructed ontology with FCA in estate domain presenting taxonomic relations of domain-specific entities [7], Alani et al attempted to build a context-based ontology in clinical domain to help identifying the relevant medical concepts and the types of their relations [8]. Besides, Li also employed FCA to construct IT-domain ontology automatically based on lexicon or corpus [9]. All these work has focused on how to select data sources and attribute sets in FCA for ontology construction. The work presented in this paper is motivated by the successful applica-

tion of FCA in automatic ontology construction and will make use of ontology as a means to evaluate the event relevance for text summarization.

3 Deriving Event Relevance

The event arguments are usually realized as named entities. Based on this observation, we represent an event approximately with a set of event terms prescribed by the associated name entities. An *event*, denoted by E , is defined as $E = \{t_i | (n_m, t_i, n_n)\}$ in our work, where t_i is the event term, either a verb or an action-denoting noun according to Word-Net's noun hierarchy [10], between the two successive name entities n_m, n_n in a sentence. The assumption behind this definition is that events are delegated by event terms and discriminated and interrelated by the associated name entities. Four types of named entities are currently under the consideration. They are <Person>, <Organization>, <Location> and <Date>.

3.1 Building Event Ontology with FCA

The events in triple patterns consisting of an event term and two name entities, $\langle n_m, t_i, n_n \rangle$, are extracted from documents. For instance, we can extract two event terms, spoke and attacking from the following illustrative sentence. They are both associated with the <Person> James Clark and <Organization> Microsoft.

<Organization> Netscape </Organization> chairman <Person> James Clark </Person> spoke boldly of attacking <Organization> Microsoft </Organization> head-on.

The hierarchical structure of event terms, which is deemed as event ontology, is constructed with Formal Concept Analysis (FCA). FCA takes two sets of data, one is called the *object* set and the other is called the *attribute* set, to find a binary relationship between the data of the two sets, and further constructs a so-called formal ontology. Attributes allow more complex relations to be modelled using the ontology.

The associated name entities of event terms conceal the relations between events. We believe that if two events are concerned with the same person or same location, or occurred at the same time, these two events are probably interrelated with each other. To construct event ontology with FCA, event terms are mapped into objects and name entities into attributes. The binary relationship between the event term t_i and the name entities n_j is determined to be 1, if t_i and n_j are associated in a triple pattern. It is 0 otherwise.

A FCA tool, called ConExp¹ (Concept Explorer) can be used to visualize the ontology by lattice, as illustrated in Figure 1. To further interpret the relationships of any two objects, we here define two kinds of relations. Objects are *equivalent* when they are associated with exactly the same attributes (such as t_3 and t_4). The object with

¹ Free downloadable from <http://sourceforge.net/projects/conexp>.

subset of attributes is considered as a *super-class* of the object with superset of attributes (such as t_1 and t_4). Otherwise, they are not directly related.

obj.\att.	n_1	n_2	n_3	n_4
t_1	1	0	0	1
t_2	0	1	1	0
t_3	1	0	1	1
t_4	1	0	1	1

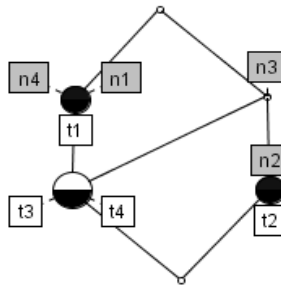


Fig. 1. Example of event ontology (event terms as objects, associated name entities as attributes)

The relations of objects are explicitly indicated in the lattice. As shown in Figure 1, the equivalent objects are denoted by a same node. The nodes in the upper levels are actually the super-classes of those in the lower levels.

3.2 Measuring Event Relevance

We propose the following event relevance measures by exploring the previously constructed ontology. The relevance between t_i and t_j is denoted by $R(t_i, t_j)$.

We first start from clusters (i.e. the nodes in ontology) provided by ontology. Event terms are assumed to be relevant only if they are in the same node (i.e. they are equivalent as ontology specifies). In such a way, the relevance is symmetrical in nature. This is where the idea of the approach Binary and Symmetrical Measure 1 (BSM1) comes from.

As the super/sub class relations are taken into the consideration, the unbalanced relations exhibit. As illustrated in Figure 1, if t_1 is the super-class of t_4 , all its attributes, n_1 and n_4 , are included in t_4 's attribute set. This relation is not hold for t_4 , because it has one more attribute n_3 . Therefore when t_i is the super-class of t_j , the relation from t_i to t_j is assumed to be stronger than from t_j to t_i , i.e. $R(t_i, t_j) > R(t_j, t_i)$. The approach Binary and Asymmetrical Measure (BAM) are therefore introduced to cope with these unbalanced relations.

To go further, we consider not only the nodes directly related but also those indirectly related, such as t_2 and t_4 . They are neither equivalent in one node and nor related by super/sub class relation. But they are indirectly liked by some common super-class, which is a virtual node in Figure 1. The indirect relevance is measured with the approach Binary and Symmetrical Measure 2 (BSM2).

Finally, the scaled approaches are extended from the binary approaches. Whereas the binary value can only represent whether two event terms are relevant or not, the scaled value indicate how strong the two event terms are related. In conclusion, based on event ontology constructed, several approaches, varied from binary to scaled and symmetrical to asymmetrical, are proposed to measure the event relevance in our work:

- Binary and Symmetrical Measure 1 (BSM1): If two event terms t_i, t_j are equivalent, $R(t_i, t_j) = R(t_j, t_i) = 1$. Otherwise R is 0.
- Binary and Asymmetrical Measure (BAM): BAM is the extension of BSM1. In addition to handle the equivalence terms in the same way as in BSM1, if the event term t_i is the super-class of the event term t_j , $R(t_i, t_j) = 1, R(t_j, t_i) = 0$. Otherwise R is 0.
- Binary and Symmetrical Measure 2 (BSM2): BSM2 is a further extension from BAM. If two event terms t_i, t_j have at least one attribute in common, $R(t_i, t_j) = R(t_j, t_i) = 1$, Otherwise R is 0. On the ontology, these two event terms are either equivalent, directly related by super/sub classes or indirectly related with at least one super-class node in common.
- Scaled and Asymmetrical Measure 1 (SAM1): SAM1 is an extended BAM assessing event relevance by decimal fraction instead of binary value. If the event term t_i is a super-class of the event term t_j , and t_i has k attributes $n_{i1}, n_{i2}, \dots, n_{ik}$, t_j has l attributes $n_{j1}, n_{j2}, \dots, n_{jl}$ ($k < l$), then $R(t_i, t_j)$ is 1 and $R(t_j, t_i)$ is k/l . Otherwise R is 0.
- Scaled and Asymmetrical Measure 2 (SAM2): Similarly, SAM2 is extended from BSM2. Suppose the event term t_i has k attributes $n_{i1}, n_{i2}, \dots, n_{ik}$ and the event term t_j has l attributes $n_{j1}, n_{j2}, \dots, n_{jl}$. If t_i and t_j have m attributes in common, then $R(t_i, t_j)$ is m/k and $R(t_j, t_i)$ is m/l . Otherwise R is 0. SAM2 is also an extension from SAM1 in the sense that the nodes with common super-classes are also considered as relevant.

The matrix representation is suitable to formalize the relevance between any two events terms. The value at the cross of column t_i and row t_j is $R(t_i, t_j)$. For instance, the matrix provided by BAM with the data given in Figure 1 is shown in Figure 2.

Relevance	t_1	t_2	t_3	t_4
t_1	-	0	1	1
t_2	0	-	0	0
t_3	0	0	-	1
t_4	0	0	1	-

Fig. 2. Example of relevance measure with BAM with the example data given in Fig.1

4 Summarization with Event Relevance

Given event term relevance, if an event term is relevant with more other event terms, it is assumed to be more significant in representing a salient concept. The event terms relevant to the significant terms are thereby more close to the salient concept than those not. We estimate term significance with *PageRank*, an efficient algorithm to exploit event term maps by linking relevant terms together [11]. It assigns the significance score to each event term according to the number of event terms linking to it as well as the strength of the links. The equation to calculate the page rank (indicated by *PR*) of a certain term *A* is shown as follows:

$$PR(A) = (1 - d) + d \left(\frac{PR(B_1)}{C(B_1)} + \frac{PR(B_2)}{C(B_2)} + \dots + \frac{PR(B_t)}{C(B_t)} \right). \quad (1)$$

In expression (1), B_1, B_2, \dots, B_t are all terms which link to term *A*. $C(B_i)$ is the number of outgoing links from term B_i . d is the factor used to avoid the limitation of loop in the map structure. The significance score of each term can be obtained recursively with this equation. The significance of each sentence to be included in the summary is then calculated from the significance of the event terms it contains.

5 Experiment, Evaluation and Discussion

5.1 Evaluation on Event-Based Summarization

To evaluate the effectiveness of integrating event relevance into multi-documents summarization, we conduct the experiments on the 50 sets of English documents from DUC 2004 multi-document summarization task. The documents are pre-processed with GATE² to recognize the previously mentioned four types of name entities³. In average, each set contains 10 documents, 149 event terms and 76 name entities.

Figure 3 shows an example of event ontology constructed with FCA based on a paragraph of real news in DUC 2004 data. This paragraph is about the Microsoft Corp.'s firm grip on the personal computer software business. As shown in Figure 3, much important information about this topic is extracted from the news, such as the

² Free downloadable from <http://gate.ac.uk>.

³ GATE also provides other types of named entities. But only four of them are recognized to fit our application.

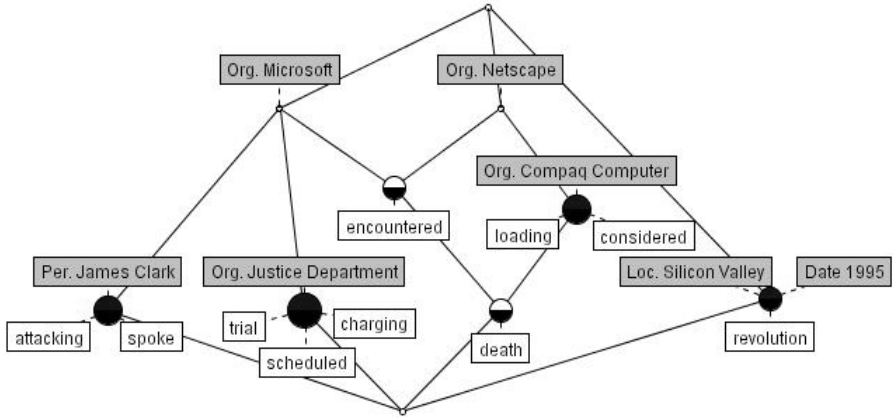


Fig. 3. Event ontology constructed from a paragraph of real news

names of IT companies and Justice Department, the time and location. The node of revolution event is associated with the location Silicon Valley and the year 1995. The efficiency of extracting events, which carry the most important information, from texts was also discussed in [2, 12]. A paragraph of news is shorter comparing to a set of topically related texts in the task of multi-document summarization where many links between events might be presented. But it somehow provides evidence that the relevant events or event terms can be discovered with FCA. For example, the verb trial and charging are equivalent in one node. The action noun death which is a probable consequence of the verb encountered is the super-class of encountered.

To evaluate the quality of summaries, we use an automatic summary evaluation metric ROUGE⁴, which has been used in DUCs. ROUGE is a recall-based metric for fixed length summaries. It bases on *N*-gram co-occurrence and compares the system-produced summaries to human judges [13]. For each DUC document set, we create a summary of length less than 665 bytes and present three of the ROUGE metrics: ROUGE-1 (unigram-based), ROUGE-2 (bigram-based), and ROUGE-W (based on longest common subsequence weighed by the length).

Table 1 compares the ROUGE evaluations of relevance-based approaches with the baseline of using event term centroid scheme as sentences selection criteria. As it

Table 1. Evaluations of event-relevance-based and independent-event-based summarization approaches

	ROUGE-1	ROUGE-2	ROUGE-W
centroid	0.28042	0.04570	0.10858
BSM1	0.28746	0.04339	0.11053
SAM1	0.29062	0.04756	0.11206
BAM	0.29760	0.04662	0.11589
BSM2	0.30166	0.05519	0.11658
SAM2	0.30192	0.05240	0.11774

⁴ <http://www.isi.edu/~cyl/ROUGE/>

indicates, the summaries created by event relevance receive a higher ROUGE score than the baseline summaries created by independent events. Better results are from SAM2 and BSM2. This is natural because they consider the terms related together indirectly. The same can also explain the improvements from BSM1 to BAM to BSM2. When name entity recognition and entity co-reference are not quite successful nowadays, the strict approaches, which consider direct relations only, are more error sensitive. Unfortunately, the asymmetrical measures proposed do not significantly outperform the symmetrical measures right now. In this first set of experiments, we do not merge the attributes. The issue of merging named entities will be discussed in the next subsection.

5.2 Discussion on Named Entity Mention Links

Our work depends on named entities to determine the relevance of events. During experiments, we observe some redundant attributes. Take the set of news about Cambodia as example. Several person names are extracted as follows,

Ranariddh
 Prince Norodom Ranariddh
 Norodom Sihanouk
 Sihanouk
 President Prince Norodom Ranariddh
 King Norodom Sihanouk

Actually, these six names mentioned above correspond to two person entities, i.e. Prince Norodom Ranariddh and King Norodom Sihanouk. However, they are considered as distinct attributes to differentiate the event terms simply because their surface texts are different. FCA provides the function to merge the redundant attributes. If the named entity mentions that represent the same or similar entities could be linked together (this is hereafter referred to as entity normalization), efficiency and precision of event relevance discovery might be improved. At present, we only consider the person’s names as an initial step to investigate the contributions of entity normalization, because of its observable repetitions in texts and its relatively straightforward variations. The clustering algorithm for linking person name mentions is given below:

Step1: For each person name $p_i = w_{i1}w_{i2}...w_{ik}$, w are the words in person name. Its person cluster $C(p_i)$ is initiated by the person name p_i .

Step2: For each person name $p_i = w_{i1}w_{i2}...w_{ik}$
 For each person name $p_j = w_{j1}w_{j2}...w_{jl}$, if $C(p_i)$ is a substring of $C(p_j)$, then $C(p_i) = C(p_j)$.

Continue Step 2 until no change occurs.

This simple algorithm can avoid merging names overly. For instance, if $A=ab$, $B=a$, $C=b$, then $A=B=a$ in iteration 1, and C can no longer merge with A or B . If $A=abc$, $B=a$, $C=ab$, then $A=B=a$ in iteration 1 and $C=A=B=a$ in next iteration.

Table 2 shows that named entity mention links affect the performance in some extent but not evidently. The most likely reason is that person names are not contained in all events. The results of these experiments also show an interesting phenomenon, i.e. entity mention links improve the performance of BSM1, have no effect on BAM and SAM1 yet cause decreases in BSM2 and SAM2. These results corroborate the previous conclusions. The automatic recognition of name entities unavoidably introduces errors. When the restriction of event relevance is getting less from BSM1 to BAM and then to BSM2, these errors are amplified gradually. When more events are related together in BSM2 and SAM2, the significance of events is indistinct with *PageRank* algorithm. In contrast, the stricter approaches benefit from the entity mention links for the same reason. The experiments suggest that the improvement of name entity recognition can help the event-based summarization.

Table 2. Result: with and without linking entity mentions

		ROUGE-1	ROUGE-2	ROUGE-W
BSM1	Without	0.28746	0.04339	0.11053
	With	0.28790	0.04453	0.11098
SAM1	Without	0.29062	0.04756	0.11206
	With	0.29243	0.04832	0.11286
BAM	Without	0.29760	0.04662	0.11589
	With	0.29760	0.04662	0.11589
BSM2	Without	0.30166	0.05519	0.11658
	With	0.30042	0.05523	0.11654
SAM2	Without	0.30192	0.05240	0.11774
	With	0.29929	0.05198	0.11584

6 Concluding Remark

In this paper, we propose a novel approach for measuring event relevance and integrating event relevance into text summarization. The experimental results indicate that event relevance is effective for extracting the salient concepts in document sets. The discussion on entity mention links shows that the improvement of named entity recognition and entity co-reference can benefit the event-based summarization.

Our approach can be further improved in the following directions. First, we consider refining the definition of event to capture the corresponding name entities more exactly. Second, we are considering prioritizing special name entities to improve the precision of event relevance. Third, we are also looking at extending the name entities to the common entities for associating events.

Acknowledgements

The work presented in this paper is supported partially by Research Grants Council on Hong Kong (reference number CERG PolyU5181/03E) and partially by National Natural Science Foundation of China (reference number: NSFC 60573186).

References

1. N. Daniel, D. Radev and T. Allison: Sub-event based Multi-document Summarization. In Proceedings of the HLT-NAACL 2003 Workshop on Text Summarization, (2003) pp9-16.
2. E. Filatova and V. Hatzivassiloglou: Event-based Extractive summarization. In Proceedings of ACL 2004 Workshop on Summarization, (2004) pp104-111.
3. R. Barzilay and M. Lapata: Modeling Local Coherence: An Entity-based Approach. In Proceedings of ACL 2005, (2005) pp141-148.
4. N. Guarino: Formal Ontology and Information Systems. In Proceedings of FOIS 98, Amsterdam, (1998) pp3-15.
5. K. Kevin: Building a Large Ontology for Machine Translation. In Proceedings of the ARPA Human Language Technology Workshop, (1993).
6. H. Alani, S. Kim, D.E. Millard, M.J. Weal, W. Hall, P.H. Lewis, and N.R. Shadbolt: Automatic Ontology-Based Knowledge Extraction from Web Documents. *IEEE Intelligent Systems*, (2003) 8(1):14-21.
7. H.M. Haav: An Application of Inductive Concept Analysis to Construction of Domain-specific Ontologies. In Proceedings of the Workshop of VLDB2003, (2003) pp63-67.
8. G. Jiang, K. Ogasawara, A. Endoh, T. Sakurai: Context-based Ontology Building Support in Clinical Domains using Formal Concept Analysis. *International Journal of Medical Informatics*, (2003) 71(1):71-81.
9. S.J. Li, Q. Lu, W.J. Li: Experiments of Ontology Construction with Formal Concept Analysis. In Proceedings of IJCNLP 05 Workshop on Ontologies and Lexical Resources, (2005).
10. C. Fellbaum, *WordNet: An Electronic Lexical Database*, MIT Press, (1998).
11. L. S. Page, B.R. Motwani, and T. Winograd. *The Pagerank Citation Ranking: Bring Order to the Web*. Technical Report, Stanford University, (1998).
12. E. Filatova and V. Hatzivassiloglou. Domain-independent Detection, Extraction, and Labeling of Atomic Events. In Proceedings of RANLP, (2003) pp145-152.
13. C.Y. Lin and E. Hovy: Automatic Evaluation of Summaries using N-gram Co-occurrence Statistics. In Proceedings of HLT-NAACL, (2003) pp71-78.

A Sentence Compression Module for Machine-Assisted Subtitling

Nadjet Bouayad-Agha, Angel Gil,
Oriol Valentin, and Victor Pascual

Universitat Pompeu Fabra,
Barcelona, Spain
nadjet.bouayad@upf.edu, firstname.lastname@upf.edu

Abstract. We present in this paper a sentence compression module used in a machine-assisted subtitling application developed in the European e-content project e-title. Our approach to compression and the architecture of the system are motivated by the commercial and multilingual nature of the project, that is, the need to output reasonable compressions and the ability to add new strategies, genres and languages easily. The compression module currently works for the Catalan and English languages and uses the Constraint Grammar engine for linguistic preprocessing and for the linguistically motivated compression rules, thus providing a homogenous format throughout the compression process. The compression rules were implemented based on a corpus of automatically aligned <script,subtitle> pairs of films for both languages. We performed for both languages an automatic quantitative evaluation of the compression using the aligned corpus and a qualitative manual evaluation of grammaticality and informativeness.

1 Motivation

We present in this paper a sentence compression module used in a machine-assisted subtitling application developed in the European e-content project e-title.¹ This application integrates speech-text synchronisation, machine translation and sentence compression to assist subtitlers in the different stages of the subtitling process. Our approach to compression and the architecture of the system are motivated by the commercial and multilingual nature of the project, that is, the need to output reasonable compressions and the ability to add new strategies, genres and languages easily. We surveyed various approaches to sentence compression developed in Natural Language Processing, for example (Jing, [6]; Zechner, [12]; Hori and Furui, [7], Knight and Marcu, [9], Vandeghinste and Pan [10]), and compiled the following list of desiderata for our system:

Grammaticality: the compression module should preserve grammaticality. We found that some approaches guarantee the grammaticality of the output but at the cost of heavier linguistic machinery such as full parsing and subcategorization information (Jing,[6]).

¹ EDC22160. Jan.2004–Jan.2006.

Content worthiness: the compression module should preserve content worthy elements in the sentences. In previous approaches this is achieved by global strategies such as global term frequency weighting (Hori and Furui, [7] or lexical cohesion relations (Jing, [6]).

Strategies variety: the compression module should allow the integration of a variety of compression strategies within the same genre and across different genres. However, in general, compression is viewed as a global problem (e.g., phrase reduction). Exceptions are Vandeghinste and Pan ([10]) who consider abbreviation and number to digit reduction and Zechner ([12]) who considers speech disfluencies removal.

Availability of linguistic resources and core technologies: full parsers and knowledge intensive resources are not always readily available for the languages considered.

Compression rate: the compression module should take into account compression rate.

Confidence score: the compression module should display a confidence score for each compressed segment.

We also found out from human subtitling literature that sentence compression strategies are characterized by their sheer variety (Díaz, [3]) and are on a continuum of semantic content loss, ranging from the more mechanical strategies (e.g., removal of repetitions or tag questions) to the more semantico-pragmatic (e.g., reduction of content that can still be conveyed by the audio-visual media) (Gottlieb,[5]). Thus, our initial aim was to start with the more mechanical strategies with a higher confidence score, working upwards.

Given these requirements, we opted for developing a compression toolkit based on shallow linguistic resources in which individual strategies can be developed and integrated independently of one another. Each has a confidence score that contributes to the overall confidence score of the compressed sentence. In section 2, we present a corpus resource which consists of <script,subtitle> pairs which are aligned in blocks and then sentences using an automated procedure. This corpus allows us to make informed decisions about which strategies to implement given a particular genre and what particular linguistic constructs are being used. It was also used to evaluate the performance of the system. In section 3, we present the architecture of the compression module followed in section 4 by an evaluation of its current performance. Finally, in section 5 we conclude with future work.

2 Corpus Resources

We gathered a corpus of 55 film pairs in Catalan and 40 film pairs in English. The Catalan corpus was obtained from a single source whilst the English corpus was obtained from different heterogeneous sources. These pairs were automatically cleaned and manually revised to remove time-stamps in the subtitles and speaker names and scene descriptions in the script and other meta-data. We then applied automatic tokenization and sentence boundary detection.

2.1 Alignment Procedure

The alignment procedure consists in the alignment of blocks followed by the alignment of sentences within each block. In addition to make sentence alignment possible, block alignment is motivated by the large discrepancies between script and subtitle: whole chunks are missing or inserted in the script or the subtitle. Block alignment is a 10-fold iterative process which consists in aligning blocks of script and subtitle using identical sentences of decreasing sizes (i.e., from 10-words to 1 word) as unique delimiters between the blocks. The result is a set of aligned blocks. The sentence alignment approach was developed to obtain 0:1, 1:0, 1:1, 1:N, N:1 and M:N alignments (where M and $N > 1$) to account for sentence deletion, sentence splitting or merging as well as for incorrect sentence boundary detection in the preprocessing phase. The sentence alignment procedure is as follows:

1. We calculate the similarity between each pair of sentences using the Cosine measure of similarity on vectors of character trigrams. Thus, we get in addition to 1:1 alignments, all the potential 1:N and N:1 alignments and by default 0:1 and 1:0 alignments using a threshold of 25% similarity.
2. If there are exact alignments (similarity = 1), then we keep only those, hence discarding possible multi-sentence alignments (1:N or N:1).
3. If we find crossing dependencies, we remove the ones with lower similarity. If alignments have equal similarity, we keep the one with the closest alignment: this is to reflect the intuition that script and subtitle are likely to reflect the same order.
4. Finally we join the related 1:N or N:1 alignments to obtain M:N alignments.

2.2 Results and Evaluation

We manually corrected about 5% of the aligned blocks in Catalan (2040 blocks out of 41708). Comparison of this manual corpus with the raw automatic alignment gives a precision of 97% and a recall of 88.5%. Table 1 shows the percentage of the different types of alignments we found for Catalan and English. It shows that 22% and 16% of sentences are compressed in Catalan and English respectively.²

3 Architecture

The general architecture of the compression module is presented in Figure 1 and consists of the following three submodules: (1) linguistic preprocessing, (2) compression candidates production and (3) compression candidates selection. The input to the compression module is a transcript with subtitle delimiters. The

² Table 1 also shows that 15% and 21% of the sentences in the scripts are fully omitted in the subtitles. However we cannot distinguish between cases when this is due to a discrepancy between script and subtitle and cases when this is due to a conscious omission of a whole sentence by the subtitler.

Table 1. Block and sentence alignments for Catalan and English

	Catalan	English
# subtitle sentences deleted	4786 (5.1%)	9825 (15.9%)
# script sentences deleted	20197 (21.6%)	9673 (15.6%)
# sentences copied/transformed	45464 (48.6%)	28230 (45.6%)
# subtitle sentences compressed	20553 (22%)	10105 (16.3%)
# script sentences compressed	2492 (2.7%)	4059 (6.6%)
# total aligned sentences	93492 (100%)	61892 (100%)
# aligned blocks	41708	9885
# script sentences	91571	54922
# subtitle sentences	75048	57607

output is a sequence of subtitles, each of which consists of a series of suggestions with compression rate and confidence score assigned. We used the Constraint Grammar (CG) Engine for linguistic preprocessing (e.g., tagging and chunking) and to write most of the compression rules for the following reasons:

- A robust CG morphological analyser had been developed for Catalan by one of the project members (GLICOM) (Alsina et al., [1]) so we had local expertise and resources.
- CG taggers have been implemented for various languages which makes the addition of new languages in the compression module more straightforward.³
- CG rules are very straightforward to implement yet powerful, taking into account rich linguistic information and context (Karlsson et al, [8]). The CG compression rules took about two-person-week to implement for each language.
- The CG Engine is fast, which makes its use in a commercial application viable.

3.1 Linguistic Preprocessing

For both English and Catalan, the linguistic preprocessing applies tokenization, part of speech tagging and chunking. The POS tagger used for Catalan is the CATCG (Constraint Grammar for Catalan) developed by GLICOM (Alsina et al., [1]). The output of this rule-based tagger aimed at unrestricted text was specifically disambiguated for this project using the decisions made by the statistical tagger TNT (Brants, [2]) trained on the 15 million word manually tagged IEC Corpus⁴. For the purposes of sentence compression, we also developed for Catalan a CG locution detector comprising more than 900 adverbial, prepositional, adjectival and conjunctive locutions compiled from online Catalan dictionaries (e.g., *fins i tot = even, a la vora de = next to*).

For English, we used ENGCG, the English Constraint Grammar provided by Connexor, which gives each word a single POS. The CG chunkers for both

³ These taggers are provided commercially by Connexor (www.connexor.com).

⁴ IEC stands for Institut d'Estudis Catalans.

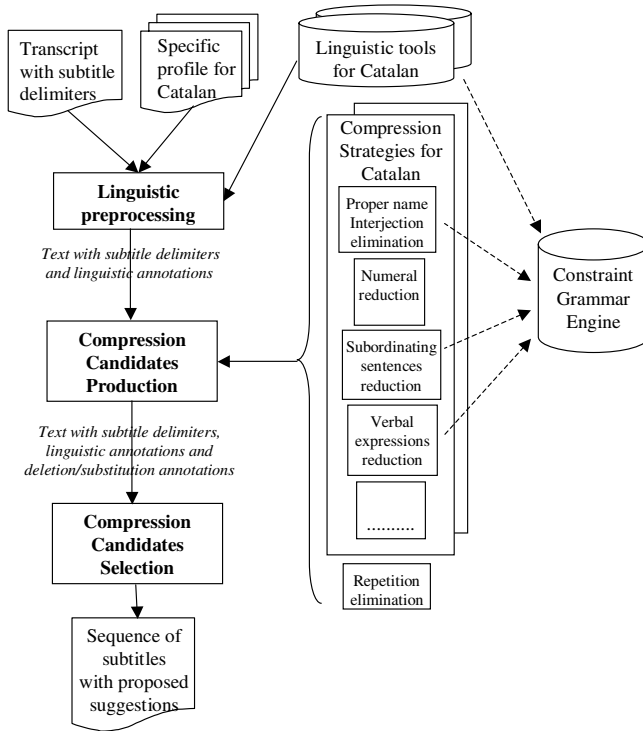


Fig. 1. Architecture of the compression module

English and Catalan consist of a set of rules to detect basic NPs, PPs and VPs to be used in the compression strategies. Basic VPs simply consist of the main verb, its clitics, modals and periphrastic constituents.

3.2 Compression Candidates Production

This submodule applies in cascade a set of CG rules and programs for detecting compression strategies in the specific language and genre. Its output is a single CG-like file with all the total and partial reductions marked on all the subtitles regardless of the compression rate. Figure 2 shows some output excerpts of the candidates production module, which incrementally assigns deletion and substitution markers (prefixed [DEL_ and [SUB_ respectively) to the linguistically annotated subtitles.

At present, the strategies implemented are all informed by strong lexical, grammatical or punctuation clues. All but number-to-digit substitution, repetition deletion and partial verbal expressions reduction are implemented as CG rules. In order to add new strategies, jar files or CG rules can simply be added to the strategies directory. The types of strategies implemented are described

"<Nois>	"noi" Nom com masc pl maj N5-MP @:[NP @:[DEL_VOCNOM
"<, >	F @:[PUNCT
"<ara>	"ara" Adv D4 @:[ADV
"<potser>	"potser" Adv D4
"<ara>	"ara" Adv D4 @:[ADV
"<potser>	"potser" Adv D4 @:[DEL_REPEAT:-2:
"<en>	"en" Prep P @:[PP
"<el>	"el" Det art masc sg EAMS @:[NP
"<segle>	"segle" Temp Nom com masc sg N5-MS
"<vint>	"vint" Nom com masc-fem sg N5-6S
"<->	F @:[PUNCT
"<i>	"i" Conj coord CC @:[COORD
"<->	F @:[PUNCT
"<u>	"u" Nom com masc-fem sg N5-6S @:[NP @:[SUB_NUMBER:-5:[21]:
"	"me" Pron person febl acus-dat 1pers masc-fem sg @:[VP @:[DEL_SUB[
"<sembla>	"semblar" Verb Mind Pres 3pers sg VDR3S-
"<que>	"que" Conj subord CS @:[SUBORD @:[DEL_SUB]
...	
"<ens>	"ens" Pron person febl acus-dat 1pers masc-fem pl REE616P @:[VP
"<dedicàvem>	"dedicar" Verb MInd Imp 1pers pl VDA1P- @:[TRANS_VPERIF[
"<a>	@:[MORPHO @:[SUB_VPERIF:+3:[treballàvem]:
"<a>	"a" Prep P @:[DEL
"<treballar>	"treballar" Verb Inf VI— @:[LEMMA @:[TRANS_VPERIF]
...	
"<, >	F @:[PUNCT
"<oi>	"oi" Interj I @:[INTERJ @:[DEL_INTERJ[
"<que>	"que" Pron rel masc-fem pl RR—6P @:[REL @:[DEL_INTERJ]
"<sí>	"sí" Adv D4 @:[ADV
"<?>	F @:[PUNCT

Fig. 2. Output of the compression candidates production module. (In English literally: *Kids, now maybe, now maybe in the twenty first century, it seems to me that...we dedicated ourselves to work...isn't it?*).

below (for both languages, unless specified otherwise). We have written a total of 158 CG rules for English and 230 for Catalan, which are grouped in different files corresponding to the type of reduction: prepositional phrases, abbreviations, ellipsis, adverbs, adjectives, etc.

