

Gennady Agre  
Josef van Genabith  
Thierry Declerck (Eds.)

LNAI 11089

# Artificial Intelligence: Methodology, Systems, and Applications

18th International Conference, AIMSA 2018  
Varna, Bulgaria, September 12–14, 2018  
Proceedings

 Springer

# Lecture Notes in Artificial Intelligence

11089

Subseries of Lecture Notes in Computer Science

LNAI Series Editors

Randy Goebel

*University of Alberta, Edmonton, Canada*

Yuzuru Tanaka

*Hokkaido University, Sapporo, Japan*

Wolfgang Wahlster

*DFKI and Saarland University, Saarbrücken, Germany*

LNAI Founding Series Editor

Joerg Siekmann

*DFKI and Saarland University, Saarbrücken, Germany*


More information about this series at <http://www.springer.com/series/1244>

Gennady Agre · Josef van Genabith  
Thierry Declerck (Eds.)

# Artificial Intelligence: Methodology, Systems, and Applications

18th International Conference, AIMSAs 2018  
Varna, Bulgaria, September 12–14, 2018  
Proceedings

*Editors*

Gennady Agre   
Institute of Information and Communication  
Technologies  
Bulgarian Academy of Sciences  
Sofia  
Bulgaria

Thierry Declerck  
DFKI GmbH  
Saarbrücken  
Germany

Josef van Genabith  
Universität des Saarlandes  
Saarbrücken  
Germany

ISSN 0302-9743                      ISSN 1611-3349 (electronic)  
Lecture Notes in Artificial Intelligence  
ISBN 978-3-319-99343-0              ISBN 978-3-319-99344-7 (eBook)  
<https://doi.org/10.1007/978-3-319-99344-7>

Library of Congress Control Number: 2018951248

LNCS Sublibrary: SL7 – Artificial Intelligence

© Springer Nature Switzerland AG 2018

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Switzerland AG  
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

# Preface

This volume contains the papers presented at the 18th International Conference on Artificial Intelligence: Methodology, Systems, and Applications (AIMSA 2018), which was held in Varna, Bulgaria, during September 12–14, 2018. Initiated in 1984, this biennial conference is a premier forum for exchanging information and research results on artificial intelligence (AI) theory and principles along with applications of intelligent system technology. The conference traditionally brings together academic and industrial researchers from all areas of AI to share their ideas and experiences and learn about the research in contemporary AI. As its name indicates, the conference is dedicated to AI in its entirety. However, AIMSA 2018 put an emphasis on deep learning as it has been used successfully in many applications, and is considered one of the most cutting-edge machine learning and AI techniques at present.

AIMSA continues to attract submissions from all over the world, with submissions from 23 countries. We received 72 submissions in total, and accepted 22 papers for oral and seven for poster presentations. Each paper was reviewed at least by two members of the Program Committee. The papers included in this volume cover a wide range of topics on AI: from natural language processing and deep learning to bioinformatics and AI, from text mining to multiagent systems, from theoretical issues to real-world applications.

AIMSA 2018 had three outstanding keynote speakers: John D. Kelleher (Dublin Institute of Technology) presented his view on the role of machine learning and deep learning in AI, Milica Gasic (University of Cambridge) discussed the necessary steps needed to deploy deep reinforcement learning for dialogue policy optimization, and Feiyu Xu (AI Lab, Lenovo Research, Lenovo Group) described a multilingual and multimodal conversational agent for real-world call centers.

We would like to thank all authors for providing an excellent set of papers. We are extremely grateful to the Program Committee for reviewing the submissions thoroughly, fairly, and very quickly. Our special thanks to our sponsors – the Bulgarian National Science Fund and the Bulgarian Artificial Intelligence Association as well as to the Institute of Information and Communication Technologies at the Bulgarian Academy of Sciences for organizational support for the conference.

July 2018

Gennady Agre  
Josef van Genabith  
Thierry Declerck

# Organization

## Program Committee

Gennady Agre	Institute of Information and Communication Technologies, Bulgarian Academy of Sciences, Bulgaria
Galia Angelova	Institute of Information and Communication Technologies, Bulgarian Academy of Sciences, Bulgaria
Roman Bartak	Charles University in Prague, Czech Republic
Christoph Beierle	FernUniversität in Hagen, Germany
Tarek R. Besold	City University of London, UK
Loris Bozzato	CIT-IRST, Fondazione Bruno Kessler, Italy
Ricardo Calix	Purdue University Northwest, USA
Diego Calvanese	Free University of Bozen-Bolzano, Italy
Thierry Declerck	DFKI GmbH, Germany
Sarah Jane Delany	Dublin Institute of Technology, Ireland
Christo Dichev	Winston-Salem State University, USA
Darina Dicheva	Winston-Salem State University, USA
Danail Dochev	Institute of Information and Communication Technologies, Bulgarian Academy of Sciences, Bulgaria
Benedict Du Boulay	University of Sussex, UK
Stefka Fidanova	Institute of Information and Communication Technologies, Bulgarian Academy of Sciences, Bulgaria
Tomasz Gambin	Institute of Computer Science, Warsaw University of Technology, Poland
Geert-Jan Houben	Delft University of Technology, The Netherlands
Maciej Kandula	Chair of Bioinformatics Research Group, Boku University Vienna, Austria
Kristian Kersting	TU Darmstadt, Germany
Matthias Knorr	Universidade NOVA de Lisboa, Portugal
Petia Koprinkova-Christova	Institute of Information and Communication Technologies, Bulgarian Academy of Sciences, Bulgaria
Amedeo Napoli	CNRS, Nancy, France
Maria Nisheva	Sofia University St. Kliment Ohridski, Bulgaria
Michal Okoniewski	IT Services, ETH Zurich, Switzerland
Dean Palejev	Institute of Mathematics and Informatics, Bulgarian Academy of Sciences, Bulgaria

Horia Pop	UBB Cluj, Romania
Allan Ramsay	The University of Manchester, UK
Ioannis Refanidis	University of Macedonia, Greece
Ute Schmid	University of Bamberg, Germany
Stefan Trausan-Matu	University Politehnica of Bucharest, Romania
Dan Tufis	Research Institute for Artificial Intelligence, Romanian Academy, Romania
Petko Valtchev	Université du Québec à Montréal, Canada
Josef van Genabith	The German Research Center for Artificial Intelligence DFKI, Germany
Dimitar Vassilev	Sofia University St. Kliment Ohridski, Bulgaria
Tulay Yildirim	Yildiz Technical University, Turkey
Paweł P. Łabaj	Chair of Bioinformatics, Boku University Vienna, Austria
Dominik Ślęzak	University of Warsaw, Poland

### **Additional Reviewers**

Gluhchev, George  
Koychev, Ivan  
Osenova, Petya  
Stoilova, Krasimira



# Contents

## Natural Language Processing

A New Approach to the Supervised Word Sense Disambiguation . . . . .	3
<i>Gennady Agre, Daniel Petrov, and Simona Keskinova</i>	
Explorations in Sentiment Mining for Arabic and English Tweets . . . . .	16
<i>Tariq Ahmad, Allan Ramsay, and Hanady Ahmed</i>	
Intent Detection System Based on Word Embeddings . . . . .	25
<i>Kaspars Balodis and Daiga Deksnė</i>	
Indirect Association Rules Mining in Clinical Texts . . . . .	36
<i>Svetla Boytcheva</i>	
Towards Automated Customer Support . . . . .	48
<i>Momchil Hardalov, Ivan Koychev, and Preslav Nakov</i>	
Evaluation of Automatic Tag Sense Disambiguation Using the MIRFLICKR Image Collection . . . . .	60
<i>Olga Kanishcheva, Ivelina Nikolova, and Galia Angelova</i>	
Echo State Network for Word Sense Disambiguation. . . . .	73
<i>Petia Koprinkova-Hristova, Alexander Popov, Kiril Simov, and Petya Osenova</i>	
Constrained Permutations for Computing Textual Similarity . . . . .	83
<i>Allan Ramsay and Amal Alshahrani</i>	
A Study on Dialog Act Recognition Using Character-Level Tokenization. . . . .	93
<i>Eugénio Ribeiro, Ricardo Ribeiro, and David Martins de Matos</i>	
Neural Methods for Cross-Lingual Sentence Compression . . . . .	104
<i>Frederico Rodrigues, Bruno Martins, and Ricardo Ribeiro</i>	
Towards Constructing a Corpus for Studying the Effects of Treatments and Substances Reported in PubMed Abstracts . . . . .	115
<i>Evgeni Stefchov, Galia Angelova, and Preslav Nakov</i>	
Recursive Style Breach Detection with Multifaceted Ensemble Learning . . . . .	126
<i>Daniel Kopev, Dimitrina Zlatkova, Kristiyan Mitov, Atanas Atanasov, Momchil Hardalov, Ivan Koychev, and Preslav Nakov</i>	

**Machine Learning and Data Mining Applications**

Improving Machine Learning Prediction Performance for Premature Termination of Psychotherapy. . . . . 141  
*Martin Bohus, Stephan Gimbel, Nora Goerg, Bernhard G. Humm, Martin Schüller, Marc Steffens, and Ruben Vonderlin*

Exploring the Usability of the Dice CAPTCHA by Advanced Statistical Analysis . . . . . 152  
*Darko Brodić, Alessia Amelio, Ivo R. Draganov, and Radmila Janković*

Time Series Analysis for Sales Prediction . . . . . 163  
*Costin-Gabriel Chiru and Vlad-Valentin Posea*

Machine Learning-Driven Noise Separation in High Variation Genomics Sequencing Datasets . . . . . 173  
*Milko Krachunov, Maria Nisheva, and Dimitar Vassilev*

Machine Learning Techniques for Survival Time Prediction in Breast Cancer . . . . . 186  
*Iliyan Mihaylov, Maria Nisheva, and Dimitar Vassilev*

**Knowledge Representation, Reasoning and Search**

Tractable Classes in Exactly-One-SAT. . . . . 197  
*Yazid Boumarafi and Yakoub Salhi*

Semantic Meta-search Using Cohesion Network Analysis. . . . . 207  
*Ionut Daniel Chelcioiu, Dragos Corlatescu, Ionut Cristian Paraschiv, Mihai Dascalu, and Stefan Trausan-Matu*

A Query Language for Cognitive Maps . . . . . 218  
*Adrian Robert, David Genest, and Stéphane Loiseau*

Approaches for Enumerating All the Essential Prime Implicants . . . . . 228  
*Yakoub Salhi*

Evolving a Team of Asymmetric Predator Agents That Do Not Compute in Predator-Prey Pursuit Problem . . . . . 240  
*Ivan Tanev, Milen Georgiev, Katsunori Shimohara, and Thomas Ray*

**Posters**

Semantic Graph Based Automatic Summarization of Multiple Related Work Sections of Scientific Articles . . . . . 255  
*Nouf Ibrahim Altmami and Mohamed El Bachir Menai*

Collective Lévy Walk for Efficient Exploration in Unknown Environments . . . . .	260
<i>Yara Khaluf, Stef Van Havermaet, and Pieter Simoens</i>	
A BLAST-Based Algorithm to Find Evenly Distributed Unique Subsequences . . . . .	265
<i>Maciej Kulawik and Robert M. Nowak</i>	
Implementation of Multilayer Perceptron in Graphics Processing Unit . . . . .	270
<i>Ventsislav Nikolov</i>	
Semantic Annotation Modelling for Protein Functions Prediction . . . . .	275
<i>Deyan Peychev and Irena Avdjieva</i>	
ReadME – Enhancing Automated Writing Evaluation . . . . .	281
<i>Maria-Dorinela Sirbu, Robert-Mihai Botarleanu, Mihai Dascalu, Scott A. Crossley, and Stefan Trausan-Matu</i>	
Feature Selection Based on Logistic Regression for 2-Class Classification of Multidimensional Molecular Data . . . . .	286
<i>Sebastian Student, Alicja Pluciennik, Michał Jakubczak, and Krzysztof Fujarewicz</i>	
<b>Author Index</b> . . . . .	291

# **Natural Language Processing**



# A New Approach to the Supervised Word Sense Disambiguation

Gennady Agre<sup>1</sup> , Daniel Petrov<sup>2</sup>, and Simona Keskinova<sup>2</sup>

<sup>1</sup> Institute of Information and Communication Technologies,  
Bulgarian Academy of Sciences, Sofia, Bulgaria  
agre@iinf.bas.bg

<sup>2</sup> Laboratory of Computer Graphics and Geographical Information Systems,  
Technical University of Sofia, Sofia, Bulgaria  
desolatora93@gmail.com, simona181212@gmail.com

**Abstract.** The paper presents a new supervised approach for solving the all-words sense disambiguation (WSD) task, which allows avoiding the necessity to construct different specialized classifiers for disambiguating different target words. In the core of the approach lies a new interpretation of the notion ‘class’, which relates each possible meaning of a word to a frequency with which it occurs in some corpora. In such a way all possible senses of different words can be classified in a unified way into a restricted set of classes starting from the most frequent, and ending with the least frequent class. For representing target and context words the approach uses word embeddings and information about their part-of-speech (POS) categories. The experiments have shown that classifiers trained on examples created by means of the approach outperform the standard baselines for measuring the behavior of all-words WSD classifiers.

**Keywords:** Word sense disambiguation · Word embedding · Classification  
Neural networks · Random forest · Deep forest

## 1 Introduction

Word sense disambiguation (WSD) is defined as a task for identifying the correct meaning of words in context in a computational manner. This task is considered as an AI-complete problem [1], which means that its difficulty is of the same order as of such central problem in AI as, for example, the Turing Test [2]. The complexity of the WSD task is due to several reasons among which are the lack of a unified representation for word senses, the use of different levels of granularity of sense inventories and a strong dependence of the task on available knowledge sources. Currently, WordNet [3] is a standard sense inventory for WSD for English texts. It represents a computational lexicon for English created and maintained at Princeton University, which is based on psycholinguistic principles. WordNet encodes concepts in terms of sets of synonyms (called *synsets*) and its latest version, WordNet 3.1, contains about 155,000 words organized in over 117,000 synsets. For the WSD task WordNet is used for constructing machine-readable dictionaries that relate each word (word lemma) to a set of WordNet sense identifiers.

The knowledge resources used are the basis for clustering the existing approaches for solving WSD task to two main groups – supervised and knowledge based. The supervised approaches explore sense-annotated collections of texts (corpora). Since the manual annotation of texts is a very time consuming task, most of the existing supervised WSD systems use the largest manually annotated corpus - SemCor [4] as their training set. SemCor comprises texts from the Brown Corpus<sup>1</sup> and contains around 234,000 words that are manually annotated with part-of-speech (POS) tags and word senses from the WordNet inventory.

The semi-supervised WSD systems try to overcome the problem with the knowledge acquisition bottleneck of supervised systems [5] by creating and using some automatically sense annotated corpora or by exploring such corpora in conjunction with manually annotated corpora. It has been shown that in some settings such semi-supervised systems outperform purely supervised WSD systems [6].

Knowledge-based approaches do not need a sense annotated corpus and rely only on lexical recourses such as a dictionary or a computational lexicon. One of the earliest approaches in this group [8] is based on the computation of overlap between the context of the target word and its definitions from the sense inventory. Several approaches explore the distributional similarity between definitions and the context of the target word [8, 9]. An important branch of knowledge-based systems are graph-based systems that use some structural properties of semantic graphs for solving the WSD task [10, 11, 21]. Such systems first create a graph representation of the input text and then traverse the graph by different graph-based algorithms (e.g. PageRank) in order to determine the most probable senses in the context. The knowledge-based systems usually have lower accuracy than the supervised systems, but they have the advantage of a wider coverage, thanks to the use of large-scale knowledge resources [2].

The structure of the paper is as follows: in the next section we discuss some related work. Section 3 is devoted to the detailed description of the proposed approach. The experimental evaluation of the approach is presented in Sect. 4. The main characteristics of the approach and our future plans are summarized in the last section.

## 2 Related Work

WSD can be considered as a classification task, in which word senses are the classes and a classifier is used to assign each occurrence of a target word in a test text to one or more classes based on some features extracted from the context of this word. There exist two variants of this task – the lexical sample WSD and the all-words WSD. In the first case only a restricted subset of target words should be disambiguated. Such words usually occur one per sentence. The all-words WSD task requires finding senses for all

---

<sup>1</sup> <http://clu.uni.no/icame/manuals/RROWN/INDEX.HTM>.

open words in a sentence (i.e. nouns, verbs, adjectives, and adverbs). In the supervised context the complexity of this task is higher because of:

1. *A huge number of classes*: a WordNet-based dictionary used for sense annotating a text contains about 150,000 open words with average number of 1.4 senses per word (from 1 to 75). Thus, assuming that each sense of a word constitutes a unique class, the overall number of possible classes is about 210,000.
2. *Data sparseness*: it is practically impossible to have a training set of adequate size that covers all open words in the dictionary, which leads to an impossibility to create a good classification model for disambiguating all words.

Because of these reasons the existing supervised WSD systems try to solve the all-words WSD task by constructing a set of classifiers (so called *word experts*) – one specialized classifier per word. A training example for such word expert is represented by a set of features extracted from the context of the corresponding word (a fixed number of words occurring left and right to the target word in a sentence) and labeled by a concrete sense of the word (class) selected from a set of possible word senses presented in the dictionary.

One of the state-of-the-art supervised all-words WSD systems is IMS [13]. It constructs a classifier for each type of open word  $w$  (noun, verb, adjective, or adverb). The representation of each example (an occurrence of  $w$  in the text) includes three set of features extracted from the context of  $w$  (a window with the size of three words left and right to  $w$ ). The first set consists of POS tags of the target word and of those context (surrounding) words, which belong to the same sentence as the target word. The second set contains a special binary representation of surrounding words in their lemma forms. The size of such a binary vector is equal to the number of all different context words for each word type found in the training set. The third feature set consists of 11 features – local collocations between context words. The representation of examples constructed by IMS can be used with different classifiers and the best results have been achieved by a linear support vector machine (SVM).

A current tendency in the development of advanced supervised WSD systems is the use of word embeddings instead of (or along with) ‘traditional’ discrete representation of words. Word embeddings are a distributional representation of words that contains some semantic and syntactic information [14]. Word embeddings have been used as a means to improve the performance of ‘traditional’ WSD systems as well as for constructing specialized WSD systems that are based on recurrent neural network models. The first approach includes different variants of integration of word embeddings with the IMS system [15–17]. In most cases different combinations of word embeddings over surrounding words were used as additional features. The extensive experiments have shown [17] that such integration is able to improve the accuracy of IMS on the all-word WSD task for several benchmark domains.

All IMS-based approaches for solving the all-words WSD task mentioned above suffer from the following shortcomings:

- Necessity to construct separate classification models for each open word (or even for each word type).

- Relying on rather complicated procedures for feature extraction and their integration with word embeddings.
- Ignoring the order of words in the context.

The second approach [7, 18–20] can be seen as an attempt to overcome the last two shortcomings. It is based on a simple idea – first, the input context is transformed into vectors in the embedding space, and then such vectors are mapped to the possible sense vectors. Such systems usually use Bidirectional LSTM as a classifier and have received promising results on the lexical sample task. However, this approach still has to construct different models (with different sets of parameters) for each target word.

### 3 The Approach

The proposed here approach tries to overcome all three deficiencies of the existing supervised WSD systems mentioned above. Similarly to the IMS system we emphasize the feature extraction step, which aims at converting available text data into sets of training and testing examples that can be used with different machine learning algorithms for solving the all-words WSD task.

We assume that three sources of knowledge are available: a WordNet-based dictionary, training text data, and a set of word embeddings. The dictionary relates open words (in their lemma forms) with their possible senses. Training text data may consist of one or several text files, where each file contains a set of sentences and each sentence is represented as a sequence of words in their lemma forms annotated by their POS category. Additionally, all open words in a sentence are assumed to be annotated by their senses<sup>2</sup>. A set of word embeddings relates words in their lemma form to their distributional representation. In our approach we use the file containing word embeddings as a parameter and are not interested in how they were created<sup>3</sup>. Testing data is assumed to have the same format as the training data.

#### 3.1 Creating Sets of Training and Testing Examples

In order to represent the all-words WSD problem as a classification task it is necessary to convert the training and test text data into sets of training and test examples relating each open word to its correct sense, as well as to a set of features extracted from the word context. In our approach the creation of such examples is implemented as a three phase process consisting of dictionary and text data analysis, generation of examples in a symbolic form, and integration of examples with word embeddings.

The first phase begins with analyzing the dictionary. First of all, we determine in it a list of *monosemous* words (i.e. words with only one meaning). It is clear, that it is not necessary to construct examples for such words since their meaning does not depend on

---

<sup>2</sup> In cases, when the training set is not available, SemCor [4], for example, may be used as a default training set.

<sup>3</sup> Moreover, we assume that all words (both in training and testing sets) that have no their embeddings are removed from the corresponding sets.



any context and can be determined directly from the dictionary. Then we analyze available text data in order to infer some statistics including the number of occurrences of *polysemous* words (i.e. words with several meanings) in the training and test data (which corresponds to the numbers of training and test examples), frequencies of different word senses for each polysemous word in the training data, etc. Finally, for each word in the dictionary we rearrange the list of word senses – from most frequent to least frequent. Such rearrangement can be done based on the sense frequency data available in the dictionary, the sense frequency data calculated over the training set, or by mixing both types of such data. In the last case the sense frequency data calculated over the training data is applied only for rearrangement of senses, which have no data on their frequencies in the dictionary.

*Generation of examples in a symbolic form* is aimed at creating training and test examples for the all-words WSD task in an intermediate form readable and easily understandable by a human. Each example is represented as a pair **<word\_context, word\_class>**, where vector **word\_context** describes the target word (and its POS category) and its context represented as words (and their POS categories) occurring left and right to the target word in the context window, which length is a system parameter:

#### **word\_context**

$$= \{w_{\text{target}} - \text{POS}_{\text{target}} \ w_{-L} - \text{POS}_{-L} \ w_{-L+1} - \text{POS}_{-L+1} \dots w_1 - \text{POS}_1 \dots w_R - \text{POS}_R \},$$

where  $L$  and  $R$  are, respectively, the left and the right boundary of the context window. We use a special symbol ( $/s$ ) for marking the end of a sentence and the system allows specifying whether the context window can include words from adjacent sentences or not. In the last case the content in the corresponding position of the context window is marked as 0-0.

We consider our representation of the class of the target word as the most important contribution to a new formulation of the all-words WSD task. Up to now each sense of a target word is considered as a unique class, which has its own encoding that makes sense only for that word. In our approach we propose a unified interpretation of a word sense class, which does not depend on the concrete word – *the class of a target word is interpreted as the place of the corresponding word sense in an ordered sequence of the relative sense frequencies associated with the word in the modified dictionary*. In other words, the most frequently used sense of each target word is marked as class 1, the next less frequent word sense is marked as class 2 and so on. The overall number of classes is defined by the maximum number of possible senses for a word in the dictionary (which is equal to 75 for our WordNet-based dictionary).

It should be noted, that according to this interpretation disambiguating an unknown open word is a two-steps process – at the first step the class associated with the sense of the tested word is determined by a classifier, and then the proper sense is retrieved from the dictionary via WordNet sense identifier that corresponds to the class for the tested word.

The last phase of the data preprocessing in our approach is intended for integrating examples created during the previous phase with word embeddings and transforming the examples into a format that may be used by different machine learning algorithms

and environments. Our approach allows using words embeddings in two different ways – in a full and in a compressed form. In the first case each word in the example (i.e. context words and the target word itself) is replaced by its embedding. ‘Null’ words (i.e. marked by 0 in the symbolic representation of the example) are replaced by ‘null’ embeddings – vectors with the same size as ‘normal’ embeddings but having all their elements set to zero. In such a way, if the size of a word embedding vector is  $d$  (i.e. 300), and the size of the context widow is  $l$  (i.e. 10), such embedding-based representation of the target word and its context will be described by a real valued vector with dimension  $d \times (l + 1)$  (i.e. 3300).

Since not all of the existing machine learning environments are able to work with data of such dimensionality, we have developed an alternative approach for utilizing word embeddings. The approach creates a ‘compressed’ real-valued representation of the target word and its context, whose dimensionality does not depend on the dimensionality of word embeddings used. In order to do this we replaced each word  $w_i$  in the context that has  $d$ -dimensional embedding  $\mathbf{Emb}(w_i)$  by a single value  $compr\_emb(w_i)$ , which measures the cosine similarity between  $\mathbf{Emb}(w_i)$  and  $\mathbf{Emb}(w_{target})$  – a  $d$ -dimensional embedding of the target word  $w_{target}$ :

$$compr\_emb(w_i) = sim(\mathbf{Emb}(w_i), \mathbf{Emb}(w_{target})) = \frac{\sum_{j=1}^d Emb_j(w_i) Emb_j(w_{target})}{\sqrt{\sum_{j=1}^d Emb_j^2(w_i)} \sqrt{\sum_{j=1}^d Emb_j^2(w_{target})}}$$

In such a way, the same context of different target words will have different ‘compressed’ embedding representations depending on the similarity between the concrete target word and the context words.

Compressed representation of the target word itself is created by measuring cosine similarity between the embedding of the word and the ‘unitary’ embedding ( $\mathbf{1}$ ) – a  $d$ -dimensional vector, whose arguments are all equal to one:

$$compr\_emb(w_{target}) = sim(\mathbf{Emb}(w_{target}), \mathbf{1}) = \frac{\sum_{j=1}^d Emb_j(w_{target})}{\sqrt{d} \sqrt{\sum_{j=1}^d Emb_j^2(w_{target})}}$$

The encoding of the POS categories of the target and context words in an example can be done in two ways. The first is to represent POS category of each word as a separate nominal attribute that can have  $k + l$  different nominal values, where  $k$  is the length of the list of POS categories to be included into context ( $k + l$ th value is the ‘null’ category assigned to ‘null’ words and to the symbol/s used to mark the end of a sentence). Another option (which is suitable for the neural network based classifiers) is to represent each POS category of a word by a  $k$ -dimensional binary vector.

### 3.2 Classification

As it has been already mentioned, our goal was to develop a flexible feature extraction system that allows converting available training and test data into a format suitable for learning effective WSD classifiers rather than to create a specialized classifier for solving the all-words WSD task. That is why, the phase of learning and testing classification models, which are based on training and test sets of examples created by our system, is not a part of the system. However, the system provides a possibility to (implicitly) define a number of classifiers needed for solving the all-words WSD task. At the final phase it is possible to select how the created training and test sets of examples should be saved – as a single file or as a set of files. Saving data into a single file allows creating a single classifier that is able to disambiguate all target words occurring in the text. On the other side is a possibility to group examples based on a concrete target word or even on a POS category of the target word. In such a way we are able to create a set of word-specific training and test files that can be used for creating ‘traditional’ word expert classifiers.

An ability to group examples according to the POS categories of target words is situated between these two extremes. These four sets of examples can be used for constructing four different classifiers specializing in solving the WSD task for nouns, verbs, adverbs and adjectives. Such a flexible approach allows for the generating of different classification models for different POS word categories depending on the quality and quantity of available examples for each category. It should be mentioned, that all such created data sets can be saved either as.txt or .arff files which makes it possible to use them directly in such machine learning environments as WEKA<sup>4</sup> or Orange<sup>5</sup>.

## 4 Experiments and Results

The conducted experiments were aimed at achieving two main goals: the first was to evaluate the performance of different classifiers trained on the sets of examples prepared according to the proposed approach in relation to such commonly used measures for evaluating WSD classifiers as Most Frequent Sense (MFS) and WordNet First Sense (WNFS) baselines [7]. MFS is the accuracy of a classifier that for each target word selects the sense that occurs most frequently in the training data for the same word. WNFS is the accuracy of a classifier that for each target word selects the most frequent sense of that word according to WordNet.3.0.

The second goal was to compare the behavior of such classifiers with a knowledge-based WSD system developed by our colleagues [21]. The system is based on different kinds of knowledge graphs and outperforms other knowledge-based WSD systems created by means of the UKB<sup>6</sup> tool and using standard knowledge graphs. The best

---

<sup>4</sup> <https://www.cs.waikato.ac.nz/ml/weka/>.

<sup>5</sup> <https://orange.biolab.si/>.

<sup>6</sup> <http://ixa2.si.ehu.es/ukb>.

reported accuracy achieved by the their system (calculated on the whole data set including monosemous words) is 68.77%

We use the same training and test data that were extracted from SemCor 3.0<sup>7</sup> corpus – one third of available files (from br-a01 to br-f44) were used for testing and the rest two third – for training. It should be noted that the files contain only open class words. The sense annotation of words in the files has been done in accordance with WordNet dictionary wnet30\_v203.lex<sup>8</sup> that contains 148,401 word lemmas with a minimum number of senses per word – 1 and maximum number of sense per word – 75 (1.40 senses per word in average). The overall number of polysemous words in the dictionary is 27,009. Table 1 and 2 summarize some features of these files at the level of words and their occurrences in the corresponding text files.

As we have already mentioned, we construct examples only for polysemous words – each occurrence of such words in the text data is converted to an example. The test data contains 1,272 different polysemous words that are not present in the training data (they occur 2,042 times in the test data - 5.8% of all test examples). Such ‘test-only’ words can be recognized by a classifier, but can not be correctly disambiguated by any classification model constructed only from the given set of training examples. The best alternative for solving such examples is to use the most frequent sense for the tested word obtained from a WordNet dictionary (WNFS). That is why we excluded such examples from the test set.

**Table 1.** Text data statistics.

Data Set	Number of sentences	Average number of words per sentence	Number of different words	Number of different monosemous words	Number of different polysemous words	Number of unknown words
Training text data	15,231	10	16,465	6,838	9,627	0
Test text data	4,669	10.9	9,584	6,302	3,282	0

**Table 2.** Some statistics on word occurrences.

Data Set	Number of word occurrences	Number of occurrences of polysemous words	Number of occurrences of monosemous words	Maximum number of senses per word	Number of occurrences of ‘test-only’ words	Number of occurrences of ‘unsol-vable’ words
Training text data	134,372	115,095	19,277	31	–	–
Test text data	49,541	42,296	7,245	23	2,042	3,594

<sup>7</sup> <http://web.eecs.umich.edu/~mihalcea/downloads/semcor/semcor3.0.tar.gz>.

<sup>8</sup> <https://github.com/asoroa/ukb/blob/master/src/README>.

We call a word occurring in the test data ‘unsolvable’ if it is present in the training data but its correct sense - is not. It is clear, that such words can never be disambiguated correctly by any classifier learnt from the given training data. Since ‘unsolvable’ words can not be recognized by any classifier, we use such data for calculating a so called ‘Best Case’ baseline that determines the upper bound of the classification accuracy that can be theoretically achieved on this test set by a classifier learnt from the given training set (for our training set this baseline is equal to 91.10%).

We used the words embeddings<sup>9</sup> with 300 dimensions, that were trained over a pseudo corpus generated from an extended WordNet-based knowledge graph (PCWN) [21] by means of Word2Vec tool<sup>10</sup> with the following settings: context window of 5 words, 7 iterations, 5 negative samples, and frequency cut sampling set to 7.

We have experimented with two types of classifiers – neural network based and ensemble-based ones. Neural networks with different architecture were built in the *TensorFlow*<sup>11</sup> - an open source machine learning framework. For the ensemble based classifiers we have selected Random Forest in its *scikit-learn* implementation<sup>12</sup> and *GCForest*<sup>13</sup> implementation of the Deep Forest model [22].

The first set of experiments was conducted on data in which the ordering of classes was done according to the frequencies of senses calculated over the training set. That is why the results were compared to the Most Frequent Sense (MFS) baseline calculated over the same training set. We have experimented with two methods for integrating word embeddings - Table 3 presents the experiment results with word embeddings in their full form and Table 4 – in their compressed form. The disambiguation was done by four classifiers trained on the training sets corresponding to the four POS categories (nouns, adverbs, adjectives and verbs). In all experiments with neural networks we applied the same set of parameters – Softmax evaluation function, ReLU activation function, learning rate set to 0.003, dropout set to 0.5 and Adam optimizer.

**Table 3.** Results from the experiments with ‘full’ word embeddings and class ordering based on the training set.

Classifier	Random Forest (RF)	Deep Forest	Linear Classifier	Neural Network	Neural Network	MFS
Architecture	100 trees	2 RF (100) + 2 ExtraTrees (100)	3340 inputs > 75 outputs	3340 > 200 > 75	3340 > 500 > 200 > 75	
Overall Accuracy (%)	64.35	<b>64.72</b>	64.09	64.17	64.20	59.81

<sup>9</sup> WN30WNGWN30glConOneGraphRelSCOne-synsetEmbeddings.bin downloaded from <http://bultreebank.org/en/DemoSem/Embeddings>.

<sup>10</sup> <https://code.google.com/archive/p/word2vec/>.

<sup>11</sup> <https://www.tensorflow.org/>.

<sup>12</sup> <http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>.

<sup>13</sup> <https://github.com/kingfengji/gcForestDeep>.

**Table 4.** Results from the experiments with ‘compressed’ word embeddings and class ordering based on the training set.

Classifier	Random Forest (RF)	Deep Forest	Linear Classifier	Neural Network	Neural Network	MFS
Architecture	100 trees	2 RF (100) + 2 ExtraTrees (100)	51 inputs > 75 outputs	51 > 200 > 75	51 > 500 > 200 > 75	
Overall Accuracy (%)	<b>64.20</b>	64.17	64.12	64.16	64.19	<i>59.81</i>

It can be seen that all classifiers easily ‘beat’ the MFS baseline as the best results have been achieved by the ensemble-based classifiers. It should be noted that the overall accuracy of classifiers trained on ‘compressed’ representation of word embedding is almost the same as for that used the ‘full’ form of word embeddings.

Tables 5 and 6 present results of similar experiments – the only difference is that the ordering of classes was done based on the sense frequencies specified in our WordNet-based dictionary. That is why we have compared the results with the WordNet First Sense (WNFS) baseline.

**Table 5.** Results from experiments with ‘full’ word embeddings and WordNet-based class ordering.

Classifier	Random Forest (RF)	Deep Forest	Linear Classifier	Neural Network	Neural Network	WNFS
Architecture	100 trees	2 RF (100) + 2 ExtraTrees (100)	3340 inputs > 75 outputs	3340 > 200 > 75	3340 > 500 > 200 > 75	
Overall Accuracy (%)	<b>70.65</b>	70.62	70.42	70.42	70.42	<i>64.89</i>

Although the baseline accuracy in this case is higher, all classifiers again outperform this baseline and again the ensemble-based classifiers have demonstrated better behavior than neural network-based ones.

In order to compare our supervised approach with the knowledge-based approach proposed in [21], we have to extend our classification schema to cover monosemous words and words from the test data that are not present in the training data. All such words are classified by means of WNFS heuristic<sup>14</sup>. As it can be seen from Table 7, the

<sup>14</sup> It is clear, that the WNFS baseline for such extended test set will be also changed.

**Table 6.** Results from experiments with ‘compressed’ word embeddings and WordNet-based class ordering.

Classifier	Random Forest (RF)	Deep Forest	Linear Classifier	Neural Network	Neural Network	WNFS
Architecture	100 trees	2 RF (100) + 2 ExtraTrees (100)	51 inputs > 75 outputs	51 > 200 > 75	51 > 500 > 200 > 75	
Overall Accuracy (%)	<b>70.49</b>	70.39	70.42	70.42	70.44	<i>64.89</i>

**Table 7.** Classification accuracy of some classifiers on all test examples.

Classifier	RF(100) + ‘full’ word embeddings + WordNet-based class ordering	RF(100) + ‘compressed’ word embeddings + WordNet-based class ordering	Knowledge-based classifier [21]	WNFS
Accuracy (%)	<b>75.80</b>	75.67	68.77	71.12

supervised-based classifiers significantly outperform as the knowledge-based ones as the WNFS baseline.

## 5 Conclusion and Future Work

In this paper we have described a new approach for solving the all-word WSD task in the supervised context. The approach allows formulating this problem as a classification task that can be solved by means of one or several classifiers that are trained on the training set or on its different subsets. In all cases the training examples are represented in a unified way, which is based on a new interpretation of the notion ‘class’ as a place of the concrete word sense in a sequence of possible word senses ordered by the sense frequencies calculated on some corpora. At the moment representation of examples includes information about POS categories of the target word and words from its context, as well as on the context itself. Both target and context words can be represented in two different ways via their embeddings. In the first case the full distributional representation of embeddings as vectors is used, while in the second – each word is coded only by a single real-valued number reflecting similarity between the target word and the given context word. The conducted experiments have shown that classifiers training on such created training examples outperform both Most Frequent Sense and WordNet First Sense baselines for the test data.

At the moment our efforts are concentrated mainly in two directions – the first is to evaluate the approach on a wider range of available data (i.e. Semeval-2, Semeval-3, etc.) and to compare the results with several state-of-the-art supervised WSD systems.

The second is to extend the proposed approach by integrating it with more knowledge sources, for example with information on context word collocations and with embeddings over word senses.

## References




1. Mallery, J.C.: Thinking about foreign policy: finding an appropriate role for artificial intelligence computers. Ph.D. dissertation. MIT Political Science Department, Cambridge, MA (1988)
2. Navigli, R. Word sense disambiguation: a survey. *ACM Comput. Surv.* **41**(2) (2009). Article 10
3. Fellbaum, C.: WordNet and wordnets. In: Brown, K., et al. (eds.) *Encyclopedia of Language and Linguistics*, 2nd edn., pp. 665–670. Elsevier, Oxford (2005)
4. Miller, G.A., Leacock, C., Teng, R., Bunker, R.T.: A semantic concordance. In: *Proceedings of the ARPA Workshop on Human Language Technology*, pp. 303–308 (1993)
5. Kuchera, H., Francis, W.N.: *Computational Analysis of Present-Day American English*. Brown University Press, Providence (1967)
6. Pilehvar, M.T., Navigli, R.: A large-scale pseudoword-based evaluation framework for state-of-the-art Word Sense Disambiguation. *Comput. Linguist.* **40**(4), 837–881 (2014)
7. Raganato, A., Camacho-Collados, J., Navigli, R.: Word sense disambiguation: a unified evaluation framework and empirical comparison. In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pp. 99–110 (2017)
8. Lesk, M. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *Proceedings of the 5th SIGDOC*, New York, NY, pp. 24–26 (1986)
9. Chen, X., Liu, Z., Sun, M.: A unified model for word sense representation and disambiguation. In: *Proceedings of EMNLP*, pp. 1025–1035 (2014)
10. Camacho-Collados, J., Pilehvar, M.H., Navigli, R.: Nasari: integrating explicit knowledge and corpus statistics for a multilingual representation of concepts and entities. *Artif. Intell.* **240**, 36–64 (2016)
11. Agirre, E., Soroa, A.: Personalizing Pagerank for word sense disambiguation. In: *Proceedings of 12th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, pp. 33–41 (2009)
12. Tripodi, R., Pelillo, M.: A game-theoretic approach to word sense disambiguation. arXiv preprint [arXiv:1606.07711](https://arxiv.org/abs/1606.07711) (2016)
13. Zhong, Z., Ng, H.T.: It Makes Sense: a wide-coverage Word Sense Disambiguation system for free text. In: *Proceedings of the ACL System Demonstrations*, pp. 78–83 (2010)
14. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient Estimation of Word Representations in Vector Space. arXiv preprint [arXiv:1301.3781](https://arxiv.org/abs/1301.3781) (2013)
15. Taghipour, K., Ng, H. T. Semisupervised word sense disambiguation using word embeddings in general and specific domains. In: *Proceedings of NAACL HLT*, pp. 314–323 (2015)
16. Rothe, S., Schutze, H.: AutoExtend: extending word embeddings to embeddings for synsets and lexemes. In: *Proceedings of ACL*, Beijing, China, pp. 1793–1803 (2015)
17. Iacobacci, I., Pilehvar, M.H., Navigli, R.: Embeddings for word sense disambiguation: An evaluation study. In: *Proceedings of ACL*, Berlin, Germany, pp. 897–907 (2016)



18. Melamud, O., Goldberger, J., Dagan, I.: Learning generic context embedding with bidirectional LSTM. In: Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning (CoNLL), pp. 51–61 (2016)
19. Kageback, M., Salomonsson, H.: Word sense disambiguation using a bidirectional LSTM. arXiv preprint [arXiv:1606.03568](https://arxiv.org/abs/1606.03568) (2016)
20. Yuan, D., Richardson, J., Doherty, R., Evans, C., Altendorf, E.: Semi-supervised word sense disambiguation with neural models. In: Proceedings of COLING, pp. 1374–1385 (2016)
21. Simov, K., Osenova, P., Popov, A.: Using context information for knowledge-based word sense disambiguation. In: Dichev, C., Agre, G. (eds.) AIMSA 2016. LNCS (LNAI), vol. 9883, pp. 130–139. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-44748-3\\_13](https://doi.org/10.1007/978-3-319-44748-3_13)
22. Zhou, Z-H., Feng, J.: Deep Forest: towards an alternative to deep neural networks. In: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI-2017), pp. 3553–3559 (2017)



# Explorations in Sentiment Mining for Arabic and English Tweets

Tariq Ahmad<sup>1</sup>(✉) , Allan Ramsay<sup>1</sup> , and Hanady Ahmed<sup>2</sup> 

<sup>1</sup> The University of Manchester, Oxford Road, Manchester M13 9PL, UK  
tariq.ahmad@postgrad.manchester.ac.uk, allan.ramsay@manchester.ac.uk

<sup>2</sup> CAS, Arabic Department, Qatar University, 2713 Al Hala St, Doha, Qatar  
hanadyma@qu.edu.qa

<http://www.manchester.ac.uk>, <http://www.qu.edu.qa>

**Abstract.** Assigning sentiment labels to documents is, at first sight, a standard multi-label classification task. As such, it seems likely that standard machine learning algorithms such as deep neural networks (DNNs) will provide an effective approach. We describe an alternative approach, involving the construction of a weighted lexicon of sentiment terms, which significantly outperforms the use of DNNs. The moral of the story is that DNNs are not a universal panacea, and that paying attention to the nature of the data that you are trying to learn from can be more important than trying out ever more powerful general purpose machine learning algorithms.

**Keywords:** Sentiment mining · Shallow learning  
Multi-emotion classification

## 1 Introduction

The key features when trying to assign sentiment labels to documents are the words that they contain, and most approaches to this task involve either explicitly or implicitly constructing a sentiment lexicon. We explore below two ways of constructing such a lexicon, either by explicitly counting the occurrences of words in documents that have been assigned a (possibly empty) set of labels drawn from a standard set of sentiment names or by building a DNN with the set of words in the training set as input features and the sentiment names as output features. In the latter case, the lexicon is implicit in the weights that link the input layer, via the hidden layers, to the output layer. Such an implicit lexicon is often referred to as a ‘word embedding’: in essence, however, it is just a lexicon where each word is connected to a set of labels through a collection of arithmetical calculations.

---

This publication was made possible by the NPRP award [NPRP 7-1334-6-039 PR3] from the Qatar National Research Fund (a member of The Qatar Foundation). The statements made herein are solely the responsibility of the author[s].

Given that the task we are undertaking involves assigning a **set** of labels, we have to be careful about how we use DNNs. There are two obvious ways to proceed. (i) Given a set of features, it would be possible to train a set of classifiers, one per feature. Each classifier will return YES/NO for the feature it was trained on. Taken together the set of classifiers can be seen as a single multi-label classifier. We refer to classifiers trained in this way as ‘multi-DNNs’ (ii) Train a single classifier with a node in the output layer for each feature and then accept every label for which the output-layer node had an excitation level above some threshold. We refer to this kind of classifier as ‘single-DNNs’.

Traditionally, emotion classification tasks have been tackled by obtaining a large corpus, constructing feature sets and pre-processing in sophisticated ways and then making use of any number of black box training algorithms [7, 9]. Indeed, this approach is still prevalent nowadays [2, 4, 6]. We demonstrate that with a small sized corpus and without using any black box algorithms our results are comparable with systems that have been trained on millions (and sometimes billions) of tweets.

We show that the approach through explicitly constructing a lexicon substantially outperforms either of the DNN algorithms outlined above, and we end with some speculation about why this should be so.

## 2 Data

We used the data provided for the SemEval-2018 Task 1: Affect in Tweets task for English and Arabic [5]. The task was to assign a possibly empty set of sentiment labels, drawn from the set {anger, anticipation, disgust, fear, joy, love, optimism, pessimism, sadness, surprise, trust}, to each member of a collection of English and Arabic tweets. This consists of a training set of around 6.8K English tweets (109K words) and 2.2K Arabic ones (62K words) and development sets of around 886 tweets (13.9K words) and 586 tweets (15.8K words) respectively<sup>1</sup>. It is worth noting that Arabic tweets typically contain more words than English ones – 27 words per Arabic tweet, 16 words per English tweet. This reflects the fact that Arabic is typically written in a very condensed form, in which the diacritics (marks corresponding to short vowels and other phonetically relevant items) are omitted. This is similar to the use of ‘textese’ in English, wherein a text like ‘*This message is heavily abbreviated*’ might be written as ‘*This msg is hvly abrvted*’. The omission of diacritics from written Arabic introduces more ambiguity than is the case for English written as textese, which makes Arabic tweets harder to analyse than English ones. At the same time, it means that it is possible for there to be significantly more words in an Arabic tweet than an English one<sup>2</sup>,

<sup>1</sup> We are using the development set rather than the official test set here because the Gold Standard annotation of the test set has not been released, and hence we cannot use it for evaluating the systems described in this paper: our own scores on the official test sets are similar to our scores on the development sets – see Table 2 – hence the assumption is that the test and development sets have very similar characteristics.

<sup>2</sup> This data was collected when tweets were limited to 140 characters.

which makes the task easier, since the more words you have the more information about what the tweet says you have. As we will see, the results we obtain for the English and Arabic sets are extremely similar, hence we believe these two effects cancel each other out.

For both datasets, we carry out a number of preprocessing steps:

1. Identify emoticons and emojis. As their names suggest, emoticons and emojis are used to convey emotions, and hence are very significant for the current task. It can be tricky to identify them, since the set is not fixed, but they are undoubtedly important.
2. Tag and stem the remaining text. Stemming the words that appear in a tweet is challenging, since the vocabulary used in tweets is very open, and includes numerous slang items and (deliberate and accidental) misspellings, so that stemmers that rely on a fixed lexicon (e.g. Madamira for Arabic [8]) do not work well on tweets. We use a version of the stemmer described in [1] for Arabic and a stemmer based on the NLTK `casual_tokenize` function for English.

### 3 Machine-Learning Algorithms

We applied three machine learning algorithms to the SemEval data, as follows.

#### Weighted Conditional Probabilities

The first algorithm, WCP, collects the conditional probabilities  $P(W_i|S_j)$  that a tweet that expresses the sentiment  $S_j$  will contain the word  $W_i$  and constructs a sentiment lexicon as follows:

*Collect Raw Conditional Probabilities.* For all words  $W_i$ , all sentiments  $S_j$  collect the conditional probability  $P(W_i|S_j)$  that a tweet that expresses the sentiment  $S_j$  will contain the word  $W_i$ .

*Normalisation.* For each  $W_i$ , calculate the average value  $A(W_i)$  of the probabilities of that word over all sentiments, i.e.  $\sum_{j=1}^k P(W_i|S_j)/k$  and subtract that from each of the individual probabilities  $P(W_i|S_j)$ . This has the effect of downplaying the significance of words that occur equally frequently in tweets expressing different emotions.

*Skew.* For each  $P(W_i|S_j)$ , multiply by the variance  $\sqrt{\sum_{j=1}^k (P(W_i|S_j) - A(W_i))^2/k}$  of the probabilities of that word for any sentiment. This increases the significance of words with very skewed distributions.

These initial steps produce a lexicon which links words to sentiments, as in Table 1.

Positive words score highly in this table for positive sentiments (*'adorable'* scores well for joy and love), negative words score well for negative emotions

**Table 1.** Extracts from the English sentiment lexicon

	anger	anticipation	disgust	fear	joy	love	optimism	pessimism	sadness
...									
admire	-0.045	0.031	-0.046	0.006	0.009	0.088	-0.031	-0.091	-0.091
adorable	-0.091	-0.091	-0.091	-0.091	0.107	0.616	0.004	-0.091	-0.091
...									
outrage	0.226	-0.057	0.169	-0.009	-0.056	-0.066	-0.032	-0.049	-0.041
...									
sick	0.045	-0.061	0.042	0.052	0.007	-0.091	0.012	0.019	0.157
...									
the	-0	0.004	-0.001	0.001	0.001	-0.012	0.005	0.005	-0.004
...									

(‘*outrage*’ scores well for **anger** and **disgust**), neutral words score near 0 for everything (‘*the*’). The entry for ‘*sick*’ is interesting since it is a word that can be used to express almost diametrically opposed sentiments. It makes a positive contribution to a range of emotions, including **joy** and **optimism**, where its contribution is comparable to or greater than that of ‘*admire*’. We return to this in Sect. 4.

*Autocorrection.* We then autocorrect this lexicon by using it to reclassify the training data and removing any words that have a positive score in the lexicon for some sentiment but which actually occur in more tweets that do not express that sentiment than ones that do. This removes a small number of words, typically between 1% and 4%. Note that we carry out the reclassification on the original training data. This is methodologically sound – we are not using the training data for testing, we are reusing it as part of the overall training process. We carried out experiments where we set aside a portion of the training data for this purpose, but it turned out to be more effective to reuse the full set. It is more important to have as much data as possible for this purpose than to keep the training and retraining data separate.

*Threshold-Setting.* Finally, we set individual thresholds for each sentiment, to reflect the fact that some sentiments are more common than others and hence we should be more generous about predicting them. Again we set the thresholds by running the classifier on the original training data and choosing the optimal threshold for that data. Again this is a sound way to use the data – problems arise only if you mix up training and test data.

This series of steps performs fairly well – as noted earlier, we came second in the SemEval multi-label emotion task for Arabic, with a Jaccard score of 0.484 just 1% behind the winner with 0.489 – but some of the steps (particularly the decision to multiply the adjusted individual probabilities by  $P(W_i|S_j)$  by the variance from the average,  $\frac{\sqrt{\sum_{j=1}^k (P(W_i|S_j) - A(W_i))^2}}{k}$ ), are motivated purely by

the fact that they work. We therefore decided to try more traditional machine learning algorithms on the same data to see how well they performed.

## Deep Neural Nets

The task we are interested in is a multi-label classification. There are a number of standard algorithms which can be exploited for such tasks, either directly or by training a series of individual classifiers. We carried out a series of experiments with SVMs, a version of Naïve Bayes and deep neural nets. The SVM and NB algorithms performed extremely poorly on the SemEval data – the SVMs in particular simply chose the majority class in each case. We therefore concentrated on the DNN models<sup>3</sup>.

There are two obvious ways of using DNNs for multi-label tasks. A single DNN is trained, the excitation levels of all the output nodes are inspected and all the classes that surpass some specified threshold are chosen; or a DNN is trained for each class and each is applied to the data being classified. It is worth noting that the second strategy implicitly assigns individual thresholds to sentiments as in the strategy outlined above.

We carried out both strategies on our data, with a DNNs with three hidden layers, with the sizes of the three layers ranging from 50 to 400, from 20 to 80 and from 5 to 20 respectively. The numbers of nodes in the hidden layers made very little difference to the outcomes: the results below were obtained with (200, 40, 10) nodes in the hidden layers for the single DNN model and (25, 20, 5) for the multi-DNN model for Arabic, and (200, 80, 10) for the single and (25, 20, 5) for the multi-DNN model for English. Increasing the numbers of nodes in the layers beyond this led to overtraining and worse performance.

The results of these three strategies for the Arabic and English data from the SemEval task are given in Table 2. Mohammed et al. [5] include an SVM-based algorithm as a baseline, and we have included their Jaccard results for that on the SemEval test data in these tables<sup>4</sup>, since we were unable to obtain any useful results using SVMs, alongside the scores for WCP on the test data.

## Support Vector Machines

The SemEval competition results included a number of baselines, one obtained by training an SVM on the distributed training data and one by making random choices. The SVM baseline was higher than anything we have managed to achieve using SVMs, and hence we are including it here for comparison with the other

<sup>3</sup> We used the SciKit Multilayer-perceptron package, [http://scikit-learn.org/stable/modules/neural\\_networks\\_supervised.html](http://scikit-learn.org/stable/modules/neural_networks_supervised.html), with sparse arrays for the DNN experiments. While TensorFlow can be faster to train if you can take advantage of multi-core processing, the functionality of the two packages is very much the same and it is unlikely that using TensorFlow would have significantly improved the performance of the DNN models.

<sup>4</sup> We do not have their precision, recall and F-measure results.

approaches. The SemEval scores were obtained on the competition test set, for which we do not have the Gold Standard labels. Our own results were better on the Gold Standard than on the development set, suggesting that in some sense the Gold Standard test set was ‘easier’ than the development set. In order to obtain a fair comparison of the various approaches, then, we are including the results for WCP and the DNN-based approaches on the development set and for WCP and the SVM on the competition test set. These results are recorded in greyed out format in Table 2 to indicate that the SVM results were obtained from an external source.

**Table 2.** Scores on Arabic and English SemEval data

	Arabic				English			
	precision	recall	F	Jaccard	precision	recall	F	Jaccard
WCP	0.620	0.632	0.626	0.455	0.589	0.658	0.622	0.451
single DNN	0.601	0.537	0.567	0.396	0.624	0.488	0.587	0.416
multi-DNNs	0.611	0.528	0.567	0.395	0.624	0.546	0.582	0.411
(WCP				0.484				(0.508)
(SVM-unigrams				0.380				(0.442)

For both datasets the two DNN-algorithms have very similar scores. The multi-DNN versions, however, take **much** longer to train, since they require 11 distinct classifiers to be trained and tested. The DNNs seem to outperform the SVM, but it should be noted that the SVM was tested on the SemEval test set, which may be easier than the development set. The ratios between WCP’s scores on the development sets and the DNN’s scores are lower than the ratios between WCP’s scores on the test sets and the SVM’s score (in plain English: WCP beats the SVM by more than it beats the DNNs, the DNNs are better than the SVM).

## 4 Interpretation and Analysis

The weighted conditional probability algorithm outperforms both the DNN algorithms and the SVM on both sets of data. It would be interesting to know why this should happen, and while it is very difficult to see inside a DNN, we do have some speculations about how the differences arise.

We carried out a number of preprocessing steps on the data. The most important of these were stemming and identification of emojis, which each had a significant impact on the results. However, since we carried out exactly the same preprocessing steps before applying WCP or either of the DNN algorithms, the discrepancy in the results cannot be caused by these steps. It is impossible to know what preprocessing steps Mohammed et al. carried out, but it is at least worth noting they obtained substantially higher scores on English than on

Arabic, where we obtain similar scores with each algorithm for both languages, suggesting that our preprocessing steps for Arabic have a significant impact.

So what does make a difference? We will consider the main steps in our algorithm in turn: the accuracy of WCP at each stage on the two datasets is given in Table 3.

**Table 3.** Stages of WCP

	Arabic	English
Raw probabilities	0.324	0.340
Normalised	0.318	0.340
Skew	0.342	0.401
Autocorrection	0.370	0.431
Threshold setting	0.452	0.455

*Collect Raw Conditional Probabilities.* At this point we are simply collecting the conditional probability  $P(W_i|S_j)$  that a tweet that expresses the sentiment  $S_j$  will contain the word  $W_i$ . Given that what we want is  $P(S_j|W_i)$ , it is hardly to be expected that simply using the information gathered at this stage will be very useful.

*Normalisation.* At this point, words which are evenly distributed across sentiments are marked down. The score for ‘*the*’ at this point, for instance, becomes very close to 0 for all sentiments, since it occurs almost equally frequently in every sentiment, so subtracting the mean across all sentiments from the value for each sentiment leads to a set of scores that are very close to 0. This plays practically the same role as dividing by document frequency (or  $\log(\text{document frequency})$ ) in algorithms that use TF-IDF. Note that we do **not** weight things in terms of document frequency – downplaying words that contribute equally to all sentiments is carried out at this point.

*Skew.* This step introduces a degree of non-linearity into the process. A word’s score across all sentiments is boosted if the difference between the mean across all sentiments and the score for each sentiment **after** the mean has been subtracted from it is high. This effectively double-counts skew, because subtracting the mean gives neutral words a score close to 0, and then multiplying the actual score for each sentiment by this variance emphasises the sentiments for which the word is significant. This step simultaneously allocates words to sentiments and gives extra weight to words that are much more important for one sentiment than for the others. This has the effect of implicitly paying attention to correlations **and anti-correlations** between sentiments.

*Autocorrection.* The steps up to this point are approximate, and the weights assigned to individual words are not reliable, any more than any other strategy



can be trusted to give appropriate scores to individual words. We therefore run a sanity check to remove words whose scores are obviously unreliable. If a word has a positive score for some emotion, but actually appears in more tweets that do not express that emotion than ones that do, we give it a fixed negative score for that emotion. This is in some ways like the self-correction steps carried out in transformation-based learning [3], though at a rather simpler level. If it looks as though the classifier has made a mistake in the dictionary it fixes it (by altering the contribution that the given word makes to the sentiment that it is wrongly contributing to). This is an easy thing to do if the lexicon is explicit – much harder if the lexicon is encoded in a combination of weights in a DNN (i.e. is a word-embedding).

*Threshold-Setting.* Choosing a separate threshold for each sentiment has a significant effect on the overall score. This step is implicitly carried out by the multi-DNN models: when a DNN is used as a classifier, i.e. when there are two output nodes, one for YES and one for NO, it calculates the optimal excitation level for the YES node. Thus, when a set of DNNs is used for multi-classification, each one will indeed have its own threshold. For the multi-DNN, it seems likely that a single threshold is chosen for all the output nodes<sup>5</sup>. This step has a significant impact on the performance of the WCP algorithm, since it implicitly weighs up the likelihood of a given sentiment occurring at all, as well as being sensitive to the fact that different sentiments may be expressed in different ways. It cannot, however, be a major contributor to the difference between the WCP algorithm’s performance on the target data and the multi-DNN’s performance, since the multi-DNNs do also carry it out.

## 5 Conclusions

General purpose machine learning algorithms are, in general, good. If you do not know much about the data you are trying to learn from, then using some standard algorithm such as DNN or SVM is the easiest way to proceed. However, in specific cases there may be information that is hard to encode for a general purpose algorithm, and there may be interactions between classes that are difficult for such algorithms to capture. In the work described here we have shown that designing an algorithm that is targeted at the specific task under investigation (assignment of sentiment labels to Arabic and English tweets in the current study) can be more effective than just applying a standard general-purpose tool.

It may be, of course, that such algorithms have parameters that can be tweaked to improve their performance, but randomly tweaking parameters is a time-consuming and unilluminating activity. We believe that it is, in at least some cases, better to think about the nature of the problem and design an algorithm that works for it than to trust in the effectiveness of some pre-built black-box learning algorithm.

<sup>5</sup> The SciKit DNN tool is open-source, but that doesn’t mean that it’s easy to read the code and work out what is going on.

## References

1. Albogamy, F., Ramsay, A.: Unsupervised stemmer for Arabic tweets. In: 2nd Workshop on Noisy User-generated Text, pp. 78–84. The COLING 2016 Organizing Committee (2016). <https://aclanthology.info/pdf/W/W16/W16-3912.pdf>
2. Baziotis, C., Athanasiou, N., Chronopoulou, A., Kolovou, A., Paraskevopoulos, G., Ellinas, N., Narayanan, S., Potamianos, A.: NTUA-SLP at SemEval-2018 Task 1: predicting affective content in tweets with deep attentive RNNs and transfer learning. arXiv preprint [arXiv:1804.06658](https://arxiv.org/abs/1804.06658) (2018)
3. Brill, E.: Transformation-based error-driven learning and natural language processing: a case study in part of speech tagging. *Comput. Linguist.* **23**(4), 543–565 (1995)
4. Kim, Y., Lee, H., Jung, K.: AttnConvnet at SemEval-2018 Task 1: attention-based convolutional neural networks for multi-label emotion classification. arXiv preprint [arXiv:1804.00831](https://arxiv.org/abs/1804.00831) (2018)
5. Mohammad, S.M., Bravo-Marquez, F., Salameh, M., Kiritchenko, S.: SemEval-2018 Task 1: Affect in Tweets. In: Proceedings of International Workshop on Semantic Evaluation (SemEval-2018), New Orleans, LA, USA (2018)
6. Nam, J., Mencía, E.L., Kim, H.J., Fürnkranz, J.: Maximizing subset accuracy with recurrent neural networks in multi-label classification. In: Advances in Neural Information Processing Systems, pp. 5419–5429 (2017)
7. Pak, A., Paroubek, P.: Twitter as a corpus for sentiment analysis and opinion mining. In: LREC vol. 10, pp. 1320–1326 (2010)
8. Pasha, A., Al-Badrashiny, M., Diab, M., Kholy, A.E., Eskander, R., Habash, N., Pooleery, M., Rambow, O., Roth, R.: Madamira: A fast, comprehensive tool for morphological analysis and disambiguation of Arabic. In: Calzolari, N., Choukri, K., Declerck, T., Loftsson, H., Maegaard, B., Mariani, J., Moreno, A., Odiijk, J., Piperidis, S. (eds.) Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC 2014). European Language Resources Association (ELRA), Reykjavik, May 2014
9. Wang, W., Chen, L., Thirunarayan, K., Sheth, A.P.: Harnessing Twitter “big data” for automatic emotion identification. In: 2012 International Conference on Privacy, Security, Risk and Trust (PASSAT) and 2012 International Conference on Social Computing (SocialCom), pp. 587–592. IEEE (2012)



# Intent Detection System Based on Word Embeddings

Kaspars Balodis<sup>1,2</sup>✉ and Daiga Deksnė<sup>1</sup>

<sup>1</sup> Tilde, Vienības gatve 75A, Rīga 1004, Latvia  
{kaspars.balodis, daiga.deksne}@Tilde.lv

<sup>2</sup> Faculty of Computing, University of Latvia, Raiņa blvd. 19, Rīga 1586, Latvia

**Abstract.** Intent detection is one of the main tasks of a dialogue system. In this paper we present our intent detection system that is based on FastText word embeddings and neural network classifier. We find a significant improvement in the FastText sentence vectorization. The results show that our intent detection system provides state-of-the-art results on three English datasets outperforming many popular services.

**Keywords:** Intent detection · Dialog system · Word embeddings

## 1 Introduction

In recent years with the development of deep learning techniques, solutions with neural networks and machine learning are replacing more traditional ones. Dialogue systems are not an exception. There are several ways how machine learning methods based on neural networks are used in dialogue systems. They can be used for end-to-end systems [21, 22, 24, 26, 29] as well as separate components such as understanding user's input, generating the dialog system response, and dialogue management [10, 13, 14, 27]. Understanding user's intent is essential for successful interaction. Therefore, intent detection is one of the main tasks of a dialog system.

The intent detection task is typically formulated in the following setting. There are several possible predefined intents according to the dialogue system domain and scope and the system should determine which one is the most relevant to the user's input. It can be solved by manually created list of patterns and comparing if the user's input matches any pattern [6, 17, 23, 25]. However, this method is relatively limited and the latest approach is to use machine learning methods based on neural networks trained on example utterances [12, 28, 30].

The specifics of dialog systems are that generally only small amount of training data is available and the utterances are relatively short. It is further complicated by the specific nature of chat language such as poorly-structured sentences, presence of grammatical errors, usage of informal slang, abbreviations, etc.

## 2 Intent Detection

The input for the intent detection system is user’s utterance (typically a sentence or a question). The task of the system is determine which of several predefined intents is the most likely for the given utterance. The intent detection task can be approached by dividing it into separate subtasks. The utterance is sequentially preprocessed, vectorized and classified.

**Table 1.** Intent detection accuracy.

Method	Words	Stopwords removed	Real	auto1fix	Fixed
Wit.ai baseline	Original	No	36.44%	–	38.14%
FastText classifier	Original	No	28.14%	29.41%	30.59%
		Yes	26.69%	28.05%	29.07%
	Lemmatized	No	27.29%	28.31%	29.41%
		Yes	32.29%	34.41%	38.14%
FastText vectors	Original	No	23.73%	23.73%	24.49%
		Yes	28.14%	28.64%	30.68%
	Lemmatized	No	25.93%	26.78%	27.71%
		Yes	36.95%	39.24%	41.86%
FastText upgraded vectors	Original	No	38.22%	38.47%	38.73%
		Yes	36.19%	37.03%	38.47%
	Lemmatized	No	35.25%	36.61%	37.97%
		Yes	<b>38.73%</b>	<b>40.76%</b>	<b>43.39%</b>

### 2.1 Preprocessing

The preprocessing step typically consists of one or more of the following tasks:

- tokenization – in this step punctuation marks, words, email addresses, links, numbers, abbreviations, etc. are properly separated into distinct tokens,
- automatic error-correction – the user grammatical errors are attempted to be programmatically corrected,
- truecasing or lowercasing – the text is converted into lowercase or true case (e.g. *usa* → *USA*),
- removal of punctuation marks and other symbols,
- lemmatization – words are transformed into canonical form,
- removal of stopwords – insignificant words (such as *a*, *and*, *I*, *or*, *to*, etc.) are removed, either from a predefined stoplist or from a dynamically-generated list based on the training data.

## 2.2 Vectorization

During vectorization the utterance is transformed from a textual form to a vector so it can be input to a machine learning algorithm. Common types of vectorization include *One-Hot* vectorization and word embeddings. In *One-Hot* vectors each dimension of the vector corresponds to a word in dictionary. The utterance is vectorized so that the value in the respective dimension is 1, if a word appears in the utterance, and 0 – otherwise.

Word embedding is a different approach to vectorization where words are mapped to vectors of real numbers. This continuous representation can then be used to perform a wide variety of natural language processing tasks.

There are several tools for generating word embeddings such as *gloVe* [18], *word2vec* [15], *fastText* [2] and others [19,20].

## 2.3 Classification

During classification the vectorized utterances are classified, typically using some machine learning algorithm such as some kind of neural network. The input for the classifier is the vectorized representation of the utterance and the output is a probability distribution over the possible intents.

# 3 Experimental Setup and Results

## 3.1 Dataset

We use an in-house dataset from a customer service system in Latvian language. It has 121 defined intent. The training set consists of expert-written examples with approx. 10 utterances per intent, in total 1231 utterances.

There are two variants of the test set. The test set *Real* is derived by 1000 randomly selected real user questions from the production system and by manually assignment the most appropriate intent for each question. In this way only for 236 utterances were classified, because others were not in the domain of the dialog system. The test set *Fixed* is the same as *Real* but with the grammatical errors manually corrected. Dataset *auto1fix* is obtained from the set *Real* by automatically correcting the errors using methods described in the subsection about automatic error correction.

## 3.2 Preprocessing

The utterances are tokenized and lowercased, and all non-alphanumeric characters are removed.

We compare several variants of preprocessing. The words in utterances are either lemmatized or not. The stopwords obtained using a method from [1] are either removed or not removed.

### 3.3 Automatic Error Correction

Type of text generated by user in a chatroom conversation or in some social media environment tend to be quite distant from the literary language. Several authors consider normalizing text prior to performing some specific NLP task [4, 7]. To check how the quality of intent detection module is influenced by the quality of the training data we have created a module for noisy chatroom text correction. In this module, we integrated three different tools. The text correction tool allows to fix text with help of regular expressions, to fix it with a spelling checker suggestions, to use a grammar checker or to use all three tools. If all three methods are chosen, at first the regular expression module is applied, then the spelling checking module, and finally the grammar checking module.

We have created a list of 770 regular expressions. The regular expression module looks for a wrong pattern in a text phrase and replaces it with a correct one. The errors covered can be divided into several groups.