A general purpose transducer engine was implemented that takes as input a transducer (written in text format using a straightforward syntax) and the linguistically tagged text which it marks with corresponding substitutions (e.g., Fig.2, @:[SUB_NUMBER:-5:[21]:), which substitutes the last 5 words by "21"). We have written transducers for ordinal and cardinal number to digits and time expressions.

We have a repetition detector (e.g., Fig.2, @:[DEL_REPEAT:-2:) that calculates inter and intra-sentence similarity using the Dice coefficient within a given window. This program is language-independent.

A set of rules was written for the detection of filler words and other constructions common to spoken languages (interjections, vocative proper nouns, sentence initial coordination, etc). These rules use punctuation in addition

to lexical and linguistic knowledge as a clue (e.g., Fig.2, @:[DEL_VOCNOM to delete the vocative noun “nois”, @:[DEL_INTERJ to delete “oi que”).

We also use separate rules for the detection of adverbial and nominal expressions, adjectives, ellipsis, prepositional phrases, specifiers. For instance, the following segments can be removed without affecting significantly informativeness (or grammaticality): specifiers in the form of *una mica de* (“a little bit of”), *un tros de* (“a piece of”), prepositional phrases with strong personal pronouns as in *[a mi] em sap greu* (literally: “to me, I feel sorry”) can be removed, noun phrases with interjective or hedging function as in *what [the heck] is this, 14 [years old]*.

We perform the detection of subordinating segments as in Figure 2, @:[DEL_SUB instructing to delete “em sembla que”. For Catalan, since we were not sure about which rules worked best for detecting such segments, we manually marked about 600 of them (verb followed by subordinating conjunction) as “can delete” or “cannot delete”. Then we applied JRIP from WEKA (Weka, [11]), a rule induction machine learning program, based on the following features which we suspected played a role in this decision: lemma, person, tense, aspect of subordinating verb, whether it has a weak pronoun, whether it is negated, whether the subordinated verb is indicative, whether the subordinated clause contains a VP, whether it contains an NP, whether the subordinating clause contains an NP. JRIP gave the following simple four rules:

1. If the subordinated verb is indicative and the lemma of the subordinating verb is “ser”, then the category is “delete”.
2. If the lemma of the subordinating verb is “semblar”, then “delete”.
3. If the subordinated verb is indicative, and the subordinating verb is 1st person, present, then “delete”.
4. Otherwise, the category is “cannot delete”.

For the “delete” category, the F-measure is 71% (precision: 76%, recall: 67%). For the “cannot delete” category, the F-measure is 75% (precision: 71%, recall: 80%). This also allows us to get an appreciation of the confidence level of this strategy.

We perform the partial reduction of periphrastic verbs in Catalan, as shown in Figure 2 with @:[SUB_VPERIF:+3:[treballàvem]: reducing *ens dedicàvem a treballar* to *treballàvem*. This is done by the marking in the preprocessing phase of all main verbs and their auxiliary elements as a VP requiring transformation (@:[TRANS_VPERIF). Within that segment, the lemma, relevant morphological features and words to delete are marked respectively as @:[LEMMA, @:[MORPHO and @:[DEL. The former two are then used to generate the new reduced form by looking up a set of verbal paradigms. This strategy is still in a beta version: some of the reductions must be prohibited since they change the meaning (e.g., “he deixat de fumar” and “he fumat” are opposites – “I quit smoking” vs “I smoked”).

```

LIST PRONFEBL = (Pron person febl) ;
LIST VP = (@:[VP] ;
LIST VIND = (Verb MInd) ;
LIST VERB = (Verb) ;
LIST COMP = ("que" Conj subord) ;

ADD (@:[DEL_SUBORDINATING[]] TARGET VP (0 PRONFEBL) (1 VERB + (Pres 1pers))
(2 COMP) (*3 VIND BARRIER VERB) ;
ADD (@:[DEL_SUBORDINATING[]] TARGET COMP (-1 (Verb) + (Pres 1pers))
(-2 PRONFEBL) (*1 VIND BARRIER VERB) ;

ADD (@:[DEL_NOUN] TARGET NOUN IF (0 ("time")) (1 ("ago")) (-1 ("long")));
ADD (@:[DEL_NULL] TARGET (det) IF (0 ("a")) (1 ("long")) (2 ("time"))
(3 ("ago")));

```

Fig. 3. A CG rule for Catalan subordinating segment reduction and another for English nominal reduction

Table 2. Applied strategies for Catalan

Filler coordinated conjunction	3784 (21.6%)
Number to Digit	2129 (12.1%)
Vocative proper noun	1659 (9.5%)
VP adverb	1332 (7.6%)
Lexical paraphrasing	1292 (7.4%)
Interjection	1211 (6.9%)
Periphrastic verbs (partial reduction)	1111 (6.3%)
Filler subordinating segment	939 (5.3%)
Adverbial	702 (4%)
Filler imperative verb	534 (3.04%)
Other	2861 (16.3%)
Total	17554 (100%)

We apply some basic abbreviations in context, for instance, replacing “Senyor” with “Sr” if it is used as the title of a proper name, replacing expressions like “the twenties” with “the 20s”. Other abbreviations include monetary symbols and measuring units.

We perform a series of lexical paraphrases, some of which we hope are relatively common and systematic in the dialogue genre whilst others almost certainly occur only rarely. For instance, we replace *would* with *'d* if used as a modal, *thank you* with *thanks*, *what's the matter* with *what's wrong*.

An example of CG rules for Catalan subordinating segment reduction and English nominal reduction is given in Figure 3. The Catalan rule is a strict implementation of the third rule given by JRIP (see above) and basically reads

Table 3. Applied strategies for English

Adverbial	2892 (26.3%)
Interjection	1743 (15.8%)
Apposition	1208 (11%)
Vocative common noun	839 (7.6%)
Nominal expressions	753 (6.8%)
Modal contraction	691 (6.3%)
Filler coordinated conjunction	499 (4.5%)
Repetition	427 (3.9%)
Filler subordinating segment	358 (3.2%)
Specifier	291 (2.6%)
Other	1313 (12%)
Total	11014 (100%)

as: enclose as a subordinating segment a VP consisting first of a weak pronoun, then a first person verb in the present tense and then a subordinating conjunction if the first verb after the conjunction is indicative.⁵ The English rules reduces “a long time ago” to “long ago”:⁶

Tables 2 and 3 show the most frequent strategies applied by the compression module on the whole corpus for Catalan and English respectively (55 Catalan films and 40 English films). The strategies applied are varied and differ between languages inasmuch as we can compare them. Some achieve little compression though they are relatively frequent (e.g., filler coordinated conjunction, interjection) whilst others achieve more (e.g., apposition for English) though many are in between.

3.3 Compression Candidates Selection

The compression candidate selection submodule calculates the compression rate given the maximum number of characters per subtitle provided by the properties file. It applies the strategies only in the case where compression is needed. At the moment, all the strategies are applied and the compression candidates selection only proposes two candidates: the original and the compressed version. The confidence score of a compressed subtitle is calculated using the sum of penalties associated with each strategy applied. These penalties can be modified in the properties file. At present these penalties are given manually, with number to digit given the lower penalty and periphrastic verbal reduction and subordinating verb reduction given the highest penalty.

⁵ We also have another rule for when the subordinating verb does not come with a weak pronoun.

⁶ Given that the reduction is discontinuous (words “a” and “time”), only one of the reductions is to be considered an application of the strategy (DEL_NOUN), the other one (DEL_NULL) is a dummy strategy with no penalty.

4 Evaluation of the Current System

4.1 Automatic Quantitative Evaluation

We performed an automatic quantitative evaluation of the compression using the compressed $\langle \text{script}, \text{subtitle} \rangle$ sentence pairs in our aligned corpora (see section 2). The script sentences were compressed by marking each sentence as a subtitle and setting the maximum number of characters per subtitle to one character so as to force the compression to apply on every sentence.⁷ Recall and precision are calculated using:

- the number of words changed (i.e., deleted or replaced) in the subtitle (W_{sub}) with respect to the original script, as the number of correct deletions,
- the number of words changed in the compressed script (W_{comp}) with respect to the original script, as the number of detected deletions, and
- the number of words changed in both compressed script and subtitle ($W_{\text{comp}} \cap W_{\text{sub}}$), as the number of correctly detected deletions.

Table 4 shows that about 60% of the compression decisions our compression system makes are also made by human subtitlers for both English and Catalan. This can be considered satisfactory given the fact that not all possible compression strategies are generally applied in human subtitles and that there might be more than one way to compress the same subtitle. Thus, the other decisions that the compression system makes might still be valid. This will be verified with the subsequent manual qualitative evaluation. On the other hand, recall is quite low (20% and 28%). However, it was not given the priority in our system. We also found that human subtitles are reduced by 21% and 27% whilst our system reduces by 12% and 10% for English and Catalan respectively.

Table 4. Automatic quantitative evaluation of the compression

	Catalan	English
# W_{sub}	63530	31150
# W_{comp}	22682	14918
# $(W_{\text{sub}} \cap W_{\text{comp}})$	13072	8834
Precision	58%	59%
Recall	20%	28%

In a similar experiment, Jing ([6]) achieves 81.3% success rate which is the percentage of automatic reduction decisions that are also made by humans. However, her system, implemented for English on a corpus of 500 Telecommunications-related news sentence pairs, uses more extensive linguistic resources, which

⁷ Typically, subtitles are limited to 70 characters, 35 per line. However, we observed in some of the $\langle \text{script}, \text{subtitle} \rangle$ sentence pairs that reduction can occur for instance to fit a single sentence onto one line.

include a full parser, Wordnet and other lexical sources and statistical information from a trained corpus. Our use of shallow linguistic resources was driven by our objective to build a commercially viable system (see section 1).

Daelmans et al. ([4]) report on the results achieved by a reduction system for English and Dutch news broadcasts that uses the combination of a memory-based learner and hand-crafted reduction rules. The precision and recall for this combined system were 26.8% and 28% for Dutch and 25.3% and 20.3% for English with a reduction by 25.2% for Dutch and 16.4% for English.

4.2 Manual Qualitative Evaluation

We performed a manual qualitative evaluation of the first 10% of automatically compressed segments for each Catalan and English film out of a set of 30 Catalan and 22 English films. There was a different evaluator for English and Catalan. For each strategy applied, we checked whether its application preserved grammaticality and informativeness. The results presented in Table 5 show that 92.3% and 86.9% of the strategies verified are correct for Catalan and English respectively, which is a high performance.⁸ A similar kind of evaluation was performed by Vandeghinste and Pan ([10]) on a system that uses reduction statistics and general syntactic reduction rules. At best only 51% of the sentences were considered a reasonable compression by the human evaluators.

Table 5. Manual qualitative evaluation of the reductions

	Catalan	English
# reductions	17554	11014
# reductions checked	890 (100%)	704 (100%)
# reductions affecting grammaticality	51 (5.7%)	23 (3.3%)
# reductions affecting informativeness	18 (2%)	69 (9.8%)

5 Conclusions and Future Work

We have achieved our objective to build a system that made *some* compression decisions but made them right. In addition we provide a framework for easily creating compression systems using shallow linguistic tools. Currently only the most mechanical procedures with the highest confidence scores are implemented in the film genre for both Catalan and English. This maybe leaves room for the more creative aspects of subtitling to be realized by human subtitlers. The compression module was integrated in the e-title system⁹, though a user evaluation of the system including the compression is still to be done. Given the low recall,

⁸ A strategy is taken to either affect grammaticality or informativeness. If a strategy affects grammaticality, then basically there is no room to check whether it affects informativeness. And if a strategy affects informativeness, then it is necessarily grammatical.

⁹ The website of etitle can be found at: <http://www.etable.co.uk/>

having a fully functioning compression candidate selection sub-module has not been essential, though in the future we must provide a mechanism to select the strategies taking into account compression rate and confidence score. In the future, we would also like to use the automatic evaluation to add more new rules, to see whether they bring significant improvement and to decide on the priority of the strategies for the compression candidates selection module: choosing in priority the strategies with more effect. We are also interested in automating the process of paraphrase finding using the aligned corpus.

References

- Alsina, A., Badia, T., Boleda, G., Bott, S., Gil, ., Quixal, M. and Valentn, O.: CATCG: a general purpose parsing tool applied. In Proceedings of Third International Conference on Language Resources and Evaluation. Las Palmas. 2002. vol. III, pp. 1130–1134.
- Brants, T.: TnT – a statistical part-of-speech tagger. In Proceedings of the 6th Applied NLP Conference, ANLP-2000, April 29 – May 3, Seattle, WA. 2000.
- Díaz Cinta, J.: Teoría y practica de la subtitulación Inglés-Español. Ariel, 2003.
- Daelemans, W., Höthker, A., Tjong Kim Sang, E.: Automatic Sentence Simplification for Subtitling in Dutch and English. Proceedings of the 4th International Conference on Language Resources and Evaluation, pp. 1045-1048, 2004
- Gottlieb, H.: Subtitling - a New University Discipline. In: Dollerup, Cay, et al. (eds.). Teaching Translation and Interpreting. Amsterdam: John Benjamins. pp. 161–170, 1992.
- Jing, H. Sentence reduction for automatic text summarization. In Proceedings of the 6th Conference on Applied Natural Language Processing. pp. 310–315, 2000.
- Hori, C. and Furui S.: Automatic Summarization of English Broadcast News speech. Notebook of HLT2002, San Diego, U.S.A., pp. 228–233, 2002.
- Karlsson, F., Voutilainen, A., Heikkilä, J. and Anttila, A. (eds): Constraint Grammar : a language-independent system for parsing unrestricted text, volume 4 of Natural Language Processing. Mouton de Gruyter, Berlin/New York. 1995.
- Knight, K. and Marcu, D.: Summarization beyond sentence extraction: a probabilistic approach to sentence compression. Artificial Intelligence Journal. Extended version of paper: Statistics-based summarization – Step 1: sentence compression, AAAI 2002.
- Vandeghinste V. and Pan, Y.: Sentence Compression for Automated Subtitling: A Hybrid Approach. In Proceedings of the 42th Annual Meeting of the Association for Computational Linguistics. Workshop: Text Summarization Branches Out, Barcelona, Spain. July 2004.
- Witten I.H. and Frank E.: Data Mining: Practical machine learning tools and techniques. 2nd Edition, Morgan Kaufmann, San Francisco, 2005.
- Zechner, K.: Automatic Summarization of Spoken Dialogues in Unrestricted Domains. Ph.D. thesis, Carnegie Mellon University, School of Computer Science, Language Technologies Institute, November 2001. Also printed as: Technical Report CMU-LTI-01-168. 2001.

Application of Semi-supervised Learning to Evaluative Expression Classification

Yasuhiro Suzuki^{1,*}, Hiroya Takamura², and Manabu Okumura²

¹ Interdisciplinary Graduate School of Science and Engineering,
Tokyo Institute of Technology,
4259 Nagatsuta Midori-ku Yokohama, Japan, 226-8503
yasu@lr.pi.titech.ac.jp

² Precision and Intelligence Laboratory, Tokyo Institute of Technology,
4259 Nagatsuta Midori-ku Yokohama, Japan, 226-8503
{takamura, oku}@pi.titech.ac.jp

Abstract. We propose to use semi-supervised learning methods to classify evaluative expressions, that is, tuples of subjects, their attributes, and evaluative words, that indicate either favorable or unfavorable opinions towards a specific subject. Due to its characteristics, the semi-supervised method that we use can classify evaluative expressions in a corpus by their polarities. This can be accomplished starting from a very small set of seed training examples and using contextual information in the sentences to which the expressions belong. Our experimental results with actual Weblog data show that this bootstrapping approach can improve the accuracy of methods for classifying favorable and unfavorable opinions.

1 Introduction

An increasing amount of work has been devoted to investigating methods of detecting favorable or unfavorable opinions towards specific subjects (e.g., companies and their products) within online documents such as Weblogs (blogs), messages in a chat room and on bulletin board (BBS) [1, 2, 7, 9, 11, 12, 18]. Areas of application for such an analysis are numerous and varied, ranging from analysis of public opinion, customer feedback, and marketing analysis to detection of unfavorable rumors for risk management. The analyses are potentially useful tools for the commercial activities of both companies and individual consumers who want to know the opinions scattered on the World Wide Web (WWW).

To analyze a huge amount of favorable or unfavorable opinions, we need to automatically detect evaluative expressions in text.

Evaluative expressions are not mere words that indicate unique (favorable or unfavorable) polarity in themselves (such as the adjectives ‘beautiful’ and ‘bad’), but rather they are tuples of the subject to be evaluated, an attribute, and an evaluative word. Tuples are necessary because the evaluative polarity of

* Yasuhiro Suzuki currently works at Fujitsu.

an individual word is often ambiguous and is determined context-dependently. In the following example ¹, which is rather artificial due to direct translation, the first sentence is positive and the second sentence is negative, although both have the same word “high”.

- The storage capacity of this HDD is high.
- The noise of this HDD is high.

We thus define an evaluative expression as a tuple of a subject, an attribute, and an evaluative word (in the above example, ‘HDD’, ‘capacity’/‘noise’, and ‘high’).

A good way to automatically acquire evaluative expressions is to first extract candidate expressions from a collection of documents and to automatically classify them as positive (favorable), negative (unfavorable), or neutral (non-evaluative). Therefore, we study classifying candidate evaluative expressions in documents into the three classes.

Much work has been done to automatically label a piece of text according to its positive or negative polarity, as detailed in the next section. Several previous papers have addressed the task by building classifiers that rely exclusively upon labeled examples [1, 2]. By training classifiers from labeled data, one can apply familiar and powerful machine learning techniques such as SVM (Support Vector Machines) [3] and the naive Bayes classifier. However, in practice, obtaining enough labeled examples to train the classifiers accurately may be difficult.

A contrasting approach relies upon only unlabeled data. This makes using a large corpus possible. However, a drawback to such an approach is its low accuracy when used on actual data.

In the area of machine learning, using labeled and unlabeled examples together has often been found effective [4]. In this paper, therefore, we explore the use of bootstrapping methods that allow evaluative classifiers to be trained on a collection of unlabeled texts. Using labeled examples as training data, we apply a supervised learning algorithm to automatically train the classifier. The trained classifier can be then used to automatically classify evaluative expressions into polarity categories and generate more labeled examples, which in turn increases the training data, and this entire process can be repeated.

To implement the idea, in this work, we adopted the EM algorithm [5] for the bootstrapping method and the naive Bayes classifier for the supervised learning method. Specifically, using various contextual information (detailed in Sec. 4.2) as features in the classifier trained with small seed training examples, evaluative expressions and their contextual information are newly identified in unannotated texts. Trained again with the labeled examples, our classifier identifies more evaluative expressions in unannotated texts. We adopted the combination of the EM algorithm and the naive Bayesian method, because Nigam et al. [4] have already shown that this combination can yield better performance in the text classification task.

¹ Though we use Japanese text data, we illustrate our method with English examples for better understanding by non-Japanese readers.

2 Related Work

The previous work on sentiment classification can be divided into the following three classes, according to the target unit of text: a word (or expression), a sentence (or clause), or a full document.

- Word sentiment classification [6, 7, 8, 9]
- Sentence sentiment classification [10, 11]
- Document sentiment classification [1, 12, 2]

From another viewpoint, the work on sentiment classification can be divided into two different approaches: those that try to learn classifiers directly from the training corpus (supervised learning method) [1, 2, 10, 11, 12] and those targeting the acquisition of evaluative lexica in an unsupervised fashion [6, 7, 8, 9].

In the latest studies on document sentiment classification, classifiers based on machine learning (e.g., [1, 2]) performed better than knowledge-intensive classifiers. One of the main obstacles to producing a sentiment classifier in a supervised fashion is a lack of training data. Targeting words (expressions) makes obtaining training data more difficult because the data must be manually annotated with polarities. Because manually producing annotated data is time consuming, the amount of available annotated data is relatively small².

As mentioned in the Introduction, much work has been done on semi-supervised learning methods [4, 14, 17]. Since unannotated texts are easy to obtain, the semi-supervised learning framework can produce a much larger collection of labeled examples than are currently available in manually created data. Consequently, we believe that sentiment classification systems can be trained on extremely large text collections by applying this framework.

We think the work of Riloff and Wiebe [13] is most relevant to ours because they also used a semi-supervised learning method. Their classifier targeted sentences, they tried to learn extraction patterns for subjective (evaluative) expressions from the training data, These extraction patterns were then used for the classification. Roughly, their work was the application of a semi-supervised learning technique to subjective expressions, which is similar to our work. However, their work concentrated on the classification of sentences as subjective or objective, while our classification targets evaluative expressions as positive, negative, or neutral (non-evaluative).

Furthermore, while Riloff and Wiebe used only extracted patterns for subjective expressions to classify sentences, we use a variety of contextual information that can be obtained from the sentence to which an evaluative expression belongs.

3 Our Method

We propose to use a semi-supervised learning method for classifying evaluative expressions (i.e., tuples of a subject, an attribute and an evaluative word) into

² Targeting full documents is easier, because more training data, in the form of reviews, can be found on the WWW.

three classes: positive, negative or neutral³. We suppose that evaluative expressions appear with certain types of context, such as ‘I am really happy, because the storage capacity is high.’ or ‘Unfortunately, the laptop was too expensive.’ We would like to extract such contexts from labeled examples and then use those contexts to re-label examples. By iterating this procedure, we would be able to accurately classify evaluative expressions and simultaneously collect evaluative words and typical contexts.

In order to achieve this bootstrapping, we used the EM algorithm [5], which has a theoretical base of likelihood maximization of incomplete data and can enhance supervised learning methods. We specifically adopted the combination of the naive Bayes classifiers and the EM algorithm. This combination has been proven to be effective in the text classification [4]. Another famous semi-supervised-training method that has been shown to be effective in text classification is co-training [14]. We however could not use co-training in this task, since we do not have conditionally independent views, which are required for co-training.

We explain the EM-based method in the following section.

3.1 Evaluative Expression Classification with Naive Bayes Classifiers

This model has been successfully applied to text categorization and its generative probability of example \mathbf{x} given a category c has the form :

$$P(\mathbf{x}|c, \theta) = P(|\mathbf{x}|)|\mathbf{x}|! \prod_w \frac{P(w|c)^{N(w, \mathbf{x})}}{N(w, \mathbf{x})!}, \quad (1)$$

where $P(|\mathbf{x}|)$ denotes the probability that a text of length $|\mathbf{x}|$ occurs, $N(w, \mathbf{x})$ denotes the number of occurrences of w in text \mathbf{x} , and θ denotes all the parameters of the model. The occurrence of a text is modeled as a set of events, in which a word is drawn from the whole vocabulary.

In evaluative expression classification, categories c are the positive category, the negative category and the neutral category. Instances \mathbf{x} are represented by features including evaluative words and their context. A detailed description of features will be given in Sec. 4.

3.2 Incorporation of Unlabeled Data with the EM Algorithm

The EM algorithm is a method to estimate a model that has the maximal likelihood of the data when some variables cannot be observed (these variables are called *latent variables*) [5]. Nigam et al. [4] proposed a combination of the naive Bayes classifiers and the EM algorithm, which we also use as a base for constructing a Fisher kernel.

³ Here, ‘evaluative expressions’ are actually candidates of evaluative expressions. Non-evaluative expressions are classified as neutral.

Ignoring the unrelated factors of Eq. (1), we obtain

$$P(\mathbf{x}|c, \theta) \propto \prod_w P(w|c)^{N(w, \mathbf{x})}, \quad P(\mathbf{x}|\theta) \propto \sum_c P(c) \prod_w P(w|c)^{N(w, \mathbf{x})}. \quad (2)$$

If we regard c as a latent variable and introduce a Dirichlet distribution as the prior distribution for the parameters, the Q -function (i.e., the expected log-likelihood) of this model is defined as :

$$Q(\theta|\bar{\theta}) = \log(P(\theta)) + \sum_{\mathbf{x} \in D} \sum_c P(c|\mathbf{x}, \bar{\theta}) \log \left(P(c) \prod_w P(w|c)^{N(w, \mathbf{x})} \right), \quad (3)$$

where $P(\theta) \propto \prod_c (P(c)^{\alpha-1} \prod_w (P(w|c)^{\alpha-1}))$; a Dirichlet distribution. α is a user-given parameter and D is the set of examples used for model estimation.

Instead of the usual EM algorithm, we use the tempered EM algorithm [15], and obtain the following EM steps :

E-step:

$$P(c|\mathbf{x}, \bar{\theta}) = \frac{(P(c|\bar{\theta})P(\mathbf{x}|c, \bar{\theta}))^\beta}{\sum_c (P(c|\bar{\theta})P(\mathbf{x}|c, \bar{\theta}))^\beta}, \quad (4)$$

M-step:

$$P(c) = \frac{g(\alpha, \bar{\theta}, c)}{\sum_c g(\alpha, \bar{\theta}, c)}, \quad P(w|c) = \frac{h(\alpha, \bar{\theta}, w, c)}{\sum_w h(\alpha, \bar{\theta}, w, c)}, \quad (5)$$

where

$$g(\alpha, \bar{\theta}, c) = (\alpha - 1) + \sum_{\mathbf{x} \in D} P(c|\mathbf{x}, \bar{\theta}), \quad (6)$$

$$h(\alpha, \bar{\theta}, w, c) = (\alpha - 1) + \sum_{\mathbf{x} \in D} P(c|\mathbf{x}, \bar{\theta})N(w, \mathbf{x}). \quad (7)$$

For labeled example \mathbf{x} , Eq. (4) is not used. Instead, $P(c|\mathbf{x}, \bar{\theta})$ is set as 1.0 if c is the category of \mathbf{x} , otherwise 0.

As can be seen from Eq. (5), the larger α is, the more uniform the distribution becomes. In practice, α is treated as a user-given parameter. By decreasing hyper-parameter β , we can reduce the influence of intermediate classification results if those results are unreliable.

Too much influence by unlabeled data sometimes deteriorates the model estimation. Therefore, we introduce a new hyper-parameter λ (≥ 0.0), which acts as weight on unlabeled data [4]. In the second term on the right-hand-side of Eq. (3), unlabeled training examples in D are weighted by λ . We can reduce the influence of unlabeled data by decreasing the value of λ .

We derived new update rules from this new Q -function. The EM computation stops if the difference in values of the Q -function is smaller than a threshold.

3.3 Hyper-Parameter Prediction

Classification results depend largely on two hyper-parameters, specifically λ and β . We would like to predict good values of λ and β . The simplest methods are leave-one-out estimation or cross-validation. However, those methods require a high computational cost, especially when we use an EM-like iterative algorithm. Therefore, we propose an efficient quasi-leave-one-out estimation method.

Our method evaluates the accuracy for classifying labeled training examples. For each training example, we add minimal modification to the estimated parameters (excluding hyper-parameters) so that we can obtain new parameters $P_k(c)$ and $P_k(w|c)$ that are estimated without using the example. Formally we use the following parameters for training example \mathbf{x}_k :

$$P_k(c) = \frac{g(\alpha, \bar{\theta}, c) - P(c|\mathbf{x}_k, \bar{\theta})}{\sum_c (g(\alpha, \bar{\theta}, c) - P(c|\mathbf{x}_k, \bar{\theta}))}, \quad (8)$$

$$P_k(w|c) = \frac{h(\alpha, \bar{\theta}, w, c) - P(c|\mathbf{x}_k, \bar{\theta})N(w, \mathbf{x}_k)}{\sum_w (h(\alpha, \bar{\theta}, w, c) - P(c|\mathbf{x}_k, \bar{\theta})N(w, \mathbf{x}_k))}. \quad (9)$$

Thus, by preserving the values of functions $g(\alpha, \bar{\theta}, c)$ and $h(\alpha, \bar{\theta}, w, c)$, we can efficiently compute the modified parameters for each labeled training example. Henceforth, we calculate the quasi-leave-one-out accuracy. We select the hyper-parameters that yield the best quasi-leave-one-out accuracy. Please notice that all the labeled training examples are used in EM iterations and therefore this procedure is not an actual leave-one-out, but a quasi-leave-one-out.

3.4 Fisher Kernel (Fisher Score)

The Fisher kernel [16] is a similarity function, which is actually the dot-product of two Fisher scores. The Fisher score of an example is obtained by partially differentiating the log-likelihood of the example with respect to parameters. The Fisher score indicates approximately how the probability model will change if the example is added to the training data that is used in the estimation of the model. That means, the Fisher kernel between two samples will be large, if the influences of the two samples are similar and large. Takamura and Okumura reported that the Fisher kernel based on a probability model estimated by the semi-supervised EM algorithm works well in text categorization [17]. One good thing about the combination of the Fisher kernel and the EM algorithm is that high-performance kernel classifiers such as SVMs can be used in a somewhat semi-supervised way. We constructed the Fisher kernel on the basis of the above EM-estimated model described above as proposed by Takamura and Okumura [17]. Please refer to their paper for a detailed explanation.

4 Data Preparation and Features

4.1 Data Preparation

As the data for the experiments, we use real Weblog (blog) data collected by the system called blogWatcher [18]. From the blog data, we obtained candidate

evaluative expressions and contextual information in the sentences to which the expressions belong, by segmenting HTML documents into sentences and applying a Japanese syntactic analyzer to the sentences to yield their syntactic structures. Hereafter, we call a pair of a candidate expression and its contextual information an example. The reason why we adopted blogs as our data source is that they contain more evaluative expressions, and they are easier to collect, than the newspaper corpora usually used in NLP research. We used as the Japanese syntactic analyzer Cabocha⁴. Sentence boundaries were detected in a heuristic way.

Then, from the sentences, candidate evaluative expressions, that is, tuples of subjects, their attributes, and evaluative words, are extracted. We extract candidate expressions only in cases where evaluative words are adjectives. For each adjective, we try to find the nouns for a subject and an attribute. If the nouns that modify the adjective in the syntactic structure satisfy some restrictions⁵, they are extracted as the nouns for the subject and the attribute. The actual phenomena of evaluations in text are more complicated than these tuples, as was discussed by Wiebe [20]. However, we believe that this tuple-based definition of evaluative expressions will give a good approximation of the actual phenomena.

By randomly sampling 200 expressions, we evaluated our method's effectiveness for extracting candidate expressions, and found that it yielded an accuracy of 64%. Therefore, we consider that some percentages of the errors in the experiments was caused by the naiveness of our method of extracting candidate expressions.

4.2 Contextual Information Used for Classification

In Sec. 3, we explained that we adopted the naive Bayes classifier. In this subsection, we describe various types of contextual information that are used as features in the classifier. Contextual information can be extracted from the sentence to which the corresponding candidate evaluative expression belongs.

We assume that evaluative expressions are accompanied by various kinds of information that are useful for deciding their polarities. For example, if we already know that 'good' is a positive expression, from a sentence 'Good, since the storage capacity of the laptop is high', we can determine that 'capacity is high' is a positive expression. We can conjecture that a causal conjunction tends to connect expressions in the same polarity. Using the knowledge, if there is a type of sentence 'A, since B', we can determine B's polarity from A's, and vice versa.