Many users do not use diacritics in a chatroom conversation although in Latvian a large part of words have them. 47% of the words contain diacritic marks measured on Latvian UD Treebank v2.0 [16]. There are four long vowels in Latvian: *ā, ē, ī, ū*. In a chatroom conversation, users might write them without a diacritic mark: *a, e, i, u*, or by doubling them: *aa, ee, ii, uu*. Also consonants *g, k, l, n, c, z, s* are written inconsistently as *g, k, l, n, c, z, s* or *gj, kj, lj, nj, ch, zh, sh*.

The next common mistake is vowel dropping. Every syllable in Latvian should contain a vowel. There is a list of words in which the regular expressions try to fix this type of error, for example, words *kpc, dzgn, mnpr, kkad* should be *kāpēc, diezgan, manuprāt, kaut kad*.

Some other errors covered by regular expressions - words written together should be written separately or vice versa, wrong letter, dropped letter, extra letter, colloquial speech, abbreviated word.

By analyzing Twitter data we have discovered some peculiarity in a lexicon. Users are very creative. There are many neologisms, unusual compounds, English words with a Latvian ending or some slang. In a future, some of the compounds should be added to the spelling checker dictionary.

The spelling checker module not only checks if a word is misspelled but also generates some suggestions for correction. Every misspelled word in a text phrase is replaced by the first suggested correction if there is one. The spelling checker is based on a general language lexicon.

The grammar checking module looks for the formatting errors, orthography errors, morphology and syntax errors, punctuation errors as well as for some style errors [5]. The grammar checker is developed as a rule-based system for correction of the general texts.

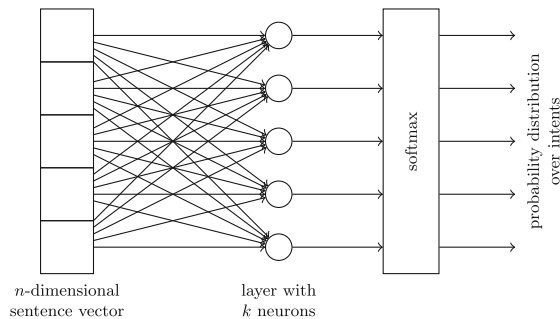
### 3.4 Baseline

As a baseline we use the Wit.ai service<sup>1</sup> which is one of few popular chatbot creation services that supports Latvian language. As seen in Table 1 the results for the *Fixed* dataset are better than for the *Real* dataset as could be expected. The overall accuracy is quite low which could be explained by the heterogenous nature of the training and test sets.

### 3.5 FastText Classifier and Its Improvements

The first system we test is the FastText tool for vectorization and classification [9]. We use 100-dimensional word embeddings that are pretrained on a large text corpus containing 82M utterances (1368M words). For word embeddings the large corpus is also preprocessed – tokenized, lowercased and nonalphanumeric characters removed.

As demonstrated in Table 1 this approach does not beat baseline. One of the problems with FastText classifier is that unseen words are not taken into consideration. However, they could possibly be misspelled versions or rarer forms of known words and could be vectorized using FastText vectorization which uses subword-level information. Therefore, we examine a method that only uses the vectorization part of FastText and makes the classification with a simple neural network with one softmax layer as shown in Fig. 1.



**Fig. 1.** The architecture of the neural network.

It can be seen in Table 1 that the accuracy has increased in the setting where stopwords are removed and decreased significantly when stopwords are not removed. Therefore it might be inferred that this solution is very sensitive to stopwords. As explained in the following, this is not a coincidence and can be explained by the implementation of the FastText sentence vectorization.

<sup>1</sup> <https://wit.ai/>.

FastText classifier calculates the sentence vector as the average of the word vectors (normalized by their length). The sentence vector  $W$  is calculated as

$$W = \frac{1}{m} \sum_{i=1}^m \frac{1}{\|w_i\|} \cdot w_i,$$

where  $w_1, \dots, w_m$  are the vectors of the sentence words and  $\|w_i\|$  is the length of the vector  $w_i$ . The normalization by the length is a large source of noise. Less important words typically have smaller vector length as seen in the Table 2. Therefore, normalizing by length assigns them a relatively higher weight in the resulting sentence vector.

**Table 2.** Lengths of the word vectors.

Word	English translation	Vector length
un	and	1.78
kāda	what	2.76
ir	is	2.27
mana	my	3.12
ip	IP	3.96
adrese	address	4.91

The accuracy can be significantly improved with a modified FastText vectorizer that does not normalize the word vectors and creates the sentence vector as  $W = \frac{1}{m} \sum_{i=1}^m w_i$ . As it demonstrated in Table 1 such modification gives a significant improvement in the intent detection accuracy.

### 3.6 FastText Increase of Dimensionality

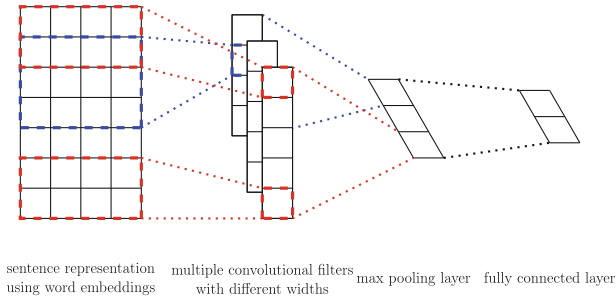
We examine whether an increase in the number of dimensions for the word embedding leads to higher accuracy. The setting we use is the original sentence without lemmatization or stopword removal. We examine word embeddings with 100, 300, 500, 1000 dimensions. As it can be seen in Table 3, increase of the number of dimensions leads to increase of the intent detection accuracy.

However, from a practical perspective for the vectorization to happen fast the word embedding model should be loaded in computer’s RAM. The size of model grows linearly with the number of dimensions. So 100-, 300-, 500-, 1000-dimensional word embeddings occupy 2.0GB, 6.0GB, 10.1GB, 20.1GB of space, respectively, therefore putting corresponding requirements for the computer’s RAM.



**Table 3.** Intent detection accuracy for different number of dimensions for the FastText word embeddings.

Dimensions	Real	Fixed
100	38.22%	38.73%
300	40.51%	43.05%
500	41.91%	44.24%
1000	<b>44.32%</b>	<b>47.46%</b>

**Fig. 2.** The architecture of the convolutional network.

### 3.7 Convolutional Network

We examine classification with a convolutional network with architecture similar to [11]. The architecture of the network we use is presented in Fig. 2.

For practical reasons 300-dimensional word embeddings are used. We examine filter widths from 1 to 5 and the number of filters from 100 to 1000. We try two regularization mechanisms – with L1 weight regularization and with a dropout layer. The regularization with a dropout layer shows better results.

In Table 4 the change in accuracy is shown for the convolutional network with dropout rate 0.5 compared to the result of the neural network classifier on 300-dimensional word embeddings (40.51%). It can be observed that depending on the parameters (and on the run) the results are different. However, on average convolutional network shows improvement in accuracy over simple neural network.

### 3.8 Language Simplification

We evaluated a method to deal with user errors in diacritical marks (omission or, for example, writing long vowels as a double letter) by simplification of the language. We replaced the letters with diacritical marks or their most common substitutions with a simple letter according to the rules in Table 5.

However, this did not give an improvement to the accuracy. For the simple neural network on 300-dimensional word embeddings on the simplified language the accuracy was 39.84% which is a little lower than for the *Real* dataset

**Table 4.** Accuracy difference of the convolutional network.

filters \ ws	1,	1,2	1,2,3	1,2,3,4	1,2,3,4,5
100	-4.70%	3.43%	0.04%	0.34%	-2.92%
200	1.31%	-1.10%	1.02%	-2.50%	0.00%
300	1.23%	1.99%	-0.93%	0.68%	4.32%
500	1.57%	-0.97%	-0.13%	2.80%	0.21%
700	3.73%	-1.82%	3.98%	-0.13%	-2.71%
1000	-0.30%	5.17%	0.55%	-0.13%	-1.23%

**Table 5.** Language simplification replacement rules.

Letters	Replace with
aa	ā a
ee	ē e
ii	ī i
uu	ū u
ch	č c
gj	g g
kj	ķ k
lj	ļ l
nj	ņ n
sh	š s
zh	ž z
	ō o
	ŗ r

(40.51%). The results for the convolutional network with dropout 0.5 can be seen in Table 6. On average the accuracy is a bit lower (compared to Table 4).

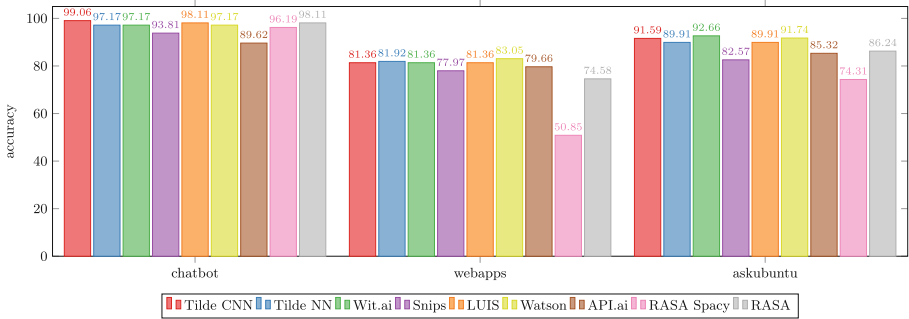
## 4 Comparison with Other Systems

In this section we compare our solution with other popular dialog systems on publicly available datasets. In the paper [3] authors compare the quality of intent and entity detection of a few popular dialog systems, namely, Microsoft LUIS, IBM Watson, Google API.ai (now called Dialogflow), and RASA, on three datasets. Furthermore, in the Snips Github repository<sup>2</sup> also Snips.ai and RASA

<sup>2</sup> <https://github.com/snipsco/nlu-benchmark>.

**Table 6.** Accuracy difference of the convolutional network on the simplified language.

filters \ ws	1,	1,2	1,2,3	1,2,3,4	1,2,3,4,5
100	-3.69%	-3.43%	-1.74%	2.46%	-0.68%
200	1.02%	0.64%	2.63%	0.59%	-2.33%
300	-0.30%	-2.71%	0.89%	-0.93%	-0.72%
500	1.40%	2.67%	-1.40%	-1.02%	-0.38%
700	4.66%	-0.34%	1.44%	-1.69%	-0.38%
1000	2.03%	2.16%	2.29%	-5.59%	0.76%

**Fig. 3.** Comparison of the intent detection accuracy of different services.

Spacy is evaluated. We also add Wit.ai to the comparison as in [3] it was excluded due to problems with automatic import.

In [3] the authors combine the results of intent detection and entity recognition and calculate the F-score on the aggregated result. However, from the raw scores it is also possible to obtain other measures. As we focus exclusively on intent detection and the relative frequency of different intents are similar, the measure of intent detection accuracy is more appropriate to compare the different systems. We use the same training and test sets as in [3]. The comparison results are shown in the Fig. 3. We can see that our system achieves results that are in line with other services.

For the vectorization we use publicly available FastText word embeddings trained on English Wikipedia [8]. As the corpus used for obtaining the word embeddings is not lowercased, but the utterances are lowercased these results theoretically could be improved by properly preprocessing also the corpus used for word embeddings.

**Acknowledgments.** The research has been supported by the European Regional Development Fund within the project “Neural Network Modelling for Inflected Natural Languages” No. 1.1.1.1/16/A/215.


## References

1. Balodis, K.: On stopword identification in dialog systems, submitted for publication
2. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. *Trans. Assoc. Comput. Linguist.* **5**, 135–146 (2017)
3. Braun, D., Hernandez-Mendez, A., Matthes, F., Langen, M.: Evaluating natural language understanding services for conversational question answering systems. In: *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pp. 174–185 (2017)
4. Damnati, G., Auguste, J., Nasr, A., Charlet, D., Heinecke, J., Béchet, F.: Handling normalization issues for part-of-speech tagging of online conversational text. In: Calzolari (Conference Chair), N., (eds.) *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. European Language Resources Association (ELRA), Paris, France, May 2018
5. Dekšne, D., Skadina, I.: Error-annotated corpus of Latvian. In: *Baltic HLT*, pp. 163–166 (2014)
6. Fonte, F., Carlos, J., Rial, B., Nistal, M.L.: TQ-Bot: an AIML-based tutor and evaluator bot. *J. Univ. Comput. Sci.* **15**(7), 1486–1495 (2009)
7. van der Goot, R., van Noord, G.: MoNoise: modeling noise using a modular normalization system. *Comput. Linguist. Neth. J.* **7**, 129–144 (2017)
8. Grave, E., Bojanowski, P., Gupta, P., Joulin, A., Mikolov, T.: Learning word vectors for 157 languages. In: *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)* (2018)
9. Joulin, A., Grave, E., Bojanowski, P., Mikolov, T.: Bag of tricks for efficient text classification. arXiv preprint [arXiv:1607.01759](https://arxiv.org/abs/1607.01759) (2016)
10. Kalchbrenner, N., Blunsom, P.: Recurrent convolutional neural networks for discourse compositionality. arXiv preprint [arXiv:1306.3584](https://arxiv.org/abs/1306.3584) (2013)
11. Kim, Y.: Convolutional neural networks for sentence classification. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1746–1751 (2014)
12. Liu, B., Lane, I.: Attention-based recurrent neural network models for joint intent detection and slot filling. arXiv preprint [arXiv:1609.01454](https://arxiv.org/abs/1609.01454) (2016)
13. Liu, C., Xu, P., Sarikaya, R.: Deep contextual language understanding in spoken dialogue systems. In: *Sixteenth Annual Conference of the International Speech Communication Association* (2015)
14. Lowe, R., Pow, N., Serban, I.V., Pineau, J.: The Ubuntu dialogue corpus: a large dataset for research in unstructured multi-turn dialogue systems. In: *16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, p. 285 (2015)
15. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint [arXiv:1301.3781](https://arxiv.org/abs/1301.3781) (2013)
16. Náplava, J., Straka, M., Straňák, P., Hajič, J.: Diacritics restoration using neural networks. In: Calzolari (Conference Chair), N., (eds.) *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. European Language Resources Association (ELRA), Paris, France, May 2018
17. Neves, A.M., Barros, F.A., Hodges, C.: iAIML: a mechanism to treat intentionality in AIML chatterbots. In: *18th IEEE International Conference on Tools with Artificial Intelligence 2006, ICTAI 2006*, pp. 225–231. IEEE (2006)
18. Pennington, J., Socher, R., Manning, C.: Glove: global vectors for word representation. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543 (2014)

19. Řehůřek, R., Sojka, P.: Software framework for topic modelling with large corpora. In: Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks, pp. 45–50. ELRA, Valletta, Malta, May 2010. <http://is.muni.cz/publication/884893/en>
20. Sales, J.E., Souza, L., Barzegar, S., Davis, B., Freitas, A., Handschuh, S.: Indra: A word embedding and semantic relatedness server. In: Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018). European Language Resources Association (ELRA), Miyazaki, Japan, May 2018
21. Serban, I.V., Sordoni, A., Bengio, Y., Courville, A.C., Pineau, J.: Building end-to-end dialogue systems using generative hierarchical neural network models. In: AAAI, vol. 16, pp. 3776–3784 (2016)
22. Shang, L., Lu, Z., Li, H.: Neural responding machine for short-text conversation. arXiv preprint [arXiv:1503.02364](https://arxiv.org/abs/1503.02364) (2015)
23. Shawar, B.A., Atwell, E.: Machine learning from dialogue corpora to generate chatbots. *Expert Update J.* **6**(3), 25–29 (2003)
24. Vinyals, O., Le, Q.: A neural conversational model. arXiv preprint [arXiv:1506.05869](https://arxiv.org/abs/1506.05869) (2015)
25. Wallace, R.: *The Elements of AIML Style*. Alice AI Foundation (2003)
26. Wen, T., et al.: A network-based end-to-end trainable task-oriented dialogue system. In: 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017-Proceedings of Conference, vol. 1, pp. 438–449 (2017)
27. Wen, T.H., Gasic, M., Mrkšić, N., Su, P.H., Vandyke, D., Young, S.: Semantically conditioned LSTM-based natural language generation for spoken dialogue systems. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp. 1711–1721. Association for Computational Linguistics (2015). <https://doi.org/10.18653/v1/D15-1199>, <http://www.aclweb.org/anthology/D15-1199>
28. Xu, P., Sarikaya, R.: Convolutional neural network based triangular CRF for joint intent detection and slot filling. In: IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU) 2013, pp. 78–83. IEEE (2013)
29. Yang, X., Chen, Y.N., Hakkani-Tür, D., Crook, P., Li, X., Gao, J., Deng, L.: End-to-end joint learning of natural language understanding and dialogue manager. In: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) 2017, pp. 5690–5694. IEEE (2017)
30. Yao, K., Peng, B., Zhang, Y., Yu, D., Zweig, G., Shi, Y.: Spoken language understanding using long short-term memory neural networks. In: Spoken Language Technology Workshop (SLT), 2014 IEEE, pp. 189–194. IEEE (2014)



# Indirect Association Rules Mining in Clinical Texts

Svetla Boytcheva<sup>(✉)</sup> 

Institute of Information and Communication Technologies,  
Bulgarian Academy of Sciences, Sofia, Bulgaria  
svetla.boytcheva@gmail.com

**Abstract.** This paper presents a method for structured information extraction from patient status description. The proposed approach is based on indirect association rules mining (IARM) in clinical text. This method is language independent and unsupervised, that makes it suitable for applications in low resource languages. For experiments are used data from Bulgarian Diabetes Register. The Register is automatically generated from pseudonymized reimbursement requests (outpatient records) submitted to the Bulgarian National Health Insurance Fund in 2010–2016 for more than 5 million citizens yearly. Experiments were run on data collections with patient status data only. The great variety of possible values (conditions) makes this task challenging. The classical frequent itemsets mining algorithms identify just few frequent pairs only even for small minimal support. The results of the proposed IARM method show that attribute-value pairs of anatomical organs/systems and their condition can be identified automatically. IARM approach allows extraction of indirect relations between item pairs with support below the minimal support.

**Keywords:** Data mining · Text mining · Health informatics

## 1 Motivation

Structured information extraction of patient status is important for many Healthcare Management tasks like diagnosis, treatment effect assessment, patient profiling, etc. Usually outpatient records (ORs) contain detailed description of patient status as free text only. Raw clinical text usually used telegraphic style of the patient status, rather than full sentences. There are also a lot of typo and abbreviations, that makes difficult usage of standard syntax parsers. Without of resources like lexicons and ontologies with medical terminology the solution of task for complex events and relations extractions is almost impossible. Still there is a lack of resources and natural language processing (NLP) tools for non-English clinical texts processing [21] albeit there is an increasing research efforts in this area. There are still non existing translations of SNOMED<sup>1</sup>,

<sup>1</sup> SNOMED, <https://www.snomed.org/>.

Medical Subject Headings (MeSH)<sup>2</sup> and Unified Medical Language System (UMLS)<sup>3</sup> for the majority of languages.

Manual annotation of many clinical text documents is time and effort consuming task. The rich vocabulary of the medical terminology is one of the major obstacles for scalability of methods developed and evaluated on the golden standard basis. Thus for NLP of clinical texts there are needed language independent methods that are unsupervised and do not rely on resources. In previous research we shown that some data mining algorithms for frequent patterns mining and frequent sequence mining can be used efficiently to extract structured information for risk factors and some complex relations between diagnosis [4, 5].

The classical frequent itemsets mining and association rules generation algorithms identify just few frequent pairs only even for small minimal support. The infrequent data contain interesting and important information. Indirect association rules mining (IARM) identify relations between items  $\{X, Y\}$  that rarely occur together in the same transaction. In case the presence in transaction of both items  $X$  and  $Y$  depend on presence of some other set  $M$ , then it is said that  $X$  and  $Y$  are *indirectly associated* via  $M$  [24]. The set  $M$  is called *mediator*. In text documents processing indirectly associated words corresponds to words, used in different context of other word, or synonyms, or antonyms. We propose a method based on IAR mining of clinical text. The aim of this research is to identify attribute-value pairs of patient status descriptions.

The paper is structured as follows: Sect. 2 briefly overviews the research in the area; Sect. 3 describes the data collection of clinical text used in the experiments; Sect. 4 presents the formal presentation of the problem and describes in details the proposed method for mining indirect association rules in clinical text; Sect. 5 shows experimental results and discusses the method applications; Sect. 6 contains the conclusion and sketches some plans for future work.

## 2 Related Work

Recently many data mining approaches were successfully used in NLP tasks [18]. Manimaran and Velmurugan show how Apriory [2] algorithm can be used for text mining with application in medical informatics.

One of the earliest comprehensive studies of IAR algorithms for mining transaction data were presented by Tan and Vipin [22–24]. They present different applications of the method for text, retail data, stock market data, Web click-stream data.

Chen et al. [6] present an algorithm MG-Growth for temporal indirect association that takes into account the lifespan of items. Ha et al. [11] demonstrate how IARM approach can be used to find hidden correlation among multimedia semantic concepts.

Some attempts to define more efficient measures in IARM task was presented. Abdullah et al. [1] introduce a new measure Definite Factor to indicate the degree

<sup>2</sup> Medical Subject Headings—MESH, <https://www.nlm.nih.gov/mesh/>.

<sup>3</sup> UMLS, <https://www.nlm.nih.gov/research/umls/>.

of certainty of association rules. Hamano and Sato [12] define a measure for mining Negative Association rules effectively without domain knowledge. Wan and An [26] describe HI-mine - an efficient algorithm, based on a novel data structure, called HIstruct. Kazienko [17] proposes IDARM\* Algorithm that uses pre-calculated direct rules for complete IARM. He presents application of the proposed algorithm for recommendation system for web pages.

There are many applications of IARM in the domain of medical informatics. Tsuruoka et al. [25] present a real-time text mining system FACTA+ for finding and visualizing indirect associations between biomedical concepts from MEDLINE abstracts. Another interesting application of IARM in medical informatics is presented by Kang and Wagacha [16]. They investigate ICD-9<sup>4</sup> disease diagnosis associations in big collection of Electronic Health Records. Wright et al. [27] apply IARM for identifying associations between medications, laboratory results and problems.

### 3 Materials

Bulgarian National Diabetes Register [3] is automatically generated from a data repository of about 262 million pseudonymized outpatient records (ORs) submitted to the Bulgarian National Health Insurance Fund (NHIF) in period 2010–2016 for more than 5 million citizens yearly. The NHIF collects for reimbursement purpose all ORs produced by General Practitioners and the Specialists from Ambulatory Care for every patient clinical visit.

ORs are stored in the repository as semi-structured files with predefined XML-format. Structured information describe the necessary data for health management like visit date and time; pseudonymized personal data and visit-related information, demographic data (age, gender, and demographic region), etc. All diagnoses are presented by ICD-10<sup>5</sup> codes and the name according to the standard nomenclature. The most important information concerning patient status and case history is provided like free text. ORs contain paragraphs of unstructured text provided as separate XML tags: “Anamnesis” (Disease history), “Status”, “Clinical tests”, and “Prescribed treatment”.

For experiments is used a data collections of ORs from Bulgarian National Diabetes Register. For all experiments are used raw ORs, without any preprocessing due to the lack of resources and annotated corpora. The text style for unstructured information is telegraphic. Usually with no punctuation and a lot of noise (some words are concatenated; there are many typos, syntax errors, etc.). The Bulgarian ORs contain medical terminology both in Latin and Bulgarian. Some of the Latin terminology is also used with Cyrillic transcription. Bulgarian language uses inflections. For some Latin medical terms in addition to the original Latin and Greek suffixes are used also prefixes and suffixes specific for Bulgarian language. This makes the task for natural language processing of the clinical text in Bulgarian quite challenging.

<sup>4</sup> <http://icd9.chrisendres.com/>.

<sup>5</sup> <http://apps.who.int/classifications/icd10/browse/2016/en#/>.



The ORs are written in telegraphic style with phrases rather than full sentences. Usually the ORs list attribute-value ( $A$ - $V$ ) pairs - anatomical organ/system and its status/condition. Attribute names contain phrases and abbreviations in Cyrillic and Latin. Values can be long descriptions in case of status complications. The order of  $A$ - $V$  pairs can vary and parts of the value descriptions can surround the attributes. It is also possible that some attributes share the same value. Sample configurations are shown below.

$$\begin{aligned} &A_1V_1, \dots, A_nV_n|V_1A_1, \dots, V_nA_n \\ &V_1\dots V_kAV_{k+1}\dots V_n \\ &A_1, A_2, \dots, A_nV|VA_1, A_2, \dots, A_n. \end{aligned}$$

## 4 Methods

### 4.1 Indirect Association Rules Mining

The vocabulary used in all ORs of a data collection  $S$  will be called *items*  $V = \{v_1, v_2, \dots, v_n\}$ . For the collection  $S$  we extract the set of all different ORs  $R = \{r_1, r_2, \dots, r_N\}$ , where  $r_i \subseteq V$ . This set corresponds to transactions; the associated unique transaction identifiers (*tids*) will be called **pids** (patient identifiers). Each patient interaction with a doctor is viewed as a single OR in  $R$ .

Preliminary analysis of N-grams in data collections with AntConc tool<sup>6</sup> show that the majority of the identified N-gram candidates are mainly cliché phrases. There are only seldom examples (about 15%) for true positive N-grams.

Therefore we treat documents as bags of words rather than sequences; they are transformed to itemsets with single word occurrences only.

Given a set of pids  $S$ , the support of an itemset  $I$  is the number of pids in  $S$  that contain  $I$ . We denote it as  $supp(I)$ . We define a threshold called *minsup* (minimum support). Frequent itemset (FI)  $I$  is one with at least minimum support count, i.e.  $supp(I) \geq minsup$ . The task of FPM of  $S$  is to find all possible frequent itemsets in  $S$ .

The following definition for indirect association rules was proposed by Tan and Vipin [24]:

**Definition 1** (*Indirect associated pair*). An itempair  $\{X; Y\}$  is indirectly associated via a mediator set  $M$  if the following conditions hold :

1.  $sup(X; Y) < minsup$  (*Itempair Support Condition*)
2. There exists a non-empty set  $M$  such that  $\forall M_i \in M$ :
  - (a)  $sup(X; M_i) \geq ts$  ;  $sup(Y; M_i) \geq ts$  (*Mediator Support Condition*).
  - (b)  $d(X; M_i) \geq conf$  ;  $d(Y; M_i) \geq conf$  where  $d(p; Q)$  is a measure of the dependence between  $p$  and  $Q$  (*Dependence Condition*).

Condition (1) is needed because an indirect association is significant only if there are seldom occurrences of both items in ORs, i.e. negatively correlated.

<sup>6</sup> Laurence, A. AntConc (Version 3.4. 4w)(Computer software). Tokyo, Japan:Waseda University. <http://www.laurenceanthony.net/> (2014).

Condition (2a) is needed to guarantee statistical significance of the mediator set  $M$ . Condition (2b) is needed to guarantee that only items highly dependent on both  $X$  and  $Y$  are used to form the mediator set  $M$ . Items in  $M$  form close neighborhood.

## 4.2 Automatic Extraction of Patient Status

The workflow of the proposed method for automatic extraction of patient status from clinical text is shown on Fig. 1. The processes are grouped in three main subtasks:

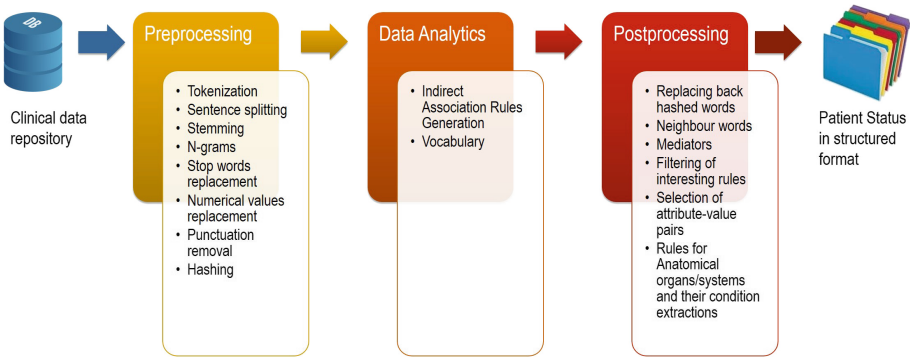


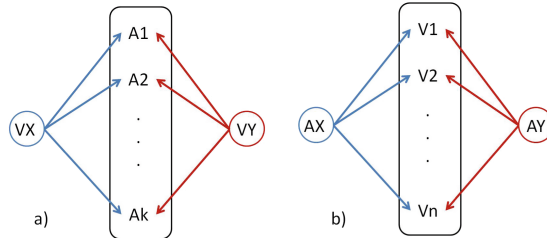
Fig. 1. Workflow

- **Preprocessing** - converts raw ORs data into itemsets. This subtask starts with tokenization of “Status” field only of ORs. The texts in “Status” section is usually short and presented in telegraphic style and without punctuation. Thus the majority of ORs are considered as single sentence only. The next step is stemming that is necessary for reducing inflected words. For this step is used Bulstem [20]. Unfortunately it doesn’t works for part of the medical terminology which is in Latin but transliterated in Cyrillic. For next step N-grams identification is used AntConc as we mentioned in the previous subsection. Some of the top ranked N-grams are replaced in the text by single item. After that stop words are identified and removed. ORs contain many numerical values in the “Status” section, like blood pressure, Body mass index, height, weight, etc. All numerical values are replaced by symbol NUM. The punctuation symbols are removed from the text, because they don’t matter when ORs are processed as bag of words. Finally itemsets are generated by applying hashing - replacing each item (word/N-gram) by unique ID and removing duplicates. The itemset is stored in increasing order of the items ID in order to fasten the data mining process.

- **Data Analytics** - applies data mining methods for IARM, FPM and Association Rules (ARs) . For experiments are used Java implementations of the algorithms IndirectRules [24], FPMMax [9], and FPGrowthARL [13] from SPMF<sup>7</sup> (Open-Source Data Mining Library) [8].
- **Postprocessing** - packs the result data. The first step is to return back the hashed items in the indirect association rules and frequent itemsets. Presenting results for all indirect pairs and the corresponding mediators. Identification of attributes and their values.

## 5 Experiments and Results

The associations of the following types are in primary interest of our task for patient status extraction (Fig. 2). In Fig. 2b AX and AY are two attributes and they share mediator set with common values. Such pairs of attributes can be the Bulgarian and Latin terms used for an anatomical organ system.



**Fig. 2.** Attribute-Value indirect association rules

### **Example 1:**

AX= черен дроб и слезка (Bulgarian) (*liver and spleen*)

AY=hepar et lien (Latin) (*liver and spleen*)

M=б.о. (*without complications*), не се палпират увеличени (*do not palpate enlarged*)

Although “liver and spleen” describes a pair of attributes they can not be split further, because there will be violation of the condition (1) of the Definition 1, both indirectly associated items not to occur frequently together in ORs. There is a direct association between them and they can be identified as frequent pair.

Another case for such interesting indirect pair is when an abbreviation and full attribute name are used.

**Example 2:** For example for Cardiovascular system (CVS) сърдечно-съдовасистема (CCC):

AX=CCC(*CVS*)

AY=сърце (*heart*), or

AY=cor (Latin)(*heart*).

<sup>7</sup> <http://www.philippe-fournier-viger.com/spmf/index.php>.

In Fig. 2a VX and VY are two values and they share mediator set with common attributes. Usually such values describe general conditions and observations.

**Example 3:**

VX = добро(*good*)

VY = увредено(*impaired*)

M= общо състояние (*general condition*)

**Example 4:**

VX = не се палпират увеличени (*do not palpate enlarged*)

VY = увеличени(*enlarged*)

M=лимфни възли (*lymph nodes*) , черен дроб и слезка (*liver and spleen*)

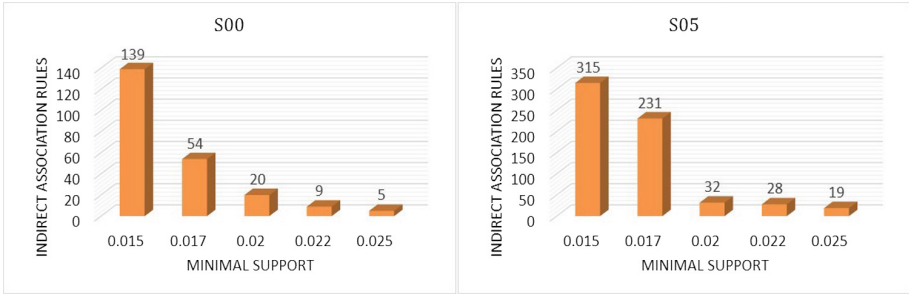
**Table 1.** Summary of data parameters and results

	S00 (General Practitioners)	S05 (Endocrinology)
Patients	10,000	10,000
ORs	123,247	14,753
Sentences	791,420	157,448
Items (vocabulary)	8,408	2,111
Minimal support (minsup)	0.011	0.011
Mediator support (ts)	0.7	0.7
Minimal confidence (conf)	0.7	0.7
Maximal frequent patterns	81	43
Indirect pairs	195	2,670
Association rules	1,236	7,121

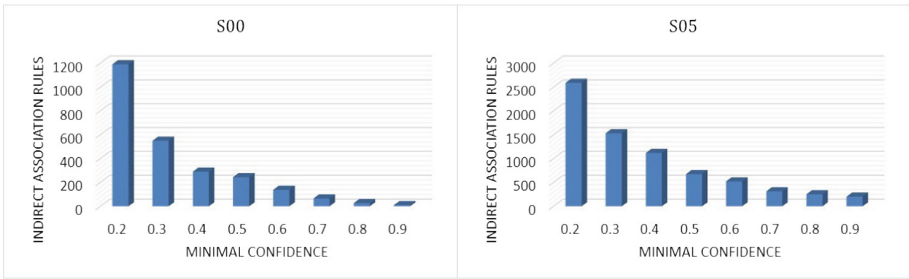
The experiments were run on 2 collections of ORs from clinical visits to different specialists in Endocrinology (S05), and General Practitioners (S00). Dataset S00 contain 791,420 sentences (transaction), and S05 - 157,448 sentences. Both datasets are sparse with huge number of items (vocabulary) (Table 1). Although the values of *minsup* are relatively small, there are only few frequent patterns due to the huge variety of values for each attribute. For most of experiments are used comparable values for minimal support (*minsup* = 0.015), mediator support (*ts* = 0.7) and minimal confidence (*conf* = 0.7), i.e. mediator dependency. For performance evaluation are used values for *minsup* in the range [0.015, 0.025] (Fig. 3), and for *conf* are used values in the range [0.2, 0.9] (Fig. 4).

Even small increase in the *minsup* value (Fig. 3) causes significant drop down of the total number of generated indirect association rules (IAR). Similarly the quantity of generated IAR is exponential decay when minimal confidence increases (Fig. 4).

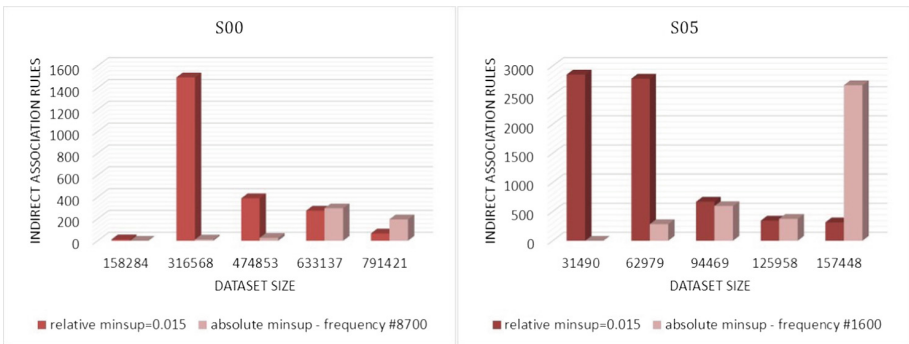
For performance evaluation the datasets S00 and S05 are fractioned on 5 equal subsets (Fig. 5). There are presented results for two experiments - with fixed relative value of *minsup*, i.e. as percentage of the dataset size.



**Fig. 3.** Minimal support vs Total number of indirect association rules for  $ts = 0.7$  and  $conf = 0.7$  for datasets S00 and S05



**Fig. 4.** Minimal confidence vs Total number of indirect association rules for  $ts = 0.7$  and  $minsup = 0.015$  for datasets S00 and S05



**Fig. 5.** S00 dataset size vs Total number of indirect association rules for  $ts = 0.7$ ,  $conf = 0.7$  for datasets S00 and S05 with different minsup values - absolute and relative

Increasing the dataset size affects the the absolute value of the threshold an it also increases. Thus the total number of IAR declines. For the second experient is used same threshold for generating IAR in all datasets. For smaller datasets the IAR have not significant support and the cumulative effect can be obtained for bigger datasets.

Some examples for extracted indirect pairs from S05 are presented below.

```
(X= 1 Y= 7 | mediator= 2 )
sup(X,mediator)= 100 sup(Y,mediator)= 68
conf(X,mediator)= 1.0 conf(Y,mediator)= 0.9685
```

Where the ID 1 = BMI (*Body Mass Index*) and 7 = сч - сърдечена честота (*heart frequency*).

The mediator set contains ID 2 = NUM - the symbol by which are replaced all numerical values in preprocessing subtask. In this example  $X$  and  $Y$  corresponds to different attributes and mediator set contains the type of their value.

The next example presents extracted indirect pair from S00, where  $X$  and  $Y$  corresponds to different possible values of the attribute presented in the mediator set.

```
(X= 7 Y= 60 | mediator= 9 )
sup(X,mediator)= 101 sup(Y,mediator)= 182
conf(X,mediator)= 0.9439 conf(Y,mediator)= 0.9479
```

The mediator set contains ID 9 = дишане (*breath*) and the ID 7 = отслабено (*weakened*) and 60 = удължено (*prolonged*).

For method accuracy evaluation are used standard metrics Precision Recall and F1-measure, where F1 measure is defined as:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Experiments are provided by non-exhaustive cross-validation (5 iterations on sets in ratio 4:1 training to test). For S00 are generated 1180 pairs of variable and mediator, and 274 pairs of indirect associated concepts. The size of the test set for S00 is 158,283 sentences. The precision is 0.85, recall is relatively low - 0.33 due to the small number of IAR and huge number of sentences in test set. This dataset is based on ORs written by General practitioners, thus the diversity of attribute-values pairs is higher. The overall evaluation for S00 is  $F1 = 0.49$ . For dataset S05 - 74 pairs of variable and mediator were generated, and 344 pairs of indirect associations. The size of the test set for S05 is 31,520 sentences. The precision is 0.69, recall is 0.5 - slightly higher than those for S00, but still low. The overall evaluation for S05 is  $F1 = 0.59$ . Better results for S05 are obtained because it contains ORs written by specialist in Endocrinology, thus the text is more consistent.

## 6 Conclusion and Further Work

This paper reports work in progress for patient status extraction from clinical text. The experimental result show that the proposed IARM based method can be successfully used for this task. All generated indirect association rules contain attribute-value pairs of anatomical organs/systems and their status.

Although relatively small number of generated IAR, and low recall, the method can be combined with direct association rules to improve results. IARM is data-driven and unsupervised method, i.e. when larger datasets are used for IAR generation the overall evaluation will be improved. The proposed method finds relations beyond simple terms only but also helps to identify attribute-value relations in patient status description.

For more detailed further analyses of the generated IAR can be used “human-in-the-loop” [15] approach. Patients phenotype will help to identify some specific status descriptions. This can help to improve the precision. In subclustering task also can be used “human-in-the-loop” approach to reduce the complexity and dimensionality of the search space. Some structured information concerning age, gender and demographic information of the patients can be used in the filtering process to determine different IAR depending on the patient phenotype. Another direction for further work is to investigate different measures for polarity between pairs of items.

In the terminology extraction tasks [14] there are used successfully many artificial intelligence (AI) approaches: linguistic, statistical, hybrid [19], neural networks, machine learning [7], etc. The main reason for choosing IARM method is that in healthcare data processing the most important characteristic of the used method is the result to be explainable, i.e. so called “Explainable AI” [10]. This will make the decision making process more transparent and will allow further generalization of the results.

**Acknowledgments.** This research is supported by the grant Specialized Data Mining Methods Based on Semantic Attributes (IZIDA), funded by the Bulgarian National Science Fund in 2017–2019. The team acknowledges the support of Medical University - Sofia, the Bulgarian Ministry of Health and the Bulgarian National Health Insurance Fund.

## References

1. Abdullah, Z., Herawan, T., Ahmad, N., Ghazali, R., Deris, M.M.: Mining indirect least association rule from students’ examination datasets. In: Murgante, B., et al. (eds.) ICCSA 2014. LNCS, vol. 8584, pp. 783–797. Springer, Cham (2014). [https://doi.org/10.1007/978-3-319-09153-2\\_58](https://doi.org/10.1007/978-3-319-09153-2_58)
2. Agrawal, R., Srikant, R., et al.: Fast algorithms for mining association rules. In: Proceedings of the 20th International Conference very large data bases, VLDB. vol. 1215, pp. 487–499 (1994)
3. Boytcheva, S., Angelova, G., Angelov, Z., Tcharaktchiev, D.: Integrating data analysis tools for better treatment of diabetic patients. CEUR Workshop Proc. **2022**, 229–236 (2017)
4. Boytcheva, S., Nikolova, I., Angelova, G.: Mining association rules from clinical narratives. In: Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017, pp. 130–138 (2017)
5. Boytcheva, S., Nikolova, I., Angelova, G., Angelov, Z.: Identification of risk factors in clinical texts through association rules. In: Proceedings of the Biomedical NLP Workshop associated with RANLP, pp. 64–72 (2017)

6. Chen, L., Bhowmick, S.S., Li, J.: Mining temporal indirect associations. In: Ng, W.-K., Kitsuregawa, M., Li, J., Chang, K. (eds.) PAKDD 2006. LNCS (LNAI), vol. 3918, pp. 425–434. Springer, Heidelberg (2006). [https://doi.org/10.1007/11731139\\_49](https://doi.org/10.1007/11731139_49)
7. Conrado, M., Pardo, T., Rezende, S.: A machine learning approach to automatic term extraction using a rich feature set. In: Proceedings of the 2013 NAACL HLT Student Research Workshop, pp. 16–23 (2013)
8. Fournier-Viger, P., Gomariz, A., Gueniche, T., Soltani, A., Wu, C.W., Tseng, V.S.: SPMF: a Java open-source pattern mining library. *J. Mach. Learn. Res.* **15**(1), 3389–3393 (2014)
9. Grahne, G., Zhu, J.: High performance mining of maximal frequent itemsets. In: 6th International Workshop on High Performance Data Mining. vol. 16, p. 34 (2003)
10. Gunning, D.: Explainable artificial intelligence (XAI). Defense Advanced Research Projects Agency (DARPA), nd Web (2017)
11. Ha, H.Y., Chen, S.C., Shyu, M.L.: Utilizing indirect associations in multimedia semantic retrieval. In: 2015 IEEE International Conference on Multimedia Big Data (BigMM), pp. 72–79. IEEE (2015)
12. Hamano, S., Sato, M.: Mining indirect association rules. In: Perner, P. (ed.) ICDM 2004. LNCS (LNAI), vol. 3275, pp. 106–116. Springer, Heidelberg (2004). [https://doi.org/10.1007/978-3-540-30185-1\\_12](https://doi.org/10.1007/978-3-540-30185-1_12)
13. Han, J., Pei, J., Yin, Y., Mao, R.: Mining frequent patterns without candidate generation: a frequent-pattern tree approach. *Data Min. Knowl. Disc.* **8**(1), 53–87 (2004)
14. Heylen, K., De Hertog, D.: Automatic term extraction. *Handb. Terminol.* **1**(01), 9–27 (2015)
15. Holzinger, A.: Interactive machine learning for health informatics: when do we need the human-in-the-loop? *Brain Inform.* **3**(2), 119–131 (2016)
16. Kang, S.M., Wagacha, P.W.: Extracting diagnosis patterns in electronic medical records using association rule mining. *Intern. J. Comput. Appl.* **108**(15), 19–26 (2014)
17. Kazienco, P.: Mining indirect association rules for web recommendation. *Intern. J. Appl. Math. Comput. Sci.* **19**(1), 165–186 (2009)
18. Manimaran, J., Velmurugan, T.: A survey of association rule mining in text applications. In: 2013 IEEE International Conference on Computational Intelligence and Computing Research (ICIC), pp. 1–5. IEEE (2013)
19. Nakagawa, H., Mori, T.: A simple but powerful automatic term extraction method. In: COLING-02 on COMPUTERM 2002: Second International Workshop on Computational Terminology-Volume 14, pp. 1–7. Association for Computational Linguistics (2002)
20. Nakov, P.: Bulstem: design and evaluation of inflectional stemmer for Bulgarian. In: Workshop on Balkan Language Resources and Tools (Balkan Conference in Informatics) (2003)
21. Névéol, A., Dalianis, H., Velupillai, S., Savova, G., Zweigenbaum, P.: Clinical natural language processing in languages other than english: opportunities and challenges. *J. Biomed. Semant.* **9**(1), 12 (2018)
22. Tan, P.N., Kumar, V.: Mining indirect associations in web data. In: International Workshop on Mining Web Log Data Across All Customers Touch Points, pp. 145–166. Springer (2001)
23. Tan, P.N., Kumar, V.: Discovery of indirect associations from web usage data. In: Zhong, N., Liu, J., Yao, Y. (eds.) Web Intelligence, pp. 128–152. Springer, Heidelberg (2003). [https://doi.org/10.1007/978-3-662-05320-1\\_7](https://doi.org/10.1007/978-3-662-05320-1_7)



24. Tan, P.-N., Kumar, V., Srivastava, J.: Indirect association: mining higher order dependencies in data. In: Zighed, D.A., Komorowski, J., Żytkow, J. (eds.) PKDD 2000. LNCS (LNAI), vol. 1910, pp. 632–637. Springer, Heidelberg (2000). [https://doi.org/10.1007/3-540-45372-5\\_77](https://doi.org/10.1007/3-540-45372-5_77)
25. Tsuruoka, Y., Miwa, M., Hamamoto, K., Tsujii, J., Ananiadou, S.: Discovering and visualizing indirect associations between biomedical concepts. *Bioinformatics* **27**(13), i111–i119 (2011)
26. Wan, Q., An, A.: An efficient approach to mining indirect associations. *J. Intell. Inf. Syst.* **27**(2), 135–158 (2006)
27. Wright, A., Chen, E.S., Maloney, F.L.: An automated technique for identifying associations between medications, laboratory results and problems. *J. Biomed. Inform.* **43**(6), 891–901 (2010). <https://doi.org/10.1016/j.jbi.2010.09.009>



# Towards Automated Customer Support

Momchil Hardalov<sup>1</sup>(✉), Ivan Koychev<sup>1</sup>, and Preslav Nakov<sup>2</sup>

<sup>1</sup> FMI, Sofia University “St. Kliment Ohridski”, Sofia, Bulgaria  
{hardalov,koychev}@fmi.uni-sofia.bg

<sup>2</sup> Qatar Computing Research Institute, HBKU, Doha, Qatar  
pnakov@qf.org.qa

**Abstract.** Recent years have seen growing interest in conversational agents, such as chatbots, which are a very good fit for automated customer support because the domain in which they need to operate is narrow. This interest was in part inspired by recent advances in neural machine translation, esp. the rise of sequence-to-sequence (seq2seq) and attention-based models such as the Transformer, which have been applied to various other tasks and have opened new research directions in question answering, chatbots, and conversational systems. Still, in many cases, it might be feasible and even preferable to use simple information retrieval techniques. Thus, here we compare three different models: (i) a retrieval model, (ii) a sequence-to-sequence model with attention, and (iii) Transformer. Our experiments with the Twitter Customer Support Dataset, which contains over two million posts from customer support services of twenty major brands, show that the seq2seq model outperforms the other two in terms of semantics and word overlap.

**Keywords:** Customer support · Conversational agents · Chatbots  
seq2seq · Transformer · IR

## 1 Introduction

The rapid proliferation of mobile and portable devices has enabled a number of new products and services. Yet, it has also laid stress on customer support as users now also expect  $24 \times 7$  availability of information about their orders, or answers to basic questions such as ‘Why is my Internet connection dead?’ and ‘What time is the next train from Sofia to Varna?’

Customer support has always been important to companies. Traditionally, it was offered primarily over the phone, but recently a number of alternative communication channels have emerged such as e-mail, social networks, forums/message boards, live chat, self-serve knowledge base, etc. As a result, it has become increasingly expensive for companies to maintain quality customer support services over a growing number of channels. First, they must find people that have both good language and communication skills. Second, each new employee must go through several training sessions before being able to operate

in the target channel, which is inefficient and time-consuming. And finally, it is difficult to have employees available for customer support  $24 \times 7$ .

Chatbots are especially fit for the task as they are automatic: fully or partially. Moreover, from a technological viewpoint, they are feasible as the domain they need to operate in is narrow. As a result, chit-chat is reduced to a minimum, and chatbots serve primarily as question-answering devices. Moreover, it is possible to train them on real-world chat logs. Here, we experiment with such logs from customer support on Twitter, and we compare two types of chatbots: (i) based on information retrieval (IR), and (ii) on neural question answering. We further explore semantic similarity measures since generic ones such as ROUGE [8], BLEU [16] and METEOR [2], which come from machine translation or text summarization, are not well suited for chatbots.

The remainder of this paper is organized as follows: Sect. 2 presents related work. Section 3 describes the dataset and the preprocessing, and offers insights about the dialogs. Section 4 focuses on the models. Section 5 describes the experiments, the results, and the evaluation measures. Section 6 discusses the results. Finally, Sect. 7 concludes, and suggests directions for future work.

## 2 Related Word

Sequence-to-sequence (seq2seq) models were first introduced in 2014 in the context of machine translation [27]. Since then, they have been successfully applied in other domains such as text summarization, question-answering, conversational agents, etc. One of the first examples of a basic seq2seq model for chatbots was proposed in 2015 by Vinyals et al. [29], who experimented with two datasets: IT helpdesk tickets and Open Subtitles. They further pointed out to the following issues: lack of context modeling for multi-turn dialogs, lack of “personality” for models trained on different sources, and need for human evaluation of the generated responses.

Another source of training data have been community Question Answering forums. In 2015, Lowe et al. [12] introduced the Ubuntu Dialog Corpus, and experimented with plain RNN vs. LSTM-based neural models, in addition to retrieval-based approaches. An extension of this study, including several new encoder-decoder architectures, was published recently [13]. In another related work, Boyanov et al. [3] explored the utility of neural models on data from SemEval-2016 task 3 on Community Question Answering [15]. They compared seq2seq models with retrieval-based ones, performing model selection using question answering measures, and studied the ability of the chatbot to answer free-form questions.

Twitter data is particularly suitable for fitting neural conversational models because of the length restriction, which encourages people to write short, more precise tweets. Thus, it was used in a number of studies. Serban et al. [23] improved seq2seq models using a hierarchical structure. Sordoni et al. [25] worked on modeling the context. Shang et al. [24] proposed a neural network response generator for short-text conversation, which was trained with a large number

of one-round conversations from a micro-blogging service, and could generate grammatically correct and content-wise appropriate responses.

Some interesting approaches for building customer support chatbots were shown in [4, 18], as a combination of retrieval and neural models. Cui et al. [4] used information from in-page product descriptions, as well as user-generated content from e-commerce web sites to improve online shopping experience. Their approach incorporated four different components (fact database, FAQs, opinion-oriented answers, and a neural-based chat-chat generator) into a meta-engine that makes a choice between them. Qiu et al. [18] proposed an open-domain chatbot engine that integrates results from IR and seq2seq models, and uses an attentive seq2seq reranker to choose dynamically between their outputs.

### 3 Dataset

Overall, data and resources that could be used to train a customer support chatbot are very scarce, as companies keep conversations locked on their own closet, proprietary support systems. This is due to customer privacy concerns and to companies not wanting to make public their know-how and the common issues about their products and services. An extensive 2015 survey on available dialog corpora by Serban et al. [22] found no good publicly available dataset for real-world customer support.

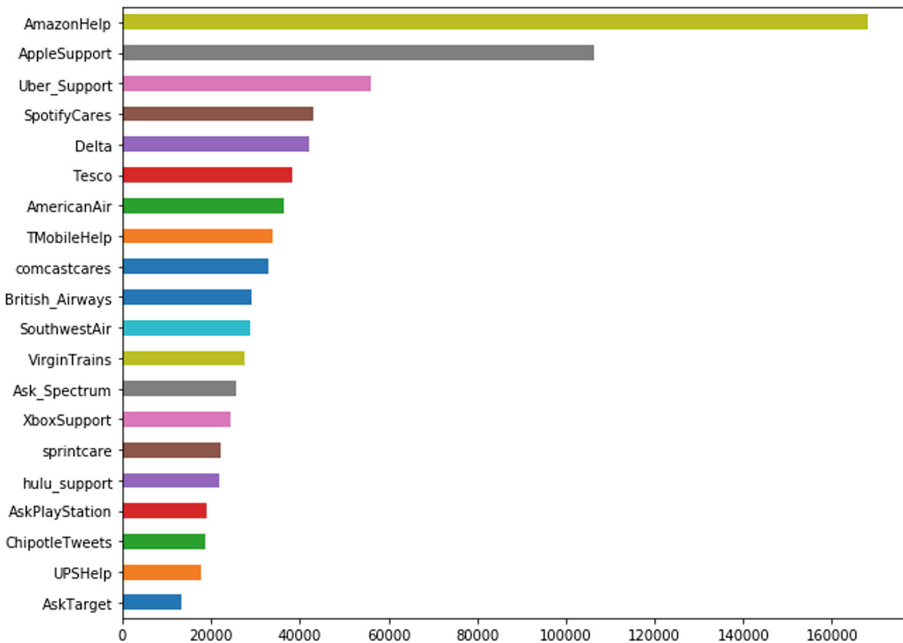


Fig. 1. Number of user tweets with replies from customer support per company.

Earlier this year, this situation has changed as a new open dataset, named *Customer Support on Twitter*, was made available on Kaggle.<sup>1</sup> It is a large corpus of recent tweets and replies, which is designed to support innovation in natural language understanding and conversational models, and to help study modern customer support practices and impact. The dataset contains 3M tweets from 20 big companies such as Amazon, Apple, Uber, Delta, and Spotify, among others. See Fig. 1 for detail.

As customer support topics from different organizations are generally unrelated to each other, we focus only on tweets related to Apple support, which represents the largest number of tweets in the corpus. We filtered all utterances that redirect the user to another communication channel, e.g., direct messages, which are not informative for the model and only bring noise. Moreover, since answers evolve over time, we divided our dataset into a training and a testing part, keeping earlier posts for training and the latest ones for testing. We further excluded from the training set all conversations that are older than sixty days. For evaluation, we used dialogs from the last five days in the dataset, to simulate a real-world scenario for customer support. We ended up with a dataset of 49,626 dialog tuples divided in 45,582 for training and 4,044 for testing.

Table 1 shows some statistics about our dataset. In the top of the table, we can see that the average number of turns per dialog is under three, which means that most of the dialogs finish after one answer from customer support. The bottom of the table shows the distribution of words in the user questions vs. the customer support answers. We can see that answers tend to be slightly longer, which is natural as replies by customer support must be extensive and helpful.

**Table 1.** Statistics about our dataset.

Overall		
# dialogs tuples		49,626
# words (in total)		26,140
Min # turns per dialog		2
Max # turns per dialog		106
Avg. # turns per dialog		2.6
Training set: # of dialogs		45,582
Testing set: # of dialogs		4,044
	Questions	Answers
Avg. # words	21.31	25.88
Min # words	1.00	3.00
1st quantile	13.00	20.00
Mode	20.00	23.00
3rd quantile	27.00	29.00
Max # words	136.00	70.00

<sup>1</sup> <https://www.kaggle.com/thoughtvector/customer-support-on-twitter>.

## 4 Method

### 4.1 Preprocessing

Since Twitter has its own specifics of writing in terms of both length<sup>2</sup> and style, standard text tokenization is generally not suitable for tweets. Therefore, we used a specialized Twitter tokenizer [14] to preprocess the data. Then, we further cleaned the data by replacing the shorthand entries, e.g., *'ll*, *'d*, *'re*, *'ve*, with the most likely literary form, e.g., *will*, *would*, *are*, *have*. We also replaced slang words, e.g., *'bout* and *'til*, with the standard words, e.g., *about* and *until*. Similarly, we replaced URLs with the special word  $\langle url \rangle$ , all user mentions with  $\langle user \rangle$ , and all hashtags with  $\langle hashtag \rangle$ .

Moreover, we tried to mitigate the effect of missing context in long conversations by concatenating all previous turns to the current question. Finally, since seq2seq models cannot be trained efficiently with a large vocabulary, we chose the top  $N$  words when building the model (see Sect. 5 for detail), and we replaced the instances of the remaining words with a special symbol  $\langle unk \rangle$ .<sup>3</sup>

### 4.2 Information Retrieval

The Information Retrieval (IR) approach can be defined as follows: given a user question  $q'$  and a list of pairs of previously asked questions and their answers  $(Q, A) = \{(q_j, a_j) | j = 1, \dots, n\}$ , find the most similar question  $q_i$  in the training dataset that a user has previously asked and return the answer  $a_i$  that customer support has given to  $q_i$ . The similarity between  $q'$  and  $q_i$  can be calculated in various ways, but most commonly this is done using the cosine between the corresponding TF.IDF-weighted vectors.

$$a' = \arg \max_{(q_j, a_j)} \text{sim}(q', q_j) \quad (1)$$

### 4.3 Sequence-to-Sequence

Our encoder uses a bidirectional recurrent neural network (RNN) based on long short-term memory (LSTM) [6]. It encodes the input sequence  $x = (x_1, \dots, x_n)$  and calculates a forward sequence of hidden states  $(\vec{h}_1, \dots, \vec{h}_m)$  and also a backward sequence  $(\overleftarrow{h}_1, \dots, \overleftarrow{h}_m)$ . The decoder is a unidirectional LSTM-based RNN, and it predicts the output sequence  $y = (y_1, \dots, y_n)$ . Each  $y_i$  is predicted using the recurrent state  $s_i$ , the previous predicted word  $y_{i-1}$ , and a context vector  $c_i$ . The latter is computed using an attention mechanism as a weighted sum over the encoder's output  $(\vec{h}_j, \overleftarrow{h}_j)$ , as proposed by Bahdanau et. al [1].

<sup>2</sup> By design, tweets have been strictly limited to 140 characters; this constrain has been relaxed to 280 characters in 2017.

<sup>3</sup> In future work, we plan to try byte-pair encoding instead [21].

## 4.4 Transformer

The Transformer model was proposed in 2017 by Vaswani et al. [28], and it has shown very strong performance for machine translation, e.g., it achieved state-of-the-art results on WMT2014 data for English-German and English-French translation. Similarly to the seqseq model, the Transformer has an encoder and a decoder. The encoder is a stack of identical layers, based on multi-head self-attention and a simple position-wise fully connected network. The decoder is similar, but in addition to the two sub-layers in the encoder, it introduces a third sub-layer, which performs multi-head attention over the encoders' stack outputs. The main advantage of the Transformer model is that it can be trained significantly faster, as compared to recurrent or convolutional networks.

## 5 Experiments

We performed three experiments using the models described in Sect. 4. Below, each model is abbreviated by its architecture name from Sect. 5.2.

*IR* is based on ElasticSearch<sup>4</sup> (ES), as it provides out-of-the-box implementation of all the components we need. We fed the pre-processed training data into an index with English analyzer enabled, whitespace- and punctuation-based tokenization, and word 3-grams. For retrieval, we used the default BM25 algorithm [19], which is an improved version of TF.IDF. For all training questions and for all testing queries, we appended the previous turns in the dialog as context. Given a user question from the testing set, we returned the customer support answer for the top-ranked result from ES.

*Seq2Seq* contains one bi-directional LSTM layer with 512 hidden units per direction (a total of 1,024). The decoder has two unidirectional layers connected directly to the bidirectional one in the encoder. The network takes as input words encoded as 200-dimensional embeddings. It is a combination of pre-trained GloVe [17] vectors learned from 27B Twitter posts<sup>5</sup> for the known words, and a positional embedding layer, learned as model parameters, for the unknown words. The embedding layers for the encoder and for the decoder are not shared, and are learned separately. This separation is due to the fact that the words used in utterances by customers are very different compared to posts by the support. In our experiments, we used the top 8,192 words sorted by frequency for both the embedding and the output. Based on the statistics presented in Sect. 3, we chose to use 60 words (time-steps) for both the encoder and the decoder. We avoid overfitting by applying dropout [26] with keep probability of 0.8 after each recurrent layer. For the optimizer, we used Adam [7] with a base value of  $1 \times 10^{-3}$  and an exponential decay of 0.99 per epoch.

*Transformer* is based on two identical layers for the encoder and for the decoder, with four heads for the self-attention. The dimensionality of the input and of the output is  $d_{model} = 256$ , and the inner dimensionality is  $d_{inner} = 512$ .

<sup>4</sup> <https://www.elastic.co/products/elasticsearch>.

<sup>5</sup> <https://nlp.stanford.edu/projects/glove/>.

The input consists of queries with keys of dimension  $d_k = 64$  and values of dimension  $d_v = 64$ . The input and the output embedding are learned separately with sinusoidal positional encoding. The dropout is set to 0.9 keep probability. For the optimization, we use Adam with varying learning rate based on Eq. (2). The hyper-parameter choice was guided by the experiments described by the authors in the original Transformer paper [28].

$$lrate = d_{model}^{-0.5} \cdot \min(step\_num^{-0.5}, step\_num \cdot warmup\_steps^{-1.5}) \quad (2)$$

## 5.1 Evaluation Measures

How to evaluate a chatbot is an open research question. As the problem is related to machine translation (MT) and text summarization (TS), which are nowadays also addressed using seq2seq models, researchers have been using MT and TS evaluation measures such as BLEU [16], ROUGE [8], and METEOR [2], which focus primarily on word overlap and measure the similarity between the chatbot’s response and the gold customer support answer to the user question. However, it has been argued [10, 11] that such word-overlapping measures are not very suitable for evaluating chatbots. Thus, we adopt three additional measures, which are more semantic in nature.<sup>6</sup>

The *embedding average* constructs a vector for a piece of text by taking the average of the word embeddings of its constituent words. Then, the vectors for the chatbot response and for the gold human one are compared using cosine similarity.

The *greedy matching* was introduced in the context of intelligent tutoring systems [20]. It matches each word in the chatbot output to the most similar word in the gold human response, where the similarity is measured as the cosine between the corresponding word embeddings, multiplied by a weighting term, which we set to 1, as shown in Eq. (3). Since this measure is asymmetric, we calculate it a second time, with arguments swapped, and then we take the average as shown in Eq. 4.

$$greedy(u_1, u_2) = \frac{\sum_{v \in u_1} weight(v) * \max_{w \in u_2} cos(v, w)}{\sum_{v \in u_1} weight(v)} \quad (3)$$

$$simGreedy(u_1, u_2) = \frac{greedy(u_1, u_2) + greedy(u_2, u_1)}{2} \quad (4)$$

The *vector extrema* was proposed by Forgues et al. [5] for dialogue systems. Instead of averaging the word embeddings of the words in a piece of text, it takes the coordinate-wise maximum (or minimum), as shown in Eq. (5). Finally, the resulting vectors for the chatbot output and for the gold human one are compared using cosine.

$$extrema(u_i) = \begin{cases} \max u_i, & \text{if } \max u_i \geq |\min u_i| \\ \min u_i, & \text{otherwise} \end{cases} \quad (5)$$

<sup>6</sup> Note that we do not use measures trained on the same data as advised by [10].



## 5.2 Results

Table 2 shows the results for the three models we compare (IR, seq2seq, and Transformer) when using word overlap measures such as BLEU@2, which uses unigrams and bigrams only, and ROUGE-L [9], which uses Longest Common Subsequence (LCS).

**Table 2.** Results based on word-overlap measures.

	Word overlap measures	
	BLEU@2	ROUGE-L
IR - BM25	13.73	22.35
Seq2Seq	<b>15.10</b>	<b>26.60</b>
Transformer	12.43	25.33

Table 3 shows the results for the same three systems, but using the above-described semantic evaluation measures, namely Embedding Average (with cosine similarity), Greedy Matching, and Vector Extrema (with cosine similarity). For all three measures, we used Google’s pre-trained word2vec embeddings because they are not learned during training, which helps avoid bias, as has been suggested in [10, 11].

**Table 3.** Results based on semantic measures.

	Semantic evaluation measures		
	Embedding Average	Greedy Matching	Vector Extrema
IR - BM25	76.53	29.72	37.99
Seq2Seq	<b>77.11</b>	<b>30.81</b>	<b>40.23</b>
Transformer	75.35	30.08	39.40

## 6 Discussion

The evaluation results show that *Seq2Seq* performed best with respect to all five evaluation measures. For the group of semantic measures, it outperformed the other systems in terms of Embedding Average by +0.58, in terms of Greedy Matching by +0.73, and in terms of Vector Extrema by +0.83 (points absolute). Moreover, SeqSeq was also clearly the best model in terms of word-overlap evaluation measures, scoring 15.10 on BLEU@2 (+1.37 ahead of the second), and 26.60 on ROUGE-L (+1.27 compared to the second best system).

The *Transformer* model was ranked second by three of the evaluation measures: Greedy Matching, Vector Extrema, and ROUGE-L. This was unexpected given the state-of-the-art results it achieved for neural machine translation.

**Table 4.** Chatbot responses. The first column shows the original question and the gold customer support answer, while the second column shows responses by our models.

Question & Answer	Response
<p>(1) my phones speaker just broke down on me thanks &lt;user&gt; do better</p> <p>&lt;user&gt; we are here to help. which iphone and ios version are you using? are you having a sound quality issue or is there no sound at all?</p>	<p><b>IR:</b> &lt;user&gt; we can help with your device . do you notice the slowness when performing a particular activity ?</p> <p><b>Seq2Seq:</b> &lt;user&gt; that is not the experience we want you to have . which iphone model do you own ?</p> <p><b>Transformer:</b> &lt;user&gt; thanks for reaching out to us . we are always happy to help . can you tell us which version of ios 11 you are using ?</p>
<p>(2) i am trying to type it not fuckin i.t, your product is absolute garbage. figure it out. &lt;user&gt; &lt;user&gt;</p> <p>&lt;user&gt; lets take a closer look at this to get your device working right. which model do you have and that is the exact os version number installed?</p> <p>have you tried any steps so far and when did this start happening?</p>	<p><b>IR:</b> &lt;user&gt; thank you for reaching out . we want to help . to start , lets go to settings &gt; general &gt; about and see which version of ios your iphone is on please .</p> <p><b>Seq2Seq:</b> &lt;user&gt; we would be happy to look into that is going on . can you tell us more?</p> <p><b>Transformer:</b> &lt;user&gt; heres what you can do to work around the issue until it is fixed in a future software update : &lt;url&gt;</p>
<p>(3) capaz que si actualizo ios me teléfono se convierta en ladrillo :(....ya no confio en sus productos &lt;user&gt;</p> <p>&lt;user&gt; we offer support via twitter in english. get help in spanish here: &lt;url&gt; or join &lt;url&gt;</p>	<p><b>All models:</b> &lt;user&gt; we offer support via twitter in english get help in spanish here &lt;url&gt; or join &lt;url&gt;</p>
<p>(4) &lt;user&gt; excuse me , i am got a refund confirmation email but nothing happened . do i have to do anythings ? → &lt;user&gt; thanks for reaching out to us about your refund ! to clarify , was this a refund for a purchase in the itunes store ? or was this a product purchase ? &lt;user&gt; thanks for reply . i am purchased music in itunes store .</p> <p>&lt;user&gt; thanks. our itunes store support team would have the best information for you on that. you can contact them here: &lt;url&gt;</p>	<p><b>IR:</b> &lt;user&gt; alright . our itunes support team will be able to look at this with you . please reach out to them here : &lt;url&gt;</p> <p><b>Seq2Seq:</b> &lt;user&gt; we are investigating and will update our system status page as more info becomes available &lt;url&gt;</p> <p><b>Transformer:</b> &lt;user&gt; thanks for reaching out . we would recommend leaving that request on our feedback page : &lt;url&gt;</p>

Higher Greedy Matching and Vector Extrema scores show that the Transformer was able to capture the semantics of the gold answer. Moreover, lower Embedding Average and BLEU@2 scores suggest that it chose different vocabulary or used different word order. This is confirmed by lower ROUGE-L, which is based on longest common subsequence.