Thus, in this work we take into account the following contextual information for an candidate evaluative expression:

1. Candidate evaluative expression itself
2. Exclamation words detected by a part-of-speech tagger
3. Emoticons in the sentence and their emotional categories

⁴ It is available at <http://chasen.org/~taku/software/cabocha>.

⁵ Roughly, the subjects should be concrete nouns, and the attributes should be abstract nouns in our thesaurus [19].

4. Words that modify the words in the tuples (candidate expressions)
5. Word that is modified by the candidate evaluative word
6. Words that are in the same ‘bunsetsu’ as the candidate evaluative word⁶

Emoticons can be considered as useful, since smileys tend to cooccur with positive expressions and sad faces tend to cooccur with negative expressions. Emoticons are automatically extracted from a sentence and classified into the six categories (happy, sad, angry, surprised, acting, and forced smile), using the method discussed in the work of Tanaka et al.’s [21].

A negation word ‘not’⁷ reverses the polarity of an evaluative word just before it. Therefore, taking into account this characteristic, if a candidate expression is followed by the negation word, the combination of the expression and the negation word is treated as a feature. Specifically, ‘not bad’ is treated as ‘bad’ + odd number of negations, ‘not not bad’ is treated as ‘bad’ + even number of negations, respectively. This definition of the scope of negation words should be discussed further in future work. Parsing results will provide us with good clues for that purpose. We will also have to collect other negation words, though we just use ‘not’ in this work.

Similarly, if a candidate expression modifies or is modified by any evaluative expression with a contrastive or adversative conjunction, the polarities of those expressions are poles apart. Therefore, in these cases, the feature ‘reverse’ is added to the contextual information.

Consider, for example, a sentence belonging to negative : ‘Phew, the noise of this HDD is annoyingly high :-()’. In the sentence, we can find a tuple of subject ‘HDD’, attribute ‘noise’, and evaluative word ‘high’. For the tuple, we can extract the following contextual information as features: the tuple itself, an exclamation ‘phew’, a modifying word ‘annoyingly’ and an emoticon ‘:-()’.

4.3 Statistics of the Data

As mentioned in Sec. 4.1, since text data on the web is noisy and our preprocessing module that uses publicly available Japanese morphological and syntactic analyzers sometimes makes errors, the data for our experiments is rather noisy. Therefore, we use the following heuristics to filter the examples that may be considered to contain errors :

- No contextual information can be obtained,
- neither subject nor attribute are extracted,
- the distance between the evaluative word and the subject and/or the attribute is more than 16 bytes⁸.

Furthermore, since the features that seldom appear are considered to be ineffective, we only used those features that appeared more than twice. Approximately

⁶ A ‘bunsetsu’ is a unit in Japanese that consists of a content word (noun, verb, adjective) and some closed words (postposition, auxiliary verb).

⁷ In the experiment, a Japanese negation word ‘nai’ is regarded as a negation word instead of ‘not’, since our dataset is in Japanese.

⁸ Examples with large distances often contain errors.

2.6 million examples were extracted from a blog collection. We obtained 35,765 examples after the filtering. Although many examples were filtered out, if a good syntactic parser trained for rather noisy text such as web documents becomes available in the near future, we would be able to use more examples.

Then, we manually labeled a subset of the examples, to use them as either training data or as test data for the evaluation. The subset were labeled as belonging to one of the following classes: neutral (non-evaluative), positive evaluation, and negative evaluation. We labeled 1,061 examples, and the proportion of the labels is as follows: neutral (69; 6.5%), positive (504; 47.5%), and negative (488; 46.0%). To check the reliability of the annotation, we compared the annotation results of two annotators. The rate of inter-annotator agreement was 91.5%.

5 Experiments

We use the 1061 labeled examples for evaluation. The number of unlabeled training examples was 34704.

As an evaluation measure, we used *accuracy*, which is defined as the number of the correctly classified examples divided by the total number of the examples. The baseline accuracy was 47.5%, which is the ratio of the examples belonging to the positive evaluation class in the 1061 labeled examples.

We conducted experiments for different values of hyper-parameters : 0.0005 to 1.0 for λ and 0.001 to 1.0 for β . We used the hyper-parameter prediction method introduced in Sec. 3.3. The user-given parameter α for the naive Bayes classifiers was fixed to 2.0. As for SVM classification, we conducted experiments with several different values of the soft-margin parameter C , and selected the value that produced the best accuracy.

5.1 Results

Comparison of methods

Table 1 shows the accuracy values for the various methods. Incorporation of unlabeled data improves classification accuracy of the naive Bayes classifiers for this task. The Fisher kernel on the probability model estimated with a semi-supervised method, which is referred to as SVM+NaiveBayes+EM in the table, also improves SVM performance.

If the actual best values of β and λ are selected for each fold of cross-validation, the accuracy reaches 79.5%. Although this is unfair, brushing up hyper-parameter selection would further improve the method's accuracy.

Though the actual best values were not selected, the proposed method for hyper-parameter prediction also worked well.

Influence of labeled training data size

The accuracy values for different sizes of labeled training data are presented in Figure 1. The values were obtained through 10, 5, 3, 2-fold cross validations and inverted cross-validations that match up the training/test dataset sizes. In

Table 1. Accuracy for each method; “NB” corresponds to the naive Bayes classifier, “NB+EM” corresponds to the naive Bayes classifier enhanced with EM, and “SVM+NB+EM” corresponds to the SVM that uses the Fisher kernel extracted from NB+EM model

Method	Accuracy(%)
Baseline	47.5
NB	76.0
SVM	76.6
NB+EM	77.1
SVM+NB+EM	77.9

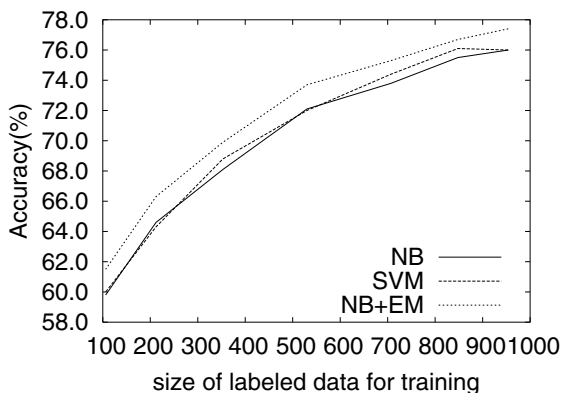


Fig. 1. Accuracy vs Labeled Training Data Size; “NB” corresponds to the result of classification with the naive Bayes classifier, and “NB+EM” corresponds to the result of classification with the naive Bayes classifier enhanced with EM

inverted cross-validations, the smaller of the two split datasets was used for training. The best values for β and λ were used in this experiment.

This result shows that our semi-supervised EM algorithm boosted accuracy, regardless of the size of labeled training data. The difference in the accuracy before and after the EM computation was statistically significant in the sign-test with a 5% significance-level.

Influence of unlabeled training data size

The accuracy values for different sizes of unlabeled training data are given in Figure 2. This result shows that even a relatively small sized unlabeled dataset (e.g., 5000 examples) improved the accuracy value. As this curve shows, although only approximately 35,000 unlabeled examples are currently available, we can expect better accuracy for a larger unlabeled training dataset.

Many of the classification errors were caused by errors in dependency analysis and failure to detect subjects and attributes. The existing dependency parsers are designed for well-formatted text such as newspaper articles, not for Web

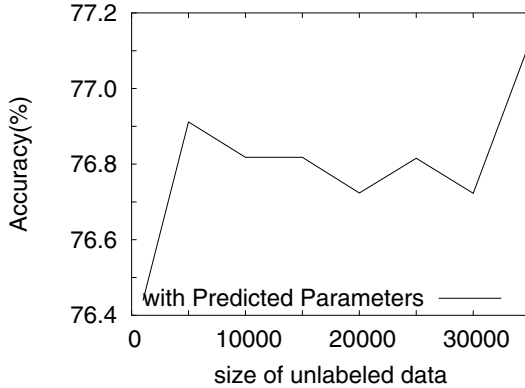


Fig. 2. Accuracy of NB+EM vs Unlabeled Data Size; note that the range for the y-axis is different from the previous figure

documents. Improvement in parsing technology would solve this problem. We currently rely on some heuristics to detect subjects and attributes. We require more sophisticated detection methods to avoid such errors.

Some errors were related to limitations of the proposed method. For example, our method still has difficulty dealing with idiomatic expressions or ambiguous words. We need to extend the method so that combinations of multiple features (words) are taken into consideration.

In order to qualitatively analyze the features, we extracted the 100 features that had the largest $P(w|positive)$ before and after EM computation. Compared with the top 100 features before EM, more contextual features were found after EM, such as, exclamations, the facemark (emoticon) category *happy*, a negation word + ‘but’, therefore + ‘interesting’, therefore + ‘comfortable’.

6 Conclusions

We proposed to use a semi-supervised method for automatically classifying evaluative expressions as positive, negative, or neutral. We adopted the EM algorithm and the naive Bayes classifiers together with a method for predicting hyper-parameters. We also used the Fisher kernel on the model that we estimated with the semi-supervised method. We empirically demonstrated that the semi-supervised method works well for classifying the evaluative expressions.

References

1. Pang, B., Lee, L., Vaithyanathan, S.: Thumbs up? sentiment classification using machine learning techniques. EMNLP’02. (2002) 76–86
2. Dave, K., Lawrence, S., Pennock, D.M.: Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. 12th WWW Conference. (2003) 519–528

3. Cristianini, N., and Shawe-Taylor, J.: *An Introduction to Support Vector Machines (and other kernel-based learning methods)*, Cambridge University Press, (2000)
4. Nigam, K., McCallum, A., Thrun, S., Mitchell, T.: Text classification from labeled and unlabeled documents using EM. *Machine Learning* **39** (2000) 103–134
5. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society Series B* **39** (1977) 1–38
6. Hatzivassiloglou, V., McKeown, K.R.: Predicting the semantic orientation of adjectives. *35th ACL*. (1997) 174–181
7. Turney, P.D.: Thumbs up? thumbs down? semantic orientation applied to unsupervised classification of reviews. *40th ACL*. (2002) 417–424
8. Kamps, J., Marx, M., Mokken, R.J., de Rijke, M.: Using wordnet to measure semantic orientations of adjectives. *4th LREC*. (2004) 1115–1118
9. Kim, S.M., Hovy, E.: Determining the sentiment of opinions. *20th COLING*. (2004) 1367–1373
10. Kudo, T., Matsumoto, Y.: A boosting algorithm for classification of semi-structured text. *EMNLP'04*. (2004) 301–308
11. Wilson, T., Wiebe, J., Hwa, R.: Just how mad are you? finding strong and weak opinion clauses. *19th AAAI*. (2004)
12. Pang, B., Lee, L.: A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. *42th ACL*. (2004) 271–278
13. Riloff, E., Wiebe, J.: Learning extraction patterns for subjective expressions. *EMNLP'03*. (2003) 105–112
14. Blum, A., Mitchell, T.: Combining labeled and unlabeled data with co-training. *Proceedings of the Workshop on Computational Learning Theory*. (1998) 92–100
15. Hofmann, T., Puzicha, J.: Statistical models for co-occurrence data. Technical Report AIM-1625, Artificial Intelligence Laboratory, Massachusetts Institute of Technology (1998)
16. Jaakkola, T., Haussler, D.: Exploiting generative models in discriminative classifiers. *NIPS 11*. (1998) 487–493
17. Takamura, H., Okumura, M.: A comparative study on the use of labeled and unlabeled data for large margin classifiers. *1st IJCNLP2004*. (2004) 620–625
18. Nanno, T., Fujiki, T., Suzuki, Y., Okumura, M.: Automatically collecting, monitoring, and mining japanese weblogs. *13th WWW Conference*. (2004) 320–321
19. Ikehara, S., Miyazaki, M., Shirai, S., Yokoo, A., Nakaiwa, H., Ogura, K., Ooyama, Y., Hayashi, Y.: *Goi-Taikei – A Japanese Lexicon*. Iwanami Shoten (1997)
20. Wiebe, J.: Instructions for annotating opinions in newspaper articles. Technical report, University of Pittsburgh Technical Report (TR-02-101) (2002)
21. Tanaka, Y., Takamura, H., Okumura, M.: Extraction and classification of facemarks with kernel methods. *IUI 2005*. (2005) 28–34

A New Algorithm for Fast Discovery of Maximal Sequential Patterns in a Document Collection

René Arnulfo García-Hernández, José Francisco Martínez-Trinidad,
and Jesús Ariel Carrasco-Ochoa

National Institute of Astrophysics, Optics and Electronics (INAOE),
Puebla, México
{rearnulfo, fmartine, ariel}@inaoep.mx

Abstract. Sequential pattern mining is an important tool for solving many data mining tasks and it has broad applications. However, only few efforts have been made to extract this kind of patterns in a textual database. Due to its broad applications in text mining problems, finding these textual patterns is important because they can be extracted from text independently of the language. Also, they are human readable patterns or descriptors of the text, which do not lose the sequential order of the words in the document. But the problem of discovering sequential patterns in a database of documents presents special characteristics which make it intractable for most of the apriori-like candidate-generation-and-test approaches. Recent studies indicate that the pattern-growth methodology could speed up the sequential pattern mining. In this paper we propose a pattern-growth based algorithm (DIMASP) to discover all the maximal sequential patterns in a document database. Furthermore, DIMASP is incremental and independent of the support threshold. Finally, we compare the performance of DIMASP against GSP, DELISP, GenPrefixSpan and cSPADE algorithms.

1 Introduction

The *Knowledge Discovery in Databases* (KDD) is defined by Fayyad [1] as “the non-trivial process of identifying valid, novel, potentially useful and ultimately understandable patterns in data”. The key step in the knowledge discovery process is the data mining step, which following Fayyad: “consisting of applying data analysis and discovery algorithms that, under acceptable computational efficiency limitations, produce a particular enumeration of patterns over the data”. This definition has been extended to *Text Mining* like: “consisting of applying text analysis and discovery algorithms that, under acceptable computational efficiency limitations, produce a particular enumeration of patterns over the text”. So, text mining is the process that deals with the extraction of patterns from textual data. This definition is used by Feldman [2] to define *Knowledge Discovery in Texts* (KDT). In both KDD and KDT tasks, special attention is required in the performance of the algorithms because they are applied on a large amount of information. In particular the KDT process needs to define simple structures that can be extracted from text documents automatically and in a reasonable time. These structures must be rich enough to allow interesting KD operations [2] having in mind that in some cases the document database is updated.

Sequential pattern mining has the goal of finding all the subsequences that are contained at least β times in a collection of sequences, where β is a user-specified support threshold. This discovered set of frequent sequences contains the *maximal frequent sequences* (MFSs), which are not a subsequence of any other frequent sequence. That is, the MFSs are a compact representation of the whole set of frequent sequences. So, the sequential pattern mining approaches play an important role in data mining tasks because these approaches allow us to identify valid, novel, potentially useful and ultimately understandable patterns in databases. In this case, we are interested in the extraction of this kind of patterns from textual databases. Due to its broad applications in text mining problems, finding textual patterns is important because they can be extracted from documents independently of the language without losing their sequential nature.

Most of the sequential pattern mining approaches have been developed for vertical databases, this is, databases with short sequences but with a large amount of sequences. A document database can be considered as horizontal because it could have long sequences. Therefore, sequential pattern mining approaches are not efficient for mining a document database. In order to guarantee human-legible and meaningful patterns we are interested in finding contiguous MFSs. Also, these special patterns could be of interest in the analysis of DNA sequences [10], data compression and web usage logs [9].

Furthermore, most of the sequential pattern mining approaches assume a short alphabet; that is, the set of different items in the database. So, the characteristics of textual patterns make the problem intractable for most of the apriori-like candidate-generation-and-test approaches. For example, if the longest MFS has a length of 100 items then GSP[3] will generate $\sum_{i=1}^{100} \binom{100}{i} \approx 10^{30}$ candidate sequences where each one must be tested over the DB in order to verify its frequency. This is the cost of candidate generation, no matter what implementation technique would be applied. For the candidate generation step, GSP generates candidate sequences of size $k+1$ by joining two frequent sequences of size k when the prefix $k-1$ of one sequence is equal to the suffix $k-1$ of other one. Then a candidate sequence is pruned if the sequence is non-frequent. Even though, GSP reduces the number of candidate sequences, it still being inefficient for mining long sequences.

Recent studies indicate that the pattern-growth methods could speed up the sequential pattern mining [4,5,6,10,11] when there are long sequences. According to empirical performance evaluations the pattern-growth methods like PrefixSpan[4], GenPrefixSpan[5] and DELISP[6] outperform GSP specially when the database contains long sequences. The basic idea is to avoid the cost of candidate generation step and to focus the search on sub-databases generating projected databases. An α -projected database is the set of subsequences in the database that are suffixes of the sequences with prefix α . In each step, the algorithm looks for frequent sequences with prefix α in the corresponding projected database. In this sense, pattern-growth methods try to find the sequential patterns more directly, growing frequent sequences, beginning with sequences of size one. Even though, these methods are faster than apriori-like methods, some of them were designed to find all the frequent sequences and not to get only the MFSs. Furthermore, none of them is incremental.

Other work related to searching of repeated substrings in a set of strings is the longest common substring (LCS) problem. From this point of view, the documents

can be taken as strings of words. The objective of LCS is to find the longest substring that is repeated in all the set of strings. The LCS problem can be solved using suffix trees, but the LCS problem looks for only one substring (the longest) which appears in all the documents. However, we need to find all the maximal substrings that appear at least in β documents.

In this paper we propose a pattern-growth based algorithm (DIMASP) to **Discover** all the **Maximal Sequential Patterns** in a document database. First, DIMASP builds a novel data structure from the document database which is relatively easy to extract. Once DIMASP has built the data structure, it can discover all the MFSs according to the threshold specified by the user. If a new threshold is specified, DIMASP avoids rebuilding the data structure for mining with this new threshold. In addition, when the document database is increased, DIMASP updates the last discovered MFSs by processing only the new documents. DIMASP assumes that the data structure can fit in the main memory.

In section 2, the problem definition is given. Section 3 describes our algorithm. In Section 4, the experiments are presented. Finally in section 5 the conclusions and future work are given.

2 Problem Definition

A *sequence* S , denoted by $\langle s_1, s_2, \dots, s_k \rangle$, is an ordered list of k elements called *items*. The number of elements in a sequence S is called the *length* of the sequence denoted by $|S|$. A k -*sequence* denotes a sequence of length k . Let $P = \langle p_1 p_2 \dots p_n \rangle$ and $S = \langle s_1 s_2 \dots s_m \rangle$ be sequences, P is a *subsequence* of S , denoted $P \subseteq S$, if there exists an integer $i \geq 1$, such that $p_1 = s_i, p_2 = s_{i+1}, p_3 = s_{i+2}, \dots, p_n = s_{i+(n-1)}$. We can consider a *document* W as a sequence of words, denoted as $\langle w_1, w_2, \dots, w_n \rangle$.

The *frequency* of a sequence S , denoted by S_f or $\langle s_1, s_2, \dots, s_n \rangle_f$, is the number of documents where S is a subsequence. A sequence S is β -*frequent* if $S_f \geq \beta$, a β -frequent sequence is also called a *sequential pattern*. A sequential pattern S is *maximal* if S is not a subsequence of any other sequential pattern.

In this paper, we are interested in the problem of discovering all the maximal sequential patterns in a document database.

3 DIMASP: A New Algorithm for Fast Discovery of All Maximal Sequential Patterns

The basic idea of DIMASP consists in finding all the sequential patterns in a data structure, built from the document database (DDB), which stores all the distinct pairs of contiguous words that appear in the documents, without losing their sequential order. Given a threshold β specified by the user, DIMASP reviews if a pair is β -frequent. In this case, DIMASP grows the sequence in order to determine all the possible maximal sequential patterns containing such pair as prefix. A possible maximal sequential pattern (PMSP) will be a maximal sequential pattern (MSP) if it is not a subsequence of any previous MSP. This implies that all MSPs which are

subsequence of the new PMSP are deleted. The proposed algorithm is composed of three steps described as follows:

In the first step, DIMASP assigns an integer number, as an identifier, for each different word (item) in the DDB. Also, the frequency for each identifier is stored *i.e.* the number of documents where it appears. These identifiers are used in the algorithm instead of the words in the DDB like in the example of the table 1.

Table 1. An example of a document database and its identifier representation

D_j	Document database	Integer identifiers
1	From George Washington to George W. Bush are 43 Presidents	<1,2,3,4,2,5,6,7,8,9>
2	Washington is the capital of the United States	<3,10,11,12,13,11,14,15>
3	George Washington was the first President of the United States	<2,3,16,11,17,18,13,11,14,15>
4	<i>the President of the United States is George W. Bush</i>	<11,18,13,11,14,15,10,2,5,6>

Step 2: Algorithm to construct the data structure from the DDB

Input: A document database (DDB) **Output:** The Array

For all the documents $D_j \in DDB$ **do**

$Array \leftarrow$ Add a document (D_j) to the array

end-for

Step 2.1: Algorithm to add a document

Input: A document D_j **Output:** The Array

For all the pairs $\langle w_i, w_{i+1} \rangle \in D_j$ **do**

$\delta_i \leftarrow$ Create a new **Pair** δ

$\delta_i.Id \leftarrow J$ //Assign the document identifier to the node δ

$index \leftarrow Array[\langle w_i, w_{i+1} \rangle]$ //Get the index of the cell where is $\langle w_i, w_{i+1} \rangle$

$\delta_i.index \leftarrow index$ //Assign the index to the node δ

$\alpha \leftarrow$ Get the first node of the list Δ

If $\delta_i.Id \neq \alpha.Id$ **then** the document identifier is new to the list Δ

$Increment C_f$ //increment the frequency

$\delta_i.NextDoc \leftarrow \alpha$ //link the node α at the beginning of list Δ

List $\Delta \leftarrow$ Add δ_i as the first node //link it at the beginning of list Δ

$\delta_{i-1}.NextNode \leftarrow \delta_i$ //link the pair to do not lose the sequential order

end-for

Fig. 1. Algorithms for steps 2 and 2.1 where is built the data structure for documents

In the second step, DIMASP builds a data structure from the DDB storing all the pairs of contiguous words $\langle w_i, w_{i+1} \rangle$ that appear in a document and some additional information to preserve the sequential order. The data structure is a special *array* which contains in each cell a pair of words $C = \langle w_i, w_{i+1} \rangle$, the *frequency* of the pair (C_f), a Boolean *mark* and a list Δ of nodes δ where a *node* δ (see Fig. 2) stores a document identifier ($\delta.Id$), an *index* ($\delta.Index$) of the cell where the pair appears in the array, a link ($\delta.NextDoc$) to maintain the list Δ and a link ($\delta.NextNode$) to preserve the sequential order of the pairs with respect to the document. Therefore, the number of

different documents presented in the list Δ is C_f . This step works as follows: for each pair of words $\langle w_i, w_{i+1} \rangle$ in the document D_j , if $\langle w_i, w_{i+1} \rangle$ is not in the array add it, and get its *index*. In the position *index* of the array, add a node δ at the beginning of the list Δ . The added node δ has j as $\delta.Id$, *index* as $\delta.index$, $\delta.NextDoc$ is linked to the first node of the list Δ and $\delta.NextNode$ is linked to the next node δ corresponding to $\langle w_{i+1}, w_{i+2} \rangle$ of the document D_j . If the document identifier ($\delta.Id$) is new in the list Δ , then the frequency of the cell (C_f) is increased. In Fig. 2 the data structure built with the step 2 algorithm the document database of table 1 is shown.

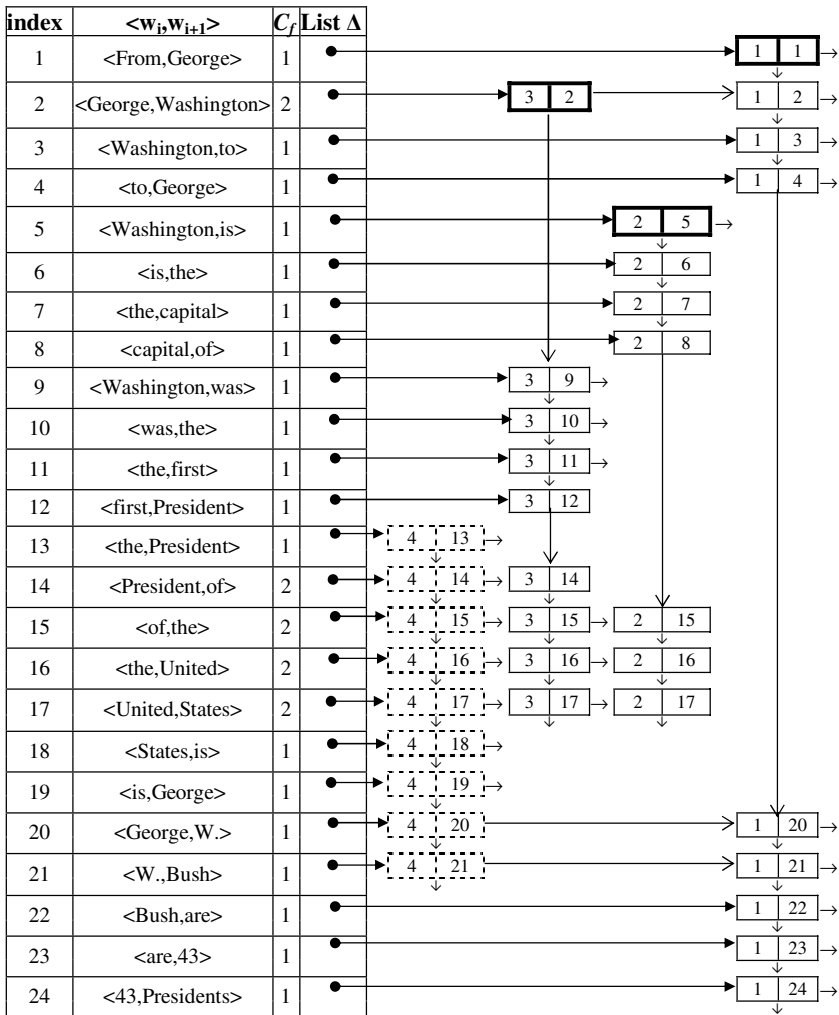


Fig. 2. Data structure built for the document database of the table 1. Note, the dotted nodes δ corresponding to D_4 will be added when D_4 would be included, of course, it will be necessary to update the frequencies C_f of the array.

Step 3: Algorithm to find all MSPs**Input:** Structure from step 2 and β threshold **Output:** MSP set**For all the documents** $D_{j \dots (\beta-1)} \in DDB$ **do**MSP set \leftarrow **Find all MSPs w.r.t. the document** (D_j)**Step 3.1: Algorithm to find all MSPs with respect to the document D_j** **Input:** A D_j from the data structure and a β threshold **Output:** The MSP set w.r.t. to D_j **For all the nodes** $\delta_{i=1 \dots n} \in D_j$ i.e. $\langle w_i, w_{i+1} \rangle \in D_j$ **do****If** Array [δ_i .index].frequency $\geq \beta$ **then** //if the pair has a frequency $\geq \beta$ PMSP \leftarrow Array [δ_i .index]. $\langle w_i, w_{i+1} \rangle$ //the initial PMSP is the pair $\langle w_i, w_{i+1} \rangle$ Δ' \leftarrow Copy the rest of the list of Δ beginning from δ_i .NextDoc Δ'_f \leftarrow Number of different documents in Δ' δ'_i \leftarrow δ_i **While** $\Delta'_f \geq \beta$ **do** the growth the PMSP Δ'' \leftarrow Array [δ'_{i+1} .index].list Δ //Denotes to Array [δ'_{i+1} .index].list Δ as Δ'' Δ' \leftarrow $\Delta' \cap \Delta''$ i.e. $\{\alpha \in \Delta' \mid (\alpha.index = \delta'_{i+1}) \wedge (\delta'_i.NextNode = \alpha)\}$ Δ'_f \leftarrow Number of different documents in Δ' **If** $\Delta'_f \geq \beta$ **then** to grow the PMSPArray [δ'_{i+1} .index].mark \leftarrow "used"PMSP \leftarrow PMSP + Array [δ'_{i+1} .index]. $\langle w_{i+1} \rangle$ δ'_i \leftarrow δ'_{i+1} i.e. $\delta'_i.NextNode$

end-while

If |PMSP| ≥ 3 **then** add the PMSP to the MSP setMSP set \leftarrow add a k-PMSP to the MSP set //step 3.1.1

end-for

For all the cells $C \in$ Array **do** the addition of the 2-MSPs**If** $C_f \geq \beta$ and $C.mark =$ "not used" **then** add it as 2-MSP2-MSP set \leftarrow add $C . \langle w_i, w_{i+1} \rangle$ **Fig. 3.** Algorithm to find all the MSPs using the data structure of step 2 and a threshold β

To prove that our algorithm finds all the MSPs we introduce the following proposition.

Proposition 1: DIMASP discovers all the maximal sequential patterns of a DDB.

Proof. To proof that DIMASP finds all the MSPs, suppose there is a k -MSP in the document database. Therefore, if there is a k -MSP then it is contained in at least β documents in the database. For $k \geq 2$ the k -MSP is $\langle w_1, w_2, w_3, \dots, w_k \rangle$ which can be separated in its frequent pairs $\langle w_1, w_2 \rangle + \langle w_2, w_3 \rangle + \dots + \langle w_{k-1}, w_k \rangle$. From step 2, we know that the pair $\langle w_i, w_{i+1} \rangle$ and the list Δ in a cell of the array are stored, denoted by $\Delta(\langle w_i, w_{i+1} \rangle)$, containing the registers of all documents that have this pair, without losing their sequential order. Therefore, as it was made in steps 3 and 3.1, we can index and get $\Delta(\langle w_1, w_2 \rangle)$, $\Delta(\langle w_2, w_3 \rangle)$, ..., $\Delta(\langle w_{k-1}, w_k \rangle)$. Also, from the array, it is clear

that the frequencies of such subsequences are $\geq \beta$. Now we have to proof that they form the k -MSP. Since the pairs do not lose their sequential order, we can establish that $\Delta(\langle w_1, w_2 \rangle) \cap \Delta(\langle w_1, w_3 \rangle) = \Delta(\langle w_1, w_2, w_3 \rangle)$ and $\|\Delta(\langle w_1, w_2, w_3 \rangle)\| \geq \beta$. Therefore, $\bigcap_{i=1}^k \Delta(\langle w_i, w_{i+1} \rangle) = \Delta(\langle w_1, w_2, w_3, \dots, w_k \rangle)$ which is actually $\Delta(k\text{-MSP})$. And it can not grow because $\|\Delta(\langle w_1, w_2, w_3, \dots, w_k \rangle) \cap \langle w_k, w_{k+1} \rangle\| < \beta$ since k -MSP is maximal. If $k=1$ then DIMASP includes this 1-MSP because in step 1 DIMASP includes all the frequent items which are not included in any other longer MSP. ■

In order to be efficient it is needed to reduce the number of comparisons when a PMSP is added to the MSP set. For such reason, a k -MSP is stored according to its length k , it means, there is a k -MSP set for each k . Also, for speed up the comparison of PMSPs, binary searches using the sum of the identifiers in a PMSP are performed. In this way, before adding a k -PMSP as a k -MSP, the k -PMSP must not be in the k -MSP set and must not be subsequence of any longer k -MSP. Two sequences might be equal only if they have the same sum. A sequence A could be a subsequence of another sequence B only if the sum of A is lesser than the sum of B . When a PMSP is added, all their subsequences are eliminated.