Finally, the retrieval (*IR*) model achieved the second-best results in terms of BLEU@2 and Embedding Average, but it was the worst according to the other three evaluation measures. This shows the superiority of the generative neural models over simple retrieval.

Table 4 shows some example responses generated by the three models. In the first example (1), the IR model is off and retrieves an answer that addresses a different customer problem. The Seq2Seq model is on the right track, because it asks the user about his device. The Transformer suggests a similar utterance, but it makes an assumption about the phone’s operating system, which was not stated in the user’s question. In the second example (2), all models propose very different ways of action to the user, compared to the original answer, and they all seem plausible in this context; yet, the Transformer is a bit off. The next example (3) illustrates the ability of the three models to distinguish between different languages, and point the user in the right direction. The last example (4) is a typical example when neural models fail. The particular question–answer tuple is hard to answer as there are very few similar examples in the training data. Thus, what the neural models generate ends up being off-topic. In contrast, the retrieval approach was able to overcome this and to propose a very good answer.

## 7 Conclusion and Future Work

We have presented a study on automating customer support on Twitter using two types of models: (*i*) retrieval-based (IR with BM25), and (*ii*) based on generative neural networks (seq2seq with attention and Transformer). We evaluated these models without the need of human judgments, using evaluation measures based on (*i*) word-overlap (BLEU@2 and ROUGE-L), and (*ii*) semantics (Embedding Average, Greedy Matching, and Vector Extrema). Our experiments have shown that generative neural models outperform retrieval-based ones, but they struggle when very few examples for a particular topic are present in the training data.

In future work, we plan to combine the three approaches into an ensemble. Another interesting direction that we would like to explore is handling questions whose correct answers evolve over time, e.g., due to service updates or to new products being released.

**Acknowledgments.** This work was supported by the EC under grant no. 763566 and by the Bulgarian National Scientific Fund as project no. DN 12/9,

## References

1. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. arXiv preprint [arXiv:1409.0473](https://arxiv.org/abs/1409.0473) (2014)
2. Banerjee, S., Lavie, A.: METEOR: an automatic metric for MT evaluation with improved correlation with human judgments. In: Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization, Ann Arbor, Michigan, pp. 65–72 (2005)
3. Boyanov, M., Nakov, P., Moschitti, A., Da San Martino, G., Koychev, I.: Building chatbots from forum data: model selection using question answering metrics. In: Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017, Varna, Bulgaria, pp. 121–129 (2017)
4. Cui, L., Huang, S., Wei, F., Tan, C., Duan, C., Zhou, M.: SuperAgent: a customer service chatbot for e-commerce websites. In: Proceedings of the Association for Computational Linguistics 2017, System Demonstrations, ACL 2017, Vancouver, Canada, pp. 97–102 (2017)
5. Forgues, G., Pineau, J., Larchevêque, J.M., Tremblay, R.: Bootstrapping dialog systems with word embeddings. In: Proceedings of the NIPS Workshop on Modern Machine Learning and Natural Language Processing, Montreal, Canada (2014)
6. Hochreiter, S., Schmidhuber, J.: Long short-term memory. *Neural Comput.* **9**(8), 1735–1780 (1997)
7. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. In: Proceedings of the 2015 International Conference on Learning Representations, ICLR 2015, San Diego, California (2015)
8. Lin, C.Y.: ROUGE: a package for automatic evaluation of summaries. In: Proceedings of the ACL Workshop on Text Summarization Branches Out, Barcelona, Spain, pp. 74–81 (2004)
9. Lin, C.Y., Och, F.J.: Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In: Proceedings of the 42nd Annual Conference of the Association for Computational Linguistics, ACL 2004, Barcelona, Spain, pp. 605–612 (2004)
10. Liu, C.W., Lowe, R., Serban, I., Noseworthy, M., Charlin, L., Pineau, J.: How NOT to evaluate your dialogue system: an empirical study of unsupervised evaluation metrics for dialogue response generation. In: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, pp. 2122–2132 (2016)
11. Lowe, R., Noseworthy, M., Serban, I.V., Angelard-Gontier, N., Bengio, Y., Pineau, J.: Towards an automatic Turing test: learning to evaluate dialogue responses. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, pp. 1116–1126 (2017)
12. Lowe, R., Pow, N., Serban, I., Pineau, J.: The Ubuntu dialogue corpus: a large dataset for research in unstructured multi-turn dialogue systems. In: Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue, SIGDIAL 2015, Prague, Czech Republic, pp. 285–294 (2015)
13. Lowe, R.T., Pow, N., Serban, I.V., Charlin, L., Liu, C.W., Pineau, J.: Training end-to-end dialogue systems with the Ubuntu dialogue corpus. *Dialogue Discourse* **8**(1), 31–65 (2017)
14. Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S., McClosky, D.: The Stanford CoreNLP natural language processing toolkit. In: Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, ACL 2014, Baltimore, Maryland, pp. 55–60 (2014)

15. Nakov, P., et al.: SemEval-2016 task 3: community question answering. In: Proceedings of the 10th International Workshop on Semantic Evaluation, SemEval 2016, San Diego, California, pp. 525–545 (2016)
16. Papineni, K., Roukos, S., Ward, T., Zhu, W.J.: BLEU: a method for automatic evaluation of machine translation. In: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, ACL 2002, Philadelphia, Pennsylvania, pp. 311–318 (2002)
17. Pennington, J., Socher, R., Manning, C.: GloVe: global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, Doha, Qatar, pp. 1532–1543 (2014)
18. Qiu, M., et al.: AliMe Chat: a sequence to sequence and rerank based chatbot engine. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, pp. 498–503 (2017)
19. Robertson, S., Zaragoza, H.: The probabilistic relevance framework: BM25 and beyond. *Found. Trends Inf. Retr.* **3**(4), 333–389 (2009)
20. Rus, V., Lintean, M.: A comparison of greedy and optimal assessment of natural language student input using word-to-word similarity metrics. In: Proceedings of the Seventh Workshop on Building Educational Applications Using NLP, Montreal, Canada, pp. 157–162 (2012)
21. Sennrich, R., Haddow, B., Birch, A.: Neural machine translation of rare words with subword units. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, pp. 1715–1725 (2016)
22. Serban, I.V., Lowe, R., Henderson, P., Charlin, L., Pineau, J.: A survey of available corpora for building data-driven dialogue systems: the journal version. *Dialogue Discourse* **9**(1), 1–49 (2018)
23. Serban, I.V., Sordoni, A., Bengio, Y., Courville, A.C., Pineau, J.: Hierarchical neural network generative models for movie dialogues. *CoRR*, abs/1507.04808 (2015)
24. Shang, L., Lu, Z., Li, H.: Neural responding machine for short-text conversation. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, ACL-IJCNLP 2015, Beijing, China, pp. 1577–1586 (2015)
25. Sordoni, A., et al.: A neural network approach to context-sensitive generation of conversational responses. In: Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2015, Denver, Colorado, pp. 196–205 (2015)
26. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **15**(1), 1929–1958 (2014)
27. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: Proceedings of the 27th Annual Conference on Neural Information Processing Systems, NIPS 2014, Montreal, Canada, pp. 3104–3112 (2014)
28. Vaswani, A., et al.: Attention is all you need. In: Proceedings of the 30th Annual Conference on Neural Information Processing Systems, NIPS 2017, Long Beach, California, pp. 5998–6008 (2017)
29. Vinyals, O., Le, Q.V.: A neural conversational model. *CoRR* abs/1506.05869 (2015)



# Evaluation of Automatic Tag Sense Disambiguation Using the MIRFLICKR Image Collection

Olga Kanishcheva<sup>1</sup>, Ivelina Nikolova<sup>2(✉)</sup>, and Galia Angelova<sup>2</sup>

<sup>1</sup> National Technical University “Kharkiv Polytechnic Institute”,  
2 Kyrpychova Street, 61002 Kharkiv, Ukraine  
kanichshevaolga@gmail.com

<sup>2</sup> Institute of Information and Communication Technologies,  
Bulgarian Academy of Sciences,  
25A Acad. G. Bonchev Street, 1113 Sofia, Bulgaria  
{iva,galia}@lml.bas.bg

**Abstract.** Automatic identification of intended tag meanings is a challenge in large image collections where human authors assign tags inspired by emotional or professional motivations. Algorithms for automatic tag disambiguation need “golden” collections of manually created tags to establish baselines for accuracy assessment. Here we show how to use the MIRFLICKR-25000 collection to evaluate the performance of our algorithm for tag sense disambiguation which identifies meanings of image tags based on WordNet or Wikipedia. We present three different types of observations on the disambiguated tags: (i) accuracy evaluation, (ii) evaluation of the semantic similarity of the individual tags with the image category and (iii) the semantic similarity of an image tagset to the image category, using different word embedding models for the latter two. We show how word embeddings create a specific baseline so the results can be compared. The accuracy we achieve is 78.6%.

**Keywords:** Image tagging · Tag sense disambiguation · Semantic relatedness WordNet · MIRFLICKR-25000 · Word embeddings

## 1 Introduction

Nowadays word sense disambiguation (WSD) remains an issue in natural language processing and computational linguistics. Usually automatic WSD of some word is performed in texts where the context supports the automatic identification of the intended meaning. Tag sense disambiguation (TSD) is a similar task, especially for images and video annotation, but it deals with isolated keyword metadata assigned to visual or multimedia objects. Systems that annotate images automatically need to be aware of multiple tag meanings in order to make a decision which keyword to assign; this is important because awareness about the correct tag senses facilitates the automatic tag translation to other languages, description aggregation etc. In the opposite direction, given images already annotated by auto-tagging or humans, one can think about automatic recognition of intended senses when the tags have multiple meanings.

Then the surrounding context, i.e. the remaining tags, are the only explicit semantic hints that may help to resolve the ambiguity. However, the large number of possible context words might also reduce the precision of sense identification, both in terms of computational effort and outcome quality.

Here the aim is to assess a tag disambiguation approach proposed in [1] using a publicly available collection with significant size – MIRFLICKR-25000<sup>1</sup> (previously it has been evaluated only using relatively small image sets collected by our team). MIRFLICKR-25000 consists of 25,000 Creative Common images downloaded from the social photography site Flickr with complete manual annotations [2]. Our TSD algorithm makes use of WordNet (WN) [3] and employs the synset information for polysemous tags for all possible senses. In this way the algorithm we proposed works on an extended tagset including the original image tags coming from MIRFLIKR as well as the synonyms from all WordNet senses. Due to lack of golden standards for TSD reference we show how word embeddings can be used to establish a specific baseline for evaluation of TSD results.

The paper is organized as follows: Sect. 2 considers related work and results suggesting certain baseline (regardless the lack of direct comparison using the same datasets). In Sect. 3 we briefly consider the MIRFLICKR-25000 collection and describe how we build our image tagsets. Section 4 presents the experimental results and their evaluation. Finally, Sect. 5 presents the conclusion.

## 2 Related Work

Integrating vision and language is a principal goal of AI since many years but there is still no considerable progress in this area of research. Ambitious projects approach language and vision from images to videos. The linguistic resources and image datasets are crucial for advancing the field. In this context, the TSD task is somewhat specific and few researchers studied it in the last decade.

The survey [4] considers datasets which enable training/testing and set benchmarks for performance evaluation. Available datasets are grouped in three categories: (i) captioned images with one or multiple tags per image; (ii) video datasets aligned with descriptions and (iii) n-gram language resources paired with scene-level understanding of an image. These datasets were used in automatic tasks such as image and video captioning, summarization, visual question answering and conversational robots grounded in the visual world. The conclusion of [4] is that “it is unclear how different methods generalize beyond the datasets they are evaluated on, and what data may be useful for moving the field beyond a single task, towards solving larger AI problems”.

A model using a dictionary and text contexts of web images to disambiguate image senses is presented in [5, 6]. Latent Dirichlet Allocation (LDA) discovers a latent sense space and makes the model robust despite the very limited dictionary definitions. The definition text is used to learn a distribution over the empirical text topics that best represents the sense. As a final step, a discriminative classifier is trained on the

---

<sup>1</sup> <http://press.liacs.nl/mirflickr/>.

re-ranked mixed-sense images that can predict the correct sense for novel images. Experiments included retrieval of the ground truth sense and classification of unseen images; evaluation was performed on over 10,000 images consisting of search results for five polysemous words. In retrieval, the dictionary model improved compared to the baseline search engine precision by re-ranking the images according to sense probability. In classification, the method outperformed a baseline algorithm that attempts to refine the search by generating sense-specific search terms from WordNet entries. The results averaged over the categories; for 300 training images the accuracy exceeds 77%.

A TSD method working in a social tagging environment is presented in [7]. The authors exploit the collective intelligence of Web 2.0 in defining the Neighbor tags by using the tag co-occurrences. They showed that TSD can be applied to the vocabulary of social tags, thereby clarifying the tag vocabulary through Wikipedia. The precision is about 80% but the evaluation settings and the test dataset remain unclear.

Sense-tagged keywords-based annotations are built by combining WordNet (a priori knowledge) and visual knowledge in [8]. On the first stage, a graph-based technique assigns a bag of keywords to a query image. Then, a WSD algorithm named Structural Semantic Interconnections is adopted to TSD to determine the sense of each keyword. The authors used the LabelMe corpus for their experiments. Terms that are not found in WordNet 3.0 were removed from the annotations. In the final dataset, some 3,118 classes (32%) of all annotations in LabelMe were aligned to WordNet synsets, by identifying words including in one or several WordNet synsets. Among these annotations, 1,735 are polysemous entries in WordNet i.e. 55% of all classes. No real assessment of accuracy is presented; experimental results are illustrated on one tag (“mouse”).

The authors of [9] proposed to make feature selection based on the Shapley Value (SV) – a coalitional game theory related metrics, measuring the importance of a component within a coalition. By including in the feature set only the words with the highest Shapley Value, they obtained good quality and performance improvements. The exponential complexity of the exact SV computation is discussed in [9]. The effectiveness of this method and sampling results are illustrated by using both a synthetic language corpus and a real linguistic corpus. The evaluation is presented as a synthesis of the performance comparison in two settings: with feature selection based on the Shapley Value of the words (top 100 Shapley values) and on the word frequency (top 100 frequency words). For the first case  $F1 = 92\%$  is reported, for the second –  $F1 = 75\%$ .

A somewhat similar work deals with image captions which in general have length of one or two sentences [10]. In contrary to tags that are assigned as isolated keywords, the captions might provide useful contexts for the WSD task. The authors show how visual features can improve the accuracy of unsupervised WSD when the textual context is very short. Their previous work in unsupervised text-only disambiguation is extended with methods that integrate text and images. The corpus is constructed by using Amazon Mechanical Turk to capture sense tagged images gathered from ImageNet. Using an Yarowsky-inspired algorithm, they showed that gains can be made over text-only disambiguation, as well as multimodal approaches such as Latent Dirichlet Allocation. The assessment is presented for 20 ambiguous tags only. The average accuracy is calculated



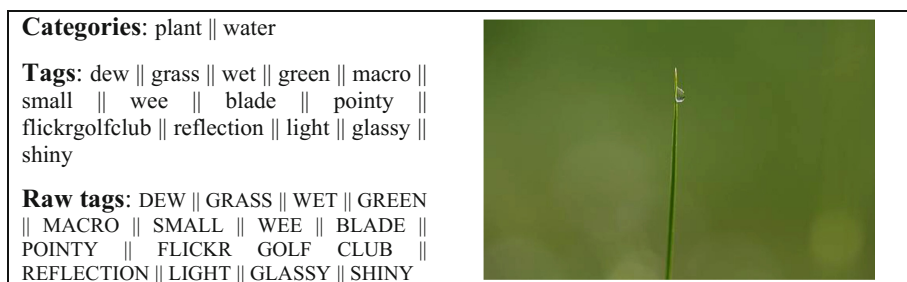
across all five sets of data. WSD based only on the text features had accuracy 76% while WSD based on combined text and visual features achieved 78% accuracy.

Research in TSD is done with different datasets by using different techniques. Due to this variety we cannot make comparisons and set a baseline. The success rates reported here seem to be around 80% for somewhat limited amount of tags. One of the major open problems is that we do not know how to integrate the knowledge about image content and the senses of the tags to be selected, especially when the keywords are not included in lexical resources such as WordNet.

New NLP trends, using deep learning, might be promising in TSD as well because they aim at a large-scale discovery of relations [11] or objects meant by a word [12].

### 3 Source Tag Set and Construction of Extensions

The collection MIRFLICKR-25000 consists of 25,000 images downloaded from Flickr.com through its public API with manual annotations mostly in English and software for similarity search and classification. There are 1,386 tags which occur in at least 20 images; average number of tags per image is 8.94. Tags are given as: (i) raw form submitted by the users and (ii) a processed form where the raw tags have been cleaned by Flickr [2]. “Cleaning” includes removal of capitalization, spaces and various special characters. In addition to the raw tags, each image is assigned categories (one or more). Figure 1 shows an image in the categories *plant* and *water* as well as its raw tags and their cleaned form (called *tags*).



**Fig. 1.** Annotation of Image-8 in MIRFLICKR-25000

Table 1 shows all general categories and subcategories in the collection. We process the ones in bold: *Animals*, *Food*, *Plant life*, *People*, *Sunset*. An image can be assigned several (sub-)categories, e.g. *Image-8* in Fig. 1 belongs to *plant life* and *water*.

MIRFLICKR-25000 was developed as a dataset primarily oriented to content-based image retrieval so the annotators chose a particular order of the tags. They were asked to interpret the content in a wide sense and narrow sense, starting from the most general topics in the wide sense [2]. Especially in this paper we do not consider the tag order but in the future we plan to integrate it and define error rankings.

**Table 1.** MIRFLICKR-25000 categories

General topic	Subcategories	Number of images
sky	clouds	7912
water	sea/ocean, river, lake	3331
<b>people</b>	<b>portrait, boy/man, girl/woman, baby</b>	<b>18222</b>
night		3380
<b>plant life</b>	<b>tree, flower</b>	<b>8763</b>
<b>animals</b>	<b>dog, bird</b>	<b>3216</b>
man-built structures	architecture, building, house, city/urban, bridge, road/street	9992
<b>sunset</b>		<b>2135</b>
indoor		8313
<b>food</b>		<b>990</b>
transport	car	2895

We used WordNet [3] to expand the tagsets, thus generating longer context needed for TSD. WordNet (WN) contains a comprehensive list of word senses with synonyms and short textual definitions called glosses. Similar senses are manually grouped in “synsets” structured in a “is-a” hierarchy. Our TSD approach combines knowledge-based methods (Lesk algorithm and Hyponym Heuristics) and semantic measures [1]. It takes as input the image tags and tag features provided by WordNet – synonyms and possible senses, glosses and hypernyms. The semantic similarity measure used for disambiguation employs the WordNet semantic network. The algorithm outputs a single WordNet sense for each image tag/word included in WordNet, which is the basic linguistic resource in this case.

## 4 Experiments and Evaluation

MIRFLICKR-25000 is a rich source of image tags to experiment with, however without any gold standard for TSD. Therefore, one of the ideas we want to explore through this study is whether we can incorporate word embeddings (WE) as an alternative to a gold standard. We demonstrate that by applying the TSD algorithm on the tags of an image, we can get semantically closer to the category of that particular image, in means of WEs. The WEs we use are vector representations of the words as they are introduced by [13]. We employ three different models of WEs, trained on distinct datasets, and comment the semantic similarity at tag and at image level.


### 4.1 Input Data

We evaluate this study on the categories *Animals*, *Food*, *People*, *Plant life*, and *Sunset*. These are in total 19,529 images but 3,498 of them have no tags, so we end up with 16,031 input images. Due to assignment of multiple categories we actually work with

20,926 image-category pairs. In addition, we made experiments at tag level, there are 85,227 tag-category pairs in the above mentioned collection.

## 4.2 TSD Output

For each tag, included in WordNet as an English word, the TSD algorithm considers also the Lemma, Synset, Gloss, and Synonyms in WordNet for all possible senses. It outputs one sense suggested as the most probable intended meaning for the tag [1]. Figure 2 illustrates the output. Image-100 has MIRFLICKR-25000 categories “*female, people, plant life*” (categories and subcategories are listed together) and raw tags “*Farmer, Cisauk, rice field, D300, 18-200VR, Nikkor, TeeJe, Serpong, West Java, Indonesia, platinumphoto, A Big Fave*”. Two of these tags are included in WordNet: “*Farmer*” and “*Indonesia*”. Our TSD algorithm suggests that “*Farmer*” is “*a person who operates a farm*”. Now another system may offer machine translation of this tag into other languages.

<p><b>Input:</b>  <b>Image tags in WN:</b> Farmer, Indonesia  <b>TSD Output:</b>  <b>Lemmas:</b> farmer, Indonesia  <b>Synsets:</b> farmer.n.01, indonesia.n.01  <b>Synonyms:</b> farmer, husbandman, granger, sodbuster, Indonesia, Republic_of_Indonesia, Dutch_East_Indies  <b>Glosses of suggested senses:</b> “a person who operates a farm”, “a republic in southeastern Asia on an archipelago including more than 13,000 islands; achieved independence from the Netherlands in 1945; the principal oil producer in the Far East and Pacific regions”</p>	
---	---

**Fig. 2.** Output of the TSD algorithm for Image-100: tags, additional features extracted from WordNet and suggested senses after TSD

## 4.3 Objective of the Experiments

We performed separate experiments with the features extracted from WordNet - lemma, synset, gloss, and synonyms. Comparing their embeddings to the image category embedding we study how much these semantic features contribute to the semantic closeness of particular tags to the image category.

As we said above, there is no gold standard built on MIRFLICKR-25000 for TSD therefore we apply two alternative approaches to measure the success of our algorithm:

- manual evaluation of the tag sense disambiguation in means of accuracy,
- assessment of the contribution of the WordNet features to the semantic closeness of image tags to the image category. Our assumption is that *if the TSD is successful, then the WEs calculated using the WordNet features (lemmas, glosses, synsets) will be closer in the vector space to the image category embedding than the embeddings calculated on tags only*. In other words, we compare the semantic relatedness of the

raw image tags to the image category and the semantic relatedness of the tags enriched with WordNet features (after TSD) to the image category, with the help of WEs.

#### 4.4 Word Embeddings Models

We used three alternative WE models, all with 300-dimensional vectors:

**Model-1:** We started our experiments with pre-trained word and phrase vectors trained on part of Google News dataset (about 100 billion words). The model<sup>2</sup> contains vectors for 3 million words and phrases [13]. It is rather generic as it was trained on news corpora, in addition on separate wordforms without lemmatization, and due to these reasons we looked for other available models integrating linguistic knowledge.

**Model-2:** Since we are assessing WordNet word senses, a natural choice is to select a WE model which is trained on the WordNet semantic network. To this purpose, we follow the methodology outlined in [14]. One of the best performing vector space models described there is called “WN30+WN30glConOne C15”. We have reused the same knowledge graph (WN30+WN30glConOne) used to generate the artificial (pseudo) corpus for training the model as well as the Word2Vec settings. The artificial corpus consists of 200 million random walks along the graph structure. Next to each synset ID in the random walks, a randomly chosen lemma from the synset is inserted, so that the artificial sentences effectively double in size. This Model-2 provides representations only for lemmas and only for open-class words (nouns, verbs, adjectives and adverbs). Therefore, some input words (mostly functional ones) do not have matching vectors.

**Model-3:** It is employed to overcome the issue with missing functional words. Model-3 is trained using the same approach as Model-2 [15]; however, knowledge from Wikipedia is also included in an attempt to enlarge the coverage of lemmas and thus include functional words as well as to add more syntagmatic knowledge. A Wikipedia dump is lemmatized and concatenated to the pseudo corpus from WordNet which is used in [14]. This allows the neural network model to explore both types of knowledge, to learn representations of words excluded from WordNet and to relate them to the synsets. This model contains embeddings for lemmas, synsets and other words included by the Wikipedia dump.

#### 4.5 Computation of Embeddings

We explored several settings in calculations of embeddings and made various comparisons in order to study how close tag and category embeddings are. To aggregate information about numerous image tags, we compute the mean vector of separate WEs. Experiments I, II and III study the behaviour of image-category pairs.

---

<sup>2</sup> <https://code.google.com/archive/p/word2vec/>.

**Experiment I:** Using Model-1 we obtained the following WEs for each image-category pair:

- the mean vector of the WEs of the raw image tags (*MRTE – Mean Raw Tags Embeddings*), this vector characterizes the image tags;
- the mean vector of the WEs (MWE) of the words in the tag’ glosses obtained after the TSD with stop words removed (*MGE – Mean Gloss Embedding*), this vector characterizes the glosses of intended senses for the image tags;
- the MWE of the raw tags and their synonyms obtained after the TSD (*MTSE – Mean Tags and Synonym Embeddings*), this vector characterizes disambiguated image tags enriched with synonyms;
- the word/phrase embedding of the image category (*CE – category embedding*).

**Experiments II and III:** Since Model-1 is limited to wordforms only, we employ Model-2 and Model-3 to extend our experiments towards the usage of lemmas and make similar calculations, with the exception that in Model-2 there are no embeddings for synsets because the synset evaluation was done with Model-3 only. We obtain the following embeddings for each image-category pair:

- the MWE of the raw image tags (*MRTE – Mean Raw Tag Embeddings*);
- the MWE of the lemmatized raw tags (*MLTE – Mean Lemmatized TE*);
- the MWE of the lemmatized words in the tag glosses obtained after the TSD with stop words removed (*MLGE – Mean of Lemmatized Gloss Embeddings*);
- the MWE of the raw tags and tag synonyms obtained after the TSD, with the synonyms from WordNet in lemmatized form (*MLTSE*);
- the MWE of the synsets of the raw tags, only for Model-3 (*MSE*);
- the word/phrase embedding of the image category (*CE*).

We found out that the values of MRTE and MLTE differ very rarely. Probably this is due to the fact that human annotators usually tag by lemmas. Since Model-2 and Model-3 are trained on lemmatized data, we used MLTE and ignored MRTE.

With Model-3 we calculate the cosine similarity between the embeddings above (*MLTE, MLGE, MLTSE, MSE*) and the category embedding (*CE*) and check whether *MLGE, MLTSE*, and *MSE* are closer to *CE* in comparison to *MLTE*. With Model-2 we do the same excluding *MSE*, because this model does not contain synset embeddings.

**Experiment IV:** With Model-3, considered as the most detailed and suitable for our task, we perform one more experiment on each tag-category pair, by calculating the following embeddings:

- the tag lemma embedding for each raw image tag (*TE*);
- the MWE of the lemmatized words of the tag gloss, obtained after the TSD and the tag lemma, with stop words removed (*MLGEI*);
- the MWE of the raw tag lemma and tag synonyms in lemmatized form, obtained after the TSD (*MLTSEI*);
- the MWE of the raw tag synset and tag lemma (*MSEI*);
- word/phrase embedding of the category (*CE*).

## 4.6 Results

We report three types of observations. At first we consider the manual accuracy evaluation of the extracted WordNet senses for raw tags. Then we present findings about cosine similarity: on the one hand, cosine similarity of raw tagset embeddings to the category embeddings and, on the other hand, cosine similarity of the embeddings of the enriched tagset with WordNet features to the category embeddings. Comparisons are summarized in two settings: at image (tagset) level – Experiments I, II and III, Tables 4 and 5, and at tag level (using the tag-category pairs) – Experiment IV, Table 7.

**Manual accuracy evaluation.** We evaluated the accuracy of the TSD algorithm for the MIRFLICKR-25000 collection on a randomly selected subset of images from each category. The algorithm performs with highest accuracy on the category *Food* – 87%, followed by *Sunset* – 86%, *Animals* – 85%, *People* – 68%, and *Plant life* – 67%, with macro-averaged accuracy of 78.6%. The precision varies strongly, it depends on the category and the inclusion of tags in WordNet. For instance many images in *Plant life* are tagged by botanic terms which are wrongly disambiguated and hence the accuracy is lower.

Manual error analysis reveals the most common recurring mistake – to assign a WordNet sense which is a verb. Table 2 presents noun tags which are interpreted as verbs by the TSD algorithm. Verbs are rarely used as image tags so we plan to improve our TSD algorithm by introducing lower weights for verb senses. Another problem is the named entity resolution – names of locations are often erroneously resolved to personal names, see Table 3. Third often occurring problem is that among the raw image tags there are camera metadata such as camera model and photograph settings, and these are get wrongly assigned WordNet meanings.

**Table 2.** Samples of wrongly disambiguated noun tags

Noun tag	Meaning of the verb proposed by the TSD algorithm
landscape	do landscape gardening
mouse	manipulate the mouse of a computer
bus	send or move around by bus
lipstick	apply lipstick to
dress	dress in a certain manner
belt	fasten with a belt

**Embeddings-based assessment of semantic relatedness of tagsets to image category.** Here we use the embeddings calculated for experiments I, II and III. Tables 4 and 5 present results about images as a whole. Instead of individual tags we evaluate there the semantic closeness between the original tagset and the image category. Table 4 (based on Model-1) shows that for 82% of the images, when we compare the MWE of the glosses (MGE) to the category embeddings (CE), we obtain higher cosine similarity than when we use the original tagset WEs only (MRTE). When instead of the

**Table 3.** Samples of wrongly disambiguated names of locations

Tag location	Personal name proposed by the TSD algorithm
Berkeley	Irish philosopher and Anglican bishop who opposed the materialism of Thomas Hobbes (1685–1753)
London	US writer of novels based on experiences in the Klondike gold rush (1876–1916)
Houston	US politician and military leader who fought to gain independence for Texas from Mexico and to make it a part of the US (1793–1863)

**Table 4.** Improvement of the similarity to the category embedding using Model-1

Model	$\cos(MGE, CE) - \cos(MRTE, CE) > 0$	$\cos(MTSE, CE) - \cos(MRTE, CE) > 0$
Model-1	82%	50%

gloss embeddings we use the tag synonyms embeddings together with the original tags (MTSE), we obtain higher similarity only in 50% of the cases. Enriching tags by glosses provides better evidence for disambiguation.

**Table 5.** Improvement of the similarity to the category embedding using Model-2 and Model-3

Model	$\cos(MLGE, CE) - \cos(MLTE, CE) > 0$	$\cos(MLTSE, CE) - \cos(MLTE, CE) > 0$	$\cos(MSE, CE) - \cos(MLTE, CE) > 0$
Model-2	88%	50%	N/A
Model-3	93%	53%	75%

Table 5 summarizes respective results for Model-2 and Model-3 (which contain lemmas, glosses and other word embeddings). Here we show similarities calculated by using lemma embeddings instead of wordform embeddings like in Model-1. We observe that using the gloss embeddings (MLGE) improves the similarity to the category embedding with respect to the lemmatized raw tags embedding (MLTE) in 88% of the cases when using Model-2, and in 93% of the cases by applying Model-3 (Table 5 column 2). When we compare how much the synonyms help to improve the semantic similarity to the category in the embedding vector space, we see that this is in 50% of the cases when using Model-2 and in 53% of the cases when using Model-3 (Table 5 column 3). With Model-3, we can also observe the synsets contribution to the similarity to the image category - and this happens in 75% of the cases. (Table 5 column 4). Again, enriching tags by glosses seems to provide better disambiguation context.

**Embeddings-based assessment of semantic relatedness of single tags to image category.** The example in Table 6 shows more details about calculation of tag embeddings and their interpretation. It is easier to follow it because a single tag is being processed. The parameters calculated for Experiment IV are used here. Table 6 contains the tag lemma, an image category and the words in a gloss as well as the

**Table 6.** Image tag features and calculation of embeddings

Image tag	Embeddings with lemmas and gloss without stop words
Raw image tag: <i>bridge</i>	$\cos(WE(bridge), WE(plant\ life)) = 0.32$
Tag gloss from WordNet, suggested after TSD: “ <i>something resembling a bridge in form or function</i> ”	$\cos(MLGE1(“something\ resembling\ bridge\ form\ function”, bridge), WE(plant\ life)) = 0.76$
Image category: <i>plant_life</i>	Interpretation: closeness increase from 32% to 76% Conclusion: TSD is correct

calculation of MLGE1 – the mean vector embedding of lemmatized words in the tag gloss, obtained after the TSD with stop words removed and the tag lemma. We see that the cosine similarity between the raw tag lemma and the category is 0.32. Enriching the tag with the lemmas of the suggested correct sense gloss, we obtain a semantic vector which is closer to the category vector (76%). In terms of the experiment objective, as stated in Sect. 4.3, we conclude that the TSD is successful.

**Table 7.** Improvement of the similarity to the category embedding using Model-3 at tag level

Model	$\cos(MLGE1, CE) - \cos(TE, CE) > 0$	$\cos(MLTSE1, CE) - \cos(TE, CE) > 0$	$\cos(MSE1, CE) - \cos(TE, CE) > 0$
Model-3	98%	98%	92%

Table 7 summarizes observations how the WordNet features – glosses, synonyms and synsets obtained after the TSD contribute to the semantic similarity of a given single raw tag to the image category. By computing the mean the WEs of the gloss terms of a given tag and the tag itself, we get semantically closer in the embeddings space to the image category embedding in 98% of the cases (column 2). Similarly, by taking the MWE of the synonyms of a given tag and the tag itself, we obtain higher cosine similarity with the category WE again in 98% of the cases. When using the synset of the tag and the tag itself, we achieve increase in the cosine similarity in 92% of the cases.

Our interpretation of the figures in Table 7 is that in case of isolated tags, the synonyms and synsets seem to contribute to the semantic closeness to the image category as much as the gloss of the correct sense.

At the end of our result report, we can make the conclusion that Model-3 is the most suitable for our task. It is trained on WordNet and also includes Wikipedia words and phrases which improves its coverage and carries additional context about the relationships between lemmas, synsets and other words. We also notice that the glosses contribute the highest for achieving higher semantic similarity of tags to the image category. This is true for experiments on single tag level and at image (tagset) level. By saying so, we consider that the TSD contributes to uncovering the correct sense of the image tags.



## 5 Conclusion

We report here about one attempt to interpret the assessment of tag sense disambiguation as a task for calculation of semantic similarity among vectors in a 300-dimensional space of word embeddings. This exercise is inspired by the belief that in the big data era we need novel scenarios for revealing latent interdependencies among entities using the relatively imprecise instruments we have at hand. This is the case with the TSD tool we have produced earlier. Its manual evaluation on MIRFLICKR-25000 suggests average accuracy of 78.6% which is somewhat insufficient per se. But combined with calculation of word embeddings it might turn to be a reasonably good component that resolves ambiguity for tags included in external linguistic resources like WordNet (alternatively, in the future, another resource providing words, definitions and senses can be used for disambiguation e.g. Wikipedia). Our results also suggest that word embeddings seem to be an alternative approach to TSD because they help to reveal the semantic relatedness of image tags and image categories. The contribution of our TSD tool is essential too, because it delivers explicit context of synonyms, gloss and synset for each successfully disambiguated tag. It would be interesting to continue our experiments with integration of the so-called sense embeddings, a further deep learning development which helps to distinguish word senses [16].

By improving the approaches for TSD and obtaining high quality synsets for the image tags, we are actually supporting the machine translation of the large image collections. A resource of the size of MIRFLICKR-25000 would be of great interest to many researchers working on image recognition for languages other than English and steps towards the automatic translation of its annotation are needed.

## References

1. Kanishcheva, O., Angelova, G.: About sense disambiguation of image tags in large annotated image collections. In: Margenov, S., Angelova, G., Agre, G. (eds.) *Innovative Approaches and Solutions in Advanced Intelligent Systems*. SCI, vol. 648, pp. 133–149. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-32207-0\\_9](https://doi.org/10.1007/978-3-319-32207-0_9)
2. Huiskes, M., Lew, M.: The MIR Flickr Retrieval Evaluation. In: *Proceedings of ACM International Conference on Multimedia IR (MIR 2008)*, pp. 39–43. ACM, New York (2008)
3. WordNet, a Lexical Database for English. <https://wordnet.princeton.edu/>. Accessed 23 June 2018
4. Ferraro, F., Mostafazadeh, N., Huang, T.-H., Vanderwende, L., Devlin, J., Galley, M., Mitchell, M.: A survey of current datasets for vision and language research. In: *Proceedings of the 2015 EMNLP Conference*, Lisbon, Portugal, pp. 207–213 (2015)
5. Saenko, K.: Image sense disambiguation: a multimodal approach. PhD thesis MIT <http://hdl.handle.net/1721.1/54651>. Accessed 11 May 2018
6. Saenko, K., Darrell, T.: Unsupervised learning of visual sense models for polysemous words. In: *Advances in Neural Information Processing Systems (NIPS 2008)*, Vancouver, Canada, vol. 21, pp. 1393–1400 (2009)

7. Lee, K., Kim, H., Shin, H., Kim, H.: Tag sense disambiguation for clarifying the vocabulary of social tags. In: International Conference on Computational Science and Engineering, Vancouver, Canada, pp. 729–734 (2009)
8. James, N., Hudelot, C.: Towards semantic image annotation with keyword disambiguation using semantic and visual knowledge. In: the IJCAI-2009 Workshop on Cross-Media Information Access and Mining. [http://liir.cs.kuleuven.be/conferences/CIAM2009/ciam2009\\_6.pdf](http://liir.cs.kuleuven.be/conferences/CIAM2009/ciam2009_6.pdf). Accessed 24 Apr 2018
9. Legesse, M., Gianini, G., Teferi, D.: Selecting feature-words in tag sense disambiguation based on their shapley value. In: Proceedings 12th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS), Naples, Italy, pp. 236–240 (2016)
10. May, W., Fidler, S., Fazly, A., Dickinson, S., Stevenson, S.: Unsupervised disambiguation of image captions. In: Proceedings of the First Joint Conference on Lexical and Computational Semantics (SemEval 2012), Montréal, Canada, vol. 1, pp. 85–89 (2012)
11. Iacobacci, I., Pilehvar, M.T., Navigli, R.: SENSEMBED: learning sense embeddings for word and relational similarity. In: Proceedings of the 53rd Annual Meeting of ACL and the 7th International Joint Conference on NLP, Beijing, China, pp. 95–105 (2015)
12. Raiman, J., Raiman, O.: DeepType: multilingual entity linking by neural type system evolution. In: Proceedings 32th AAAI Conference on AI (AAAI-2018), February 2018, New Orleans, Louisiana, USA (2018). <https://arxiv.org/abs/1802.01021>. Accessed 24 Apr 2018
13. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Proceedings of the 26th International Conference on Neural Information Processing Systems (NIPS 2013), Nevada, USA, vol. 2, pp. 3111–3119 (2013)
14. Simov, K., Osenova, P., Popov, A.: Comparison of word embeddings from different knowledge graphs. In: Gracia, J., Bond, F., McCrae, John P., Buitelaar, P., Chiarcos, C., Hellmann, S. (eds.) LDK 2017. LNCS (LNAI), vol. 10318, pp. 213–221. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-59888-8\\_19](https://doi.org/10.1007/978-3-319-59888-8_19)
15. Popov, A.: Word sense disambiguation with recurrent neural networks. In: Kovatchev, V., et al. (eds.) Proceedings of the Student Research Workshop Associated with RANLP 2017, Varna, Bulgaria, pp. 25–34 (2017)
16. Camacho-Collados, J., Taher Pilehvar, M.: From Word to Sense embeddings: a survey on vector representations of meaning. Submitted to JAIR, [arXiv:1805.04032](https://arxiv.org/abs/1805.04032), May 2018. <http://adsabs.harvard.edu/abs/2018arXiv180504032C>. Accessed 22 June 2018



# Echo State Network for Word Sense Disambiguation

Petia Koprinkova-Hristova, Alexander Popov, Kiril Simov,  
and Petya Osenova<sup>(✉)</sup>

Institute of Information and Communication Technology,  
Bulgarian Academy of Sciences, Akad. G. Bonchev. 25A, 1113 Sofia, Bulgaria  
pkoprinkova@bas.bg, {alex.popov,kivs,petya}@bultreebank.org

**Abstract.** The current developments in the area report on numerous applications of recurrent neural networks for Word Sense Disambiguation that allowed the increase of prediction accuracy even in situation with sparse knowledge due to the available generalization properties. Since the traditionally used LSTM networks demand enormous computational power and time to be trained, the aim of the present work is to investigate the possibility of applying a recently proposed fast trainable RNN, namely Echo state networks. The preliminary results reported here demonstrate the applicability of ESN to WSD.

**Keywords:** Word sense disambiguation · Word embedding  
Sense embedding · Echo state network

## 1 Introduction

Word Sense Disambiguation (**WSD**) is a very important task for NLP and thus it enjoys constant interest. Only recently Neural Networks became predominant approaches in this area. In our paper we report on preliminary experiments from having applied Echo state networks (ESN) to WSD. Echo state networks are a member of the reservoir computing family [1] proposed first in [2]. The main aim was the development of fast trainable recurrent neural networks (RNN) and it was achieved by generating a random and sparsely connected recurrent reservoir of neurons and linear readout that can be tuned in one shot (presenting each training sample only once).

ESN were widely used for modeling of dynamic systems [3]. Applications of ESN in NLP started recently so there are only few works in this area. In [4, 5] ESNs were applied for semantic role labeling in a multi-modal robotic architecture with natural language feedback. Other NLP applications are in the area of speech processing—[6, 7], and in the area of language modeling—[8]. To the best of our knowledge, there are not yet examples for applications of ESNs for WSD since another promising RNN architecture (LSTM) was widely explored [9]. However, the LSTM networks training has been done by gradient algorithm (usually backpropagation). It is much slower and demanding more

computational resources than the training algorithm (one shot least squares) of ESN. Hence the aim of the present work is to investigate the possibility of applying the ESN instead of LSTM to the WSD task, and observe whether the former is a good alternative to the latter.

The experiments we report here use an artificially created corpus (pseudo-corpus) for training as well as for testing. The motivation for this is that we could control the size of both corpora and also the coverage over the vocabulary. For the generation of the pseudo-corpus we used the UKB System.<sup>1</sup>

The structure of the paper is as follows: the next section discusses the related work. Section 3 presents the data used in the experiments and the basics of Echo state networks. In Sect. 4 we describe the Echo state network model for WSD and the results from our preliminary experiments. The last section concludes the paper.

## 2 Related Work

Here we present two of the most popular approaches to WSD – the knowledge-based one and the supervised one. For more details on different approaches the interested reader is referred to [10]. The WSD task means assignment of the correct meaning of a word form within its specific context of use. The most used source for English in that respect is the Princeton English WordNet – [11]. WordNet represents meaning in terms of synonymic sets (synsets) containing lexical items that share the same meaning. The meaning is determined by the relations between the synsets within WordNet. Such relations are hyperonymy, meronymy, antonymy, etc. Additionally, the meaning is explicated by a gloss (definition of the meaning) and an example of usages of the lexical items in the synset. The lexical items are represented by their lemma. The supervised systems require a reasonable amount of manually annotated data which is very expensive to be built. As an alternative approach to the manually annotated data is the automatically constructed one like the one-million-word corpus created for the training of the IMS system—[12] which uses an SVM algorithm to perform the disambiguation. Currently the preferred framework for the supervised WSD systems are the Neural Networks including deep ones—[9]. In our work we also use the Neural Network approach, but in the special form of Echo state networks as discussed in the introduction.

Knowledge-based systems for WSD have proven to be a good alternative to the supervised systems. The knowledge-based systems require only a knowledge base and no additional corpus-dependent information. An especially popular knowledge-based disambiguation approach has been the use of popular graph-based algorithms known under the name of “Random Walk on Graph” [13]. Most approaches exploit variants of the PageRank algorithm [14]. Agirre and Soroa (2009) [15] apply a variant of the algorithm to the WSD by translating WordNet into a graph in which the synsets are represented as nodes and the relations between them are represented as arcs. The resulting graph is called a

<sup>1</sup> <http://ixa2.si.ehu.es/ukb/>.

*knowledge graph* in this paper. In our work reported here we exploit the idea of Random walk on graphs in order to generate the pseudo-corpus used for the training and testing phases in our experiments.

### 3 Experimental Set-Up

In this section we present the creation of the pseudo corpus used in our experiments and the basics of the Echo state networks.

#### 3.1 The Creation of the Pseudo Corpus

For the generation of pseudo corpus we exploit the UKB system, which provides graph-based methods for WSD and for measuring the lexical similarity. In the experiments reported here we exploit it as a generator of pseudo corpora. Such pseudo corpora can be the output from the Random Walk algorithm, when it is set to the mode of selecting sequences of nodes from a knowledge graph (KG)—see [16] for the generation of pseudo corpora from a WordNet knowledge graph and [17] for the generation of pseudo corpora from RDF knowledge graphs such as DBPedia, GeoNames, FreeBase. The UKB system uses a set of random-walk-on-graph algorithms, described in [15]. The tool can also build a knowledge graph over a set of relations that can be induced from different types of resources, such as WordNet or DBPedia. The UKB tool requires two resources. First is a lexicon in which each record is represented as a lemma and a list of associated sense identifiers, taken from WordNet in our case:

```
predicate 06316813-n:0 06316626-n:0 01017222-v:0
```

After each sense identifier a number following a colon indicates the frequency of the word sense, calculated on the basis of a tagged corpus. The second resource represents the relations between the different senses. These relations are used to form the knowledge graph where the nodes are the sense identifiers and the arcs are the relations between them. We use the resource files for WordNet version 3.0. The standard lexical relations from WordNet are hypernymy, meronymy, etc. In addition to the standard relations we use other relations that were extracted from different sources—[18]. The format of the relations in the KG is as follows:

```
u:SynSetId01 v:SynSetId02 s:Source d:w
```

where *SynSetId01* is the identifier of the first synset in the relation, *SynSetId02* is the identifier of the second synset, *Source* is the source of the relation, and *w* is the weight of the relation in the graph. In the experiments reported in the paper, the weight of all relations is set to 0. The generation of the pseudo corpus was done by the system in the following way:

1. An empty sequence is created.
2. A node in the graph is selected randomly.
3. The assigned to the node synset ID is added to the sequence.

4. A probability for the sequence is calculated.
5. If the probability is less than a threshold, the sequence is printed and made empty. The execution proceeds with step 2.
6. If the probability surpasses the threshold, then if there are arcs going out from the current node, one of these arcs is selected randomly and traversed to the corresponding node. The execution proceeds with step 3.
7. After the construction of N sequences the process stops.

The result from this procedure is called *sense pseudo corpus*. From a sense pseudo corpus we generate a *lemma pseudo corpus* selecting randomly for each synset a lemma from the synset. In this way we have a corpus of pseudo sentences such that for each lemma the corresponding sense is known. We consider each sequence in the pseudo corpus a *lexical chain* corresponding to some real sentence. Here is an example of such a pseudo sentence:

goldbrick dupery take\_in gull dupe person laugh\_at

Executing the system several times we produce several pseudo corpora on which we train lemma and sense embeddings and also train the Echo state network model for WSD. Needless to say, different pseudo corpora are used for the different tasks.

### 3.2 Echo State Network Basics

The structure of ESN is shown on Fig. 1. It incorporates a dynamic reservoir of neurons with randomly generated recurrent connections and linear readout:

$$out(k) = W^{out}[in(k), R(k)] \quad (1)$$

Here,  $in(k)$  is the vector of network inputs and  $R(k)$  the vector of the reservoir neuron states;  $W^{out}$  is a  $n_{out} \times (n_{in} + n_R)$  trainable matrix (here  $n_{out}$ ,  $n_{in}$  and  $n_R$  are the sizes of the corresponding vectors  $out$ ,  $in$  and  $R$ ).

The neurons in the reservoir have a simple sigmoid output function, usually hyperbolic tangent, that depends on both the ESN input  $in(k)$  and the previous reservoir state  $R(k - 1)$ :

$$R(k) = (1 - a)R(k - 1) + a \tanh(W^{in}in(k) + W^{res}R(k - 1)) \quad (2)$$

Here  $W^{in}$  and  $W^{res}$  are  $n_{in} \times n_R$  and  $n_R \times n_R$  matrices that are randomly generated and are not trainable; the parameter  $a$ , called leaking rate, influences the reservoir short term memory and in many applications was omitted, i.e.  $a$  was set to 1. The only trainable parameters of ESN are the output connection weights contained in the  $n_{out} \times (n_{in} + n_R)$  dimensional matrix  $W^{out}$ .

According to recipes in [2] the reservoir connection matrix  $W^{res}$  should be generated so as to guarantee “echo state property” of the ESN, i.e. the changes in the input vector must be reflected like “echo” at the output vector (that means the response effect should vanish gradually with time). This is achieved

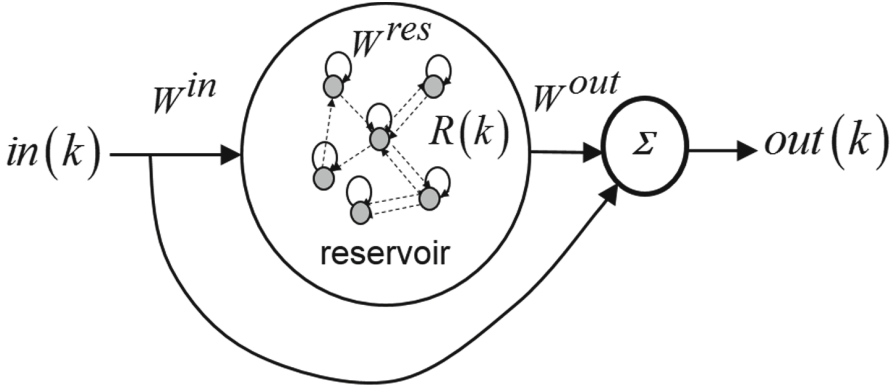


Fig. 1. Echo state network structure.

by proper normalization of the matrix  $W^{res}$  so that its spectral radius become smaller than 1.

The output weights can be tuned by solving the linear regression equation (least squares approach) in one shot after single presentation of all input/output training samples or iteratively using its recursive version (RLS) presenting each training input/output sample one by one [2].

## 4 Echo State Network for WSD

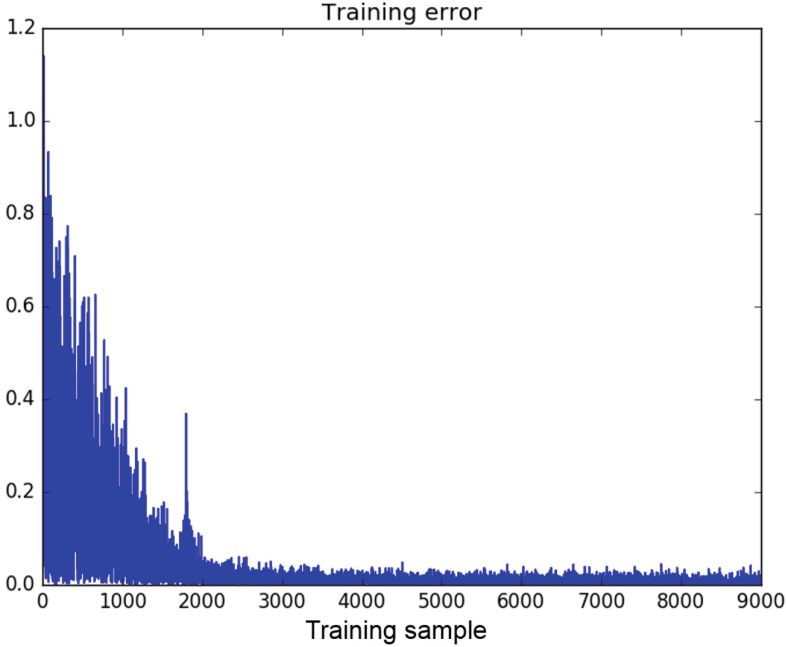
The input vector of our ESN for WSD was composed by embeddings of a sequence of words taken from training set of lemmas using a fixed-length (three words) window omitting the middle word. The ESN was trained to predict the embedding of the middle word synonyms, thus reflecting its sense. The size of word embedding vectors was 300. Thus the input vector of our ESN contained 600 elements while the output vector had 300 elements.

For the aim of ESN training and simulation, a software on python was developed. It includes both LS and RLS tuning algorithms but since the size of the training dataset was quite big (for the first tests we generated about 9000 examples for training and testing respectively), we used the recursive version.

Figure 2 represents the changes of the root mean square error (RMSE):

$$RMSE(k) = \frac{1}{n_{out}} \sqrt{\sum_{i=1}^{n_{out}} (out^{ESN}(i) - out_k^{train}(i))^2} \quad (3)$$

during RLS training. Each training sample  $k$  was presented once and the corresponding RMSE was calculated. It was observed that after first 3000 samples the training error remains almost the same although it continued to decrease slightly.



**Fig. 2.** RMSE during RLS training of ESN.

We trained reservoirs of different sizes (from 100 up to 1200 neurons). The other parameters of the ESN that were investigated are the leaking rate  $a$  and sparsity (percentage of non-zero elements) of the reservoir weight matrix  $W^{res}$ . Table 1 presents mean values of RMSE for all training/testing data:

$$RMSE = \frac{1}{n_{data}} \sum_{k=1}^{n_{data}} RMSE(k) \quad (4)$$

for different set of ESN parameters and Table 2 - the variances of the corresponding errors from Table 1.

From Table 1 we conclude that the bigger the reservoir size, the smaller the achieved training and testing errors. However, for the biggest reservoir size (1200 neurons) the testing error slightly increased in comparison to the other investigated cases.

The differences between the errors are not too big but we observed that with the increase of the reservoir matrix sparsity (that means more recurrent connections in the reservoir), the training and testing errors slightly decreased. The same was the effect of the decreased leaking rate ( $a = 0.5$ ) that in practice led to increased duration of the ESN short memory.

From Table 2 we conclude that the variance of the achieved testing and training errors decreases with the increase of the reservoir size no matter what was



**Table 1.** RMSE from training and testing of ESN for WSD having different leaking rate and reservoir connection matrix sparsity.

$n_R$	$W^{res} \text{ sparsity}$	$a$	$RMSE^{train}$	$RMSE^{test}$
100	0.2	1	0.001807	0.001981
100	0.8	1	0.001804	0.001979
100	0.5	0.5	0.001794	0.001969
300	0.2	1	0.001751	0.001969
600	0.2	1	0.001685	0.001969
1200	0.2	1	0.001575	0.001996

**Table 2.** Variances of the training and testing RMSE of ESN for WSD having different leaking rate and reservoir connection matrix sparsity.

$n_R$	$W^{res} \text{ sparsity}$	$a$	$RMSE_{var}^{train}$	$RMSE_{var}^{test}$
100	0.2	1	$4.2466e^{-9}$	$4.5899e^{-9}$
100	0.8	1	$3.9920e^{-9}$	$4.7130e^{-9}$
100	0.5	0.5	$3.8663e^{-9}$	$4.5110e^{-9}$
300	0.2	1	$3.0145e^{-9}$	$3.7606e^{-9}$
600	0.2	1	$2.4161e^{-9}$	$3.1849e^{-9}$
1200	0.2	1	$1.7818e^{-9}$	$2.7443e^{-9}$

the leaking rate or reservoir connection matrix sparsity. Since all variances are quite small, we can expect good disambiguation of the predicted word synonyms.

Next we trained an ESN using as input vector the embeddings of all three words from the window, thus having input vector with size 900 and output vector again of size 300. The obtained testing and training errors decreased significantly as it is shown in Tables 3 and 4 even in the case of a smaller reservoir size (100 neurons only).

**Table 3.** RMSE from training and testing of ESN for WSD using the full window of three words as input.

$n_R$	$W^{res} \text{ sparsity}$	$a$	$RMSE^{train}$	$RMSE^{test}$
100	0.5	1	0.001469	0.001679

Next we tested the last ESN model with data generated from 1581 sentences taken from SemCor (See [19])—16827 testing input/output data pairs. Since the text contained phrases like “*in fact*”, “*in common*” etc. for which there were no synonym embeddings, we replaced them with zero vectors. The achieved testing RMSE and its variance were similar to those from previous test data set, namely  $RMSE^{test} = 0.001341$  and  $RMSE_{var}^{test} = 1.7852e^{-9}$  respectively.

**Table 4.** Variances of the training and testing RMSE of ESN for WSD using the full window of three words as input.

$n_R$	$W^{res} sparsity$	$a$	$RMSE_{var}^{train}$	$RMSE_{var}^{test}$
100	0.5	1	$0.8383e^{-9}$	$1.1423e^{-9}$

**Table 5.** Evaluation over 1581 sentences from SemCor.

Baseline	ESN
70.68 %	51.75 %

Our ESN model maps the input embeddings for the lemmas to the output embeddings for the corresponding senses (synsets in WordNet). Of course, during the test the mapping is not exact. Thus, for the actual task of WSD we need to interpret the output of ESN as embeddings for the synsets. We have used the cosine measure for similarity between the gold sense embeddings and the output vector of our ESN model. In this evaluation we have used the model trained on complete three grams reported in Tables 3 and 4. We first evaluated the model over the test set from the pseudo corpus. The precision is 69.41 %. The results over SemCor are represented in Table 5. The baseline was calculated on the basis of the most frequent sense in WordNet. The results show that the first experiments with ESN, despite giving relatively good results with respect to the RMSE measure, perform very poorly. In order to gain some insights about the reasons for this, we performed error analysis. The main errors can be classified into the following two groups:

- a wrong POS (noun instead of verb or vice versa);
- a wrong meaning in words with high polysemy (as *to be*, *man*).

The words with good prediction represent:

- more specific lexicalizations (*witness*, *doubt*);
- named entities (*Atlanta*, *Georgia*; *names of the week and months*, *professions like attorney*, *mayor*, *governor*).

We noted that among the 29 examples with the highest dissimilarity (more than 0.5) there are the abstract verbs, such as: *have*, *state*, *act*, *being*, as well as cognitive verbs, such as: *concern*, *matter*, *refer* prevail. From this we conclude that although ESN demonstrate good performance with respect to RMSE, it do not capture good estimation on lexical items with a high degree of polysemy.

There, however, also some abstract nouns are spotted, such as: *thing*, *object*, *animal*, *sound*, *abstraction*. Here the wrong POS is not only between a noun and a verb (*matter*, *concern*, *act*), but also between an adjective and a verb (*sound*; *live*); an adjective and a noun (*green*, *whole*).

## 5 Conclusion

In the paper we report preliminary results from experiments with Echo state network for WSD. Although the results over the SemCor are very pessimistic, we think that some useful insights for future work can be derived. In the first place, the generation of the pseudo corpus always follows the paths of relations within WordNet knowledge graph with step one. This was useful for training word embeddings (see [16,20]). Our initial intuition that the pseudo sentences represent potential lexical chains in texts failed. The pseudo sentences contain the lexical items that could be found in a lexical chain, but the word order in the pseudo sentences reflects the one-step paths of relations in WordNet. Such a word order is rare in real texts. For that reason, ESN performs well on the test set extracted from the pseudo corpus, but fails on the test set extracted from real texts. Another reason is that the three-word context is very small for the task of WSD. Knowledge-based approaches and LSTM architectures consider the whole sentence as a context or a window of at least 20 words.

Our plans are to train the model on manually annotated real text data. With respect to the generation of pseudo corpus more work is necessary in order to generate pseudo sentences that approximate in a better way the lexical chains in the texts.

Also we plan to construct a larger model comprising subnetworks with several reservoirs for solving several tasks simultaneously including POS tagging, lemmatization, coreference resolution and WSD. In this way we hope to rule out some of the errors that follow from the pipeline model.

**Acknowledgements.** This research has received partial support by the grant 02/12—*Deep Models of Semantic Knowledge (DemoSem)*, funded by the Bulgarian National Science Fund in 2017–2019. We are grateful to the anonymous reviewers for their remarks, comments, and suggestions. All errors remain our own responsibility.

## References

1. Lukosevicius, M., Jaeger, H.: Reservoir computing approaches to recurrent neural network training. *Comput. Sci. Rev.* **3**, 127–149 (2009)
2. Jaeger, H.: Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the echo state network approach. GMD Report 159, German National Research Center for Information Technology (2002)
3. Jaeger, H.: Adaptive nonlinear system identification with echo state networks. In: *Advances in Neural Information Processing Systems 15 (NIPS 2002)*, pp. 593–600. MIT Press, Cambridge (2003)
4. Twiefel, J., Hinaut, X., Wermter, S.: Semantic role labelling for robot instructions using echo state networks. In: *Proceedings of the European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*, Bruges, Belgium, pp. 695–700 (2016)
5. Twiefel, J., Hinaut, X., Borghetti, M., Strahl, E., Wermter, S.: Using natural language feedback in a neuro-inspired integrated multimodal robotic architecture. In: *Proceedings of the 25th IEEE International Symposium on Robot and Human Interactive Communication (ROMAN)*, New York City, USA (2016)

6. Skowronski, M., Harris, J.: Minimum mean squared error time series classification using an echo state network prediction model. In: 2006 IEEE International Symposium on Circuits and Systems, IEEE (2006)
7. Squartini, S., Cecchi, S., Rossini, M., Piazza, F.: Echo state networks for real-time audio applications. In: Liu, D., Fei, S., Hou, Z., Zhang, H., Sun, C. (eds.) ISNN 2007. LNCS, vol. 4493, pp. 731–740. Springer, Heidelberg (2007). [https://doi.org/10.1007/978-3-540-72395-0\\_90](https://doi.org/10.1007/978-3-540-72395-0_90)
8. Tong, M.H., Bickett, A.D., Christiansen, E.M., Cottrell, G.W.: Learning grammatical structure with echo state networks. *Neural Networks* **20**(3), 424–432 (2007). *Echo State Networks and Liquid State Machines*
9. Popov, A.: Neural network models for word sense disambiguation: an overview. *Cybern. Inf. Technol.* **18**, 139–151 (2018)
10. Navigli, R.: Word sense disambiguation: a survey. *ACM Comput. Surv. (CSUR)* **41**(2), 10 (2009)
11. Fellbaum, C. (ed.): *WordNet: An Electronic Lexical Database*. The MIT Press, Cambridge, London (1998)
12. Zhong, Z., Ng, H.T.: It makes sense: a wide-coverage word sense disambiguation system for free text. In: *Proceedings of the ACL 2010 System Demonstrations*, pp. 78–83. Association for Computational Linguistics (2010)
13. Agirre, E., López de Lacalle, O., Soroa, A.: Random walks for knowledge-based word sense disambiguation. *Comput. Linguist.* **40**(1), 57–84 (2014)
14. Brin, S., Page, L.: The anatomy of a large-scale hypertextual web search engine. *Comput. Netw.* **56**(18), 3825–3833 (2012)
15. Agirre, E., Soroa, A.: Personalizing PageRank for word sense disambiguation. In: *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pp. 33–41 (2009)
16. Goikoetxea, J., Soroa, A., Agirre, E.: Random walks and neural network language models on knowledge bases. In: *HLT-NAACL*, pp. 1434–1439. The Association for Computational Linguistics (2015)
17. Ristoski, P., Paulheim, H.: RDF2Vec: RDF graph embeddings for data mining. In: Groth, P. (ed.) *ISWC 2016*. LNCS, vol. 9981, pp. 498–514. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-46523-4\\_30](https://doi.org/10.1007/978-3-319-46523-4_30)
18. Simov, K., Osenova, P., Popov, A.: Using context information for knowledge-based word sense disambiguation. In: Dichev, C., Agre, G. (eds.) *AIMSA 2016*. LNCS (LNAI), vol. 9883, pp. 130–139. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-44748-3\\_13](https://doi.org/10.1007/978-3-319-44748-3_13)
19. Miller, G.A., Leacock, C., Teng, R., Bunker, R.T.: A semantic concordance. In: *Proceedings of HLT 1993*, pp. 303–308 (1993)
20. Simov, K., Osenova, P., Popov, A.: Comparison of word embeddings from different knowledge graphs. In: Gracia, J., Bond, F., McCrae, J.P., Buitelaar, P., Chiarcos, C., Hellmann, S. (eds.) *LDK 2017*. LNCS (LNAI), vol. 10318, pp. 213–221. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-59888-8\\_19](https://doi.org/10.1007/978-3-319-59888-8_19)



# Constrained Permutations for Computing Textual Similarity

Allan Ramsay<sup>(✉)</sup> and Amal Alshahrani

University of Manchester, Manchester, UK

Allan.Ramsay@manchester.ac.uk, amal.alshahrani@postgrad.manchester.ac.uk

**Abstract.** A wide range of algorithms for computing textual similarity have been proposed. Much recent work has been aimed at calculating lexical similarity, but in general such calculations have to be treated as components in larger algorithms for computing similarity between sentences.

In the current paper we describe a refinement of the well-known dynamic-time warping (DTW) algorithm for calculating the string edit distance between a pair of texts. The refined version of this algorithm allows for a range of constrained permutations without increasing the complexity of the underlying algorithm.

**Keywords:** Text similarity · Alignment

## 1 Introduction

Spotting that two sentences are similar is useful in a wide range of applications – news aggregation services, where it is desirable to avoid repeating closely matched texts obtained from multiple sources, information retrieval tasks, plagiarism detection algorithms, . . . Techniques for carrying out this task vary considerably in computational complexity, from simple bag-of-words algorithms, where the time taken to match a pair of sentences is proportional to the sum of their lengths, to systems that carry out deep reasoning over formal paraphrases of the two texts, which can be anywhere from exponential in the lengths of the texts to semi-decidable (if the translation is to first-order logic) or undecidable (if the translation is to something more expressive).

In the current paper we propose an extension to the standard ‘weighted dynamic time warping’ algorithm. The standard algorithm finds the least cost sequence of string edits (i.e. **insertion**, **deletion** or **exchange** of items) to turn one string into another. If the cost of **exchange** is made sensitive to the similarity of the words being exchanged, as suggested by [8], then this algorithm can detect sentences that have similar meanings. Consider, for instance, #1 and #2 (the elided elements of #1 are identical and hence have been deleted to keep the pairings printable):

- (1) ... had argued that he had never personally killed or beaten anyone  
       ... had discussed that he had never killed or beaten anyone
- (2) the rock icon is said to have cheated death on numerous occasions  
       the rock singer is said to have cheated death on many occasions

In #1, most of the words are direct matches (indicated by the dotted lines connecting them to their partners), the words ‘discussed’ and ‘argued’ are reasonably similar, though not identical, and the word ‘personally’ has to be deleted from the first sentence to make it match the second. In #2 we have two pairs of non-identical but similar words, ‘icon’/‘singer’ and ‘numerous’/‘many’. If we make the cost of exchanging one for another depend on the ‘similarity’ of the two words then this technique works reasonably well for calculating the similarity of the two sentences: the similarity of a pair of sentences is their string-edit distance where the cost of exchange depends on the similarity of the matched words. Any similarity measure can be plugged into this definition. In the examples in this paper we are using Wu-Palmer similarity [6] for simplicity, but other kinds of measure, e.g. one based on word embeddings [3, 4], could be used. The work reported here relates to the use of such measures within the calculation of the string-edit distance, not to the measures themselves.