Step 3.1.1: Algorithm to add a PMSP to the MSP set

```

Input: A  $k$ -PMSP, MSP set           Output: MSP set
If ( $k$ -PMSP  $\in$   $k$ -MSP set) or
If ( $k$ -PMSP is subsequence of some longer  $k$ -MSP) then do not add anything
    return MSP set
Else
     $k$ -MSP set  $\leftarrow$  add  $k$ -PMSP //add as a MSP
    {del S  $\in$  MSP set | S  $\subseteq$   $k$ -PMSP }
    
```

Fig. 4. Algorithm to add a PMSP to the MSP set

Since the *array* has only the distinct pair of words $\langle w_i, w_{i+1} \rangle$ the performance for comparing two sequences can be improved if instead of adding the identifiers w_i and w_{i+1} to PMSP only the *index* of the array where the pair appears is added. In this way, a sequence A is a subsequence of B only if the last and the odds items of A are contained in B . For example, if the array structure of Fig. 2 is used with the sequence $A = \langle \text{the, President, of, United, States} \rangle$ and $B = \langle \text{the, President, of, United, States, is} \rangle$ then the sequences $A = \langle 13, 14, 15, 16, 17 \rangle$ and $B = \langle 13, 14, 15, 16, 17, 18 \rangle$. Therefore it is enough, checking that the last and odds items of $A = \langle 13, \square, 15, \blacklozenge, 17 \rangle$ are contained in $B = \langle 13, \square, 15, \blacklozenge, 17, \bullet \rangle$, to guarantee that $A \subseteq B$ because only the items 14 and 16 can fit in \square and \blacklozenge , respectively.

With the objective of do not repeat all the work to discover all the MSPs when one or a set of new documents are added to the database, DIMASP only preprocesses the part corresponding to these new documents. After the identifiers of these new documents were defined in step 1, DIMASP would only use the step 2.1 to add them to the array. Then, the step 3.1 is applied on the new documents and on the old MSP set, to discover the new MSP set. This strategy works only for the same β , however with a different β only the discovery step (step 3) must be applied, without rebuilding the data structure. For example, Fig. 2 shows with dotted line the new part of the data structure when D_4 of table 1 is added as a new document. Then, using $\beta=2$ for the al-

gorithm of the step 3, the PMSPs $\langle \textit{President,of,the,United,States} \rangle$ and $\langle \textit{George,W.,Bush} \rangle$ are discovered. The first PMSP eliminates the previous discovered maximal sequential pattern $\langle \textit{of,the,United,States} \rangle$ because it is not maximal.

4 Experiments

The next experiments were accomplished using the well-known reuters-21578 document collection [7]. After a prune of 400 stop-words, this collection has 21578 documents with around 38,565 different words from 1.36 million words used in the whole collection. The average length of the documents is 63 words. In all the experiments the first 5000, 10000, 15000 and 20000 documents were used. Excepting for GSP, the original programs provided by the authors were used. In Fig. 5a the performance comparison of DIMASP, cSPADE[8], GenPrefixSpan, DELISP and GSP algorithms with $\beta=15$ is shown. Fig. 5b shows the same comparison of Fig. 5a but the worst algorithm (GSP) is eliminated, here it is possible to see that DELISP is not as good as it seems to be in Fig. 5a. In this case GenPrefixSpan had memory problems, so it was only possible to test with the first 5000 and 10000 documents. Fig. 5c compares DIMASP against the fastest algorithm cSPADE, the time of the steps 2 and 3 of DIMASP are also compared. Fig. 5d draws a linear scalability of DIMASP whit respect to β . An additional experiment with the lowest $\beta=2$ was performed, in this experiment DIMASP found a MSP of length 398, Fig. 5e shows the results. To evaluate the incremental scalability of DIMASP, 4000, 9000 14000 and 19000 documents were processed, and 1000 documents were added in each experiment. Fig. 5f shows the results and compares them against cSPADE which needs to recompute all the MSPs. Fig. 5g shows the distribution of the MSPs according to their length. Finally, Fig. 5h shows the number of MSPs when $\beta = 1\%$ of the documents in the collection was used.

5 Conclusions

In this paper, DIMASP a pattern-growth memory-based algorithm to discover all the maximal sequential patterns MSPs in a document database was proposed. To do that, DIMASP builds a data structure for the document database which speeds up the mining of MSPs. Our algorithm allows working with different support thresholds without rebuilding the data structure. Moreover, DIMASP is incremental with respect to document addition. According to the empirical evaluations, DIMASP outperforms GSP, DELISP GenPrefixSpan and cSPADE algorithms in discovering all MSPs in a document database and has a good scalability regarding to β . One of the reasons for which DIMASP is more efficient is because the algorithm begins to discover MSPs longer than 2 and the 1-MSPs and 2-MSPs, which are the majority of the MSPs, are discovered in one-pass. For example, Fig. 5g shows that $\sum_{i=1}^2 |i - \text{MSP}| > \sum_{i=3}^{14} |i - \text{MSP}|$. For our experiments with the whole reuters-21578 collection DIMASP used around 30 Mbytes of main memory for the data structure built in step 2 which is able to be handled by most of the computers. This shows that, even though DIMASP needs the whole data structure to fit in main memory, it might process bigger document collections.

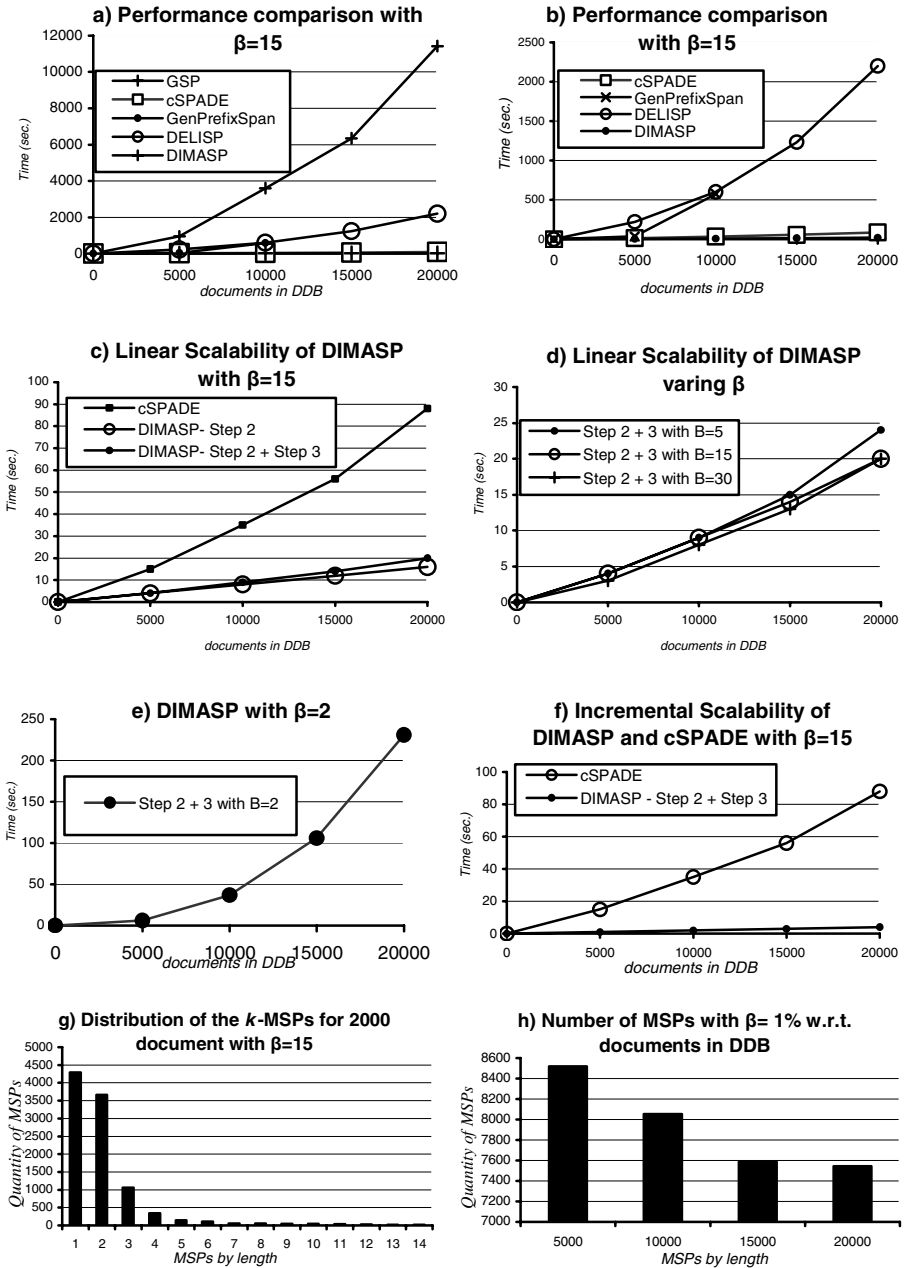


Fig. 5. Results of the performance experiments using the collection Reuters-21578

As future work we will extend the idea of DIMASP to be able of manage a gap constraint which allows a controlled separation between the items that form a sequential pattern. Also we are going to apply DIMASP on other kind of data like web logs or DNA sequences.

References

- [1] Fayyad, U., Piatetsky-Shapiro G. "Advances in Knowledge Discovery and Data mining". AAAI Press, 1996.
- [2] Feldman, R and Dagan, I. "Knowledge Discovery in Textual Databases (KDT)", *In Proceedings of the 1st International Conference on Knowledge Discovery (KDD-95)* 1995.
- [3] Srikant, R., and Agrawal, R. Mining sequential patterns: Generalizations and performance improvements. *In 5th Intl. Conf. Extending Database Discovery and Data Mining*, 1996.
- [4] Pei, J, Han, et all: "PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth" in *Proc International Conference on Data Engineering (ICDE 01)*, 2001.
- [5] Antunes, C., Oliveira A. Generalization of Pattern-growth Methods for Sequential Pattern Mining with Gap Constraints. *Third IAPR Workshop on Machine Learning and Data Mining MLDM 2003*, 2003.
- [6] Ming-Yen Lin, Suh-Yin Lee, and Sheng-Shun Wang, "DELISP: Efficient Discovery of Generalized Sequential Patterns by Delimited Pattern-Growth Technology," *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD02)*, Taipei, Taiwan, pp. 189-209, 2002.
- [7] <http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html>
- [8] Mohammed J. Zaki, Sequence Mining in Categorical Domains: Incorporating Constraints, in *9th International Conference on Information and Knowledge Management*, pp 422-429, Washington, DC, November 2000.
- [9] Amir H. Youssefi, David J. Duke, Mohammed J. Zaki, "Visual Web Mining". *13th International World Wide Web Conference* , New York, NY, 2004.
- [10] Jiawei Han and Micheline Kamber, *Data Mining: Concepts and Techniques*, Morgan Kaufmann Publishers, August 2000. c.9 &10.
- [11] Jian Pei, Jiawei Han, et. al. "Mining Sequential Patterns by Pattern-Growth: The Prefix-Span Approach", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 16, No. 10, October 2004.

A Machine Learning Based Approach for Separating Head from Body in Web-Tables*

Sung-Won Jung and Hyuk-Chul Kwon

Korean Language Processing Lab., Department of Computer Science and Engineering,
Pusan National University, Busan, Korea
{swjung, hckwon}@pusan.ac.kr

Abstract. This study aims to separate the head from the data in web-tables to extract useful information. To achieve this aim, web-tables must be converted into a machine readable form, an attribute-value pair, the relation of which is similar to that of head-body. We have separated meaningful tables and decorative tables in our previous work, because web-tables are used for the purpose of knowledge structuring as well as document design, and only meaningful tables can be used to extract information. In order to extract the semantic relations existing between language contents in a meaningful table, this study separated the head from the body in meaningful tables using machine learning. We (a) established features observing the editing habit of authors and tables themselves, and (b) established a model using machine learning algorithm, C4.5 in order to separate the head from the body. We obtained 86.2% accuracy in extracting the head from the meaningful tables.

1 Introduction

Information extraction encounters various text types. Generally, editors produce three types of text: free text, structured text, and semi-structured text. Among those, free text, composed of natural language sentences, is the most frequently found. To extract information from free text, a computer must analyze the text using natural-language-processing techniques. However, practical application of natural language understanding is still far from being achieved. On the contrary, authors make structured text for specific aims such as a database or a file. These texts contain restricted and well-formed rules. Computers can easily analyze them even though they do not contain structured information apart from that which is predefined. Semi-structured text falls between structured and free text. We can include tables and charts in this type. These texts are easier to analyze and contain more useful and dense information than free text, because of their structural features. This paper focuses on the table among the semi-structured texts, because the table is usually used in HTML documents and easily extracted from HTML documents.

A table is composed of row(s) and column(s) in the structural view; and those row(s) and column(s) can be divided into head and body in the semantic view.

* This work was supported by the Regional Research Centers Program(Research Center for Logistics Information Technology), granted by the Korean Ministry of Education & Human Resources Development.

According to these characteristics of tables, data are arranged in rows and columns and located in the head and the body. That is, the head abstracts the data. Thanks to this structural characteristic, a table intrinsically provides relational semantic information among data sets. For easy processing by computer, these tables should be converted into machine-readable form. That is, semi-structural texts should be converted into structural texts. Accordingly, the head and body of a table are converted into attribute-value pairs, in exactly the same way as in a database. Once a structural text with attribute-value pairs is constituted, we can extract, from a table, the hierarchical relations existing between words, which are our ultimate aim.

To achieve our aim, two stages of investigation are necessary: (1) preprocessing raw tables on the web (hereafter, web-tables) and (2) extracting information from the tables. The preprocessing stage includes extraction of meaningful tables. As is well known, several types of tables, whether meaningful or not, exist on the web, and information extraction is possible only from meaningful tables. Our previous work's main focus was distinguishing meaningful from meaningless tables. In order to isolate meaningful tables, we set features which interacted in defining the meaningfulness of a table; we built a separation model that utilized, with those features, a machine-learning algorithm. The aim of the present study was to separate the head from the body in web-tables in order to extract information. To achieve this aim, we used the results of our previous work, in which we established the method of determining the meaningfulness of a web-table. This method provided us with meaningful tables by preprocessing web-tables, from which we can now distinguish the head from the body. The separation of these is the prime clue for reconstituting hierarchical relations between 'words', which are candidates of attribute-value pairs.

This paper is organized as follows. Section 2 briefly summarizes our previous work, which investigated preprocessing web-tables together with several recent studies undertaken to develop information extraction from web-tables. Section 3 briefly describes our preprocessing method together with some necessary predefined concept for determining the meaningfulness of a web-table. Section 4 describes the method of extracting the head. Finally, Section 5 illustrates the experiments and offers concluding comments.

2 Related Works

Table information extraction is a sub-domain of information extraction. Having started in the late 1990s, table information extraction is a relatively fresh area and related research is scarce. A previous researcher [1] named table information extraction "table mining," which term has recently been adopted by many researchers. Research in the web table mining field can be classified into two categories: (1) domain-specific research and (2) domain-independent research. Domain-specific research is based on wrapper induction [8], which performs particular information extraction using extraction rules. Using these extraction rules, several studies [1, 2, 4] have extracted table information according to a special tabular form. Because these studies dealt only with the special tabular form, the researchers experienced difficulty in coping with the various web document formats.

Domain-independent approaches have recently been introduced into the table mining field. Wang [5] attempted to implement a general table mining system, using a machine-learning algorithm. He applied Information Retrieval (IR) strategies to his table mining using term frequency and table frequency (which correspond to document frequency in IR). This strategy can hardly cope with new tables that contain unknown words. This limitation can be overcome, if we renew the calculation of the term frequency and table frequency according to the change of application domain. Yang [7] conceives the database as being constructed with attribute-value pairs, and table mining as a reverse process of table publishing from a database. He extracts the attribute-value pairs using entity-patterns and extraction rules. Yang's method shows the same weak point as Wang's. We need to repetitively and manually update those rules and patterns with linguistic bias, which is rather far from domain independence. Both of those researchers [5, 7] neglected to provide explicit definitions of table types and to acquire a domain-independent strategy. Although they proposed table types such as *genuine table*, *non genuine table* [5], *table (data table)*, and *non-table (non-data table)* [7], none of these table types is explicitly defined.

The objective of this study was to apply table mining to general HTML documents. In order to satisfy this objective, we applied a machine-learning method, using only structural information to avoid domain-specific information.

3 Separating Meaningful Table from Decorative Table

Generally, the table is conceived as means to present certain types of data to users in a rows-and-columns formatted way. However, tables on web pages should be conceived differently from the general concepts. Our previous research [9] has defined the particularity of those web-tables and distinguished meaningful tables from decorative tables on the Internet. The following sub-sections briefly report the results of our previous work.

3.1 Definition of Table

The web-table can be defined as one of the components of an HTML document. The tags of HTML are divided into two groups: (1) tags to express display contents of HTML documents (contents-components tags); (2) tags to express the structure of HTML documents (structural-components tags). The structural-components tags demonstrate the author's intention while clarifying the contents-components. A table-like structure on web pages can be composed of various structural-components tags. But the present paper deals with web-tables constructed with a particular series of tags: `<table>`, `<tr>`, `<td>`, `<th>`, and others (hereafter, table tags). The following is the definition of the web-table only with respect to tags.

Definition 1: A web-table is an area enclosed between the two specific tags in HTML source code: `<table>` and `</table>`.

HTML does not separate presentation and structure, whereas tables on the Internet are used for the genuine purposes, together with the purpose of constructing the

layout of an HTML document. For solving this, we firstly defined two-types of web-table as below.

Definition 2:

- (1) A **meaningful table** is that used for genuine table purposes (conveying the contents’ information), and its contents’ meaning depends on its structure (providing deductive information).
- (2) A **decorative table** is that used in constructing the layouts of HTML documents and its contents’ meaning does not depend on its structure.

We need formal and specific definitions of the two kinds of table for machine readability while extracting from web-tables. Our previous work gave an explicit definition of a web-table according to the fact that the presence of an abstraction level (i.e. a head) determines the meaningfulness, regardless of how many abstraction levels exist. However, few meaningful tables seem to have lack of heads as shown in Figure 1.

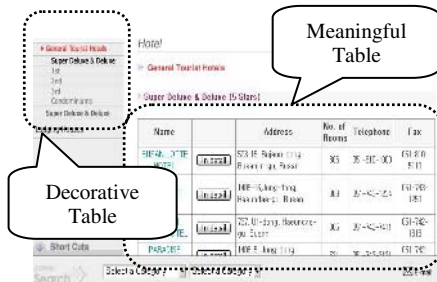


Fig. 1. Meaningful and decorative tables

New York	7/14 C
London	9/18 C
Paris	8/18 C
Mumbai	26/33 C
Beijing	11/22 C
Tokyo	9/18 C
Sydney	14/21 C
Johannesburg	11/27 C
Mexico City	10/25 C

Fig. 2. Example of a table with no head

In the table shown in Figure 2, any column or row doesn’t abstract the other(s). However we can easily estimate the abstraction of the first and second columns as ‘City’ and ‘Temperature’, respectively, using specific domain knowledge and linguistic clues. Although the table is missing an obvious head, we can not deny that the existence of an eclipsed abstractions of ‘City’ and ‘Temperature’ by referring to the relationship between the modified and modifier in these abstractions, e.g., ‘Temperature of city’. This eclipsed abstraction can be conceived as weak abstraction.¹ Considering this weak abstraction, we can define that all meaningful tables are equipped with at least one abstraction level, a head.

Definition 3: A web-table with a head is a meaningful table and that without a head is a decorative table.

3.2 Making Classification Model

Our previous research describes the method for extracting the tables that conform to Definition 3, from web pages. The four components of a table including title, head,

¹ For more discussion of the weak abstraction, we refer the reader to [9].

body and foot, have been defined as HTML tags: <caption>, <thead>, <tbody>, and <tfoot>. However, in some practical cases, those tags are not used distinctly for each component. In many cases <tr> and <td> are used for every component without distinction. This tendency causes the difficulty of separating the table's components. Our previous work estimated those components through considering the patterns of tables produced by many authors.

Decorative tables and meaningful tables require different techniques in editing, because they are edited with different ultimate aims: making layouts of HTML documents and communicating information more clearly, respectively. Based on these differences revealed by the different usage of HTML tags and by the difference of content type, we can formulate appearance. Those usage patterns should be extracted by considering not only (1) the appearance of the tables expressed by the table-tags (appearance features), but also (2) the contents' instance-type of each cell and their cross-cell distribution (consistency features). 'Consistency features' determine whether a table has repetitiveness and a similarity of cell contents' instance-type and cross-cells structure (i.e. cell distribution). A head in a meaningful table is the origin of this repetition and similarity of cell contents: cell contents adjust to the head. That is, a meaningful table has structural repetitiveness because of the head. This repetition can be computed by standard deviation, which is a well-known method to determine the degree of data distribution. If a table has repetitive span tags, its standard deviation is low. Thus it is estimated to be a meaningful table. We set 26 features based on the above observations. We refer readers to our previous work [3] for the details of these features.

The web-tables were converted using the 26 features for the machine-learning algorithm. Each feature value was converted into the value of one vector element. Thereby, a table was converted into a 26-dimensional vector. Once this input data was obtained, we constructed the classification model using the decision tree classifier, C4.5 [6]. In our previous work, we obtained a 94% F-measure value in distinguishing meaningful tables from decorative tables.

4 Extracting Head

This level separates the head from the body in a meaningful table. As we stated in the introduction, the aim of information extraction from web-tables is to establish machine-readable information, and this can be achieved by converting a table to attribute-value pairs. Because a table head abstracts related data in body, the head can be a strong candidate-attribute and the body can be a strong candidate-value. The table head is defined as a row(s) or a column(s). The extraction of a head starts with reanalyzing these features for rows and columns, instead of the entire table. For this, we first describe clues for separating the head from the body and institute features based on these clues. Then, the meaningful table set is converted to an input data set for machine learning algorithm, C4.5 using these features. The extracting model is constructed using this input data.

4.1 Instituting Features for Extracting Heads

We can observe two important factors that are pertinent to separating the head from the data in a web-table while observing the editing habits of authors themselves. First, authors often use specific techniques for separating the head from the data in order that readers understand a table more clearly. Second, the head abstracts related data in a table. Therefore, the row or column related to a head contains repetition. Both of these observations concern (1) rows and columns' appearance characteristics and (2) their inter-relations. While analyzing rows and columns as parts of a head in meaningful tables and considering their appearance and relations, the following features were formulated.

Observation 1. The `<th>` tag, which is a head tag, is not always respected in practical use. Generally, the `<td>` tag is used instead of the `<th>` tag. Therefore, if an author uses the dedicated head tags in question, he is purposely expressing a head.

Feature 1. If a row or column is expressed by `<th>` tags, the feature value is 1 otherwise 0.

Observation 2. When we edit web-tables, we often use decorative methods for separating the head from the data. The most remarkable methods among those separate the head from the body using background color and font attributes. In the web-table, background color is expressed by the tag attribute, 'bgcolor'. Font is more complex than background color. Font attributes include font face, font size, and font color. Font effects such as bold, italic and underscore are expressed by other tags, ``, ``, `<i>` and `<u>`.

We considered eight tags and attributes that are most commonly used for expressing the content of a head (hereafter, ECH): `<td bgcolor>`, ``, ``, ``, ``, ``, `<i>`, and `<u>`. If we consider the absence of ECH in a table as one of its values, then ECH can have one value or more than one value in a table. Generally, when an author wishes to structure a table as head and body, one ECH value is used, so that a 'binary distinction' is possible: (1) the absence or presence of an ECH, or (2) the presence of ECH with two distinct values in a given table. This binary 'distinctivity' is weakened if a head does not appear at all in the whole table or if there are more than two values.

Feature 2. The degree of possibility of the presence of a head (*DPH*) based on background color.

Feature 3. The degree of possibility of the presence of a head (*DPH*) based on font attributes.

The value of each ECH set (ECHV) can vary according to the table. Each element of the ECHV defines an 'area' in a table. We propose an 'area' to be a sequence of more than one cell, and that a single cell cannot be an area. The following is the result of observations of the ECHV and the areas defined by related characteristics.

- (a) An ECH contains at least two values.
- (b) A table can be divided into several areas according to the values of the tag.

- (c) The most unambiguous meaningful table is one containing tags having only two values, each of them defining an area.
- (d) If an area in a table has only one cell, or its shape is not rectangular, then it is indicative of a decorative table.
- (e) If all of the areas defined by each element of the ECHV are nonuniform in size and shape, then the table in which those areas are enclosed has a high probability of being a decorative table.

The following equations can be used to estimate the degree of presence of a head based on the previous five characteristics of the ECH:

$$DPH = H_{pres} - H_{abs} \tag{1}$$

$$H_{abs} = P_{area} + P_{rep} \tag{3}$$

$$H_{pres} = \begin{cases} 0 & \text{if } dn = 1 \\ \frac{1}{dn-1} & \text{otherwise} \end{cases} \tag{2}$$

$$P_{area} = \sum_{i \in ECHV} \sum_{j=1}^{an_i} p_{ij} \tag{4}$$

$$P_{rep} = \sum_{i \in ECHV} ran_i \tag{5}$$

where

an_i	= No. of areas in a table defined by the i -th element of the ECHV
p_{ij}	= The penalty that is assigned to the j -th area defined by the i -th element of ECHV ('1' when the j -th area has decorative characteristic (e.g., see Section 4.2.5), or '0' otherwise)
P_{area}	= The sum of penalties, p_{ij}
ran_i	= No. of nonuniform areas that share the same i -th element of the ECHV
P_{rep}	= The penalties considering nonrepetitive areas
H_{abs}	= The degree of absence of a HEAD considering the characteristics of the areas defined by the i -th element of the ECHV
dn	= No. of elements of the ECHV in a table
H_{pres}	= The degree of presence of a HEAD considering the number of elements of the ECHV

The degree of presence of a head (DPH) consists of two parts: H_{pres} and H_{abs} . H_{pres} obtains the degree of presence of a head by considering the number of elements of the ECHV. H_{abs} obtains the degree of absence of a head by considering the characteristics of the areas defined by each element of the ECHV.

According to the (c), if the number of elements of an ECH in a table (dn) is $dn = 2$, then this favors a high degree of presence of a head. This is a natural result, because of the structural characteristics of a table that is composed of a head and a body, and, as $dn = 2$, there is the least degree of ambiguity in distinguishing the two components. If $dn > 2$, then the value of H_{pres} is lower than when $dn = 2$, even if a table still has the possibility of having a head, because of the ambiguity in distinguishing two components. If $dn = 1$, then a table contains only one attribute value. Therefore, we cannot recognize the head by considering the ECHV.

The appearance characteristics of areas, as noted in the fourth and (d), do not offer any clues to the meaningfulness of a web table. Nevertheless, they can serve as criteria to determine whether a table is decorative. Applying these criteria, the DPH value can be estimated by means of the penalty value, which is high when the possibility of a table being decorative is high. Thus, H_{abs} consists of two parts: P_{area} and P_{rep} . P_{area} is

the sum of the penalties, p_{ij} , which consider the decorative characteristics of all the areas according to (d). If an area has only one cell, or if its shape is not rectangular, then the evaluation value of the area is assigned ‘1’ as the penalty; otherwise it is given the value ‘0’. P_{rep} is the penalty assigned when the areas of a table are nonuniform according to (e). In Equation 5, ran denotes the number of nonuniform areas that share the same i -th element of the $ECHV$. Therefore, if these areas are uniform, there is no penalty; otherwise the number of nonuniform areas is assigned a penalty. H_{abs} provides a negative value when determining the meaningfulness of a table. Thus, we obtain the value of DPH by subtracting H_{abs} from H_{pres} .

Table 1. Estimation of the DPH and its parameters

Equation	Parameter	Values	Estimation
H_{pres}	$dn = 1$	0	No determination
	$dn = 2$	1	Preponderant weight for presence of a head
	$dn \geq 3$	$0 < H_{pres} < 1$	Preponderant weight for presence of a head
H_{abs}	–	0	No determination
	–	$H_{abs} > 0$	Preponderant weight for absence of a head
DPH	–	$0 < DPH \leq 1$	Preponderant weight for presence of a head
	–	0	No determination
	–	$DPH < 0$	Preponderant weight for absence of a head

Units From	Up to	Price Per Unit
1	10	115.00 €
11	24	85.00 €
25	49	65.00 €
50	99	55.00 €
100	unlimited	45.00 €

Fig. 3. Example of meaningful table with contents types

Cut in length	Gamma Energy in PbWO4	e- Energy in PbWO4
50 micron	12 keV	128 keV
100 micron	20 keV	210 keV
500 micron	80 keV	642 keV
1 mm	85 keV	1.18 MeV
5 mm	137 keV	5.39 MeV
1 cm	219 keV	11.5 MeV

Fig. 4. Example of meaningful table with contents patterns

Observation 3. In many meaningful tables, cell-contents types (hereafter, CCT) such as link, image, digit and words are repetitive in some order in a body. In our observation, a head’s representative CCT is different from a table’s representative CCT (hereafter, TRT). In Figure 3, the TRT is the digit; however the head’s representative CCT is the words. Therefore, the fraction of the TRT in a row or column is the clue for extracting a head. Additionally, a cell can contain complex contents types. If a cell contains one or several contents types, it is converted to an integer value that contains contents type information using flag bits.

Feature 4. The fraction of TRT in a row or column.

In Figure 3, the CCTs of each row are uniform; however the CCTs of each column are not uniform because each column contains a part of a head. Therefore, an additional feature is needed which is Feature 5.

Feature 5. The fraction of a row's or column's representative CCT in a row or column.

We also consider the CCT pattern (hereafter, CP) in a row or column. For example, in Figure 3, the first row's CP is {words, words, words} and the second row's CP is {digit, digit, digit}. This difference is the clue that the first row is a head. We do not consider the undermost row or the rightmost column for this comparison because most heads are located in the uppermost row or leftmost column.

Feature 6. If a row's or column's CP is different of its successive row's or column's CP, the feature value is 1 otherwise 0. The undermost row and the rightmost column are always 0.

Observation 4. Cells have a particular sequence of token types. A token is the part of a sentence separated by specific delimiters such as space and punctuation marks, among others. We divide them into four types: word, digit, tag, and specific character, and a cell-content assumes a pattern according to these token types. We term it the cell-contents pattern (hereafter, CCP). In the meaningful table, we can often observe a repetitive CCP in a data area such as that shown in Figure 4. This example contains a 'digit-word' pattern in the data area. If a CCP is the majority of CCPs among a table, it is termed the table's representative CCP (TRP). Observation 4 is similar to Observation 3. Therefore we institute the following three features as Observation 3.

Feature 7. The fraction of TRP in a row or a column.

Feature 8. The fraction of a row or a column's representative CCP in a row or a column.

Feature 9. If a row or column's CP based on CCP is different from its successive row's or column's CP based on CCP, the feature value is 1 otherwise 0. The undermost row and the rightmost column are always 0.

Observation 5. Web tables are often composed of merged cells, expressed with the 'rowspan' or the 'colspan' attribute of the <td> tag, which offers higher-level abstraction for a head.

Feature 10. If a row or column contains a span tag, the feature value is 1 otherwise 0.

Observation 6. If the uppermost row and the leftmost column are part of a head, the crossed cell has a high possibility of being empty. In the case of an off-line document, this crossed cell contains an abstracted index of the uppermost row and that of the leftmost column, divided by a diagonal line. However, because HTML does not support expressing a diagonal line in a cell, the authors left the crossed cell vacant.

Feature 11. If a table has an empty cell in the first row, first column, the row and column that includes that empty cell is 1, otherwise 0.

Observation 7. The head of a web table is almost always located on the upper-side of the table or the left-side of table.

Feature 12. The index-number of a row or column in a table.

Observation 8. A cell of a meaningful table almost always contains a word or a short-length phrase.

Feature 13. The average number of characters in a cell of a row or a column.

Feature 14. The standard deviation of the number of characters in a cell of a row or a column.

4.2 Constructing Model

In the machine-learning step, we constructed a classification model. From the previous work, we acquired candidates for meaningful tables that have a formal structure. We converted these meaningful tables to an input data set for the machine-learning algorithm. As we mentioned above, the table head can be a row(s) or a column(s). Therefore, the rows and columns in the tables were converted using the 14 features in Section 4.1, as in Figure 5. Each feature value was converted into a value of a vector element. In this way, each row and column was converted into a 14-dimensional vector. From this input data, we constructed the classification model using a decision tree classifier, C4.5 [6]. This classifier was chosen because of the nature of our feature values and our need to observe the classification model.

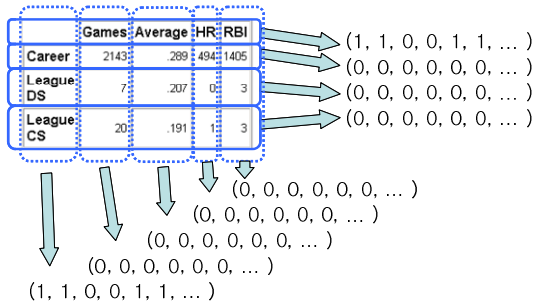


Fig. 5. Converting rows and columns to vectors

5 Experiment

For an experiment, we randomly chose HTML documents from the Internet as our training data set. We also used a part of Wang’s data [5].

We randomly collected our data from web pages. Wang’s was retrieved and downloaded from web pages, using the Google search engine, according to a set of key words likely to indicate documents containing tables. We mostly adopted Wang’s suggestion in order to mark-up our data set with three attributes: ‘tabid’, ‘genuine table’, and ‘table title’. The possible values of the ‘genuine table’ were ‘yes’ or ‘no’, and the others were a string. We added our own attribute, ‘head’, to the ‘<th> or <td>’ tag, to indicate whether a cell is a part of a head. The possible values of this attribute were ‘yes’ or ‘no’.

Table 2. The characteristics of data sets

Items	Our training data	Wang’s data	Total
No. of meaningful tables	964	969	1,933
No. of decorative tables	2,249	2,009	4258
Total	3,213	2,978	6,191

In chapter 4.1, we instituted 14 features for separating the head from the data and made an input data set using these features. In the machine-learning step, we applied the decision tree algorithm, C4.5 to the input data set. We randomly divided the data set into 10 parts. The decision tree was trained on nine parts, and then tested on the remaining part. This procedure was repeated 10 times. Then, the combined 10 parts were averaged to arrive at the overall performance measure. Table 3 shows the result of ranking by information gain and accumulative performance by the ranking.

Table 3. Accumulative performance by feature-ranking

Rank	Feature	Body			Head		
		Precision	Recall	F-measure	Precision	Recall	F-measure
1	12	0.905	1	0.953	0	0	0
2	4	0.989	0.967	0.978	0.739	0.898	0.811
3	7	0.988	0.968	0.978	0.746	0.888	0.811
4	5	0.982	0.980	0.981	0.813	0.823	0.818
5	9	0.981	0.981	0.981	0.818	0.817	0.818
6	2	0.983	0.980	0.981	0.817	0.837	0.827
7	8	0.980	0.986	0.983	0.857	0.805	0.830
8	11	0.985	0.983	0.984	0.840	0.858	0.849
9	1	0.985	0.984	0.984	0.847	0.861	0.854
10	6	0.985	0.983	0.984	0.846	0.862	0.854
11	14	0.982	0.988	0.985	0.878	0.833	0.855
12	13	0.983	0.988	0.986	0.883	0.842	0.862
13	3	0.984	0.988	0.986	0.879	0.845	0.862
14	10	0.984	0.987	0.986	0.872	0.852	0.862

According to the precisions and recalls in Table 3, all features are good criteria for separating the head from the data. Table 4 shows the final performance of head classifier: we obtained 86.2% accuracy in extracting the head.

Table 4. Performance of our head classifier

		Assigned class		Precision	Recall	F-Measure
		Body	Head			
True class	Body	24,587	329	0.984	0.987	0.986
	Head	389	2,237	0.872	0.852	0.862

6 Conclusions

This paper developed a method of separating the head from the data in tables. Although several studies have dealt with table information, most of them used a certain level of linguistic bias. From this fact arises domain dependency in extracting information from tables. In order to overcome this limitation, this paper used (1) cells' appearance characteristics (which distinguish head and body) and (2) their inter-relationship (founded on the relation between head and body) in order to cope with general HTML documents. With analysis on cells being parts of a head in meaningful tables and in consideration of their appearance and relations, we formulated 14 features for separating the head from the body, which serve the basis of information extraction. The method combining these features showed approximately 86.2% accuracy in separating the head from the body for the test set comprising general HTML documents.

The ultimate goal of our research is to convert semi-structured text in tables into structured information via the hierarchical semantic relations between words. In our future work, we will define the semantic relation between the head and the body in order to expand the application domain to information retrieval systems, the construction of primary data for ontology, and other fields and areas.

References

1. Chen, H.H., Tsai, S.C., Tsai, J.H.: Mining Tables from Large Scale HTML Texts. Proceedings of 18th International Conference on Computational Linguistics, Saarbrücken, Germany, July (2000)
2. Hurst, M.: Layout and Language: Beyond Simple Text for Information Interaction - Modeling the Table. Proceedings of the 2nd International Conference on Multimodal Interfaces, Hong Kong, (1999)
3. Jung, S.W., Park, D.W., Kwon, H.C.: Extracting Web-Table Information Using Decision Tree and Rule Based Approach, Asia Information Retrieval Symposium, China, October (2004) 281-284
4. Ning, G., Guowen, W., Xiaoyuan, W., Baile, S.: Extracting web table information in cooperative learning activities based on abstract semantic model. Computer Supported Cooperative Work in Design, The Sixth International Conference (2001) 492-497
5. Wang, Y., Hu, J.: A Machine Learning Based Approach for Table Detection on The Web in Proceedings of The Eleventh International World Wide Web Conference WWW2002, Sheraton Waialili Honolulu, Hawaii, USA (2002) 7-11
6. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations, Morgan Kaufmann Pub. (2000)
7. Yang, Y.: Web Table Mining and Database Discovery. M.Sc. thesis, Simon Fraser University, August (2002)
8. Kushmerick, N., Weld, D. S., Doorenbos, R.: Wrapper Induction for Information Extraction, 15th International Joint Conference on Artificial Intelligence(IJCAI-97), Nagoya, August (1997)
9. Jung, S.W., Kwon, H.C.: A Scalable Hybrid Approach for Extracting Head Components from Web Tables, accepted and to be appeared in IEEE transaction on knowledge and data engineering, vol. 18. No. 2.

Clustering Abstracts of Scientific Texts Using the Transition Point Technique*

David Pinto^{1,2}, Héctor Jiménez-Salazar¹, and Paolo Rosso²

¹ Faculty of Computer Science, BUAP, Puebla 72570,
Ciudad Universitaria, Mexico
{davideduardopinto, hgimenezs}@gmail.com

² Department of Information Systems and Computation,
UPV, Valencia 46022,
Camino de Vera s/n, Spain
{dpinto, proso}@dsic.upv.es

Abstract. Free access to scientific papers in major digital libraries and other web repositories is limited to only their abstracts. Current keyword-based techniques fail on narrow domain-oriented libraries, e.g., those containing only documents on high energy physics like those of the *hep-ex* collection of CERN. We propose a simple procedure to cluster abstracts which consists in applying the transition point technique during the term selection process. This technique uses the mid-frequency terms to index the documents due to the fact that they have a high semantic content. In the experiments we have carried out, the transition point approach has been compared with well known unsupervised term selection techniques. Transition point technique shown that it is possible to obtain a better performance than traditional methods. Moreover, we propose an approach to analyse the stability of transition point term selection method.

1 Introduction

Nowadays, very short text clustering on narrow domains has not received too much attention by the computational linguistic community. This is derived from the high challenge that this problem implies, since the obtained results are very unstable or imprecise when clustering abstracts of scientific papers, technical reports, patents, etc. But, as we can see, most digital libraries and other web-based repositories of scientific and technical information nowadays provide free access only to abstracts and not to the full texts of the documents. Moreover, some institutions, like the well known CERN¹, receive hundreds of publications every day that must be categorized on some specific domain with an unknown number of categories. This led to construct novel methods for treating this real problem.

* This work was partially supported by BUAP-VIEP 3/G/ING/05, R2D2 (CICYT TIC2003-07158-C04-03), ICT EU-India (ALA/95/23/2003/077-054), and Generalitat Valenciana Grant (CTESIN/2005/012).

¹ Centre Européen pour la Recherche Nucléaire.

Clustering of very short texts implies to deal with very low frequencies; moreover, if this kind of texts belong to scientific papers, the difficulty increases, due to the continue use of some words like, for instance: “in this paper we present...”, etc.; as a matter of fact, in [1], it is said that:

When we deal with documents from one given domain, the situation is cardinally different. All clusters to be revealed have strong intersections of their vocabularies and the difference between them consists not in the set of index keywords but in their proportion. This causes very unstable and thus very imprecise results when one works with short documents, because of very low absolute frequency of occurrence of the keywords in the texts. Usually only 10% or 20% of the keywords from the complete keyword list occur in every document and their absolute frequency usually is 1 or 2, sometimes 3 or 4. In this situation, changing a keywords frequency by 1 can significantly change the clustering results.

Some related work was presented in [9], where simple procedures in order to improve results by an adequate selection of keywords and a better evaluation of document similarity was proposed. The authors used as corpora two collections retrieved from the Web. The first collection was composed by a set of 48 abstracts (40 Kb) from the CICLing 2002 conference; the second collection was composed by 200 abstracts (215 Kb) from the IFCS-2000² conference. The main goal in this paper was to stabilize results in this kind of task; a 10% of differences among different clustering methods were obtained, taking into account different broadness of the domain and combined measures.

In [1] an approach for clustering abstracts in a narrow domain using Stein’s MajorClust Method for clustering both keywords and documents was presented. Here, Alexandrov et al. used the criterion introduced in [8] in order to perform the word selection process. The authors based their experiments on the first CICLing collection used by Makagonov et al. [9], and they succeeded in improving those results. In the final discussion, Alexandrov et al. stated that abstracts cannot be clustered with the same quality as full texts, though the achieved quality is adequate for many applications; moreover, they suggested that, for an open access via Internet, digital libraries should provide document images of full texts for the papers and not only abstracts.

More recently, in [6] a third experiment with the CICLing collection was carried out. In this paper, a novel method for keyword selection was proposed, claiming improving results on clustering abstracts for that collection. Jiménez-Salazar et al. based their comparisons with different mechanisms of term selection by using the evaluation of feature selection employed in the text categorization task [7].

After reviewing these works, we have observed that the feature selection process is the key of the clustering of abstracts task for narrow domains. Moreover, a bigger collection of abstracts is needed in order to confirm previously

² International Federation of Classification Societies; <http://www.Classification-Society.org>

obtained results. In the following Section we present a brief description of the Transition Point technique. The third Section describes the term selection methods used in the experiments we carried out. The fourth Section shows the data set and the performance measure formulas used. A comparison of the results obtained is presented in Section five. Finally, the conclusions of our experiments are given.

2 The Transition Point Technique

The Transition Point (TP) is a frequency value that splits the vocabulary of a document into two sets of terms (low and high frequency). This technique is based on the Zipf Law of Word Occurrences [22] and also on the refined studies of Booth [2], as well as Urbizagástegui [20]. These studies are meant to demonstrate that terms of medium frequency are closely related to the conceptual content of a document. Therefore, it is possible to form the hypothesis that terms whose frequency is closer to TP can be used as indexes of a document. A typical formula used to obtain this value is given in equation 1:

$$TP_V = \frac{\sqrt{8 * I_1 + 1} - 1}{2}, \quad (1)$$

where I_1 represents the number of words with frequency equal to 1 in the text T [15] [20]. Alternatively, TP_V can be localized by identifying the lowest frequency (from the highest frequencies) that it is not repeated; this characteristic comes from the properties of Booth's law for low frequency words [2].

Let us consider a frequency-sorted vocabulary of a text T ; i.e.,

$$V = [(t_1, f_1), \dots, (t_n, f_n)],$$

with $f_i \geq f_{i-1}$, then $TP_V = f_{i-1}$, iif $f_i = f_{i+1}$. The most important words are those that obtain the closest frequency values to TP, i.e.,

$$V_{TP} = \{t_i | (t_i, f_i) \in V, U_1 \leq f_i \leq U_2\}, \quad (2)$$

where U_1 is a lower threshold obtained by a given neighbourhood value of the TP, thus, $U_1 = (1 - NTP) * TP_V$ ($NTP \in [0, 1]$). U_2 is the upper threshold and it is calculated in a similar way ($U_2 = (1 + NTP) * TP_V$).

The TP technique has been used in different areas of Natural Language Processing (NLP) like: clustering of short texts [5], categorization of texts [12] [13], keyphrases extraction [14] [19], summarization [3], and weighting models for information retrieval systems [4]. Thus, we believe that there exists enough evidence to use this technique as a term selection process.

3 Term Selection Methods

Up to now, different term selection methods have been used in the clustering task; however, as we mentioned in Section 1, clustering abstracts for a narrow

domain implies the well known problem of the unidentified number of categories to be used in the clustering process. This led us to use unsupervised methods instead of supervised ones, as well as the identification of new categories, which is very usual in the domain of digital libraries. In this section we will describe the unsupervised term selection methods used in our experiments.

1. *Document Frequency (DF)*: This method assigns the value df_t to each term t , where df_t means the number of texts, in a collection, where t occurs. This method assumes that low frequency terms will rarely appear in other documents, and therefore, they will not have significance on the prediction of the class for this text.
2. *Term Strength (TS)*: The weight given to each term t is defined by the following equation:

$$ts_t = Pr(t \in T_i | t \in T_j), \text{ with } i \neq j,$$

where $sim(T_i, T_j) \geq \beta$, and β is a threshold that must be tuned by reviewing the similarity matrix. A high value of ts_t means that the term t contributes to the texts T_i and T_j to be more similar than β . A more detailed description can be found in [21].

3. *Transition Point (TP)*: A higher value of weight is given to each term t , as its frequency is closer to the TP frequency, named TP_V . The following equation shows how to calculate this value:

$$idtp(t, T) = \frac{1}{|TP_V - freq(t, T)| + 1},$$

where $freq(t, T)$ is the frequency of the term t in the document T .

The unsupervised methods presented here are the most succesful in the clustering area. Particulary, DF is an effective and simple method, and it is known that this method obtains comparable results to the classical supervised methods like χ^2 (CHI) and Information Gain (IG) [17]. TP also has a simple calculation procedure, and as it was seen in Section 2, it can be used in different areas of NLP. The DF and TP methods have a temporal linear complexity with respect to the number of terms of the data set. On the other hand, TS is computationally more expensive than DF and TP, because it requires to calculate a similarity matrix of texts, which implies this method to be in $O(n^2)$, where n is the number of texts in the data set.

4 Clustering of Abstracts in a Narrow Domain

As was mentioned in Section 1, previous works for clustering abstracts in a narrow domain (see [9], [1], and [6]) used a very small collection (only 48 abstracts and 6 categories). Therefore, there exists a need of a bigger sized real corpus in order to verify the results obtained. Following, we introduce *hep-ex* collection, a real corpus obtained from the CERN.

4.1 Data Set

In our experiments we used two corpora based on the collection of abstracts compiled and provided to us by the University of Jaén, Spain [11], named *hep-ex*. The first corpus was built by extracting a subset of documents from the full collection. We used the full collection as a second corpus, which is composed by 2,922 abstracts from the *Physics* domain originally stored in CERN³. The distribution obtained for both corpora is shown in Table 1. The distribution of the categories for each corpus is better described in Table 2.

We have preprocessed these collections by eliminating stopwords and by applying the Porter stemmer. Due to their average size per abstract (aprox. 47 words), the preprocessed collections are suitable for our experiments.

Table 1. Collections (preprocessed) features

Feature	Subset of <i>hep-ex</i>	Full collection	<i>hep-ex</i>
Size of the corpus (bytes)	165,349		962,802
Number of categories	7		9
Number of abstracts	500		2,922
Total number of terms	23,500		135,969
Vocabulary size (terms)	2,430		6,150
Term average per abstract	47		46.53

Table 2. Categories in corpora

Category	Number of texts	Subset of <i>hep-ex</i>	Full collection
Information Transfer and Management	1	NO	YES
Particle Physics - Phenomenology	3	YES	YES
Particle Physics - Experimental Results	2,623	YES	YES
XX	1	YES	YES
Nonlinear Systems	1	YES	YES
Accelerators and Storage Rings	18	YES	YES
Astrophysics and Astronomy	3	YES	YES
Other Fields of Physics	1	NO	YES
Detectors and Experimental Techniques	271	YES	YES

4.2 Performance Measurement

We used *F*-measure (commonly used in information retrieval [16]) in order to determine which method obtains the best performance. Given a set of clusters $\{G_1, \dots, G_m\}$ and a set of classes $\{C_1, \dots, C_n\}$, the *F*-measure between a cluster *i* and a class *j* is given by the following formula.

$$F_{ij} = \frac{2 \cdot P_{ij} \cdot R_{ij}}{P_{ij} + R_{ij}}, \quad (3)$$

³ <http://library.cern.ch>

where $1 \leq i \leq m$, $1 \leq j \leq n$. P_{ij} and R_{ij} are defined as follows:

$$P_{ij} = \frac{\text{Number of texts from cluster } i \text{ in class } j}{\text{Number of texts in cluster } i}, \quad (4)$$

and

$$R_{ij} = \frac{\text{Number of texts from cluster } i \text{ in class } j}{\text{Number of texts in class } j}. \quad (5)$$

The global performance of the clustering is calculated using the values of F_{ij} . This measure is named F measure and it is shown as follows:

$$F = \sum_{1 \leq i \leq m} \frac{|G_i|}{|D|} \max_{1 \leq j \leq n} F_{ij}, \quad (6)$$

where $|D|$ is the number of documents in the collection.

5 Experimental Results

Our main concern was to evaluate the term selection methods described above, in the clustering of abstracts task, specifically in a narrow domain. Thus, we have used only one clustering method, k -NN based on Jaccard similarity function [18], which is consider as unsupervised, and therefore it complies with our requirements.

5.1 Test over a Subset of *hep-ex*

In order to obtain a first glance of the behaviour of each term selection method used in our experiments, we performed a first test over a subset of *hep-ex*, composed by 500 abstracts taken randomly from the original collection; in the case of those categories with only one instance, we randomly choose two categories. The threshold used as the minimum similarity accepted in the k -NN clustering method was tuned over this collection. The average of similarities was used as a threshold.

Figure 1 shows F values for every term selection method executed over different percentages of the collection's vocabulary (from 600 to 2,000 terms).

Given a percentage of the collection vocabulary, DF and TS methods selected the higher score terms. TP method selected terms in a local fashion; i.e. it took a given number of terms from each text. Therefore, comparison among methods must be done through the vocabularies obtained in each selection of terms carried out by the methods. DF and TS methods used from 2% to 70% of the vocabulary terms. This range corresponds from 21 to 1,700 of the total terms in the collection. The TP selection method took from 5 to 30 terms from each text, given a similar range of total terms. In Fig. 1, the results of these three methods are shown; the horizontal axis represents the number of terms and the vertical axis the F values (eq. 6). In order to apply TS method, similarity matrix was calculated as 3-tuples (T_i, T_j, sim_{ij}) and sorted according sim_{ij} , then ts_t

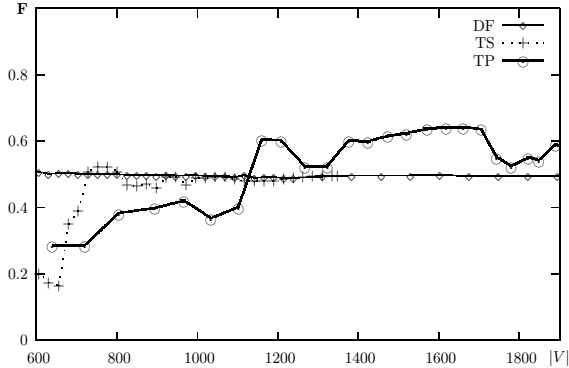


Fig. 1. Behaviour of DF, TS and TP methods in a subset of *hep-ex*

was computed for all terms. Since only 1,349 terms were obtained, threshold β was fixed to 0.

DF method was very stable but it did not help to the clustering task. From the beginning, DF included the most frequent terms in the texts, and this contributed to maintain a minimum level of similarity during the clustering task. Baseline, i.e. the clustering done without term selection ($F = 0.5004$), indicates that DF selects terms to represent texts that maintain resemblance with the original ones. On the other hand, TS method reached the maximum F value after 700 terms, and after 900 terms it obtained stability as well as the DF method did.

TP method outperformed the other two methods. The maximum F value for TP method was 0.6415. This value was reached with a vocabulary size of 1,661 terms which corresponds to only 22 terms per text. The unstability of TP method is derived from noisy words that are difficult to detect because of their low frequencies. Next subsection presents an analysis of the TP selection process, in order to control the unstability.

Analysis of the Unstability of TP: Although the TP method obtained the high F values, it did not allowed to decide the best quantity of terms to be used in the clustering task. It would be desirable to determine the best selection through an indicator based on characteristics of the collection. First of all, clustering method we have used has shown better performance when the number of clusters diminishes. This fact may be used in combination with $d\bar{f}_{V_i}$, which is explained in the following paragraph.

Let C_i be the text collection composed by the texts whose terms have been obtained by applying the TP method and by including the i terms with frequency value closer to TP_V from each original text. Let V_i be the vocabulary of C_i and $d\bar{f}_{V_i}$ the average of df_t for terms t that belong to V_i but do not belong to V_{i-1} . $d\bar{f}_{V_i}$ value is linked to the similarity among the texts. Clearly, the lowest value of $d\bar{f}_{V_i}$ is 1, and it means that the new terms added to V_{i-1} are not shared by

the texts of C_i . In our experiments it was observed that a decreasing in the df_{V_i} value ($df_{V_i} < df_{V_{i-1}}$) contributed to change instances from an incorrect cluster to a correct one. Therefore, terms with low df_{V_i} help to distribute texts into the clusters. Now, we can define an indicator of the goodness of a selection C_i .

Whenever the number of clusters (N_i) decreases after applying clustering to C_i , a lower df_{V_i} value means that new terms added to vocabulary V_i will provide a rising of similarity between texts in C_i . In such conditions df_{V_i} indicates a good selection. A way to express the above description is by saying that a good clustering supposes that df_{V_i} should be greater than $df_{V_{i-1}}$ and N_i should be greater than N_{i-1} . We define the goodness of selection C_i as:

$$dfN_i = \frac{(N_i - N_{i-1}) \times (df_{V_i} - df_{V_{i-1}})}{N_i}. \quad (7)$$

In Table 3 a neighbour of the maximum value of dfN_i is shown. Row 1 shows the i number of terms selected by the TP method; row 2, the size of the vocabulary of C_i ; row 3, the normalized values of dfN_i ; and row 4, the F measure. As we can see, dfN_i obtains the maximum value at $i = 22$, as also F does. Thus, independently of unstability of TP method, dfN_i can be used in order to determine what collection C_i must be used in the clustering task.

Table 3. Some normalized values of dfN_i

i	20	21	22	23	24
$ V_i $	1,572	1,619	1,661	1,706	1,744
dfN_i	0.573	0.621	1.027	0.584	0.990
F	0.637	0.6411	0.6415	0.636	0.551

5.2 Test over the Whole *hep-ex* Collection

An experiment was performed using the entire collection and applying the three methods described in Section 3. In this case, the noisy words had a notably effect, mainly in the TP method. Since TP method selects one term per time for each text, a wrong selection may be crucial in the clustering task. In some cases, this iterative process includes words that change dramatically the composition of texts. Thus, a term with very low DF value changes threshold used in the clustering task. We tried to face this problem with an enrichment of terms selected by TP. It is not possible to solve this task using related terms dictionaries like WordNet, since the terminology of texts is very specialized (see [6]). The problem was solved using n -grams as an approximation to related words.

Improving Transition Point Approach: A refined method based on the Transition Point technique was proposed in order to improve the results obtained over the whole collection of *hep-ex*. This method was named *Transition Point and Mutual Information* (TPMI), and basically uses $idtp(t, T)$ and mutual information. Thus TPMI is a refinement of the selection method provided by TP.

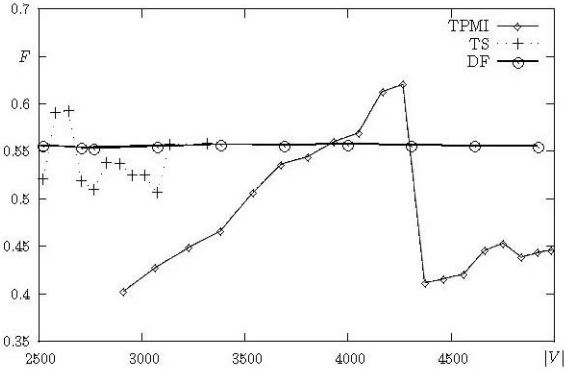


Fig. 2. Behaviour of DF, TS and TPMI term selection methods

Let TP_V be the transition point of the text $T = [t_1, \dots, t_k]$. We can calculate MI score of each term t_i as $MI(TP_V, t_i)$. The TPMI will assigns the final score:

$$tpmi(t_i, T) = idtp(t_i, T) * MI(TP_V, t_i) \tag{8}$$

$MI(x, y)$ was computed considering n -grams of x , where y appears at a distance of 2 words from x , and the frequency of both x and y was greater than 2.

The results obtained by using this refined method are shown in Figure 2. There we can see that this approach obtains the best value of F measure. Very similar results of clustering on the whole collection were obtained for DF and TS methods, with respect to the subset of *hep-ex*. Anyway, TS method reached the maximum F value (0.5925) with 43% of terms, which corresponds to a collection vocabulary size of 2,644 terms, and only 3,318 terms hold the threshold β . Whereas the DF method is very stable, it maintains its F values below of the baseline (0.5919). TPMI method had a good high peak ($F = 0.6206$) taking 20 terms, and giving a vocabulary size of 4,268 terms

6 Conclusions

In this paper we have proposed a new use of the Transition Point technique in the task of clustering of abstracts in a narrow domain. We used as a corpus a set of documents originally stored at CERN, in the *High Energy Physics* domain, which led to experiment with real collections conformed by very short texts (*hep-ex*). Findings after the execution of three unsupervised methods (DF, TS and TP) were that TP outperforms the other two methods over a subset of *hep-ex*. However, when the whole collection was used, a new filtering method had to be developed in order to improve the previous results. This method was named TPMI, and it used a dictionary of related terms, constructed over the same collection by using mutual information, since common dictionaries are not able to solve this case due to the very specialized vocabulary of this particular

domain. After the calculation of a baseline in both experiments was carried out, we could verify that this value was outperformed by our approaches.

We observed that there are not methods to determine the number of terms that a term selection method must obtain, in order to carry out the clustering task. Due to the unstability of TP, we carried out an analysis for explaining this behaviour and therefore to be able to determine the number of terms needed in such task. It is very important to continue with the study of the stability control for this methods, since, this is in fact the key in the clustering of very short texts.

Clustering abstracts in a narrow domain has received not too much attention by the computational linguistic community, and therefore it is very important to continue with the experiments in this area.

Acknowledgments

We thank to Davide Buscaldi by his useful comments on the present work. Also to Arturo Ráez and Alfonso Ureña who kindly provided the *hep-ex* collection.

References

1. M. Alexandrov, A. Gelbukh, P. Rosso: *An Approach to Clustering Abstracts*, A. Montoyo et al. (Eds.): NLDB 2005, LNCS 3513, pp. 275–285, 2005.
2. A. Booth: *A Law of Occurrences for Words of Low Frequency*, Information and control, 1967.
3. C. Bueno, D. Pinto, H. Jimenez: *El párrafo virtual en la generación de extractos*, Research on Computing Science Journal, 2005.
4. R. Cabrera, D. Pinto, H. Jimenez, D. Vilarino: *Una nueva ponderación para el modelo de espacio vectorial de recuperación de información*, Research on Computing Science Journal, 2005.
5. H. Jimenez, D. Pinto, P. Rosso, *Selección de Términos No Supervisada para Agrupamiento de Resúmenes*, In proceedings of Workshop on Human Language, ENC05, 2005.
6. H. Jiménez-Salazar, D. Pinto & P. Rosso: *Uso del punto de transición en la selección de términos índice para agrupamiento de textos cortos*, Journal: Procesamiento del Lenguaje Natural, Num. 35, pp. 114–118, 2005.
7. T. Liu, S. Liu, Z. Chen, W.-Y. Ma: *An Evaluation on Feature Selection for Text Clustering*, In Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003), Washington DC, 2003.
8. P. Makagonov, M. Alexandrov, K. Sboychakov: *A toolkit for development of the domain oriented dictionaries for structuring document flows*, In: Data Analysis, Classification, and Related Methods, Studies in classification, data analysis, and knowledge organization, Springer, pp. 83–88, 2000.
9. P. Makagonov, M. Alexandrov, A. Gelbukh: *Clustering Abstracts instead of Full Texts*, Text, Speech and Dialogue (TSD-2004). Lecture Notes in Artificial Intelligence, N 3206, Springer-Verlag, pp. 129–135, 2004.
10. C. Manning, H. Schütze: *Foundations of Statistical Natural Language Processing*, MIT Press. Cambridge, MA, May 1999.

11. A. Montejo-Ráez, L. A. Ureña-López, R. Steinberger: *Text Categorization using bibliographic records: beyond document content*, Journal: Procesamiento del Lenguaje Natural, Num. 35, pp. 119–16, 2005.
12. E. Moyotl, H. Jiménez: *An Analysis on Frequency of Terms for Text Categorization*, Proceedings of XX Conference of Spanish Natural Language Processing Society (SEPLN-04), 2004.
13. E. Moyotl-Hernández, H. Jiménez-Salazar: *Enhancement of dtp feature selection method for text categorization*. Lecture Notes in Computer Science 3406, Gelbukh (Ed.), pp. 719–722, 2005.
14. D. Pinto, F. Pérez: *Una Técnica para la Identificación de Términos Multipalabra*, Proceedings of 2nd. National Conference on Computer Science, México, 2004.
15. Edgar Moyotl Hernández: *DTP, un metodo de selección de términos para agrupamiento de textos*, Tesis de maestría: Facultad de Ciencias de la Computación, BUAP, 2005.
16. C. J. van Rijsbergen: *Information Retrieval*, London, Butterworths, 1999.
17. F. Sebastiani: *Machine Learning in Automated Text Categorization*, ACM Computing Surveys, Vol. 34(1), pp 1–47, 2002.
18. K. Shin, S. Y. Han: *Fast clustering algorithm for information organization*, In A. F. Gelbukh, editor, CICLing, Lecture Notes in Computer Science, Volume 2588, pp. 619–622, 2003.
19. M. Tovar, M. Carrillo, D. Pinto, H. Jimenez, *Combining Keyword Identification Techniques*, Journal: Research on Computing Science, 2005.
20. R. Urbizagástegui: *Las posibilidades de la Ley de Zipf en la indización automática*, Research report of the California Riverside University, 1999.
21. Y. Yang: *Noise Reduction in a Statistical Approach to Text Categorization*, in *Proc. of SIGIR-ACM*, pages 256–263, 1995.
22. G. K. Zipf: *Human Behavior and the Principle of Least-Effort*, Addison-Wesley, Cambridge MA, 1949.