The time and space complexity of the standard dynamic time warping algorithm are  $N^2 \times sim$ , where  $N$  is the length of the shorter of the two sequences to be matched and  $sim$  is the time taken to compare two words. This bound can be improved if we assume that the pair of sentences being matched do have something in common, since it is then possible to concentrate on a ‘corridor’ of fixed width, in which case the complexity is  $N \times C \times sim$ , where  $N$  is the length of the shorter sentence and  $C$  is the width of the corridor [5]. The standard algorithm, however, is order-preserving. Given a pair of sentences such as the ones in #3, it cannot spot that the words have been swapped around:

- (3) the raised money will go to the Mount Vernon cancer centre  
       all the money raised will go to the Mount Vernon cancer centre

The best that the standard algorithm can do is to match one of the swapped words, ‘money’ and ‘money’ in the current example, and delete a copy of the other from one sentence and insert one into the other.

The standard algorithm attempts to use the set of recurrence relations in Eq. 4 to find the cheapest set of edit operations for transforming a source sequence  $S^1$  into a target sequence  $S^2$  by systematically traversing a grid  $A_{i,j}$

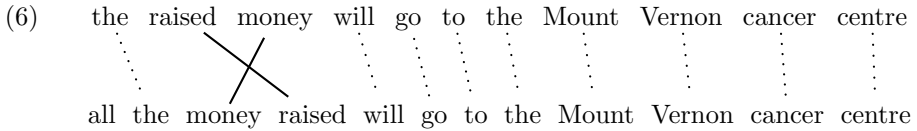
whose axes are defined by the elements of the two sequences:

$$cost(A_{i,j}) = \min \left\{ \begin{array}{l} cost(A_{i-1,j-1}) + \text{exch}(S_i^1, S_j^2) \\ cost(A_{i-1,j}) + \text{insert}(S^1, i) \\ cost(A_{i,j-1}) + \text{delete}(S^2, j) \end{array} \right\} \quad (4)$$

Damerau [2] suggested adding a new edit step, **swap**, to the standard three. **swap** involves exchanging  $S_i^1$  for  $S_{i+1}^2$  and  $S_{i+1}^1$  for  $S_i^2$ , where  $S_k^i$  denotes the  $k$ th word of sentence  $i$ . The cost of this operation is the cost of doing each of the exchanges plus some fixed penalty  $K$  for the two words being out of order. This leads to the set of recurrence relations in Eq. 5:

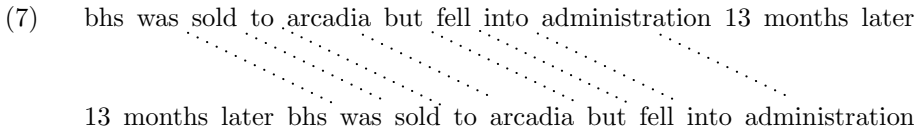
$$cost(A_{i,j}) = \min \left\{ \begin{array}{l} cost(A_{i-1,j-1}) + \text{exch}(S_i^1, S_j^2) \\ cost(A_{i-1,j}) + \text{insert}(S^1, i) \\ cost(A_{i,j-1}) + \text{delete}(S^2, j) \\ cost(A_{i-2,j-2}) + \text{exch}(S_{i-1}^1, S_j^2) \\ \quad + \text{exch}(S_i^1, S_{j-1}^2) + K \end{array} \right\} \quad (5)$$

Note that the cost of a **swap** here is based on the cost of a standard **exchange**. Given that we are calculating the cost of **exchanges** in terms of some measure of lexical similarity, **swaps** will also favour cases where both the pairs of words being swapped are similar. In the case of #3 this leads to the following alignment, at a lower cost than in #3 since both the exchanges are zero cost so the cost of the **swap** is just  $K$ :

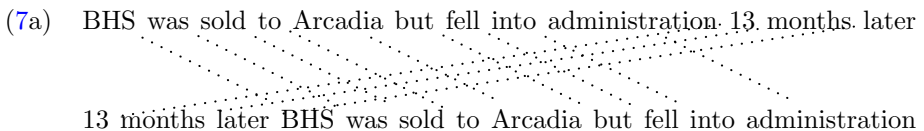


Equation 5 contains an extra step, but as with **insert**, **delete** and **exchange** it is a fixed cost step that is performed once for every element of the array and hence has no effect on the complexity of the algorithm.

Swaps do not, however, only involve adjacent items. Consider #7:



We propose a further extension to the basic algorithm which makes it possible to match swapped sequences, rather than simply swapped words, thus enabling us to deal with cases like #7 as in #7a:



It is easier to explain the extension to DTW that allows us to cope with [#7a by looking at it algorithmically than by adapting the recurrence relations above.

The standard DTW algorithm involves making a 2D array, where the axes are the sequences being compared, and then trying to find the cheapest route through this array. This can be done backwards: systematically visit each point and look at the ways you could have arrived at this point, or forwards: systematically visit each point and look at all the places you could get to from here. The version employed here runs forwards: a slightly abstracted version is given in Fig. 1. `S1` and `S2` are the two sequences, `costInsert`, `costDelete` and `costExchange` are functions for calculating the cost of inserting or deleting an item or exchanging two items (these are often taken to be fixed values, but greater flexibility can be obtained by making them functions).

```
def dtw(S1, S2):
    A = array(S1, S2)
    for i in range(len(S1)):
        for j in range(len(S2)):
            extend(A, S1, S2, i, j)

def extend(A, S1, S2, i, j):
    insert = A[i][j]+costInsert(S1[i+1])
    if insert < A[i+1][j]:
        A[i+1][j] = insert
    delete = A[i][j]+costDelete(S2[j+1])
    if delete < A[i][j+1]:
        A[i][j+1] = delete
    exchange = A[i][j]+costExchange(S1[i+1], S2[j+1])
    if exchange < A[i+1][j+1]:
        A[i+1][j+1]
```

Fig. 1. Basic DTW algorithm

This algorithm visits every cell in the array `A` and carries out three fixed cost operations at each. The time complexity is hence  $len(S1) \times len(S2)$ . The correctness is proved inductively. Assume that the value at `A[i][j]` represents the least cost sequence of inserts, deletes and exchanges that will turn `S1[0:i]` into `S2[0:j]`. After `extend(A, S1, S2, i, j)` has been carried out, `A[i+1][j]`, `A[i][j+1]` and `A[i+1][j+1]` will contain the lowest costs that can be obtained by extending this sequence by inserting, deleting or exchanging something. Eventually every point in the array will have been explored as the result of an insert, a delete or an exchange, and it will have a score that reflects which of these was the cheapest.

It is also useful to keep a backpointer showing where the last step that led to each point came from. Figure 2 shows a couple of intermediate steps during the execution of this algorithm for the sequences `abcxyz` and `abyrz`, using fixed costs of 2 for insert and delete and 3 for exchange if the two items are different.



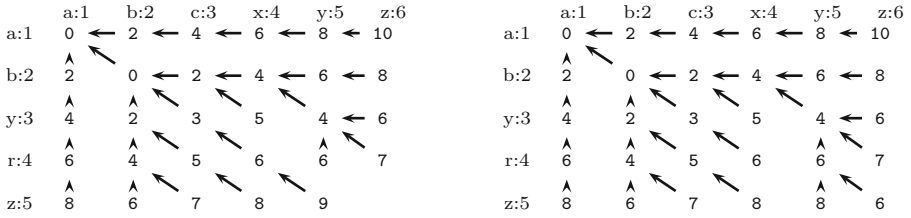


Fig. 2. Stages in aligning abcxyz and abyxz

In the left-hand picture, we have reached the point (5, 4) in the array. The cheapest sequence of edits to get to here is `match(a,a)`, `match(b,b)`, `delete(c)`, `delete(x)`, `match(y,y)`, `insert(r)`, at a total cost of 6. We can get from here to (6, 4), by deleting z, but that would cost 6+2, which is greater than the cost of the current cheapest route to this point, so we do not do it; to (6, 5), by matching z and z, at a cost of 6+2. Since we have not visited (6, 5) at all at this point. this is the best value so far for getting to here, so we will update the array; and to (5, 5), by inserting z, at a cost of 6+2. Since this is cheaper than the existing value at (5, 5), we also update the array for this point.

This is all, of course, completely standard. But we now add a new edit operation to `extend`. This operation comes in two parts:

1. Find the longest possible sequence  $S2_j, \dots, S2_{j+k}$  such that

$$\sum_{i=1}^k \text{costExchange}(S1_i, S2_{j+i}) \leq 1$$

2. Find the **standard** string edit distance between  $S1_j, \dots, S2_{j+k}$  and  $S2_i, \dots, S2_j$

Step 1 looks for sequences in S2 that have been right-shifted, step 2 matches the material in S2 over which they were shifted with the material in S1 that follows the matched segment.

When this step applies an entry will be made in the array that is some way away from the cell under inspection. Consider, for instance, the sequences `abcdxypq` and `abxyzcdpq`, where `cd` has been shifted over `xy`, which has itself morphed into `xyz`. At the point when we have matched the occurrences of `a` and `b` and are considering the places it is possible to get to, we will spot this match and hence the array will look as in Fig. 3.

The dotted line indicates that the cost of 2 at (6, 7) arises from doing a swap that started at (2, 2): this cost is obtained by the cost of getting to (2, 2) in the first place + the cost of aligning the sequences `x`, `y` and `x`, `y`, `z`. The final array for this pair is given in Fig. 4.

The final cost of aligning the two sequences is 2, using a sequence that includes the exchange of `cd|xy` and `xyz|cd`. Note that when we get to (5, 6) and

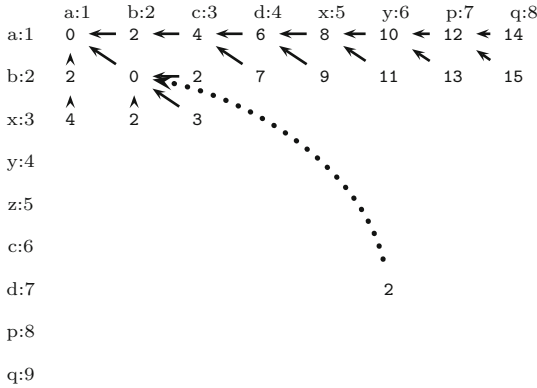


Fig. 3. Spotting the swap between  $cd|xyz$  and  $xyz|cd$

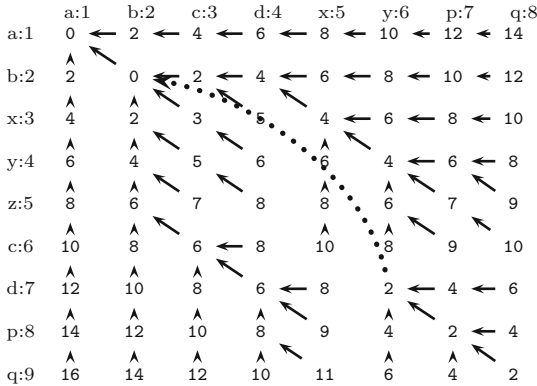


Fig. 4. Final alignment of  $abcdxypq$  and  $abxyzcdpq$

try the diagonal match of  $y$  and  $d$  we get a score of  $8+3$  which is greater than the score of  $2$  that we already have, and hence we do not choose this option, and similarly for inserting  $d$  at  $(5, 7)$  or deleting  $y$  at  $(6, 6)$ .

## 2 Textual Similarity

We have applied this extended version of DTW, which we will refer to as XDTW, to texts collected from different news RSS feeds. We collected material from a number of RSS feeds<sup>1</sup>, matched articles in the various sources by cosine/TF-IDF similarity, and then matched sentences within the matched articles by the same

<sup>1</sup> BBC English [//www.bbc.co.uk/news](http://www.bbc.co.uk/news), The Guardian [www.theguardian.com/uk](http://www.theguardian.com/uk), Independent <http://www.independent.co.uk/news/uk/rss>, Reuters [uk.reuters.com/news/uk](http://uk.reuters.com/news/uk), and Express [feeds.feedburner.com/daily-express-uk](http://feeds.feedburner.com/daily-express-uk).

technique. This gave us numerous potentially matching sentence pairs, which we then applied the standard and extended DTW algorithms to.

There were a number of striking examples where XDTW produced good matches which were missed by the standard version. #8–#10 show a number of such cases:

- (8) Rebecca Hilsenrath the EHRC chief executive added we know ...  
 the EHRC chief executive Rebecca Hilsenrath added we know ...
- (9) she stopped messaging me last night about 3:00pm Parkin said  
 Parkin said she stopped messaging me last night about 3:00pm
- (10) ... identified by Newham CCG and the community as key health issue  
 ... identified as key health issue by Newham CCG and the community
- 

The swapped material generally consists of modifiers (#8 and #10), but there are examples, such as #9, where the items that have been swapped are arguments (*‘she stopped messaging me about 3:00pm’* is the complement of *‘said’*).

### 3 Conclusions

The algorithm described above extends the standard dynamic time warping algorithm so that it can detect chunks of text that have been moved around wholesale. As such, it makes it possible to assess textual similarity between sentence pairs that would not have scored well under the standard DTW algorithm, even with the extra recurrence relation added in Eq. 4. Two questions remain: how common are examples like #8 to #10], and what is the complexity of the new version of the algorithm? If examples like these are very rare, or if the extension has a catastrophic effect on the complexity, then it may be that while it does catch some examples the cost/benefit ratio is too poor to make it worthwhile.

**How common are examples like #8 to #10?** We tested the algorithm on a subset of the corpus described above. This corpus contained just under 3000 sentence pairs, but these included a substantial number of pairs which were clearly not related. We therefore found the cosine TF-IDF score at which around 50% appeared to be related, which gave us 299 pairs, and had these annotated as being mutually entailing or not by a team of annotators (each sentence was annotated by five annotators to try to ensure that the annotation was robust).

Applying XDTW to these produced twelve examples which scored 0 (i.e. were regarded as identical), in addition to the four which scored 0 under the standard DTW by virtue of containing a pair of synonyms, as in #11 and #12:

(11) around 50 civilians have been trapped in Fallujah which has ...  
 | .....  
 about 50 civilians have been trapped in Fallujah which has ...

(12) concerns have been raised about the participation of UK science ...  
 / .....  
 fears have been raised about the participation of UK science ...

There are, however, also a few examples of sentences in this data where a block of text has been moved **and** altered, either by substitution of a near synonym or by insertion or deletion of a single word:

(6) the raised money will go to the Mount Vernon cancer centre  
 .....  
 all the money raised will go to the Mount Vernon cancer centre

(13) Labour MP for Barnsley Central Dan Jarvis said know ...  
 .....  
 Dan Jarvis *the* Labour MP for Barnsley Central said know ...

It seems, then, that at least for news articles, which are commonly obtained from a shared source and then slightly rewritten<sup>2</sup>, looking for cases where chunks of text have been shifted does produce a useful improvement in recall without damaging precision (every case that was found by the extended algorithm was unambiguously marked as an instance of textual equivalence by the annotators).

**What is the complexity of XDTW?** The new algorithm adds a new step to the three standard operations that are applied at every cell  $(i, j)$  in the array, i.e.  $len(S1) \times len(S2)$  times. To analyse the complexity of this step it is convenient to use  $S1_i$  and  $S2_j$  to denote the subsequences of  $S1$  and  $S2$  starting at  $i$  and  $j$  respectively, and  $S1_{[i:j]}$  to denote the slice of  $S$  from  $i$  to  $j$ . This step has two stages: (i) look for the maximal sequence in  $S2_{j+1}$  which matches a prefix of  $S1_i$ . To do this we initialise a table showing the locations of every word type in  $S2$ , e.g. if  $S2$  were *the cat sat on the mat* then this table would be  $\{ 'on': [3], 'the': [4, 0], 'sat': [2], 'mat': [5], 'cat': [1] \}$ . Then when we are searching for occurrences of prefixes of  $S1_i$  in  $S2_j$ , we know exactly where they might start. Since the average number of places where a given word from  $S1$  appears in  $S2$  is barely more than 1, the task of finding the longest sequence  $S2_{j+o:j+o+k}$  which matches a prefix  $S1_{[i:i+k]}$  of  $S1_i$  is linear in the length of the longest such prefix. (ii) when we find such a prefix, we have to align  $S2_{[j:j+o]}$  (the part of  $S2$  that we skipped over when finding the matched segment) with a portion  $S1_{[i+k:??]}$  of  $S1$  starting at  $i+k$ . In almost all cases, the difference in the

<sup>2</sup> Just like student essays!.

lengths of  $S2_{[j:j+o]}$  and  $S1_{[i+k:??]}$  is no more than 1. We therefore only need to look at, at most,  $S1_{[i+k:i+k+o-1]}$ ,  $S1_{[i+k:i+k+o]}$  and  $S1_{[i+k:i+k+o+1]}$ . Furthermore, we can safely assume that there is at most one insertion or deletion involved in aligning these sequence, since the swapped element never turns out to contain multiple insertions and deletions, which means that we only need to look at very restricted corridor when doing the alignment. This step therefore also turns out to be linear in the length of the length of the swapped item. If we insist that the swapped sections have the same length we can use a corridor of size 0 (i.e. we just look at the diagonal elements). This improves the speed substantially while having very little effect on the recall – XDTW 1 in Table 1<sup>3</sup> requires the swapped segments to be the same length, XDTW 2 allows their lengths to differ by 2. ‘Close matches’ in this table are cases where a single word has been inserted or deleted or where two similar<sup>4</sup> words have been matched.

**Table 1.** Timing on news stories (299 sentences, 4960 words)

	Time (seconds)	Identical matches	Close matches
Basic algorithm	0.88	4	26
XDTW 1	1.84	16	27
XDTW 2	4.12	16	28

Both versions of XDTW discovered a useful set of sentences where a word or phrase has been shifted without change (16/299, i.e. 5% of the entire collection). They also discovered a small number of cases where a phrase was shifted and there was one other change (these are in fact #8 and #13 above). XDTW thus carries out the task that tree-edit distance algorithms [1,7] aim to solve, at a much lower theoretical cost and indeed at very little extra practical cost over standard DTW, without requiring the text to be parsed. The key here is that groups of words that get shifted around intact do generally constitute phrases: after all, moving a group that is **not** a phrase will generally lead to gibberish. XDTW exploits this observation to allow us to spot paraphrases which would be missed by the standard DTW algorithm, at much lower cost than is incurred when using tree-edit distance.

<sup>3</sup> Timings are average of 10 runs, MacBook Pro 2.8 Ghz processor.

<sup>4</sup> according to WUP, which all three algorithms use for calculating the cost of exchanging one word for another.

## References

1. Alabbas, M., Ramsay, A.M.: Natural language inference for Arabic using extended tree edit distance with subtrees. *J. Artif. Intell. Res.* **48**, 1–22 (2013). <https://www.jair.org/media/3892/live-3892-7342-jair.pdf>
2. Damerau, F.J.: A technique for computer detection and correction of spelling errors. *Commun. ACM* **7**(3), 171–176 (1964)
3. Mikolov, T., Chen, K., Carrado, G., Dean, J.: Efficient Estimation of Word Representations in Vector Space, 1st edn. (2013). <http://arxiv.org/pdf/1301.3781.pdf>
4. Pennington, J., Socher, R., Manning, C.D.: Glove: Global Vectors for Word Representation (2014)
5. Sakoe, H., Chiba, S.: Dynamic programming optimization for spoken word recognition. *IEEE Trans. Acoust. Speech Sign. Proces.* **26**, 43–49 (1978)
6. Wu, Z., Palmer, M.: Verb semantics and lexical selection. In: 32nd Annual Meeting of the Association for Computational Linguistics (1994). <http://www.aclweb.org/anthology/P94-1019>
7. Zhang, K., Shasha, D.: Simple fast algorithms for the editing distance between trees and related problems. *SIAM J. Comput.* **18**(6), 1245–1262 (1989)
8. Zhao, S., Lan, X., Liu, T., Li, S.: Application-driven statistical paraphrase generation. In: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2, pp. 834–842. Association for Computational Linguistics (2009)



# A Study on Dialog Act Recognition Using Character-Level Tokenization

Eugénio Ribeiro<sup>1,2(✉)</sup>, Ricardo Ribeiro<sup>1,3</sup>, and David Martins de Matos<sup>1,2</sup>

<sup>1</sup> L<sup>2</sup>F – Spoken Language Systems Laboratory, INESC-ID, Lisboa, Portugal  
eugenio.ribeiro@l2f.inesc-id.pt

<sup>2</sup> Instituto Superior Técnico, Universidade de Lisboa, Lisboa, Portugal

<sup>3</sup> Instituto Universitário de Lisboa (ISCTE-IUL), Lisboa, Portugal

**Abstract.** Dialog act recognition is an important step for dialog systems since it reveals the intention behind the uttered words. Most approaches on the task use word-level tokenization. In contrast, this paper explores the use of character-level tokenization. This is relevant since there is information at the sub-word level that is related to the function of the words and, thus, their intention. We also explore the use of different context windows around each token, which are able to capture important elements, such as affixes. Furthermore, we assess the importance of punctuation and capitalization. We performed experiments on both the Switchboard Dialog Act Corpus and the DIHANA Corpus. In both cases, the experiments not only show that character-level tokenization leads to better performance than the typical word-level approaches, but also that both approaches are able to capture complementary information. Thus, the best results are achieved by combining tokenization at both levels.

**Keywords:** Dialog act recognition · Character-level  
Switchboard dialog act corpus · DIHANA corpus · Multilinguality

## 1 Introduction

Dialog act recognition is important in the context of a dialog system, since it reveals the intention behind the words uttered by its conversational partners [24]. Knowing that intention allows the system to apply specialized interpretation strategies, accordingly. Recently, most approaches on dialog act recognition focus on applying different Deep Neural Network (DNN) architectures to generate segment representations from word embeddings and combine them with context information from the surrounding segments [9, 11, 14, 15]. However, all of these approaches look at the segment at the word level. That is, they consider that a segment is a sequence of words and that its intention is revealed by the combination of those words. However, there are also cues for intention at the sub-word

---

This work was supported by national funds through Fundação para a Ciência e a Tecnologia with reference UID/CEC/50021/2013 and by Universidade de Lisboa.

level. These cues are mostly related to the morphology of words. For instance, there are cases, such as adverbs of manner and negatives, in which the function, and hence the intention, of a word is related to its affixes. On the other hand, there are cases in which considering multiple forms of the same lexeme independently does not provide additional information concerning intention and the lemma suffices. Thus, it is interesting to explore dialog act recognition approaches that are able to capture this kind of information. In this paper, we explore the use of character-level tokenization with different context windows surrounding each token. Although character-level approaches are typically used for word-level classification tasks, such as Part-of-Speech (POS) tagging [23], they have also achieved interesting results on short-text classification tasks, such as language identification [8] and review rating [27]. In addition to the aspects concerning morphological information, using character-level tokenization allows us to assess the importance of aspects such as capitalization and punctuation. Additionally, we assess whether the obtained information can be combined with that obtained using word-level tokenization to improve the performance on the task. In this sense, in order to widen the scope of our conclusions, we performed experiments on two corpora, the Switchboard Dialog Act Corpus [10] and DIHANA [3], which have different characteristics, including domain, the nature of the participants, and language – English and Spanish, respectively.

In the remainder of this paper we start by providing an overview of previous approaches on dialog act recognition, in Sect. 2. Then, in Sect. 3, we discuss why using character-level tokenization is relevant for the task. Section 4 describes our experimental setup, including the used datasets, classification approach, and word-level baselines. The results of our experiments are presented and discussed in Sect. 5. Finally, Sect. 6 states the most important conclusions of this study and provides pointers for future work.

## 2 Related Work

Automatic dialog act recognition is a task that has been widely explored over the years, using multiple machine learning approaches, from Hidden Markov Models (HMMs) [25] to Support Vector Machines (SVMs) [6]. The article by Král and Cerisara [13] provides an interesting overview of most of those approaches on the task. However, recently, similarly to many other Natural Language Processing (NLP) tasks [7, 16], most approaches on dialog act recognition take advantage of different DNN architectures.

To our knowledge, the first of those approaches was that by Kalchbrenner and Blunsom [11]. They used a Convolutional Neural Network (CNN)-based approach to generate segment representations from randomly initialized 25-dimensional word embeddings and a Recurrent Neural Network (RNN)-based discourse model to combine the sequence of segment representations with speaker information and output the corresponding sequence of dialog acts.

Lee and Deroncourt [14] compared the performance of a Long Short-Term Memory (LSTM) unit against that of a CNN to generate segment representations



from 200-dimensional Global Vectors for Word Representation (GloVe) embeddings [21] pre-trained on Twitter data. Those segment representations were then fed to a 2-layer feed-forward network that combined them with context information from the preceding segments. The best results were obtained using the CNN-based approach combined with information from two preceding segments in the form of their representation.

Ji et al. [9] used a Discourse Relation Language Model (DRLM) with a hybrid architecture that combined a Recurrent Neural Network Language Model (RNNLM) [19] with a latent variable model over shallow discourse structure. This way, the model can learn vector representations trained discriminatively, while maintaining a probabilistic representation of the targeted linguistic element which, in this context, is the dialog act. In order to function as a classifier, the model was trained to maximize the conditional probability of a sequence of dialog acts given a sequence of segments.

The previous studies explored the use of a single recurrent or convolutional layer. However, the top performing approaches use multiple of those layers. On the one hand, Khanpour et al. [12] achieved their best results by combining the outputs of a stack of 10 LSTM units, in order to capture long distance relations between tokens. On the other hand, Liu et al. [15] combined the outputs of three parallel CNNs with different context window sizes, in order to capture different functional patterns. Both studies used Word2Vec [20] embeddings as input to the network. However, their dimensionality and training data varied.

Additionally, Liu et al. [15] explored the use of context information concerning speaker changes and from the surrounding segments. Concerning the latter, they used approaches that relied on discourse models, as well as others that combined the context information directly with the segment representation. Similarly to our previous study using SVMs [22], they concluded that providing that information in the form of the classification of the surrounding segments leads to better results than using their words. Furthermore, both studies have shown that the first preceding segment is the most important and that the influence decays with the distance.

### 3 Character-Level Tokenization

It is interesting to explore character-level tokenization because it allows us to capture morphological information that is at the sub-word level and, thus, cannot be directly captured using word-level tokenization. Considering the task at hand, that information is relevant since it may provide cues for identifying the intention behind the words. When someone selects a set of words to form a segment that transmits a certain intention, each of those words is typically selected because it has a function that contributes to that transmission. In this sense, affixes are tightly related to word function, especially in fusional languages. Thus, the presence of certain affixes is a cue for intention, independently of the lemma. However, there are also cases, such as when affixes are used for subject-verb agreement, in which the cue for intention is in the lemmas and, thus, considering multiple forms of the same lexeme does not provide additional information.

Information concerning lemmas and affixes cannot be captured from single independent characters. Thus, it is necessary to consider the context surrounding each token and look at groups of characters. The size of the context window plays an important part in what information can be captured. For instance, English affixes are typically short, but in other languages, such as Spanish, there are longer commonly used affixes. Furthermore, to capture the lemmas of long words, and even inter-word relations, wider context window sizes must be considered. However, using wide context windows impairs the ability to capture information from short groups of characters, as additional irrelevant characters are considered. This suggests that, in order to capture all the relevant information, multiple context windows should be used.

Using character-level tokenization also allows us to consider punctuation, which is able to provide both direct and indirect cues for dialog act recognition. For instance, an interrogation mark provides a direct cue that the intention is related to knowledge seeking. On the other hand, commas structure the segment, indirectly contributing to the transmission of an intention.

Additionally, character-level tokenization allows us to consider capitalization information. However, in the beginning of a segment, capitalization only signals that beginning and, thus, considering it only introduces entropy. In the middle of a segment, capitalization is typically only used to distinguish proper nouns, which are not related to intention. Thus, capitalization information is not expected to contribute to the task.

Finally, note that previous studies have shown that word-level information is relevant for the task. In this sense, it is interesting to assess whether that information can be captured using character-level tokenization or if there are specific aspects that require specialized approaches.

## 4 Experimental Setup

In order to assess the validity of the hypotheses proposed in the previous section, we performed experiments on different corpora and compared the performance of word- and character-level tokenization. The used datasets, classification approach, and word-level baselines are described below.

### 4.1 Datasets

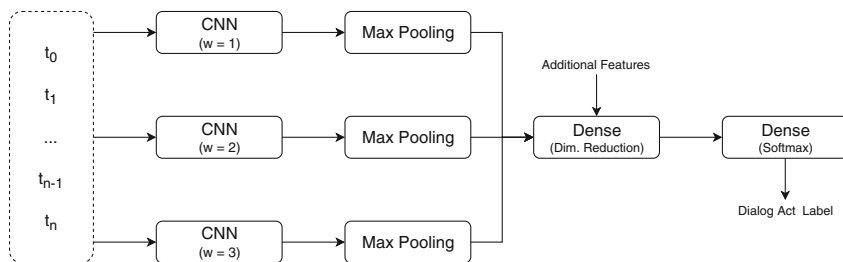
In order to widen the scope of the conclusions drawn in the study, we selected two corpora with different characteristics to perform our experiments on. On the one hand, the Switchboard Dialog Act Corpus [10], henceforth referred to as Switchboard, is the most explored corpus for dialog act recognition. It features 1,155 manually transcribed human-human dialogs in English, with variable domain, containing 223,606 segments. The set is partitioned into a training set of 1,115 conversations, a test set of 19 conversations, and a future use set of 21 conversations [25]. In our experiments, we used the latter as a validation set.

In terms of dialog act annotations, we used the most used version of its tag set, which features 42 domain-independent labels.

On the other hand, the DIHANA corpus [3] consists of 900 dialogs in Spanish between human speakers and a Wizard of Oz (WoZ) telephonic train information system. The total number of annotated segments is 23,542, with 9,712 corresponding to user segments and 13,830 to system segments [2]. The set is partitioned into five folds to be used for cross-validation [17]. The dialog act annotations are hierarchically decomposed in three levels [18]. The first level represents the domain-independent intention of the segment, while the remaining are task-specific. In our experiments we focused on the first level, which has 11 different tags, out of which 5 are common to user and system segments.

## 4.2 Classification Approach

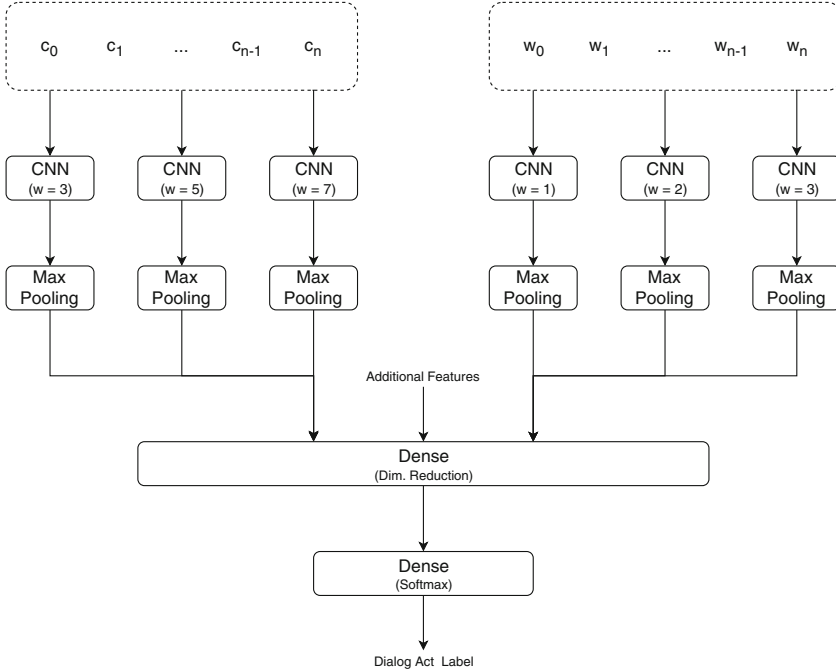
As a classification approach, we adapted the state-of-the-art word-level approach by Liu et al. [15] to use characters instead of words as tokens. As shown in Fig. 1, the token embeddings are passed through a set of parallel temporal CNNs with different context window sizes followed by a max pooling operation. The results of those operations are then concatenated to form a representation of the segment. To achieve the state-of-the-art results, additional features concerning context information are appended to that representation before it is passed through a dimensionality reduction layer. Since our study focuses on the difference between using character- and word-level tokenization, we only included that information in a final experiment for comparison with the state-of-the-art. Finally, the reduced segment representation is passed through a dense layer with the softmax activation to obtain its classification.



**Fig. 1.** The generic architecture of the network used in our experiments.  $t_i$  corresponds to the embedding representation of the  $i$ -th token.  $w$  corresponds to the context window size of the CNN. The number of parallel CNNs and the respective window sizes vary between experiments. Those shown in the figure correspond to the ones used by Liu et al. [15] in their experiments.

In order to assess whether the character- and word-level approaches capture complementary information, we also performed experiments that combined both approaches. In that scenario, we used the architecture shown in Fig. 2. In this

case, two segment representations are generated in parallel, one based on the characters in the segment and other on its words. Those representations are then concatenated to form the final representation of the segment. The following steps do not differ from the architecture with a single branch. That is, context information can be added to the segment representation before it is passed to the two dense layers.



**Fig. 2.** The architecture of the network that combines the character- and -word-level approaches.  $c_i$  corresponds to the embedding representation of the  $i$ -th character while  $w_i$  corresponds to the embedding representation of the  $i$ -th word. The context window sizes of the CNNs in the character-level branch refer to those that achieved best performance in our experiments. Those on the word-level branch correspond to the ones used by Liu et al. [15] in their experiments.

We used Keras [5] with the TensorFlow [1] backend to implement the networks. The training phase stopped after 10 epochs without improvement on the validation set. The results presented in the next section refer to the average ( $\mu$ ) and standard deviation ( $\sigma$ ) accuracy values over 10 runs.

### 4.3 Baselines

In order to assess the performance of the character-level approach, in comparison to the word-level approach, we defined two baselines. One of them uses randomly

initialized word embeddings that are adapted during the training phase, while the other uses fixed pre-trained embeddings. The latter were obtained by applying Word2Vec [20] on the English Wikipedia<sup>1</sup> and the Spanish Billion Word Corpus [4]. Additionally, we defined a third baseline that replicates the state-of-the-art approach by Liu et al. [15]. It consists of the baseline with pre-trained embeddings combined with context information from three preceding segments in the form of their gold standard annotations and speaker change information in the form of a flag. Similarly to Liu et al. [15], we used three parallel CNNs with context window sizes one, two, and three in all the baselines.