Sense Cluster Based Categorization and Clustering of Abstracts

Davide Buscaldi¹, Paolo Rosso¹, Mikhail Alexandrov², and Alfons Juan Ciscar¹

¹ Dpto. Sistemas Informáticos y Computación (DSIC),
Universidad Politécnica de Valencia, Spain
{dbuscaldi, proso, ajuan}@dsic.upv.es

² Center for Computing Research,
National Polytechnic Institute, Mexico
dyner1950@mail.ru

Abstract. This paper focuses on the use of sense clusters for classification and clustering of very short texts such as conference abstracts. Common keyword-based techniques are effective for very short documents only when the data pertain to different domains. In the case of conference abstracts, all the documents are from a narrow domain (i.e., share a similar terminology), that increases the difficulty of the task. Sense clusters are extracted from abstracts, exploiting the WordNet relationships existing between words in the same text. Experiments were carried out both for the categorization task, using Bernoulli mixtures for binary data, and the clustering task, by means of Stein's MajorClust method.

1 Introduction

Typical approaches to document clustering and categorization in a given domain are to transform the textual documents into vector form, by using a list of index keywords. This kind of approaches has also been used for clustering heterogeneous short documents (e.g. documents containing 50-100 words) with good results. However, term-based approaches usually give unstable or imprecise results when applied to documents from one narrow domain.

Previous works on narrow-domain short document classification obtained good results by using supervised methods and set of keywords (*itemsets*) as index terms [3].

In this work, we exploited the linguistic information extracted from WordNet in order to extract key *concept clusters* from the documents, using the method proposed by Bo-Yeong Kang *et al.* [5], which is based on semantic relationships between the terms in the document. Concept clusters are used as index words.

Various methods have been tested for the categorization and clustering task, including Bernoulli mixture models, which have been investigated for text categorization in [4]. Text categorization procedures are based on either binary or integer-valued features. In our case, due to the low absolute frequency observable in short documents, we used only the information if an index term was or not in the abstract, thus obtaining a binary representation of each document.

2 The MajorClust Clustering Method

We use the MajorClust method described by B.Stein [6], with the standard vector model for document representation. To evaluate the closeness between two documents, the well-known cosine measure is used, with some modifications for term weighting discussed in [1] in order to take into account the fact that abstracts usually introduce the reader to the possibilities of a suggested approach or method, while the full papers give its more or less detailed explanation.

The idea of the MajorClust method is very simple: it distributes objects to clusters in such a way that the similarity of an object to the assigned cluster exceeds its similarity to any other cluster. MajorClust method works as follows: first, every object is considered a separate cluster. Then the objects are joined to the nearest cluster. In the process of cluster construction, the objects can change their cluster in contrast, for instance, to the nearest neighbor method.

3 Bernoulli Mixture-Based Classifiers

A finite mixture model is a probability (density) function of the form:

$$p(\mathbf{x}) = \sum_{i=1}^I p(i)p(\mathbf{x}|i) \quad (1)$$

where I is the number of mixture components and, for each component i , $p(i)$ is its prior or coefficient and $p(i)$ is its component-conditional probability (density) function. It can be seen as a generative model that first selects the i -th component with probability $p(i)$ and then generates \mathbf{x} in accordance with $p(\mathbf{x}|i)$.

A Bernoulli mixture model is a particular case of (1) in which each component i has a D -dimensional Bernoulli probability function governed by its own vector of parameters or *prototype* $p_i = (p_{i1}, \dots, p_{iD})^t \in [0, 1]^D$,

$$p(\mathbf{x}|i) = \prod_{d=1}^D p_{id}^{x_d} (1 - p_{id})^{1-x_d} \quad (2)$$

As with other types of mixtures, Bernoulli mixtures can be used as class-conditional models in supervised classification tasks. Let C denote the number of supervised classes. Assume that, for each supervised class c , we know its prior $p(c)$ and its class-conditional probability function $p(\mathbf{x}|c)$, which is a mixture of I_c Bernoulli components,

$$p(\mathbf{x}|c) = \sum_{i=1}^{I_c} p(i|c) p(\mathbf{x}|c, i). \quad (3)$$

Then, the optimal Bayes decision rule is to assign each pattern vector (\mathbf{x}) to a class $c^*(\mathbf{x})$ giving maximum a posteriori probability, or, equivalently,

$$c^*(\mathbf{x}) = \arg \max_c \left(\log p(c) + \sum_{i=1}^{I_c} p(i|c) p(\mathbf{x}|c, i) \right) \quad (4)$$

Maximum likelihood estimation of class-conditional mixture parameters (component coefficients and Bernoulli prototypes) can be reliably accomplished by the well-known EM algorithm [4].

4 Experiments and Results

The experiments have been conducted on the set of CiCling2002¹ conference abstracts, consisting in 48 abstracts related to computational linguistics the-matics grouped into the following 4 categories: *linguistic*, *ambiguity*, *lexicon*, *text processing*. The intersection of vocabulary for the documents from the most different second and fourth groups was about 70%. This implies that the selected domain is narrow.

The semantic indexing consists in extracting *concept clusters* from the doc-uments. A concept cluster is composed by two or more document nouns that are connected by one or more of the following relations (R): identity, synonymy, hypernymy, meronymy. Except for identity, that is, word count, the others are defined in WordNet. Each cluster obtain a weight proportional to the number of nouns in the cluster and depending on the type of the relations connecting them, according to [5].

4.1 Categorization

Each document was represented as a bit vector, with 1 indicating the keyword (or key-sense cluster) was in the document and 0 elsewhere. The size of the vocabulary (d) was $d = 465$ when using full-text indexing and $d = 331$ for the semantic indexing technique. Due to the limited size of the corpus, the testing method was the leaving-one-out, using each document vector as test set and all the remaining documents as training.

Each average was computed from 4 runs, each one entailing a randomly ini-tialised EM-based learning of a Bernoulli mixture per class. For simplicity, we did not try classifiers with class-conditional mixtures of different number of com-ponents.; i.e., an I -component classifier means that a mixture of $I_c = I$ Bernoulli components was trained for each abstract c . The average error obtained using the standard indexing and the Bernoulli mixtures classifier was 81.2%, whereas the average error obtained using sense clusters as indices was 48.8%. However, the use of mixtures does not seem to be useful since the errors do not change significantly with respect to the number of components in the mixture. The reason could be due to the small size of the corpus, that does not allow to estimate accurately the probabilities.

4.2 Clustering

The procedure of evaluating the clustering quality is called cluster validation. For testing cluster validity we used the index of expected density of clustering

¹ <http://www.cicling.org/2002/>

($\bar{\rho}$) defined in [6], and the F -measure in the form presented in [2] in order to evaluate the clusters usability (i.e., the correspondance between the results of automatic and human clustering).

Each abstract was represented with a feature vector, constituted by the weights of the index sense clusters (0 if the sense cluster is not present in the abstract). Results were compared with the indexing method based on words [1], according to well-known tf and $tf-idf$ techniques.

The obtained F -measure was 0.44 with the sense cluster indexing and 0.64 with the standard $tf-idf$, whereas the obtained $\bar{\rho}$ was 0.08 and 0.56, respectively. Therefore, sense cluster indexing did not improve the results for the text clustering task as it did for the categorization one.

5 Conclusions

The use of semantic indexing seems to improve results in the case of categorization, although the small size of the corpus does not allow to appreciate the use of the multivariate Bernoulli mixture model. Semantic indexing did not allow to obtain better results for clustering. Further investigation will be done over the weights assigned to the WordNet relationships, and using larger collections like the Medline² one.

Acknowledgments

We would like to thank R2D2 CICYT (TIC2003-07158-C04-03) and ICT EU-India (ALA/95/23/2003/077-054) research projects for partially supporting this work.

References

1. Alexandrov, M., Gelbukh, A., Rosso, P.: An Approach to Clustering Abstracts. NLDB05, Alicante, Spain, 2005.
2. Eissen, S., M., B. Stein: Analysis of Clustering Algorithms for Web-based Search. Practical Aspects of Knowledge Management, LNAI N 2569, Springer, 2002, pp.168178.
3. Hynek, J., Jezek, K., Rohlik, O.: Short Document Categorization Itemsets Method. PKDD-2000, Springer, LNCS N 1910, 2000, 6 pp.
4. A. Juan and E. Vidal: On the use of Bernoulli mixture models for text classification. Pattern Recognition, 35(12):2705-2710, 2002.
5. Kang, B., Kim, H., Lee, S.: Performance Analysis of Semantic Indexing in Text Retrieval. CICLing 2004, Lecture Notes in Computer Science, Vol. 2945. Springer-Verlag, 2004
6. Stein, B., S. M. Eissen, F. Wissbrock: On Cluster Validity and the Information Need of Users. Proc. 3-rd IASTED Intern. Conf. on Artificial Intelligence and Applications (AIA'03), Acta Press, 2003, pp. 216221.

² <http://www.ncbi.nlm.nih.gov/entrez/query.fcgi>

Analysing Part-of-Speech for Portuguese Text Classification

Teresa Gonçalves¹, Cassiana Silva², Paulo Quaresma¹, and Renata Vieira²

¹ Dep. Informática, Universidade de Évora, 7000 Évora, Portugal
{tcg, pq}@di.uevora.pt

² Unisinos, CEP 93.022-000 São Leopoldo, RS, Brasil
{cassiana, renata}@exatas.unisinos.br

Abstract. This paper proposes and evaluates the use of linguistic information in the pre-processing phase of text classification. We present several experiments evaluating the selection of terms based on different measures and linguistic knowledge. To build the classifier we used Support Vector Machines (SVM), which are known to produce good results on text classification tasks.

Our proposals were applied to two different datasets written in the Portuguese language: articles from a Brazilian newspaper (Folha de São Paulo) and juridical documents from the Portuguese Attorney General's Office. The results show the relevance of part-of-speech information for the pre-processing phase of text classification allowing for a strong reduction of the number of features needed in the text classification.

1 Introduction

Machine learning techniques are applied to document collections aiming at extracting patterns that may be useful to organise or retrieve information from large collections. Tasks related to this area are text classification, clustering, summarisation, and information extraction. One of the first steps in text mining tasks is the pre-processing of the documents, as they need to be represented in a more structured way to be fed to machine learning algorithms. In this step, words are extracted from the documents and, usually, a subset of words (stop words) is not considered, because their role is related to the structural organisation of the sentences and does not have discriminating power over different classes. This shallow and practical approach is known as bag-of-words. Usually, to reduce semantically related terms to the same root, a lemmatiser is applied.

Finding more elaborated models is still a great research challenge in the field; natural language processing increases the complexity of the problem and these tasks, to be useful, require efficient systems. Our proposal considers that there is still lack of knowledge about how to bring natural language and traditionally known techniques of data mining tasks together for efficient text mining. Therefore, here we make an analysis of different word categories (nouns, adjectives, proper names, verbs) for text mining, and perform a set of experiments of

text classification over Brazilian and European Portuguese data. Our goal is to investigate the use of linguistic knowledge in text mining.

As classifier we used Support Vector Machines (SVM), which are known to be good text classifiers [8]. Other learning algorithms have been also applied such as decision trees [16], linear discriminant analysis and logistic regression [13], and naïve Bayes algorithm [10].

A method for incorporating natural language processing into existing text classification procedures is presented in [1] and a study of document representations based on natural language processing in four different corpora and two languages (English and Italian) is reported in [11]. Although they strongly claim against the union of NLP and text mining their experiments present just a few combinations of linguistic information. We believe that there is still much space for research in this area, and in this paper we show some interesting results of text classification regarding the simple linguistic knowledge of word categories.

In [6], SVM performance is compared with other Machine Learning algorithms and in [7] a thorough study on some preprocessing techniques (feature reduction, feature subset selection and term weighting) is made over European Portuguese and English datasets. The impact of using linguistic information on the preprocessing phase is reported in [15] over a Brazilian dataset.

This paper is organised as follows: in Section 2, a description of the used techniques and datasets is presented while Sections 3 and 4 describe the experiments. Conclusions and future work are pointed out in Sections 5 and 6.

2 Methods and Materials

In this section we describe the Support Vector Machines paradigm, the natural language tools applied for pre-processing the documents, the datasets studied and, at the end, the experimental setup is explained.

2.1 Support Vector Machines

Support Vector Machines (SVM) is a learning algorithm introduced by Vapnik and coworkers [4], which was motivated by the theoretical results from the statistical learning theory. It joins a kernel technique with the structural risk minimisation framework. A *kernel technique* comprises two parts: a module that performs a mapping into a suitable feature space and a learning algorithm designed to discover linear patterns in that space.

The *kernel function*, that implicitly performs the mapping, depends on the specific type and domain knowledge of the data source. The *learning algorithm* is general purpose and robust; it's also efficient, since the amount of computational resources required is polynomial with the size and number of data items, even when the dimension of the embedding space grows exponentially [14]. Key aspects of the approach can be highlighted as follows (illustrated in Figure 1):

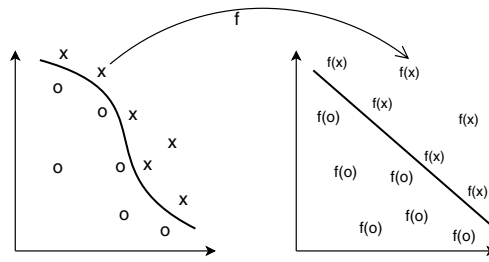


Fig. 1. Kernel function: Data's nonlinear pattern transformed into linear feature space

- Data items are embedded into a vector space called the feature space.
- Linear relations are discovered among images of data items in feature space.
- Algorithm is implemented in a way that the coordinates of the embedded points are not needed; only their pairwise inner products.
- Pairwise inner products can be computed efficiently directly from the original data using the kernel function.

The *structural risk minimisation* (SRM) framework creates a model with a minimised VC (Vapnik-Chervonenkis) dimension. This developed theory[17] shows that when a model's VC dimension is low, the expected probability of error is also low, which means good performance on unseen data.

2.2 Natural Language Processing Tools

We applied a Portuguese stop-list (set of non-relevant words such as articles, pronouns, adverbs and prepositions) and POLARIS, a lexical database [9], to generate the lemma for each Portuguese word.

The POS tags were obtained through the syntactic analysis performed by PALAVRAS [2] parser, which was developed in the context of the VISL project (Visual Interactive Syntax Learning – <http://www.visl.sdu.dk/>) in the Institute of Language and Communication of the University of Southern Denmark. Possible morpho-syntactic tags are:

- adjective (*adj*),
- adverb (*adv*),
- article (*det*),
- conjunction (*conj*),
- interjection (*in*),
- noun (*n*),
- numeral (*num*),
- preposition (*prp*),
- pronoun (*pron*),
- proper noun (*prop*) and
- verb (*v*).

Portuguese is a morphological rich language: while nouns and adjectives have 4 forms (two *genders* – masculine and feminine and two *numbers* – singular and plural), a regular verb has 66 different forms (two *numbers*, three *persons* – 1st, 2nd and 3rd and five *modes* – indicative, conjunctive, conditional, imperative and infinitive, each with different number of *tenses* ranging from 1 to 5).

PALAVRAS parser is robust enough to always produce an output even for incomplete or incorrect sentences (which might be the case for the type of documents used in text mining tasks). It has a comparatively low percentage of errors (less than 1% for word class and 3-4% for surface syntax)[3].

It's output is the syntactic analysis of each phrase and the POS tag associated with each word. For example, the morphological tagging of the phrase 'O Manuel ofereceu um livro ao seu pai./Manuel gave a book to his father.' is:

```
o [o] <artd> <dem> DET M S
Manuel [Manuel] PROP M S
ofereceu [oferecer] V PS 3S IND VFIN
um [um] <quant> <arti> DET M S
livro [livro] N M S
a [a] <prp>
o [o] <artd> <dem> DET M S
seu [seu] <pron-det> <poss> M S
pai [pai] N M S
```

2.3 Dataset Description

As already mentioned, we performed the experiments over two datasets: FSP, a Brazilian Portuguese dataset of newspaper articles from "Folha de São Paulo" and PAGOD – Portuguese Attorney General's Office Decisions, an European Portuguese dataset of juridical documents.

FSP Dataset. FSP is a subset of the NILC corpus (Núcleo Inter-institucional de Linguística Computacional – <http://www.nilc.icmc.usp.br/nilc/>) containing 855 documents from the year of 1994.

These documents are related to five newspaper sections, each one having 171 documents: informatics, property, sports, politics and tourism. Since each document belongs to one of the five possible classes, we have a multi-class problem. From all documents, there are 19522 distinct words, and, on average, 215 running words (tokens) and 124 unique words (types) per document.

PAGOD Dataset. On the other hand, PAGOD has 8151 juridical documents, represents the decisions of the Portuguese Attorney General's Office since 1940 and delivers 96 MBytes. All documents were manually classified by juridical experts into a set of categories belonging to a taxonomy of legal concepts with around 6000 terms. Each document is classified into multiple categories so, we have a multi-label classification task. Normally, it is solved by splitting into a set of binary classification tasks and considering each one independently.

For all documents, we found 68877 distinct words and, on average, 1608 tokens and 366 types per document. A preliminary evaluation showed that, from all potential categories only about 3000 terms were used and from all 8151 documents, only 6773 contained at least one word in all experiments. Table 1 presents the top ten categories (the most used ones) and the number of documents belonging to each one.

Table 1. PAGOD’s top ten categories

category (Portuguese)	category (English)	# docs
pensão por serviços excepcionais	excepcional services pension	906
deficiente das forças armadas	army injured	678
prisioneiro de guerra	war prisoner	401
estado da Índia	India state	395
militar	military	388
louvor	praise	366
funcionário público	public officer	365
aposentação	retirement	342
competência	competence	336
exemplar conduta moral e cívica	exemplary moral and civic behaviour	289

2.4 Experimental Setup

Now we present the choices made in our study: the kind of kernel used, the representation of documents and the used measures of learners’ performance.

The linear SVM was run using the WEKA [18] software package from Waikato University, with default parameters (complexity parameter equal to one and normalised training data) and performing a 10-fold cross-validation procedure.

To represent each document we chose the bag-of-words approach, a *vector space model* (VSM) representation: each document is represented by the words it contains, with their order and punctuation being ignored. From the bag-of-words we removed all words that contained digits.

Learner’s performance was analysed through precision, recall and F_1 measures [12] of each category (obtained from contingency table of the classification – prediction *vs.* manual classification). For each one, we calculated the micro- and macro-averages and made significance tests regarding a 95% confidence level.

3 Baseline Experiments

In this section, we first present the IR techniques used, the experiments made and the results obtained.

We considered three typical information retrieval preprocessing techniques: feature reduction/construction, feature subset selection and term weighting. For each technique, we considered several experiments as described below.

Feature Reduction/Construction. On trying to reduce/construct features we used some linguistic information: we applied a Portuguese stop-list and POLARIS to generate the lemma for each Portuguese word. We made three sets of experiments:

- rdt_1 : consider all words of the original documents
- rdt_2 : consider all words but the ones that belong to the stop-list (stop words)
- rdt_3 : all words (except the stop words) are transformed into its lemma

Feature Subset Selection. For selecting the best features we used a filtering approach, keeping the ones with higher scores according to different functions:

- scr_1 : *term frequency*. The score is the number of times the feature appears in the dataset; only the words occurring more frequently are retained;
- scr_2 : *mutual information*. It evaluates the usefulness of an attribute by measuring the Mutual Information with respect to the class. Mutual Information is an Information Theory measure [5] that ranks the information received to decrease the uncertainty. The uncertainty is quantified through the Entropy measure.

For each filtering function, we tried different threshold values. This threshold is given by the number of times the word appears in all documents – thr_n means that all words appearing less than n times are eliminated. For each threshold we looked at the number of words retained and used it to select the features.

Term Weighting. Finally, for the term weighting experiments, we made two different experiments:

- wgt_1 : uses $TF(w_i, d_j)$ normalised to unit length. $TF(w_i, d_j)$ is the number of times word w_i occurs in document d_j .
- wgt_2 : *TFIDF representation*. It's $TF(w_i, d_j)$ multiplied by $\log(N/DF(w_i))$, where N is the total number of documents and $DF(w_i)$ is the number of documents in which w_i occurs. The measure is normalised to unit length.

3.1 Experiments

For the FSP dataset, we performed experiments for all options of feature reduction/construction, scoring function and term weighting (rdt_1 , rdt_2 and rdt_3 ; scr_1 and scr_2 ; wgt_1 and wgt_2) and tried the following threshold values: thr_1 , thr_5 , thr_{10} , thr_{20} , thr_{30} , ..., thr_{90} , totalling a number of 132 experiments.

For the PAGOD dataset, we performed experiments for rdt_2 and rdt_3 options of feature reduction/construction, scr_1 and scr_2 scoring functions and wgt_1 and wgt_2 term weighting schemes. We tried the threshold values thr_1 , thr_{50} , thr_{100} , thr_{200} , ..., thr_{900} (88 experiments).

Table 2 presents the number of words ($\#words$) and per document averages of token (avg_{tok}) and type (avg_{typ}) for each feature reduction/construction setup and Table 3 shows the number of features obtained for each threshold value.

Table 2. Baseline experiments: number of words and averages for each dataset

	FSP			PAGOD		
	#words	avg _{tok}	avg _{typ}	#words	avg _{tok}	avg _{typ}
<i>rdt</i> ₁	19522	215	124	68877	1608	366
<i>rdt</i> ₂	19352	134	100	68679	963	331
<i>rdt</i> ₃	13317	128	91	42399	921	258

Table 3. Baseline experiments: number of features for each threshold value

FSP	<i>thr</i> ₅	<i>thr</i> ₁₀	<i>thr</i> ₂₀	<i>thr</i> ₃₀	<i>thr</i> ₄₀	<i>thr</i> ₅₀	<i>thr</i> ₆₀	<i>thr</i> ₇₀	<i>thr</i> ₈₀	<i>thr</i> ₉₀
	4420	2315	1153	745	529	397	334	265	222	199
PAGOD	<i>thr</i> ₅₀	<i>thr</i> ₁₀₀	<i>thr</i> ₂₀₀	<i>thr</i> ₃₀₀	<i>thr</i> ₄₀₀	<i>thr</i> ₅₀₀	<i>thr</i> ₆₀₀	<i>thr</i> ₇₀₀	<i>thr</i> ₈₀₀	<i>thr</i> ₉₀₀
	9477	6435	4236	3226	2577	2198	1897	1678	1514	1369

3.2 Results

Table 4 presents the minimum, maximum, average and standard deviation of all experiments.

Table 4. Baseline experiments summarising values

	FSP						PAGOD					
	μP	μR	μF_1	<i>MP</i>	<i>MR</i>	<i>MF</i> ₁	μP	μR	μF_1	<i>MP</i>	<i>MR</i>	<i>MF</i> ₁
min	.863	.863	.863	.865	.863	.864	.497	.742	.606	.491	.687	.559
max	.982	.982	.982	.983	.982	.982	.878	.789	.816	.799	.741	.758
avg	.947	.947	.947	.947	.947	.947	.837	.768	.800	.768	.716	.734
stdev	.026	.026	.026	.026	.026	.026	.043	.013	.023	.034	.015	.022

FSP Dataset. From all 132 FSP experiments, there were 19 ‘best’ ones with no significant difference for all six performance measures (precision, recall and F_1 micro- and macro-averages). The distribution of these experiments on each setup was the following:

- for *rdt*₁, *rdt*₂ and *rdt*₃ there were 4, 6 and 9 ‘best’ experiments,
- for *scr*₁ and *scr*₂ there were 0 and 19 ‘best’,
- for *wgt*₁ and *wgt*₂ there were 9 and 10 ‘best’ and finally,
- for *thr*₅, *thr*₁₀, *thr*₂₀, *thr*₃₀ and *thr*₄₀ there were 6, 6, 4 2 and 1 ‘best’ values.

From these results, one can say that the most suited setup is lemmatisation along with mutual information scoring function and TFIDF weighting scheme. The *thr*₄₀ threshold is the one with less features from the set of the ‘best’ ones.

PAGOD Dataset. Table 5 presents, for the PAGOD dataset, the number of experiments with no significant difference with respect to the best one and the distribution of these experiments on each setup (for example, macro- F_1 have 16 best experiments: 7 belong to the *rdt*₂ setup and 9 to the *rdt*₃ one).

Table 5. Baseline PAGOD experiments: number belonging to the set of best results

	μP	μR	μF_1	MP	MR	MF_1
<i>best</i>	5	22	34	14	21	16
<i>rdt₂</i>	2	10	14	4	7	7
<i>rdt₃</i>	3	12	20	10	14	9
<i>scr₁</i>	2	21	19	2	20	14
<i>scr₂</i>	3	1	15	12	1	2
<i>wgt₁</i>	1	16	26	10	16	13
<i>wgt₂</i>	4	6	8	4	5	3
<i>thr₁</i>	0	2	0	0	2	0
<i>thr₅₀</i>	0	2	0	0	2	0
<i>thr₁₀₀</i>	0	2	0	0	2	0
<i>thr₂₀₀</i>	0	4	1	0	3	0
<i>thr₃₀₀</i>	0	4	2	1	4	3
<i>thr₄₀₀</i>	0	2	3	1	2	2
<i>thr₅₀₀</i>	0	2	6	1	2	3
<i>thr₆₀₀</i>	0	2	6	1	2	3
<i>thr₇₀₀</i>	0	1	5	3	1	2
<i>thr₈₀₀</i>	0	1	5	3	1	2
<i>thr₉₀₀</i>	5	0	6	4	0	1

One can say that both *rdt₂* and *rdt₃* produce similar results and that term frequency scoring function along with term frequency weighting scheme is the setup with best results. The *thr₈₀₀* threshold is the biggest with good results.

Table 6 shows the precision, recall and F_1 for the best setups of both datasets; the values that belong to the set of best ones are bold faced. From these figures we can say that *rdt₃.scr₁.wgt₁.thr₈₀₀* is the best setup for the PAGOD dataset since it has more significant best values than the other one.

Table 6. Baseline experiments: precision, recall and F_1 micro- and macro-averages for the best setups

	μP	μR	μF_1	MP	MR	MF_1
FSP. <i>rdt₃.scr₂.wgt₂.thr₄₀</i>	.975	.975	.975	.976	.975	.975
PAGOD. <i>rdt₂.scr₁.wgt₁.thr₈₀₀</i>	.846	.772	.807	.776	.720	.743
PAGOD. <i>rdt₃.scr₁.wgt₁.thr₈₀₀</i>	.846	.782	.813	.782	.732	.753

4 POS Tag Experiments

This section presents the POS tag experiments made and the results obtained. From all possible parser tags (see Section 2.2), we just considered *n*, *prop*, *adj* and *v*. We tried all possible combinations of these tags.

For both datasets, we made experiments for the best baseline setup and three more obtained by reducing the number of features through new threshold values

– thr_{40} , thr_{50} , thr_{60} , thr_{70} for FSP and thr_{800} , thr_{900} , thr_{1000} , thr_{1100} for PAGOD, totalling a number of 60 experiments for each dataset.

PAGODS’ thresholds thr_{1000} and thr_{1100} have 1259 and 1160 features, respectively. Table 7 presents the per document averages of token (avg_{tok}) and type (avg_{typ}) for each POS tag (number and percent).

The proportion of verbs is similar in both datasets, but FSP has 2 times more percentage of proper nouns than PAGOD. This could be a reason for the different best baseline setups obtained in the previous section.

Table 7. POS experiments: averages (number and percent) of token and type

	FSP				PAGOD			
	# avg_{tok}	# avg_{typ}	% avg_{tok}	% avg_{typ}	# avg_{tok}	# avg_{typ}	% avg_{tok}	% avg_{typ}
<i>adj</i>	11	9	9.8%	10.8%	115	41	14.4%	16.4%
<i>nn</i>	52	37	46.4%	44.6%	423	110	52.8%	44.0%
<i>prop</i>	26	18	23.2%	21.7%	90	27	11.2%	10.8%
<i>vr̄b</i>	23	19	20.5%	22.9%	173	72	21.6%	28.8%

4.1 Results

We compared all 60 experiments along with the best setup obtained from the baseline experiments.

FSP Dataset. For all six measures, there were 6 ‘best’ experiments with no significant difference in the thr_{40} and thr_{50} thresholds. They were:

- for thr_{40} : *rdt3.scr2.wgt2* (baseline experiment), **nn+prop**, **nn+adj+prop** and **nn+adj+prop+vr̄b**
- for thr_{50} : **nn+prop+vr̄b** and **nn+adj+prop+vr̄b**.

From these, we can say that although we could not enhance the classifier using POS tags for selecting features, it was possible to reduce their number with no reduction on performance if we use nouns, proper nouns and verbs or these along with adjectives.

Table 8 shows the values of precision, recall and F_1 for the baseline experiment (529 features) and the best combinations of tags for the highest threshold value thr_{50} (397 features). Once again, bold faced figures have no significant difference with the best one obtained.

Table 8. POS FSP experiments: precision, recall and F_1 micro- and macro-averages for baseline and best setups

	μP	μR	μF_1	MP	MR	MF_1
baseline	.975	.975	.975	.976	.975	.975
nn+prop+vr̄b	.965	.965	.965	.965	.965	.965
nn+adj+prop+vr̄b	.967	.967	.967	.968	.967	.967

Table 9. POS PAGOD experiments: number belonging to the set of best results

	μP	μR	μF_1	MP	MR	MF_1
best	2	14	36	5	10	19
baseline	0	1	1	0	1	1
nn	0	0	4	1	0	0
nn+adj	0	0	4	1	0	0
nn+vr b	0	0	4	0	0	3
nn+prop	0	0	4	2	0	1
adj+prop	2	0	0	0	0	0
nn+adj+prop	0	4	4	1	2	4
nn+adj+vr b	0	2	4	0	1	2
nn+prop+vr b	0	3	4	0	2	4
adj+prop+vr b	0	0	3	0	0	0
nn+adj+prop+vr b	0	4	4	0	4	4

PAGOD Dataset. Table 9 presents, the number of experiments with no significant difference with respect to the best one and the distribution for each combination of POS tags experiments. The combinations `adj`, `prop`, `vrb`, `adj+vrb` and `prop+vrb` had no experiment in the set of best ones.

From the Table, we can say, again, that POS tags do not enhance the classifier but help feature selection by reducing the number of needed features. The best results were obtained using nouns combined with two of the other POS tags.

Table 10 shows the values of precision, recall and F_1 for the baseline experiment and those POS tags combinations for the highest threshold value (thr_{1100}) with results in the best set. We can say that `nn+adj+prop` and `nn+adj+prop+vrb` combinations are the best ones, since they have more best significant values.

Table 10. POS PAGOD experiments: precision, recall and F_1 micro- and macro-averages for best setups

	μP	μR	μF_1	MP	MR	MF_1
baseline	.846	.782	.813	.782	.732	.753
nn+adj+prop	.868	.773	.818	.791	.720	.746
nn+adj+vr b	.860	.765	.810	.783	.710	.738
nn+prop+vr b	.865	.770	.815	.788	.716	.743
nn+adj+prop+vr b	.860	.776	.816	.788	.723	.749

5 Conclusions

This paper presents a series of experiments aiming at comparing our proposal of pre-processing techniques based on linguistic information with usual methods adopted for pre-processing in text classification. We find in the literature other

alternative proposals for this pre-processing phase. Our approach differs from those since we propose single term selection based on different POS information.

From the results we were able to identify which setup is more suited for each dataset:

- for the newspaper articles, lemmatisation with mutual information scoring function and TFIDF weighting scheme, and
- for the juridical collection, lemmatisation with term frequency scoring function and normalised term frequency weighting scheme.

Selecting just some kind of tagged words allowed us to decrease the number of features (around 24%) without affecting learner's performance:

- for FSP, a decrease from 529 to 397 features was obtained using just the words tagged as **noun**, **proper noun** and **verb**.
- for PAGOD, a decrease from 1514 to 1160 was obtained using **noun**, **adjective** and **proper noun** tags.

As conclusion, the presented results support the claim that part-of-speech information can be, in fact, relevant in classification, allowing for a complexity reduction of the problem.

6 Future Work

Regarding future work, we intend to perform further tests on different collections and languages. It will be important to evaluate if these results are binded to the Portuguese language and/or the kind of dataset domain.

Aiming to develop better classifiers, we intend to address the document representation problem by trying more powerful representations than the bag-of-words allowing to use word order and syntactical and/or semantical information in document representation. To achieve this goal we plan to use other kind of kernels such as the string kernel (see, for example, [14]).

References

1. A. Aizawa. Linguistic techniques to improve the performance of automatic text categorization. In *NLPRS*, pages 307–314, 2001.
2. E. Bick. *The Parsing System PALAVRAS – Automatic Grammatical Analysis of Portuguese in a Constraint Grammar Framework*. Aarhus University Press, 2000.
3. E. Bick. A constraint grammar based question answering system for portuguese. In *Proceedings of the 11th Portuguese Conference of Artificial Intelligence – EPIA'03*, pages 414–418. LNAI Springer Verlag, 2003.
4. C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.
5. Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley Series in Telecommunication. John Wiley and Sons, Inc, New York, 1991.

6. T. Gonçalves and P. Quaresma. A preliminary approach to the multilabel classification problem of Portuguese juridical documents. In F. Moura-Pires and S. Abreu, editors, *11th Portuguese Conference on Artificial Intelligence, EPIA 2003*, LNAI 2902, pages 435–444, Évora, Portugal, December 2003. Springer-Verlag.
7. T. Gonçalves and P. Quaresma. Evaluating preprocessing techniques in a text classification problem. In *ENIA'05: Encontro Nacional de Inteligência Artificial*, São Leopoldo, RS, Brasil, August 2005. (to appear).
8. T. Joachims. *Learning to Classify Text Using Support Vector Machines*. Kluwer academic Publishers, 2002.
9. J.G. Lopes, N.C. Marques, and V.J. Rocio. Polaris: POrtuguese lexicon acquisition and retrieval interactive system. In *The Practical Applications of Prolog*, page 665. Royal Society of Arts, 1994.
10. D. Mladenić and M. Grobelnik. Feature selection for unbalanced class distribution and naïve Bayes. In *Proceedings of the 16th International Conference on Machine Learning – ICML'99*, pages 258–267, 1999.
11. A. Moschitti and Roberto Basili. Complex linguistic features for text classification: A comprehensive study. In *ECIR*, volume 2997. Proceedings Editors: Sharon McDonald, John Tait, 2004.
12. G. Salton and M. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.
13. H. Schütze, D. Hull, and J. Pedersen. A comparison of classifiers and document representations for the routing problem. In *Proceedings of the 18th International Conference on Research and Development in Information Retrieval – SIGIR'95*, pages 229–237, Seattle, WA, 1995.
14. J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.
15. C.F. Silva, R. Vieira, F.S. Osorio, and P. Quaresma. Mining linguistically interpreted texts. In *Proceedings of the 5th International Workshop on Linguistically Interpreted Corpora*, Geneva, Switzerland, August 2004.
16. R. Tong and L.A. Appelbaum. Machine learning for knowledge-based document routing. In Harman, editor, *Proceedings of the 2nd Text Retrieval Conference*, 1994.
17. V. Vapnik. *Statistical learning theory*. Wiley, NY, 1998.
18. I. Witten and E. Frank. *Data Mining: Practical machine learning tools with Java implementations*. Morgan Kaufmann, 1999.

Improving kNN Text Categorization by Removing Outliers from Training Set*

Kwangcheol Shin, Ajith Abraham, and Sang Yong Han**

School of Computer Science and Engineering, Chung-Ang University,
221, Heukseok-dong, Dongjak-gu, Seoul 156-756, Korea
kcshin@archi.cse.cau.ac.kr,
ajith.abraham@ieee.org, hansy@cau.ac.kr

Abstract. We show that excluding outliers from the training data significantly improves kNN classifier, which in this case performs about 10% better than the best know method—Centroid-based classifier. Outliers are the elements whose similarity to the centroid of the corresponding category is below a threshold.

1 Introduction

Since late 1990s, the explosive growth of Internet resulted in a huge quantity of documents available on-line. Technologies for efficient management of these documents are being developed continually. One of representative tasks for efficient document management is text categorization, called also classification: given a set of training examples assigned each one to some categories, to assign new documents to a suitable category.

A well-known text categorization method is kNN [1]; other popular methods are Naive Bayesian [3], C4.5 [4], and SVM [5]. Han and Karypis [2] proposed the Centroid-based classifier and showed that it gives better results than other known methods.

In this paper we show that removing outliers from the training categories significantly improves the classification results obtained with kNN method. Our experiments show that the new method gives better results than the Centroid-based classifier.

2 Related Work

Document representation. In both categorization techniques considered below, documents are represented as keyword vectors according to the standard vector space model with *tf-idf* term weighting [6, 7]. Namely, let the document collection contains in total N different keywords. A document d is represented as an N -dimensional vector of term weight t with coordinates

* Work 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).

** Corresponding author.

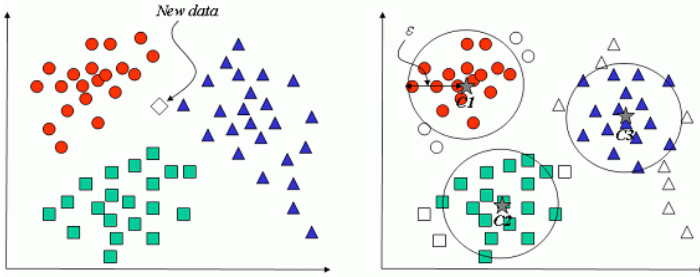


Fig. 1. Example of classification

$$w_{td} = \frac{f_{td}}{\max_t f_{td}} \log \frac{n_t}{N}, \tag{1}$$

where f_{td} is the frequency of the term t in the document d and n_t is the number of the documents where the term t occurs. The similarity between two vectors d_i and d_j is measured as the cosine of the angle between them:

$$s(d_i, d_j) = \cos(\theta(x_i, x_j)) = \frac{d_i^T d_j}{\|d_i\| \|d_j\|}, \tag{2}$$

where θ is the angle between the two vectors and $\|d\|$ is the length of the vector.

kNN classifier [1]. For a new data item, k most similar elements of the training data set are determined, and the category is chosen to which a greater number of elements among those k ones belong; see Figure 1, left.

Centroid-based classifier [2]. Given a set S_i of documents—the i -th training category, its center is defined as its average vector:

$$\bar{C}_i = \frac{1}{|S_i|} \sum_{\vec{d} \in S_i} \vec{d} \tag{3}$$

where $|S_i|$ is the number of documents in the category. For a new data item the category is chosen that maximizes the similarity between the new item and the centers of each category. This was reported as the best known classifier so far [2].

3 Proposed Method

We observed that the training data items that are far away from the center of its training category reduce the accuracy of classification. Our hypothesis is that those items represent noise and not useful training examples and thus decrease the classification accuracy. Thus we exclude them from consideration; see Figure 1, right. Specifically, at the training stage we calculate the center C_i of each category S_i using (2). Then we form new categories by discarding outliers:

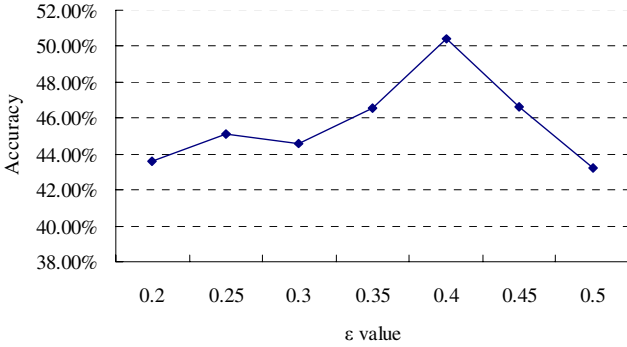


Fig. 2. Different accuracy according to ϵ value at using 80% of test dataset

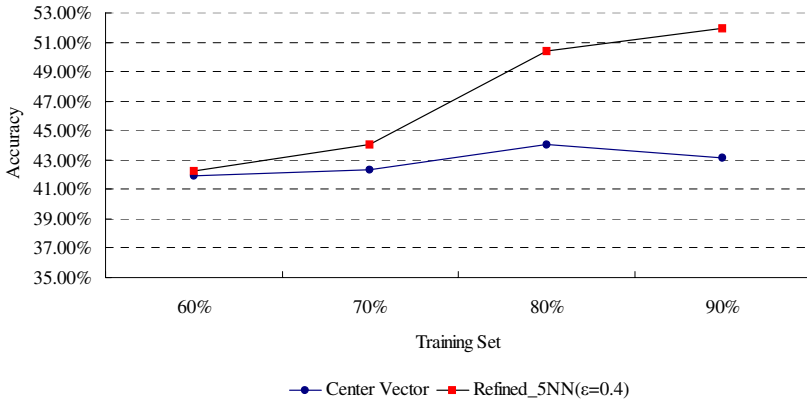


Fig. 3. Test Results

$$S'_i = \{d \in S_i : \text{Sim}(d, \vec{C}_i) > \epsilon\}; \quad (4)$$

in the next section we discuss the choice of the threshold ϵ . Finally, we apply the kNN classifier using these modified categories.

4 Experimental Results

We used the 20-newsgroup dataset to evaluate the performance of the proposed method. The dataset consists of 20 classes of roughly 1000 documents each. We used MC [8] program to build the document vectors. We implemented our modified kNN method with $k = 5$ and compared it with the Centroid-based classification. As Figure 2 shows, our method provides the best performance with $\epsilon \approx 0.4$. Figure 3 shows how the classification accuracy depends on the percentage of training dataset over total dataset. We obtain 9.93% improvement over the original Centroid-based classification.

5 Conclusion

We have presented an improved kNN classifier, combining it with the idea of the Centroid-based method. The improvement consists in removing outliers from the categories of the training dataset. Our method shows almost 10% better accuracy than the original Centroid-based classifier, which was reported in [2] as the most accurate text categorization method. In the future, automatic choice of the threshold value ε is to be considered.

References

1. W. W. Cohen and H. Hirsh. Joins that generalize: Text Classification using WHIRL. In Proc. of the Fourth Int'l Conference on Knowledge Discovery and Data Mining, 1998.
2. E. Han and G. Karypis. Centroid-Based Document Classification: Analysis & Experimental Results. *Principles of Data Mining and Knowledge Discovery*, p. 424–431, 2000.
3. D. Lewis and W. Gale. A sequential algorithm for training text classifiers. SIGIR-94, 1994.
4. J. Ross Quinlan. C4.5: Programs for Machine Learning. Morgan Kaufmann, 1993.
5. V. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.
6. G. Salton and M. J. McGill, *Introduction to Modern Retrieval*. McGraw-Hill, 1983.
7. R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley, 1999.
8. Dhillon I. S., Fan J., and Guan Y. Efficient Clustering of Very Large Document Collections. *Data Mining for Scientific and Engineering Applications*, Kluwer, 2001.

Writing for Language-Impaired Readers

Aurélien Max

LIMSI-CNRS & Université Paris Sud,
Bâtiment 508, F-91405 Orsay Cedex, France
`aurelien.max@limsi.fr`

Abstract. This paper advocates an approach whereby the needs of language-impaired readers are taken into account at the stage of text authoring by means of NLP integration. In our proposed system architecture, a simplification module produces candidate simplified rephrasings that the author of the text can accept or edit. This article describes the syntactic simplification module which has been partly implemented. We believe the proposed approach constitutes a framework for the more general task of authoring NLP-enriched documents obtained through validations from the author.

1 Introduction

The goal of NLP, as it seems, is mainly to do some processing on existing text. Another domain of application that we believe has a great potential is the use of NLP during text creation, where it can help authors write better documents in a more efficient way. A lot of the difficulties when processing real-life documents arise from the inherent complexity of natural language, which requires word-sense and syntactic structure disambiguation, to name just two. In fact, rule-based and statistical NLP systems are rather good at finding hypotheses, but they often fail when it comes to ranking them and finding the appropriate solution in context.

Some cases can certainly justify the extra cost of annotating the text with the result of the correct analysis, thus permitting much better results on NLP tasks. This concept has already been investigated, for example in the Dialogue-based Machine Translation paradigm [1] whereby a monolingual writer answers ambiguity questions. This process yields a disambiguated analysis for each sentence that is then sent to a machine translation engine.

The kinds of annotation that can be obtained through interaction can be of very different natures. One kind is a transformation of the initial text: for example, annotations at the paragraph level can be assembled to constitute a summary. Transformations at the sentence level include paraphrasing and its different uses. Among them, text simplification has attracted significant interest in the past years [4,3,5]. The most notable application of text simplification has been as an assistive technology for people suffering from aphasia, a loss of language that can result in severe comprehension disorders. It is very unlikely

that a text transformation system could produce a coherent text conveying the closest possible meaning of an original text while significantly simplifying its complexity. Except in a few well-known cases, whenever Machine Translation is used, translations are revised (or post-edited) by professional translators to ensure that the resulting text is an accurate translation. The same requirement holds for text simplification. But because the author, who possesses all the required knowledge to express his ideas, has already performed the task of writing text to express them, that same author seems to be the appropriate person to validate simplifications produced by an automatic system. This is why we advocate an integration of NLP components in typical word processors with the aim of helping authors create high quality annotations.

In the next section we present an architecture for an interactive text simplification system, with a particular focus on the syntactic simplification module.

2 Interactive Text Simplification System

Figure 1 shows the different elements that constitute our system. We have chosen a rule-based approach to text simplification as in other works [4,3].

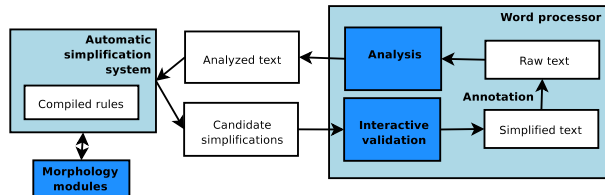


Fig. 1. General architecture of the interactive simplification system

We proposed a formalism that a linguist could use to describe rewriting rules. The rule on figure 2 rewrites passive sentences with overt agent as active sentences¹. The development of such a ruleset can be incremental, allowing to control the impact of new additions on performance. When no rule applies, an input sentence is left untouched and therefore retains its initial complexity.

The level of description of the rules depends on the output format of the parser used. An important constraint is that the information required for the generation of the simplified sentences must be derived from the input sentences. Rewriting rules allow to reuse elements in simplified sentences, or create new elements based on conditions on the rules. Access to a morphology module is required when different surface forms of words are needed (e.g. when activising a sentence, the verb from the main clause has to agree with the agent that was

¹ Finding the correct agents and patients in semantically reversible passive sentences can be very difficult for aphasic subjects [2].

previously in the *by* prepositional phrase: *The Prime Minister is named by the President* → *The President names the Prime Minister*).

Templates describing syntactic structures are incrementally searched in the output of the parser, and when matches are found and conditions are fulfilled, the simplified structures are produced (with the possibility of inserting new sentences to split up complex structures). Rules are recursively applied on the simplified sentences as long as matches are found, the assumption being that the more rules are applied, the more syntactic complexity is “broken”.

```

define Activise passive sentences with overt agent

if [be, InflBe] is analyzeVerb(TagBe, [Be]);
  ?OptAdvs contains only advp rb; ?OptPart contains only prt;
rewrite [s [
  ?Opt1 [np-Index NPTheme] ?Opt2
  [vp [
    [TagBe [Be]]
    [vp [
      ?OptAdvs [vbn Verb] ?OptPart
      [np [[none [Trace-Index]]] ?Opt3
      [pp [[prep [by]]
        [np/lgs NPAgent]]]
      ?Opt4]]]]
    ?Opt5]]]
as [s [
  ?Opt1 [np NPAgent] ?Opt2
  [vp [[SurfaceTag [SurfaceVerb]]
    ?OptPart [np NPTheme] ?Opt3 ?Opt4]]
  ?Opt5]]]
where [Number, Person] is number(NPAgent);
  [BaseForm, Infl] is analyzeVerb(vbn, Verb);
  [SurfaceVerb, SurfaceTag] is generateVerb(BaseForm, InflBe, Number, Person);

```

Fig. 2. Simplification rule for activizing sentences with overt agent

A non-deterministic search in the space of possible rewritings yields a collection of competing rewritings for each sentence. Because we want the writer to choose (and possibly edit) the best candidate, we do not need to attempt to identify it automatically, but we can help the writer by finding a good ordering. Such ordering should ideally take into account a score of syntactic complexity, as well as an indication of the shift in meaning incurred by the rewriting. As the degree of simplification depends on the number of rules applied to obtain a given result, we have chosen to take this number into account for ranking candidate simplifications, as well as the average length of sentence (borrowed from *readability* measures).

The simplification module in the user word processor functions in non-preemptive mode so that the user is not interrupted when typing. At his request, a given sentence is analyzed, and candidate simplifications are displayed by decreasing score. The one that is chosen can then be edited, in order to guarantee that the initial meaning is preserved as much as possible.²

² We also plan to investigate different types of user interfaces for the integration of this module within a word processor, for example using interlines to display the corresponding simplified text.

3 Perspectives

Our initial experiments with a ruleset designed for aphasic readers based on results from psycholinguistics [2] have shown that sentence transformation by rewriting rules almost always alters meaning.³ However, we believe that involving the author in this task is certainly the fairest way to obtain a good rephrasing that is at best easier to understand than the original text, and at least never more difficult.

The following example shows a possible rewriting obtained by applying several rules of our ruleset to create a separate sentence from a noun phrase in apposition and to split up conjoined clauses: *S., a deputy minister, launched a no-smoking week and urged other schools to ban on-campus smoking.* → *S. is a deputy minister. S. launched a no-smoking week. S. urged other schools to ban on-campus smoking.*

By implementing several optimization techniques (for example, pre-analyzing sentences while the user is working on other sentences, using simplification memories, etc.), we hope to provide a practical tool that can efficiently help the author, so that choosing a rephrasing among the system outputs and editing it would take significantly less time than writing a simplified version from scratch. More work is needed on this, and we look forward to experimenting these ideas.

Finally, we believe it is important to have at least some authors agree on the importance of this extra work in their authoring task, and we hope that the integration of NLP will gradually lead to more *linguistics-aware* authoring tools.

References

1. Boitet, C. and Blanchon, H. Multilingual Dialogue-Based MT for monolingual authors: the LIDIA project and a first mockup. In *Machine Translation*. Vol. 9(2) : pp 99-132 (1995).
2. Caplan, D. *Neurolinguistics and linguistic aphasiology*. Cambridge University Press, Cambridge, United Kingdom (1987)
3. Carroll, J., Minnen, G., Canning, Y., Devlin, S. and Tait, J. Practical simplification of English newspaper text to assist aphasic readers. *Proceedings of AAAI-98 workshop on integrating artificial intelligence and assistive technology*, Madison, USA (1998)
4. Chandrasekar, R., Doran, C. and Srinivas, B. Motivations and methods for text simplification. *Proceedings of COLING'96*, Copenhagen, Denmark (1996)
5. Siddhartan, A. *Syntactic simplification and text cohesion*. PhD thesis, University of Cambridge (2003)

³ For the purpose of text simplification for readers with language impairments, we intend to extend our work to cover lexical simplification as well as anaphora resolution [5].

Document Copy Detection System Based on Plagiarism Patterns*

NamOh Kang and SangYong Han**

School of Computer Science and Engineering,
ChungAng University, Seoul, Korea
kang@archi.cse.cau.ac.kr, hansy@cau.ac.kr

Abstract. Document copy detection is a very important tool for protecting author's copyright. We present a document copy detection system that calculates the similarity between documents based on plagiarism patterns. Experiments were performed using CISI document collection and show that the proposed system produces more precise results than existing systems.

1 Introduction

For protecting author's copyright, many kinds of intellectual property protection techniques have been introduced; copy prevention, signature and content based copy detection, etc. Copy protection and signature-based copy detection can be very useful to prevent or detect copying of a whole document. However, these techniques have some drawbacks that they make it difficult for users to share information and can not prevent copying of the document in partial parts [1].

Huge amount of digital documents is made public day to day in Internet. Most of the documents are not supported by either copy prevention technique or signature based copy detection technique. This situation increases the necessity in content based copy detection techniques. So far, many document copy detection (DCD) systems based on content based copy detection technique have been introduced, for example COPS [2], SCAM [1], CHECK [3], etc. However, most DCD systems mainly focus on checking the possibility of copy between original documents and a query document. They do not give any evidence of plagiaristic sources to user. In this paper, we propose a DCD system that provides evidence of plagiarism style to the user.

* 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).

** Corresponding author.

2 Comparing Unit and Overlap Measure Function

DCD system divides documents efficiently in comparing unit (chunking unit) for checking the possibility of copy. In this paper, we select the comparing unit as a sentence because the similarity comparison between sentences becomes a good norm to calculate the local similarity and can provide plagiarism pattern information between them.

Overlap measure function is used to get copy information of the comparing units extracted from documents. Traditionally, many DCD systems use vector space model or cosine similarity model. It is no problem to calculate the similarity between two objects but it is not enough to calculate the degree of copy. In this research, we suggest the overlap measure function which can quantify the overlap between comparing units and give information about plagiarism.

Let S_o come from an original document and S_c from a query document. The $Sim(S_o, S_c)$ can be calculated as follows.

$$\begin{aligned}
 S_o &= \{w_1, w_2, w_3, \dots, w_n\} & S_c &= \{w_1, w_2, w_3, \dots, w_m\} \\
 Comm(S_o, S_c) &= S_o \cap S_c & Diff(S_o, S_c) &= S_o - S_c \\
 Syn(w) &= \{\text{The synonym of } w\} \\
 SynWord(S_o, S_c) &= \{w_i \mid w_i \in Diff(S_c, S_o) \cap Syn(w_i) \in S_o\} \\
 WordOverlap(S_o, S_c) &= \frac{|S_o|}{|Comm(S_o, S_c)| + 0.5 \times |SynWord(S_o, S_c)|} \\
 SizeOverlap(S_o, S_c) &= \sqrt{|Diff(S_o, S_c)| + |Diff(S_c, S_o)|} \\
 Sim(S_o, S_c) &= |S_o| \times \left(\frac{1}{WordOverlap(S_o, S_c) + SizeOverlap(S_o, S_c)} \right)
 \end{aligned}$$

Calculation of $Sim(S_o, S_c)$ gives not only similarity between S_o and S_c but also the plagiarism information. The following table 1 shows how to decide plagiarism patterns.

Table 1. Plagiarism patterns and their decision parameters

Plagiarism pattern	Decision parameters
Sentence copy exactly	$WordOverLap(S_o, S_c) = 1, SizeOverlap(S_o, S_c) = 0$
Word insertion	$SizeOverlap(S_o, S_c) \neq 0, Diff(S_o, S_c) > 1$
Word remove	$SizeOverlap(S_o, S_c) \neq 0, Diff(S_c, S_o) > 1$
Changing word	$1 < WordOverLap(S_o, S_c) < \infty, SizeOverlap(S_o, S_c) = 0$
Changing sentence	$WordOverLap(S_o, S_c) = 1, SizeOverlap(S_o, S_c) = 0$

3 System Design and Algorithm

All original documents are stored in document data base. When the query document is input, the system divides the query document and the original documents into comparing units - sentences. The divided sentences are then used to calculate the overlap and the plagiarism information in local_similarity_extractor by using the overlap measure function defined in section 2. The extracted information is used to calculate the degree of copy in original documents from each other, and the ordered information is supplied to user. The algorithm of the proposed system is followed.

Algorithm

Input:

$Document_DB = \{D_1, D_2, D_3, \dots, D_n\}$ and each

$D_i = \{S_{i1}, S_{i2}, S_{i3}, \dots, S_{im}\}$

$QueryDocument = \{QS_1, QS_2, QS_3, \dots, QS_t\}$

Output:

Decreasing ordered document list in document similarity value

for $i = 1$ to n

 for $j = 1$ to t

 localsimilarity[1..j] = 0

 for $k = 1$ to m

 if $|Comm(S_{ik}, QS_j)| \geq \frac{|S_{ik}|}{2}$ then

 localsimilarity[j] = max {localsimilarity[j],
 $Sim(S_{ik}, QS_j)$ }

 documentsimilarity[i] = $\sum_j localsimilarity[j]$

return sort(documentsimilarity)

4 Experiment and Discussion

We generated the test document set from CISI as follow.

1. 11 relevant documents related to a specific query are selected from CISI document set.
2. One document is selected as an original document. The others 10 documents are selected as candidate document for plagiarism.
3. A partial part extracted from the original document is transformed (exact copy, changing synonym, changing sentence structure) and it is inserted into the candidate documents for plagiarism to make plagiarized document.
4. The plagiarized documents are returned into the CISI document set. Selected original document is removed from the CISI document set and becomes the query document.

For comparison with the proposed system (P_System), we made DCD system based on word similarity of document (WD_System) and of sentence (WS_System). For performance checking, we chose R-precision as the evaluation norm and R is set to 10.

Table 2. Copy detection test (R = 10)

	WD_System	WS_System	P_System
Exact copy	2	6	8
Synonym	2	6	8
Changing structure	1	4	4

The experimental results show that the proposed P_System produces more precise results in exact copy and changing synonym. It shows that in the proposed method overlap measure function is more useful to check the copy of document than the normalized comparison value like cosine similarity. And if user decides the copy of document with the consideration of plagiarism pattern information produced in comparison, the more precise decision could be made.

References

1. Shivakumar, N. and Garcia-Monlina, H. SCAM: A Copy Detection Mechanisms for Digital Documents. In Proceedings of International Conference on Theory and Practice of Digital Libraries, Austin, Texas. June 1995.
2. Brin, S., Davis, J., and Garcia-Molina, H. Copy Detection Mechanisms for Digital Documents. In Proceedings of ACM SIGMOD Annual Conference, San Jose, CA, 1995
3. Si, A., Leong, H., and Lau, R. CHECK: A Document Plagiarism Detection System. In Proceedings of ACM Symposium for Applied Computing, pp. 70-77, Feb 1997.
4. Bao Jun-Peng, Shen Jun-Yi, Liu Xiao-Dong, Liu Hai-Yan, Zhang Xiao-Di. Document Copy Detection Based On Kernel Method. In Proceedings of 2003 International Conference on Natural Language Processing and Knowledge Engineering.
5. Karen Fullam, J. Park. Improvements for Scalable and Accurate Plagiarism Detection in Digital Documents. 2002

Regional vs. Global Robust Spelling Correction*

Manuel Vilares Ferro, Juan Otero Pombo, and Víctor Manuel Darriba Bilbao

Department of Computer Science, University of Vigo,
Campus As Lagoas s/n, 32004 Orense, Spain
{vilares, jop, darriba}@uvigo.es

Abstract. We explore the practical viability of a regional architecture to deal with robust spelling correction, a process including both unknown sequences recognition and spelling correction. Our goal is to reconcile these techniques from both the topological and the operational point of view. In contrast to the global strategy of most spelling correction algorithms, and local ones associated with the completion of unknown sequences, our proposal seems to provide an unified framework allowing us to maintain the advantages in each case, and avoid the drawbacks.

1 Introduction

In describing human performance in spelling correction, as compared to machine performance, we should try to take into account both the computational efficiency and the quality achieved in order to equal, or even do better than humans. This translates into a trade-off between the study of the often complex linguistic phenomena involved and the efficiency of the operational mechanisms available for implementation. In order to attain this goal, simple proposals can be sufficient to overcome most limits to providing an efficient strategy, even in the case of interactive applications. In fact, most approaches are oriented to improving first-guess accuracy [1] and/or to considering filter-based solutions to speed up the process [8]. So, system developers expect to reduce the time needed to ensure an adequate coverage of the problem, before taking into account more sophisticated linguistic information.

The state of the art techniques mainly focus on approximate string matching proposals, often firstly oriented to searching [2], although they can be easily adapted to robust spelling correction tasks [4]. Essentially, these algorithms apply a metric [5] to measure the minimum number of unit operations necessary to convert one string into another, sometimes embedding this task in the recognizer [10] in order to improve the computational efficiency. In this context, we identify a set of objective parameters in order to evaluate different approaches and algorithms in dealing with robust spelling correction.

* This research has been partially supported by the Spanish Government under projects TIN2004-07246-C03-01, and the Autonomous Government of Galicia under projects PGIDIT05PXIC30501PN, PGIDIT05SIN059E, PGIDIT05SIN044E, PGIDIT04SIN065E and PGIDIT03SIN30501PR.

Our interest, in this paper, centers only around morphological aspects, including the treatment of unknown sequences and taking into account both quantitative and qualitative criteria. The treatment of syntactic or semantic information is a part of our future work. Given that we consider practical systems, we focus on the most efficient global and regional proposals embedded in the recognition process, to prove that regional methods seem to be promised in the field of robust spelling correction.

Succintly, global algorithms [9,10] extend the repair region to the entire string, expending equal correction effort on all parts of the word and providing the best repair quality with a high computational cost. Regional algorithms [12] try to determine how far to validate each repair, taking into account that a short validation may fail to gather sufficient information and in a long one most of the effort can be wasted. The goal of a regional method is to obtain a repair quality comparable to that attained by a global one, with a significant saving in time and space.

2 The Recognizer

Our aim is to parse a word $w_{1..n} = w_1 \dots w_n$ according to a regular grammar $\mathcal{G} = (N, \Sigma, P, S)$. 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 [7] generated by GALENA [6]. 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 $|q.a| \leq 1, \forall q \in \mathcal{Q}, a \in \Sigma$. The notation is transitive, $q.w_{1..n}$ denotes the state $(\overset{n-2}{q.w_1} \overset{n-2}{.} w_n)$. 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 $\alpha = \{q_1, \dots, q_n\}$, such that $\forall i \in \{1, \dots, n-1\}, \exists a_i \in \Sigma, q_i.a_i = q_{i+1}$.

We also apply a minimization process [3]. In this sense, we say that 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 with 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 $q \in \mathcal{Q}$.

Our description uses *parsing schemata* [11], a triple $\langle \mathcal{I}, \mathcal{H}, \mathcal{D} \rangle$, with $\mathcal{H} = \{[a, i], a = w_i\}$ an initial set of items called *hypothesis* that encodes the word to be recognized¹, and \mathcal{D} a set of *deduction steps* that allow items to be derived

¹ A word $w_{1..n} \in \Sigma^+, n \geq 1$ is represented by $\{[w_1, 1], [w_2, 2], \dots, [w_n, n]\}$.

from previous ones. These are of the form $\{\eta_1, \dots, \eta_k \vdash \xi/\mathcal{C}\}$, meaning that if all antecedents η_i are present and the conditions \mathcal{C} are satisfied, then the consequent ξ is generated. In our case, $\mathcal{D} = \mathcal{D}^{\text{Init}} \cup \mathcal{D}^{\text{Shift}}$, where:

$$\mathcal{D}^{\text{Init}} = \{\vdash [q_0, 1]\} \quad \mathcal{D}^{\text{Shift}} = \{[p, i] \vdash [q, i + 1] / \exists [a, i] \in \mathcal{H}, q = p.a\}$$

We associate a set of items S_p^w , called *itemset*, to each $p \in \mathcal{Q}$; and apply these deduction steps until no new item is generated. The word is recognized iff a *final item* $[q_f, n + 1]$, $q_f \in \mathcal{Q}_f$ has been generated. We can assume 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 which become the new initial and final states, and are linked to the original ones through empty transitions, our only concession to the notion of minimal FA.

3 An Unified Robust Framework

We define a *singularity* in a word to mean the difference between what was intended and what actually appears, and we call *point of singularity* the position at which that difference occurs. In a robust interpretation, the singularity can be a spelling *error* or a *distortion* in the acquisition process from the text. Whichever the case, a *repair* should be understood as a modification allowing us both to recover the recognition and to avoid cascaded errors, that is, errors precipitated by a previous erroneous repair diagnosis. In order to compute the *edit distance* [9], we extend the structure of items, as a pair $[p, i]$, with an error counter e ; resulting in a new structure $[p, i, e]$.

The spelling correction strategies compared revolve around two contrasting alternatives and are concerned with the natural extension of pure spelling correction algorithms [10, 12], which are embedded in the recognizer in order to increase the computational efficiency. We describe them using parsing schemata, which allows us to establish an unified framework as well as to justify the validity of the results shown. To emphasize this aspect, we shall first decompose, as far as possible, this description to later introduce the particular conditions associated with each case.

3.1 On Spelling Incomplete Strings

We extend the input alphabet with two new symbols. The first, “?”, stands for one unknown character; and “*” stands for an unknown sequence of input characters. In dealing with sequences of unknown characters, different paths in the FA can resolve the same “*” symbol. Although this could be useful for subsequent processing, an uncontrolled over-generation is not of interest in most cases. We solve this by using the error counter in items to tabulate the number of characters used to rebuild the word, and later apply the principle of optimization. Once the recognizer detects that the next symbol to be read from the input string denotes an unknown sequence, we apply the set, $\mathcal{D}_{\text{incomplete}}$, of steps:

$$\begin{aligned} \mathcal{D}_{incomplete}^{Shift} &= \{[p, i, e] \vdash [q, i + 1, e + I(a)] / \exists [?, i] \in \mathcal{H}, q = p.a\} \\ \mathcal{D}_{incomplete}^{Loop_shift} &= \{[p, i, e] \vdash [q, i, e + I(a)] / \exists [*, i] \in \mathcal{H}, q = p.a, \nexists q.w_{i+1}\} \\ \mathcal{D}_{incomplete}^{Loop_shift_end} &= \{[p, i, e] \vdash [q, i + 1, e + I(a)] / \exists [*, i] \in \mathcal{H}, q = p.a, \exists q.w_{i+1}\} \end{aligned}$$

where $I(a)$ is the insertion cost for $a \in \Sigma$, and we have to adapt the previous deduction steps to deal with counters:

$$\mathcal{D}^{Init} = \{ \vdash [q_0, 1, 0] \} \quad \mathcal{D}^{Shift} = \{ [p, i, e] \vdash [q, i + 1, e] / \exists [a, i] \in \mathcal{H}, q = p.a \}$$

Intuitively, $\mathcal{D}_{incomplete}^{Shift}$ applies any shift transition independently of the current lookahead available, provided that this transition is applicable with respect to the FA configuration and that the next input symbol is an unknown character. In relation to $\mathcal{D}_{incomplete}^{Loop_shift}$, it simulates shift actions on items corresponding to FA configurations for which the next input symbol denotes an unknown sequence of characters, when no standard shift action links up to the right-context. Given that in this latter case new items are created in the same itemset, these transitions may be applied any number of times to the same computation thread, without scanning the input string. These steps are applied until a recognition branch links up to the right-context by using a shift action, resuming the standard recognition mode, as it is described by $\mathcal{D}_{incomplete}^{Loop_shift_end}$. So, we extend, in a natural manner and faithfully safeguarding the operational kernel, both original spelling correction algorithms [10, 12] to repair incomplete strings.

3.2 On Spelling Correction

When we deal with an error, we define as *point of detection* the position at which we attempt to find its origin. Associated with the point of error (resp. detection) w_j (resp. w_i), we consider the corresponding *error* (resp. *detection*) *item* $[q, j, -]$ (resp. $[p, i, -]$). In this case, both original spelling correction proposals [10, 12] apply the same set of error hypothesis, that we can define as follows:

$$\begin{aligned} \mathcal{D}_{error}^{Insert} &= \{[r, l, e] \vdash [r, l + 1, e + I(a)], \mathcal{C}^{Insert}\} \\ \mathcal{D}_{error}^{Delete} &= \{[r, l, e] \vdash [s, l, e + D(w_l)], \mathcal{C}^{Delete}\} \\ \mathcal{D}_{error}^{Replace} &= \{[r, l, e] \vdash [s, l + 1, e + R(w_l, a)], \mathcal{C}^{Replace}\} \\ \mathcal{D}_{error}^{Transpose} &= \{[r, l, e] \vdash [s, l + 2, e + T(w_l, w_{l+1})], \mathcal{C}^{Transpose}\} \end{aligned}$$

where the conditions \mathcal{C}^{Insert} , \mathcal{C}^{Delete} , $\mathcal{C}^{Replace}$ and $\mathcal{C}^{Transpose}$ depend on each particular strategy and will be detailed later. On the other hand, since we want to filter out undesirable repairs, we introduce criteria to select those with minimal cost. So, for each $a, b \in \Sigma$ we assume the cost of insertion is $I(a)$; the cost of deletion is $D(a)$, the cost of replacement is $R(a, b)$, and the cost of transposition is $T(a, b)$. In order to take the edit distance as the error metric for measuring the quality of a repair, it is sufficient to consider discrete costs $I(a) = D(a) = 1, \forall a \in \Sigma$ and $R(a, b) = T(a, b) = 1, \forall a, b \in \Sigma, a \neq b$.

4 A Regional Approach

Our goal now is to integrate the regional spelling corrector described in [12], in order to obtain the first robust technique we are going to consider. This will allow us to avoid cascaded errors reducing the impact of the repair in the input.

4.1 Spelling Correction

Given that the aim is essentially practical, we first succinctly remember the concepts associated to the notion of *regional repair* introduced in [12], where the reader can find formal definitions and proofs.

Working on acyclic FA, we define an order relation $p < q$, with $p, q \in \mathcal{Q}$ iff there is a path in the FA from p to q . A pair of states (p, q) is a *region* in the FA when it defines a sub-automaton with initial state p and final state q , taking all states and edges occurring in any path from p to q . So, we say that a state r is in the region defined by the pair (p, q) , denoted by \mathcal{R}_p^q , iff there exists a path α in \mathcal{R}_p^q , such that $r \in \alpha$. Given $r \in \mathcal{Q}$, it can be proved that there is only one *minimal region*² in the FA containing it.

To begin with, we assume that we are dealing with the first error. Given a *point of error* w_j , the associated *point of detection* is the initial state of the minimal region, $\mathcal{M}(w_j) = \mathcal{R}_p^q$, containing w_j . We apply, from the detection item, the steps $\mathcal{D}_{\text{error}} = \mathcal{D}_{\text{error}}^{\text{Shift}} \cup \mathcal{D}_{\text{error}}^{\text{Insert}} \cup \mathcal{D}_{\text{error}}^{\text{Delete}} \cup \mathcal{D}_{\text{error}}^{\text{Replace}} \cup \mathcal{D}_{\text{error}}^{\text{Transpose}}$, with:

$$\begin{aligned} \mathcal{C}^{\text{Insert}} &= \emptyset, & \mathcal{C}^{\text{Delete}} &= \{\mathcal{M}(q_0.w_{1..j}) = \mathcal{R}_{q_s}^{q_d}, r.w_l = q \in \mathcal{R}_{q_s}^{q_d} \text{ or } s = q_d\} \\ \mathcal{C}^{\text{Replace}} &= \{\mathcal{M}(q_0.w_{1..j}) = \mathcal{R}_{q_s}^{q_d}, r.a = s \in \mathcal{R}_{q_s}^{q_d} \text{ or } s = q_d\} \\ \mathcal{C}^{\text{Transpose}} &= \{\mathcal{M}(q_0.w_{1..j}) = \mathcal{R}_{q_s}^{q_d}, r.w_{l+1}.w_l = s \in \mathcal{R}_{q_s}^{q_d} \text{ or } s = q_d\} \end{aligned}$$

where $w_{1..j}$ is the prefix for the point of error. In any case, the error hypothesis apply on transitions in the repair region. The process continues until a repair covers that region, accepting a character in the remaining string. When no repair is possible, the process extends to the next region, taking the final state of the previous one as the new point of error. We apply a principle of optimization, saving only those items with minimal counters.

When the current error is not the first one, we can modify a previous repair in order to avoid cascaded errors, by adding the cost of the new error hypothesis to profit from the experience gained from previous ones. This allows us to get a quality close to global methods [12].

4.2 Robust Spelling

We are now ready to introduce robust construction. We must now guarantee the capacity to recover the recognizer from an unexpected situation derived either from gaps in the recognizer or from errors. To deal with this, it is sufficient to combine the rules previously introduced. More precisely, we have that the new set of deduction steps, $\mathcal{D}_{\text{robust}}$, is given by:

² That is, the smallest region containing the state r .

$$\mathcal{D}_{\text{robust}} = \mathcal{D}^{\text{Init}} \cup \mathcal{D}^{\text{Shift}} \cup \mathcal{D}^{\text{Insert}}_{\text{error}} \cup \mathcal{D}^{\text{Delete}}_{\text{error}} \cup \mathcal{D}^{\text{Replace}}_{\text{error}} \cup \mathcal{D}^{\text{Transpose}}_{\text{error}} \cup \mathcal{D}^{\text{Shift}}_{\text{incomplete}} \cup \mathcal{D}^{\text{Loop_shift}}_{\text{incomplete}} \cup \mathcal{D}^{\text{Loop_shift_end}}_{\text{incomplete}}$$

where there is no overlapping between the deduction subsets. The final robust recognizer has a time complexity, in the worst case

$$\mathcal{O}\left(\frac{n!}{\tau! * (n - \tau)!} * (n + \tau) * 2^\tau * \text{fan-out}_\mu^\tau\right)$$

where τ and fan-out_μ are, respectively, the maximal error counter computed and the maximal fan-out of the FA in the scope of the repairs; and n the length of the ill-formed sentence. The input string is recognized iff a final item $[q_f, n + 1, e]$, $q_f \in \mathcal{Q}_f$, is generated.

4.3 Pruning Strategies

Looking to reduce the recognition schemata to be explored, our proposal includes a set of cut-off strategies, combining the repair hypothesis in order to allow the user to implement human-like correction strategies.

Path-Based Pruning. This consists of pruning repair branches on items with an error below a given threshold, ρ . If the counter is greater than ρ , we stop any action on that item, this translates into a simple test condition:

$$\forall I \vdash [r, l, e] \in \mathcal{D}_{\text{robust}}, e < \rho$$

Sequence-Based Pruning. We limit the number of consecutive errors in a path, pruning them on items with a quality below a given threshold, σ . We introduce a counter, e_l , representing the local error count accumulated along a sequence of repair hypothesis in the path we are exploring. Items take the structure $[p, i, e_g, e_l]$, where the counter e_g is the same as that considered in the robust mode. We then re-define

$$\mathcal{D}^{\text{Shift}} = \{[r, l, e_g, e_l] \vdash [s, l + 1, e_g, 0], \exists [a, l] \in \mathcal{H}, s = r.a\}$$

So, when a shift is performed, a sequence of repair hypothesis is broken and no pruning can be considered. We only have to test that no sequence in $\mathcal{D}_{\text{robust}}$ exceeds the threshold σ . A complete previous deduction step

$$[r, l, e_g] \vdash [s, j, e_g + \Delta] \in \mathcal{D}_{\text{robust}}$$

is now replaced by another one of the form

$$[r, l, e_g, e_l] \vdash [s, j, e_g + \Delta, e_l + \Delta] \in \mathcal{D}_{\text{robust}}, e_l + \Delta < \sigma$$

Type-Based Pruning. We may be interested in detecting some particular hypothesis in a path of the FA or even in a sequence of it. Taking, for example, the case of $\mathcal{D}^{\text{Insert}}_{\text{robust}}$ and assuming a threshold τ to locate the pruning action on a path, the deduction steps are now:

$$\forall I \vdash [r, l, e_g, e_l] \in \mathcal{D}_{\text{robust}}^{\text{Insert}}, e_g < \tau$$

and, if we deal with a sequence on a path, we have that:

$$[r, l, e_g, e_l] \vdash [s, j, e_g + \Delta, e_l + \Delta] \in \mathcal{D}_{\text{robust}}^{\text{Insert}}, e_l + \Delta < \tau$$

assuming that standard shift actions re-initialize local counters to zero.

5 A Global Approach

We describe now an alternative global technique [10] that provides the best repair quality avoiding cascaded errors, but expending equal effort on all parts of the word including those containing no errors.

5.1 Spelling Correction

Although the author does not define them, the original algorithm differentiates between an *exploration* phase and an *exhaustive correction* one. In the first one, we look for a repair candidate by applying a delete hypothesis when no shift actions are allowed, considering the principle of optimization from an initial user-defined error threshold.

Even the generation of this error candidate is not guaranteed, once the input string has been exhausted, the algorithm goes into the exhaustive correction phase and becomes a global technique. We then apply all possible repair hypothesis on all input positions. This translates into the steps $\mathcal{D}_{\text{error}} = \mathcal{D}_{\text{error}}^{\text{explore}} \cup \mathcal{D}_{\text{error}}^{\text{exhaustive}}$, where $\mathcal{D}_{\text{error}}^{\text{explore}} = \mathcal{D}^{\text{Shift}} \cup \mathcal{D}_{\text{error}}^{\text{Delete}}$, with $\mathcal{C}^{\text{Delete}} = \{\bar{A} p.w_i\}$. For the exhaustive correction, $\mathcal{D}_{\text{error}}^{\text{exhaustive}} = \mathcal{D}^{\text{Shift}} \cup \mathcal{D}_{\text{error}}^{\text{Insert}} \cup \mathcal{D}_{\text{error}}^{\text{Delete}} \cup \mathcal{D}_{\text{error}}^{\text{Replace}} \cup \mathcal{D}_{\text{error}}^{\text{Transpose}}$, with

$$\mathcal{C}^{\text{Insert}} = \emptyset, \quad \mathcal{C}^{\text{Delete}} = \emptyset, \quad \mathcal{C}^{\text{Replace}} = \emptyset, \quad \mathcal{C}^{\text{Transpose}} = \emptyset$$

applying, in any case, the principle of optimization. Given that we are dealing with a global spelling correction approach, error and detection points are located at the beginning of the input string, during the exhaustive correction phase.

5.2 Robust Spelling

Although the Savary's algorithm does not deal with incomplete sentences, we have simply extended it by simulating insertions on unknown substrings, defining

$$\mathcal{D}_{\text{robust}} = \mathcal{D}^{\text{Init}} \cup \mathcal{D}_{\text{error}}^{\text{explore}} \cup \mathcal{D}_{\text{error}}^{\text{exhaustive}} \cup \mathcal{D}_{\text{incomplete}}^{\text{Shift}} \cup \mathcal{D}_{\text{incomplete}}^{\text{Loop_shift}} \cup \mathcal{D}_{\text{incomplete}}^{\text{Loop_shift_end}}$$

The final robust recognizer has a time complexity, in the worst case

$$\mathcal{O}\left(\frac{n!}{\tau! * (n - \tau)!} * (n + \tau) * 2^\tau * \text{fan-out}_\mu^\tau\right)$$

where τ and fan-out_μ are now considered on the global FA. Recognition occurs iff an item $[q_f, n + 1, e]$, $q_f \in \mathcal{Q}_f$, is generated out of the exploration phase.

6 Experimental Results

We consider a lexicon for Spanish built from GALENA [6], which includes 514,781 different words, to illustrate our work. The lexicon is recognized by an FA containing 58,170 states connected by 153,599 transitions, of sufficient size to allow us to consider it as a representative starting point for our purposes. In order to compare the approaches we have introduced, we look for tests that will show the influence of the operational kernel in a robust recognition process.

Three different kinds of patterns are considered for modeling ill-formed input strings. The first pattern, which we call *unknown*, is given by words which do not include spelling errors, but only unknown symbols. We call the second as *error-correction*, gathering words including only errors. The last, called *overlapping*, groups words combining both unknown symbols and spelling errors.

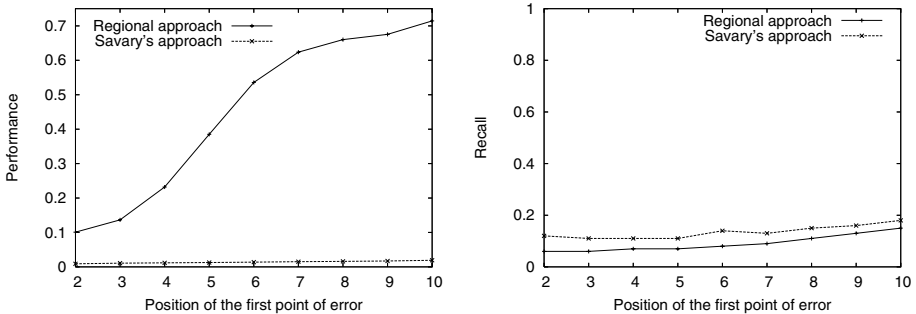


Fig. 1. Performance and recall for the *unknown* example

The results are shown, for the *unknown*, *error-correction* and *overlapping* examples in Figs. 1, 2 and 3; respectively. In all cases, we have started from the same sample of words, which has the same distribution observed in the original lexicon in terms of lengths of the strings dealt with. On these words and for each length category, we have randomly generated errors and unknown sequences in a number and position in the input string. This is of some importance since, as the authors claim, the efficiency of previous proposals depends on these factors [9, 10]. No other morphological dependencies have been detected.

Even it can be argued that testing on randomly generated errors does not yield significantly similar results to testing on human misspellings, this allows us to compare our proposal with Savary's³ one [10] on a common experimental framework⁴, which was the aim of this work. In addition, although the origin of an human error can be typographic⁵ or cognitive⁶, this does not influence

³ The most performance global repair approach, in the best of our knowledge.

⁴ Savary [10] tests her original proposal on randomly generated errors.

⁵ Caused, for example, by a keyboarding problem.

⁶ Involving, for example, phonological similarity between the intended word and the output word.

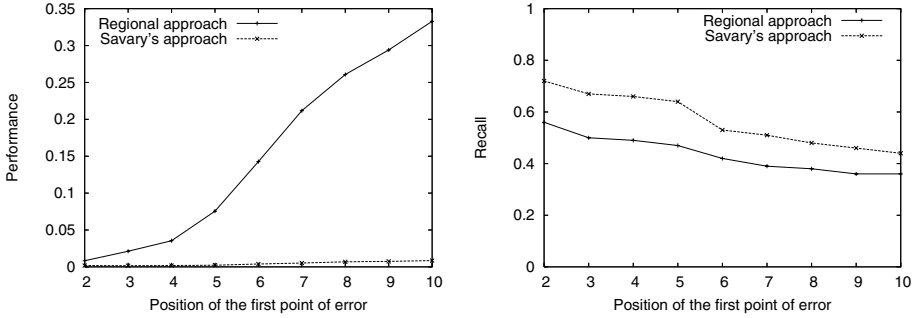


Fig. 2. Performance and recall for the *error correction* example

the structure of the FA implementing the recognizer, which only depends on the language. So, randomly generated errors seem to be valid to test an exclusively structural concept, as it is the case of the computational method used to validate the repair region.

In relation to the pruning criteria, we consider a specific one for each example. So, in the *unknown* case, path and sequence thresholds are set to 3. Type-based ones, which are only considered for delete hypothesis, are also 3. For the *error correction* example, path and sequence thresholds are, respectively, 3 and 2. Here, type-based ones are considered for all error hypothesis and set to 1. In the overlapping case, path and sequence thresholds are 4, and type-based ones are also fixed for all error hypothesis. In dealing with deletions they have a value of 3, and in the case of insertion, replacement and transposition their value is 1.

We consider the concept of item in order to measure the computational effort, disregarding purely temporal criteria, which are more dependent on the implementation. Since we are interested in computational and quality aspects, we must take into account data from both the user's and the system's viewpoint. For a given ill-formed word, w , we introduce:

$$performance(w) = \frac{useful\ items}{total\ items} \qquad recall(w) = \frac{proposed\ corrections}{total\ corrections}$$

complemented with the *precision* of the recognition process, that is, the rate reflecting when the algorithm provides the correction that could be expected from a user. We use the term *useful items* to refer to the number of generated items that finally contribute to the obtaining of a recognition, 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. We take here into account that, without any restriction, the latter is equivalent to the total number of paths recognized by the FA, which cannot be considered as a real framework. So, in a more practical sense, we consider a candidate for correction only when the number of repair hypothesis applied on the ill-formed string is not greater than half of its length.

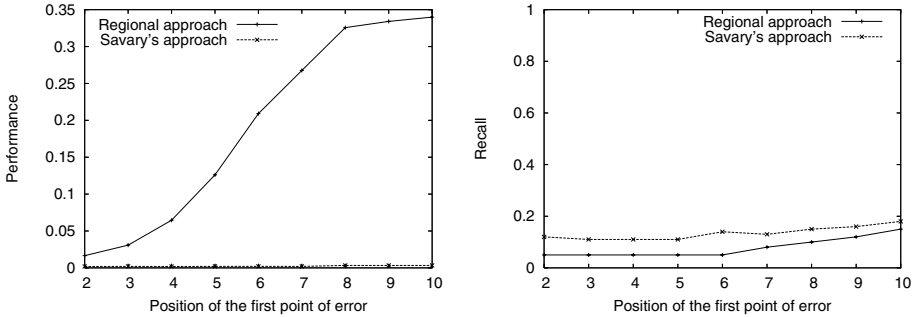


Fig. 3. Performance and recall for the *overlapping* example

In each figure, we compare two graphs corresponding to our regional proposal and the Savary's global one [10], using in the latter case the same path-threshold of our pruning schemata. In any case, all tests demonstrate a better performance and a more moderate recall for the regional approach. In particular, results on performance show a linear-like behavior for Savary's, while the graph associated with the regional approach seems to be of polynomial type. On the other hand, Savary's reaches double the recall of the regional strategy and, in the best case, it surpasses our proposal by 25%. Taking into account that the Savary's method is, to the best of our knowledge, the most efficient global proposal, this seems to corroborate the validity of our approach.

7 Conclusions

We have built a formal testing framework in order to compare global and regional robust spelling correction approaches. We have chosen to work with the most representative global strategy, defined on an unified description/operational model. We also use a collection of tests randomly generated on a corpus for Spanish, a language with a non-trivial morphology, to apply on all of the possible configurations. The collection of experimental results we have obtained in this manner seems to be promising in relation to our regional approach, justifying our future work in this domain. In particular, we are interested in going deeper into the application of more sophisticated cut-off techniques based on linguistic information.

References

1. E. Agirre, K. Gojenola, K. Sarasola, and A. Voutilainen. Towards a single proposal in spelling correction. In C. Boitet and P. Whitelock, editors, *Proc. of the 36th Annual Meeting of the ACL*, pages 22–28, 1998.
2. R.A. Baeza-Yates and G. Navarro. Faster approximate string matching. *Algorithmica*, 23(2):127–158, 1999.

3. 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.
4. A. Dermouche. A fast algorithm for string matching with mismatches. *Information Processing Letters*, 55(2):105–110, July 1995.
5. A.R. Golding and Y. Schabes. Combining trigram-based and feature-based methods for context-sensitive spelling correction. In *Proc. of the 34th Annual Meeting of the ACL*, 1996.
6. 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.
7. C.L. Lucchesi and T. Kowaltowski. Applications of finite automata representing large vocabularies. *Software-Practice and Experience*, 23(1):15–30, January 1993.
8. K. Min and W.H. Wilson. Integrated correction of ill-formed sentences. *Lecture Notes in Computer Science*, 1342:369–378, 1997.
9. K. Oflazer. Error-tolerant finite-state recognition with applications to morphological analysis and spelling correction. *Computational Linguistics*, 22(1):73–89, 1996.
10. 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.
11. K. Sikkel. *Parsing Schemata*. PhD thesis, Univ. of Twente, The Netherlands, 1993.
12. M. Vilares, J. Otero, and J. Graña. Regional finite-state error repair. *Lecture Notes in Computer Science*, 3317:269–280, 2005.

Author Index

- Abdel Fattah, Mohamed 97
Abraham, Ajith 563
Aceves-Pérez, Rita M. 453
Akama, Hiroyuki 55
Alexandrov, Mikhail 547
Alonso-Lavernia,
 María de los Ángeles 311
Amaro, Raquel 28
Angheluta, Roxana 208
- Ben Ahmed, Mohamed 121
Ben Othmane Zribi, Chiraz 121
Bollegala, Danushka 223
Bolshakov, Igor A. 93
Boonthum, Chutima 196
Bouayad-Agha, Nadjet 490
Brew, Chris 41
Burget, Lukáš 410
Buscaldi, Davide 192, 547
- Carrasco-Ochoa, Jesús Ariel 514
Castellanos, Hayde 331
Castellanos-Nieves, Dagoberto 71
Černocký, Jan 410
Chan, Samuel W.K. 263
Chang, Chia-Hui 144
Chaves, Rui Pedro 28
Chen, Xiaojun 378
Choi, Bumghi 406
Coria Olguin, Sergio Rafael 331, 355
Csomai, Andras 429
Cui, Yuzhen 245
- Darriba Bilbao, Víctor Manuel 575
De-la-Cruz-Rivera, Argelio Víctor 311
de Rijke, Maarten 180
Duan, Huiming 235
- Estrada, Varinia 331
- Fang, Alex Chengyu 168
Fapšo, Michal 410
Fazly, Afsaneh 81
Feldman, Anna 41
Fernández-Breis, Jesualdo Tomás 71
- Galicia-Haro, Sofia Natalia 93
García-Hernández, René Arnulfo 514
Gil, Angel 490
Gonçalves, Teresa 551
Gonzalez, Marco 394
Graesser, Arthur C. 287
Guntur, Bharadwaja Kumar 156
Gupta, Rakesh 343
- Han, SangYong 563, 571
Han, Shuang 235
Hana, Jirka 41
He, Yanxiang 470
Hennacy, Ken 343
Hovy, Eduard 1
Hu, Haiqing 458
Hu, Jianhua 245
Hu, Qinan 245
Huang, Joshua 378
- Ide, Nancy 13
Infante-Lopez, Gabriel 180
Inui, Nobuo 315
Ishizuka, Mitsuru 223
- Ji, Donghong 470
Jiménez-Salazar, Héctor 536
Juan Ciscar, Alfons 547
Jung, Jaeyoung 55
Jung, Sung-Won 524
Jung, Youngim 366
- Kaestner, Celso A.A. 132
Kang, Bo-Yeong 389
Kang, NamOh 571
Karafiát, Martin 410
Kavi, Narayana Murthy 156
Kim, Dae-Won 389
Kit, Chunyu 117
Komiya, Kanako 315
Kosseim, Leila 441
Kotani, Yoshiyuki 315
Kozareva, Zornitsa 208
Kulkarni, Anagha 208
Kuroiwa, Shingo 97, 458

- Kwon, Hyuk-Chul 366, 524
 Kwon, Namhee 1
- Lee, Ju-Hong 406
 Lee, Yue-Shi 144
 Levinstein, Irwin 196
 Li, Wei 417, 480
 Li, Wenjie 480
 Li, Yan 378
 Lieberman, Henry 319
 Liu, Dexi 470
 Liu, Hugo 319
 Liu, Xiaoyue 117
 López, Fernanda 331
 López, Isabel 331
 Lu, Huaming 105
- Marcu, Daniel 59
 Marrafa, Palmira 28
 Martínez-Trinidad, José Francisco 514
 Matsuo, Yutaka 223
 Max, Aurélien 567
 McCarthy, Philip M. 287
 Medina-Urrea, Alfonso 101
 Mendes, Sara 28
 Meza, Ivan 331
 Mihalcea, Rada F. 249, 319, 429
 Miyake, Maki 55
 Montes-y-Gómez, Manuel 453
 Moreno, Iván 331
- North, Ryan 81
- Okumura, Manabu 502
 Otero Pombo, Juan 575
- Pan, Haihua 245
 Paraboni, Ivandré 299
 Park, So-Young 51
 Park, Sun 406
 Park, Tae-Su 406
 Pascual, Victor 490
 Pedersen, Ted 208
 Peñas, Anselmo 275
 Pérez, Patricia 331
 Pineda Cortés, Luis Albreto 331, 355
 Pinto, David 536
 Pla, Ferran 192
- Quaresma, Paulo 551
- Ren, Fuji 97, 458
 Rodrigo, Álvaro 275
 Rodríguez, Carlos 331
 Rosso, Paolo 192, 536, 547
 Rus, Vasile 287
- Salgueiro Pardo, Thiago Alexandre 59
 Sanchis Arnal, Emilio 192
 Schwarz, Milan 410
 Schwarz, Petr 410
 Segarra, Encarna 192
 Shin, Dongha 51
 Shin, Kwangcheol 563
 Sidorov, Grigori 311
 Silla, Carlos N., Jr. 132
 Silva, Cassiana 551
 Smrž, Pavel 410
 Solorio, Thamar 208
 Song, Ui-Sung 51
 Stevenson, Suzanne 81
 Strube de Lima, Vera Lúcia 394
 Strunk, Jan 132
 Sun, Maosong 105, 417
 Suzuki, Yasuhiro 502
 Szöke, Igor 410
- Tajima, Yasuhiro 315
 Takamura, Hiroya 502
 Toida, Shunichi 196
 Torjmen, Aroua 121
 T'sou, Benjamin Ka-Yin 105
- Valdeni de Lima, José 394
 Valencia-García, Rafael 71
 Valentin, Oriol 490
 van Deemter, Kees 299
 Verdejo, Felisa 275
 Vieira, Renata 551
 Vilares Ferro, Manuel 575
 Villaseñor-Pineda, Luis 453
 Vivancos-Vicente, Pedro José 71
 Volpe Nunes, Maria das Graças 59
- Wang, Houfeng 235
 Wang, Zhimin 235
 Webster, Jonathan J. 117
 Wu, Mingli 480
 Wu, Yu-Chieh 144

Xu, Wei 480
Xu, Xiaofei 378

Yang, Hua 470
Ye, Yunming 378
Yoon, Aesun 366

Yousefi, Jamileh 441
Yu, Shiwen 235
Yuan, Chunfa 480

Zhang, Shuwu 458
Zhang, Zhengcao 105