## 5 Results

Starting with the word-level baselines, in Table 1 we can see that, in comparison to using randomly initialized word embeddings, using fixed pre-trained embeddings led to an average accuracy improvement of .64 and .88 percentage points on the validation (SWBD-V) and test (SWBD-T) sets of the Switchboard corpus, respectively. However, that was not the case on the DIHANA corpus, where the improvement was negligible. This can be explained by the difference in the nature of the dialogs between corpora. Since the Switchboard dialogs have a large variability in terms of style and domain, the performance on the validation and test sets is impaired when overfitting to the training data occurs. On the other hand, since most DIHANA dialogs are similar, the cross-validation performance is not impaired and may actually benefit from it. The improvement provided by context information is in line with that reported by Liu et al. [15]. Our results on the Switchboard corpus vary from those reported in their paper mainly because they did not use the standard validation and test partitions.

**Table 1.** Accuracy results of the word-level baselines.

	SWBD-V		SWBD-T		DIHANA	
	$\mu$	$\sigma$	$\mu$	$\sigma$	$\mu$	$\sigma$
Random	.7617	.0019	.7223	.0020	.9196	.0013
Pre-trained	.7681	.0032	.7311	.0026	.9198	.0012
Pre-trained + Context	.8129	.0030	.7835	.0036	.9826	.0004

Regarding the character-level experiments, in Table 2 we can see that, as expected, considering each character individually is not the appropriate approach to capture intention. By considering pairs of characters, the performance improved by over 5 percentage points on both corpora. Widening the window up to five characters leads to a nearly 3 percentage point improvement on the Switchboard corpus, but less than 1 percentage point on the DIHANA corpus. However, it is important to note that while the results are above 90% accuracy

<sup>1</sup> <https://dumps.wikimedia.org/enwiki/>.

on the DIHANA corpus, they are below 80% on the Switchboard corpus. Thus, improvements are expected to be less noticeable on the first. Using a window of seven characters still improves the results on the Switchboard corpus, but is not relevant on the DIHANA corpus. Considering wider windows is harmful on both corpora. However, note that, similarly to what Liu et al. [15] have shown at the word level, different context windows are able to capture complementary information. Thus, it is beneficial to combine multiple windows. In our experiments, the best results on both corpora were achieved using three context windows, which considered groups of three, five, and seven characters, respectively. The sizes of these windows are relevant, since the shortest window is able to capture most affixes in English and the small affixes in Spanish, the middle window is able to capture the larger Spanish affixes and most lemmas in both languages, and the widest window is able to capture larger words and inter-word information. Finally, it is relevant to note that the results on the DIHANA corpus are already above the word-level baselines.

**Table 2.** Accuracy results using different token context windows.

Window Size(s)	SWBD-V		SWBD-T		DIHANA	
	$\mu$	$\sigma$	$\mu$	$\sigma$	$\mu$	$\sigma$
1	.6542	.0017	.6081	.0023	.8571	.0029
2	.7221	.0047	.6752	.0054	.9154	.0014
3	.7432	.0055	.7000	.0035	.9217	.0010
4	.7456	.0019	.7064	.0049	.9222	.0014
5	.7509	.0052	.7091	.0038	.9228	.0011
7	.7535	.0023	.7086	.0034	.9224	.0013
10	.7510	.0036	.7097	.0035	.9216	.0013
(3, 5, 7)	.7608	.0033	.7208	.0042	.9244	.0012

In Sect. 3, we hypothesized that capitalization is not relevant for dialog act recognition. In Table 3, we can see that the hypothesis holds for the Switchboard corpus, as the results obtained when using capitalized segments do not significantly differ from those obtained using uncapitalized segments. However, on the DIHANA corpus, using capitalized segments led to an average improvement of 1.81 percentage points. Since this was not expected, we looked for the source of the improvement. By inspecting the transcriptions, we noticed that, contrarily to user segments, the system segments do not contain mid-segment capitalization. Thus, proper nouns, such as city names which are common in the dialogs, are capitalized differently. Since only 5 of the 11 dialog acts are common to user and system segments, identifying its source reduces the set of possible dialog acts for the segment. Thus, the improvement observed when using capitalization information is justified by the cues it provides to identify whether it is a user or system segment.

**Table 3.** Accuracy results using different segment preprocessing approaches.

	SWBD-V		SWBD-T		DIHANA	
	$\mu$	$\sigma$	$\mu$	$\sigma$	$\mu$	$\sigma$
Capitalized	.7604	.0028	.7194	.0026	.9425	.0015
Punctuated	.7685	.0021	.7317	.0032	.9371	.0007
Capitalized + Punctuated	.7673	.0025	.7314	.0040	.9548	.0004
Lemmatized	.7521	.0027	.7140	.0012	.9239	.0006

In Table 3, we can also see that, as expected, punctuation provides relevant information for the task, improving the performance around 1 percentage point on both corpora. Using this information, the character-level approach surpasses the randomly initialized word-level baseline and is in line with the one using pre-trained word embeddings on the Switchboard corpus. Also expectedly, the decrease in performance observed when using lemmatized segments proves that affixes are relevant. However, that decrease is not drastic, which suggests that most information concerning intention can be transmitted using a simplified language that does not consider variations of the same lexeme and that those variations are only relevant for transmitting some specific intentions.

**Table 4.** Accuracy results using the combination of word and character-level representations.

	SWBD-V		SWBD-T		DIHANA	
	$\mu$	$\sigma$	$\mu$	$\sigma$	$\mu$	$\sigma$
Char + Word	.7800	.0016	.7401	.0035	.9568	.0003
Char + Word + Context	.8200	.0027	.7901	.0016	.9910	.0004

Finally, Table 4 shows the results obtained by combining the word and character-level approaches. We can see that the performance increases on both corpora, which means that both approaches are able to capture complementary information. This confirms that information at the sub-word level is relevant for the task. When using context information, the combination of both approaches leads to results that surpass the state-of-the-art word-level approach on the Switchboard corpus by around .7 percentage points and to a nearly perfect score on the DIHANA corpus. Concerning the latter, it is not fair to compare our results with those of previous studies, since the only one that focused on Level 1 labels did not rely on textual information [26].

## 6 Conclusions

We have shown that there is important information for dialog act recognition at the sub-word level which cannot be captured by word-level approaches. We used

character-level tokenization together with multiple context windows with different sizes in order to capture relevant morphological elements, such as affixes and lemmas, as well as long words and inter-word information. Furthermore, we have shown that, as expected, punctuation is important for the task since it is able to provide both direct and indirect cues regarding intention. On the other hand, capitalization is irrelevant under normal conditions. Finally, our experiments revealed that the character- and word-level approaches capture complementary information and, consequently, their combination leads to improved performance on the task. In this sense, by combining both approaches with context information we achieved state-of-the-art results on the Switchboard corpus and a nearly perfect score on the DIHANA corpus.

It is important to note that while one of the corpora used in our experiments features variable-domain human-human interactions in English, the other features fixed-domain interactions in Spanish between a WoZ dialog system and its users. Thus, the importance of information at the sub-word level is not domain-dependent and it is not limited to a single language.

In terms of morphological typology, although English has a more analytic structure than Spanish, both are fusional languages. Thus, as future work it would be interesting to assess whether the conclusions of this study hold for analytic languages, such as Chinese, and agglutinative languages, such as Turkish.

## References

1. Abadi, M., et al.: TensorFlow: large-scale machine learning on heterogeneous systems (2015). <https://www.tensorflow.org/>
2. Alcácer, N., Benedí, J.M., Blat, F., Granell, R., Martínez, C.D., Torres, F.: Acquisition and labelling of a spontaneous speech dialogue corpus. In: SPECOM, pp. 583–586 (2005)
3. Benedí, J.M., Lleida, E., Varona, A., Castro, M.J., Galiano, I., Justo, R., de Letona, I.L., Miguel, A.: Design and acquisition of a telephone spontaneous speech dialogue corpus in Spanish: DIHANA. In: LREC, pp. 1636–1639 (2006)
4. Cardellino, C.: Spanish billion words corpus and embeddings (2016). <http://crscardellino.me/SBWCE/>
5. Chollet, F., et al.: Keras: the python deep learning library (2015). <https://keras.io/>
6. Gambäck, B., Olsson, F., Täckström, O.: Active learning for dialogue act classification. In: INTERSPEECH, pp. 1329–1332 (2011)
7. Goldberg, Y.: A primer on neural network models for natural language processing. *J. Artif. Intell. Res.* **57**, 345–420 (2016)
8. Jaech, A., Mulcaire, G., Hathi, S., Ostendorf, M., Smith, N.A.: Hierarchical character-word models for language identification. In: International Workshop on Natural Language Processing for Social Media, pp. 84–93 (2016)
9. Ji, Y., Haffari, G., Eisenstein, J.: A latent variable recurrent neural network for discourse relation language models. In: NAACL-HLT, pp. 332–342 (2016)
10. Jurafsky, D., Shriberg, E., Biasca, D.: Switchboard SWBD-DAMSL Shallow-Discourse-Function Annotation Coders Manual. Tech. Rep. Draft 13, University of Colorado, Institute of Cognitive Science (1997)



11. Kalchbrenner, N., Blunsom, P.: Recurrent convolutional neural networks for discourse compositionality. In: Workshop on Continuous Vector Space Models and their Compositionality, pp. 119–126 (2013)
12. Khanpour, H., Guntakandla, N., Nielsen, R.: Dialogue act classification in domain-independent conversations using a deep recurrent neural network. In: COLING, pp. 2012–2021 (2016)
13. Král, P., Cerisara, C.: Dialogue act recognition approaches. *Comput. Inform.* **29**(2), 227–250 (2010)
14. Lee, J.Y., Deroncourt, F.: Sequential short-text classification with recurrent and convolutional neural networks. In: NAACL-HLT, pp. 515–520 (2016)
15. Liu, Y., Han, K., Tan, Z., Lei, Y.: Using context information for dialog act classification in DNN framework. In: EMNLP, pp. 2160–2168 (2017)
16. Manning, C.D.: Computational linguistics and deep learning. *Comput. Linguist.* **41**(4), 701–707 (2015)
17. Martínez-Hinarejos, C.D., Benedí, J.M., Granell, R.: Statistical framework for a Spanish spoken dialogue corpus. *Speech Commun.* **50**(11–12), 992–1008 (2008)
18. Martínez-Hinarejos, C.D., Sanchis, E., García-Granada, F., Aibar, P.: A labelling proposal to annotate dialogues. *LREC* **5**, 1566–1582 (2002)
19. Mikolov, T., Karafit, M., Burget, L., Cernock, J., Khudanpur, S.: Recurrent neural network based language model. In: INTERSPEECH, pp. 1045–1048 (2010)
20. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: NIPS, pp. 3111–3119 (2013)
21. Pennington, J., Socher, R., Manning, C.D.: GloVe: global vectors for word representation. In: EMNLP, pp. 1532–1543 (2014)
22. Ribeiro, E., Ribeiro, R., de Matos, D.M.: The influence of context on dialogue act recognition. CoRR abs/1506.00839 (2015). <http://arxiv.org/abs/1506.00839>
23. Santos, C.D., Zadrozny, B.: Learning character-level representations for part-of-speech tagging. In: ICML, pp. 1818–1826 (2014)
24. Searle, J.R.: *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press, Cambridge, London (1969)
25. Stolcke, A., Coccaro, N., Bates, R., Taylor, P., Van Ess-Dykema, C., Ries, K., Shriberg, E., Jurafsky, D., Martin, R., Meteer, M.: Dialogue act modeling for automatic tagging and recognition of conversational speech. *Comput. Linguist.* **26**(3), 339–373 (2000)
26. Tamarit, V., Martínez-Hinarejos, C.D.: Dialog act labeling in the DIHANA corpus using prosody information. In: V Jornadas en Tecnología del Habla, pp. 183–186 (2008)
27. Zhang, X., Zhao, J., LeCun, Y.: Character-level convolutional networks for text classification. *NIPS* **1**, 649–657 (2015)



# Neural Methods for Cross-Lingual Sentence Compression

Frederico Rodrigues<sup>1,2(✉)</sup>, Bruno Martins<sup>1,2</sup>, and Ricardo Ribeiro<sup>1,3</sup>

<sup>1</sup> INESC-ID, Lisbon, Portugal

<sup>2</sup> Instituto Superior Técnico (IST), University of Lisbon, Lisbon, Portugal  
`frederico.i.rodrigues@tecnico.ulisboa.pt`

<sup>3</sup> Instituto Universitário de Lisboa (ISCTE-IUL), Lisbon, Portugal

**Abstract.** Sentence compression produces a shorter sentence by removing redundant information, preserving the grammaticality and the important content. We propose an improvement to current neural deletion systems. These systems output a binary sequence of labels for an input sentence: one indicates that the token from the source sentence remains in the compression, whereas zero indicates that the token should be removed. Our main improvement is the use of a Conditional Random Field as final layer, which benefits the decoding of the best global sequence of labels for a given input. In addition, we also evaluate the incorporation of syntactic features, which can improve grammaticality. Finally, this task is extended into a cross-lingual setting where the models are evaluated on English and Portuguese. The proposed architecture achieves better than or equal results to the current state-of-the-art systems, validating that the model benefits from the modification in both languages.

**Keywords:** Sentence compression · Deep neural networks  
Cross-language Learning

## 1 Introduction

Sentence compression is a Natural Language Processing (NLP) task that returns a shorter version of a sentence, maintaining its readability and the most important information. Sentence compression systems can be applied, for example, in news digests, subtitle generation, and automatic summarization systems.

A compression system is often formulated as deletion-based, which indicates that the output is a sequence of labels, namely zeros and ones, representing if a token should be deleted or not from the source sentence. This approach is often called extractive sentence compression and it will be the focus of this work. These type of systems are mostly used in a monolingual setting because

---

This work was supported by national funds through Fundação para a Ciência e Tecnologia (FCT) with reference UID/CEC/50021/2013.

each language has its grammatical rules and it may be difficult to generate readable sentences across languages. Developing a method that performs well across languages is a challenge. This challenge can be addressed by learning cross-lingual representations that project different languages into a shared space.

Tree-based methods are often used to sentence compression: a sentence is parsed by a dependency parser and the process of compression is done by removing dependency edges from the parse tree. Although this approach often works, if a sentence is ambiguous and have multiple possible parses, the compression can lead to grammatical errors. Thus, instead of using the parse tree as output, the former systems prefer to use it a feature. Another popular approach is to formulate sentence compression as an Integer Linear Programming (ILP) problem using dependency trees features as constraints in addition to other linguistic features to ensure the grammaticality of the output [4]. Recent work focus on deep neural networks, which, even without any linguist features, generate compressions grammatically correct [7]. The incorporation of syntactic features has been shown to improve these models [23].

In this paper, the proposed architecture uses a combination of a bi-directional Long Short-Term Memory (BiLSTM) with a Conditional Random Field (CRF) as final layer, which finds globally the best label sequence, instead of a softmax layer used by the current systems [7, 23]. In addition, we evaluate the proposed model in a cross-lingual setting using English and Portuguese. We do not rely on a parallel corpus to learn the cross-lingual word embeddings. Instead, we follow the work of Conneau et al. [5] that build a synthetic bilingual dictionary and learns a mapping from a source to a target space to learn the cross-lingual embeddings. Our models are evaluated on the same sentence pairs used by the previous work [7, 23], achieving similar results or better. In addition, we want to research the following question: is it relevant to incorporate syntactic features across models? While models with syntactic features had achieved competitive results in a monolingual setting, we demonstrate, when evaluating on Portuguese data, that the model does not benefit from the incorporation of syntactic features. However, both languages benefit from the use of a CRF as the last layer.

The structure of the paper is as follows. First, in Sect. 2, we describe the related work. In Sect. 3, we introduce our proposed method. In Sect. 4, we describe our experiments and summarize the results, and in Sect. 5, we state our conclusions and discuss future work.

## 2 Related Work

Early work on this task relies heavily on syntactic trees. Knight and Marcu [14] propose two alternative methods: a probabilistic noisy-channel model and a deterministic decision-based, both using syntactic parse trees. McDonald [16] presents a discriminative large-margin learning framework, which makes use of a large set of syntactic features as soft evidence in their model. Clarke and Lapata [4] formulate sentence compression as an optimization problem and solve it using ILP. The objective function includes a word importance score

and a trigram language model score. In order to assure the grammaticality of the output, some linguistic constraints are included. Filipova and Strube [9] present an unsupervised approach that relies on a dependency tree. A sentence is parsed by a dependency parser, which is then transformed in order to guarantee grammaticality when pruning. The compression task is formulated as an ILP problem and the best sub-tree is the one with highest score given by the objective function. Finally, the words of the compressed sentence are presented in the same order as the source sentence. Filipova and Altun [8] use previous work [9] to create a compression corpus of about two hundred thousand instances of sentence-compression pairs. Moreover, they also improve the compression method by adding structured prediction based on lexical, syntactic, and other types of features. A joint framework is also proposed by Berg-Kirkpatrick et al. [3] and Almeida and Martins [2], where they use sentence compression for multi-document summarization. Their model jointly extracts and compresses sentences using an ILP approach.

Recent work focus on neural network models. Filipova et al. [7] propose a deletion-based Long Short-Term Memory (LSTM) model which outputs readable and informative compressions without any linguistic features. Wang et al. [23] expand the latter architecture by using a BiLSTM which allows to capture contextual information of a sequence in both directions. Moreover, they propose two alternatives to compress sentences: to incorporate syntactic information in order to use this model in a cross-domain setting and to introduce syntactic constraints through ILP making the model more robust across different domains.

Ive and Yvon [12] propose to extend the task to a bilingual context, using two alternative methods: dynamic programming and ILP. Their goal was to generate parallel compressions of parallel sentences. Cross-lingual systems need to perform well across languages which is always a challenge. One way to tackle this problem is by projecting the word embeddings of each language to a common space. There are various models for learning cross-lingual representations [21]. The approach that we follow is the monolingual mapping which consists in training monolingual word embeddings on a large monolingual corpora and then learn a linear mapping from a source to a target space. Conneau et al. [5] leverage on adversarial training to learn this mapping and also contribute with a bilingual synthetic dictionary that results from the shared embedding space.

Deletion-based sentence compression can be considered a sequence labeling problem such as named entity recognition (NER) and part-of-speech (POS) tagging. State-of-the-art sequence labeling systems leverage on neural networks architectures. The combination of a BiLSTM with a CRF produced the most promising results among different architectures [11, 15, 20] for the above mentioned tasks. In this work, we extend the sentence compression task into a cross-lingual setting without parallel data. Instead of using softmax as final output [7, 23] we focus on the same architecture applied in other sequence labeling tasks which is a combination of a BiLSTM with a CRF classifier as the final layer.

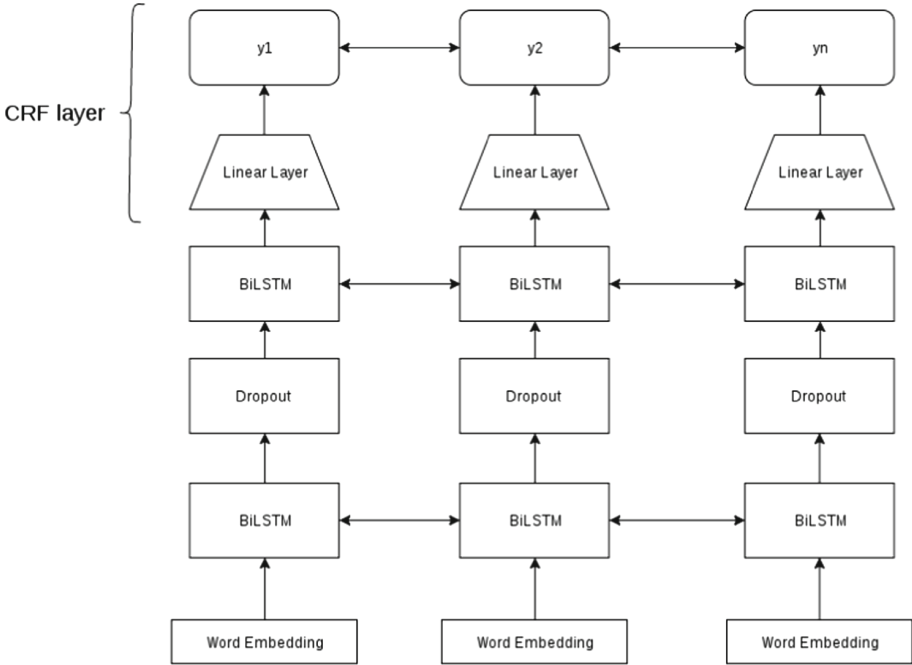
### 3 Sentence Compression

Inspired by the work of Filipova et al. [7], we also consider the task of sentence compression a deletion-based problem. The output of a system is a sequence of binary labels which represents if a word is deleted or not from the original sentence. In addition, we extend our approach for a cross-lingual setting where we use English and Portuguese datasets to evaluate our model. We propose an architecture based on a combination of two stacked BiLSTMs followed by a CRF classifier as the last layer, whereas previous work [7, 23] use softmax as the output layer. The use of a CRF as the output layer has achieved state-of-art results [20] in other sequence labeling tasks, such as POS and NER [11, 15]. Sequence labeling tasks benefit from the use of BiLSTMs [6], since they process information from past and future into two hidden states which are concatenated to form the final output. The use of a CRF as final layer helps to guarantee that the sequence of labels is globally consistent, jointly decoding the best chain of labels for a certain input. Consider an input sequence of words as  $s = (w_0, w_1, \dots, w_n)$ . Each  $w_i$  belongs to a vocabulary,  $w_i \in V$ , which contains English and Portuguese words. Since the output is a shorter or equal length sequence, there are words in  $s$  that may be deleted, thus the compressed sentence is represented by a sequence of binary labels  $y = (y_0, y_1, \dots, y_n)$ , where  $y_i \in \{0, 1\}$  ( $y_i = 0$  represents a token that should be deleted from the original sequence and  $y_i = 1$  indicates that the token remains). We use 80 tokens as the input sequence maximum length. Each  $w_i$  is mapped to a 300-dimension pre-trained embedding. We use fastText embeddings [17]: each word is represented by a sum of representations of character n-grams, allowing a better representation of misspelled and rare words.

For cross-lingual systems, learning cross-lingual embeddings is essential to represent different languages in a shared space. In this way, words from different languages but meaning the same are close to each other. We rely on a bilingual dictionary of Portuguese-English pairs that was released by Conneau et al. [5], aligning monolingual word embedding spaces in an unsupervised way. The cross-lingual embeddings are learned using a method that leverages adversarial training to learn a linear mapping from a source to a target space [5]. These embeddings are fed into a BiLSTM, one at a time, processing a sequence from left to right and in reverse, capturing contextual information from both directions. The hidden vectors go through a dropout layer to prevent overfitting [22] and are fed into another BiLSTM. The concatenated output of the last BiLSTM is mapped with a dense layer and then a linear-chain CRF maximizes the best sequence of labels for each input sequence, as shown in Fig. 1.

On sequence labeling tasks such as POS tagging and NER [11, 15], this architecture has achieved state-of-art results [20]. The incorporation of syntactic features into neural network models has shown improvements [7, 23]. Considering the set of Universal POS tags illustrated on Table 1, we used the spaCy<sup>1</sup> parser [10] to POS tag each input sentence, which has an embedding vector associated that needs to be learned during training. For each sequence, it is also

<sup>1</sup> <https://spacy.io/>.



**Fig. 1.** Architecture of the network.

**Table 1.** Universal POS tags.

Open-class words	Closed-class words	Other
ADJ	ADP	PUNCT
ADV	AUX	SYM
INTJ	CCONJ	X
NOUN	DET	
PROPN	NUM	
VERB	PART	
	PRON	
	SCONJ	

performed dependency parsing. Each word is replaced by the dependency relation connecting to its head. During the training, the weights of this embedding are learned. We also experimented different combinations between these features: Word + POS embeddings; Word + POS + Dependency relation embeddings; Word + POS + Parent word + POS parent embeddings; and, Word + POS + Parent word + POS parent + Dependency relation embeddings.

## 4 Experimental Evaluation

In this section, we describe the datasets used, and summarize the results.

## 4.1 Data

The English dataset was publicly<sup>2</sup> released by Filipova and Altun [8]. It contains 200,000 sentence compression pairs for training and 10,000 sentence pairs for testing. A pre-processing step was applied before using the raw data. Some sentence pairs were removed from the training set, because there were compressions which contained words that were not in the same order as in the original sentence. The numbers were replaced by a special *NUMB* token and a word embedding is assigned to it. In addition, tokens not found in the vocabulary of the pre-trained embeddings are replaced by an *UNK* token.

The Portuguese dataset was publicly<sup>3</sup> released by Almeida et al. [1]. This corpus contains 801 documents split in 80 topics. Each topic has two human-made summaries of about 100 words which were built performing only sentence and word deletion. Although created for multi-document summarization, following some ideas of Nóbrega and Pardo [19] is possible to transform it for the sentence compression task. Considering that each human-made summary is a document composed by several sentences, each sentence is compressed using word deletion from the source documents. In order to create sentence compression pairs some heuristics were followed: a compressed sentence must be smaller or equal length in respect to the original sentence; the compression must be a sub-sequence of words from the source sentence; the words present in the compression sentence must be in the same order. After applying these rules, a new sentence compression dataset was created with 799 sentence compression pairs.

## 4.2 Automatic Evaluation

The system was evaluated by taking the first 1,000 sentences pairs from the English test set, following the same practise as Filipova et al. [7] and Wang et al. [23]. We compare our approaches with the current neural deletion-based baselines that were evaluated on the same data set:

- LSTM: This is the basic model of Filipova et al. [7] using a sequence to sequence paradigm. In short, the architecture of the network is based on three stacked LSTM layers with a softmax output layer;
- LSTM+: Filipova et al. [7] propose a version of their model which uses the same architecture but the input concatenates the current word embedding, parent word embedding of the current word in the dependency tree, and three bits indicating if the parent word has been seen in the compression;
- BiLSTM: In this setting, Wang et al. [23] use as base model an architecture of three-layered BiLSTM with a softmax as the last input layer;
- BiLSTM+SynFeat: Wang et al. [23] also propose an advanced version where they incorporate syntactic features into their model. In the input layer, they combine word, POS, and dependency embeddings into a single vector.

<sup>2</sup> <https://github.com/google-research-datasets/sentence-compression>.

<sup>3</sup> <http://labs.priberam.pt/Resources/PCSC.aspx>.

We consider four metrics for automatic evaluation: per-sentence accuracy, word-based F1-score, compression rate, and accuracy. The first represents the percentage of compressions that were fully reproduced; the second computes the recall and precision in terms of tokens kept in the reference and the generated compression; compression rate is the number of characters in the compression divided by the number of characters in the original sentence; finally, accuracy is defined as the percentage of total tokens correct in the compression.

The word embeddings used were the 300-dimension fastText pre-trained embeddings [18]. The POS and dependency embeddings have a 10-dimensional and a 40-dimensional vector, respectively, and their weights are updated during training. The main architecture has two stacked BiLSTM interleaved with a dropout [22] layer whose value is 0.2. We defined the value 80 as maximum for an input sequence.

The model was trained, with early stopping, using Adam [13] as optimizer, with a learning rate initialized as 0.001, and a batch size of 32. The dimension of the hidden-layers of BiLSTM is 200. The majority of the parameters above were selected based on the work of Reimers and Gurevych [20], which describes the best parameters to achieve a good performance on sequence labeling tasks. Before evaluating the previous model on Portuguese, due to the small data set it was re-trained using a strategy of 5-fold cross validation with the same architecture and parameters.

The architecture of the model LSTM+Softmax consists in two stacked LSTM with a softmax classifier as the last layer. For the next model, BiLSTM+Softmax, we just replaced the LSTM with a BiLSTM keeping the same structure. BiLSTM+CRF model uses a combination of a BiLSTM with a CRF classifier as the last layer. The previous models use only word embeddings as input. The set of features that achieved the best results was composed by word, POS, and dependency relation embeddings, which resulted in the following model: BiLSTM+CRF+SynFeat. The results of the automatic evaluation on the Google News dataset are reported in Table 2.

**Table 2.** Automatic evaluation of the systems on the Google News data set.

	Word F1	Per-sentence accuracy	Accuracy	Compression ratio
LSTM [7]	0.8	0.3	-	0.39
LSTM+PAR+PRES [7]	0.82	0.34	-	0.38
BiLSTM [23]	0.75	-	0.76	0.43
BiLSTM+SynFeat [23]	0.8	-	0.82	0.43
LSTM+Softmax	0.79	0.16	0.83	0.41
BiLSTM+Softmax	0.83	0.25	0.86	0.39
BiLSTM+CRF	0.83	0.29	0.86	0.40
BiLSTM+CRF+SynFeat	0.84	0.31	0.87	0.41



The previous models were also evaluated on the Portuguese dataset, which contains 799 sentence compression pairs. Due to the size of the data, the re-training of the model in this language uses a 5-fold cross validation. The Portuguese and English embeddings were projected into a shared space, making it possible to use in Portuguese the previously trained models on the English dataset. The results are reported in Table 3.

**Table 3.** Automatic evaluation of the models on the Portuguese dataset.

	Word F1	Per-sentence accuracy	Accuracy	Compression ratio
BiLSTM+Softmax	0.73	0.07	0.73	0.57
BiLSTM+CRF	0.75	0.19	0.74	0.57
BiLSTM+CRF+SynFeat	0.7	0.11	0.7	0.57

### 4.3 Discussion

When evaluating the proposed architecture, BiLSTM+CRF, on English, even the model with no syntactic features outperforms the current neural deletion sentence compression systems in terms of word-based F1 score and accuracy. Although the incorporation of syntactic features could lead to a better performance of the model by capturing grammatical relations, the results are not markedly better. Some compressions predicted by our model can be seen at Table 4.

Even though the majority of compressions seems to be grammatically correct, there are some examples in which the model decides to remove all the words from the source sentence. This might happen because there is not any constraint to ensure a minimum size. Although the word-based F1 score is better, the per-sentence accuracy metric was not better or equal: we think this is due to the size of the training dataset. While Filipova et al. [7] trained with about 2,000,000 sentence compression pairs, the training of our models were made with 200,000 instances, leading to an inferior result on this metric.

Since the Portuguese dataset was developed in this work, there are not previous models to which we can compare our results. However, it is possible to verify that the models with a CRF layers benefit when fully reproducing the compressions on the test set. It is interesting to verify that the model which performs better is the one without any syntactic features. Although previous work reports, in most cases, better performances using syntactic features, here we can verify that across models there is no benefit in adding these type of features. One of the reasons could be the size of training data, which did not allow the model to capture some important grammatical rules when compressing.

The results achieved demonstrated that this task benefits from the use of the CRF classifier as the last layer. The ability to decode the best global sequence of labels taking into account the correlations between labels allows to output a better compression on both languages.

**Table 4.** Sentences and compressions from the English data set. S: Input. G: Ground truth. P: Compressed sentences predicted by BiLSTM+CRF+SynFeat model.

---

S: In response to a question from NDP Treasury Board critic, Mathieu Ravignat on Tuesday, Clement told the House of Commons Tuesday that contracting out government services reduces costs

G: Clement told the House of Commons Tuesday that contracting out government services reduces costs

P: Contracting out government services reduces costs

---

S: Floyd Mayweather is open to fighting Amir Khan in the future, despite snubbing the Bolton-born boxer in favour of a May bout with Argentine Marcos Maidana, according to promoters Golden Boy.

G: Floyd Mayweather is open to fighting Amir Khan in the future

P: Floyd Mayweather is open to fighting Amir Khan

---

S: Studies and surveys have found that men and women dream differently

G: Men and women dream

P: Men and women dream differently

---

S: Interethnic relations are not a field for political games, Kazakhstan’s President Nursultan Nazarbayev declared in his speech at the Assembly of People of Kazakhstan, Tengrinews reports

G: Interethnic relations are not a field for political games

P: Interethnic relations are not a field for political games

---

## 5 Conclusions and Future Work

We propose a different neural architecture for sentence compression and evaluate it in a cross-lingual setting. We show that the current neural deletion sentence compression systems benefit from the use a CRF classifier. The proposed architecture achieves better or close results when compared to the current neural deletion approaches. Although the incorporation of syntactic features improved the architecture, the results are not significant better than the base model with only word embeddings. Across languages, the proposed architecture also shows a better performance than baseline models. However, it is interesting to verify that the inclusion of syntactic features has a negative impact on the Portuguese results. This could be due to the size of the dataset which did not enable the model to learn enough syntactic information.

In the future, it would be interesting to modify the method proposed by Filipova and Altun [8] to build a larger corpus of sentence compression pairs in Portuguese. Furthermore, the use of a larger corpus for the English language may increase the performance of the per-sentence accuracy metric, generating more readable and comprehensible compressions. One way to improve the prediction of the neural compression system, and avoid the problem of sentences with all the words removed, would be the use of an auxiliary loss function which would take into account the size of each sentence. This function could help the grammaticality of a compression.

## References

1. Almeida, M., Almeida, M.S., Martins, A., Figueira, H., Mendes, P., Pinto, C.: Priberam compressive summarization corpus: a new multi-document summarization corpus for European Portuguese. In: Proceedings of the 9th International Conference on Language Resources and Evaluation, pp. 146–152 (2014)
2. Almeida, M., Martins, A.: Fast and robust compressive summarization with dual decomposition and multi-task learning. In: Proceedings of the 51st Annual Meeting of the ACL, pp. 196–206 (2013)
3. Berg-Kirkpatrick, T., Gillick, D., Klein, D.: Jointly learning to extract and compress. In: Proceedings of the 49th Annual Meeting of the ACL, pp. 481–490 (2011)
4. Clarke, J., Lapata, M.: Global inference for sentence compression: an integer linear programming approach. *J. Artif. Intell. Res.* **31**, 399–429 (2008)
5. Conneau, A., Lample, G., Ranzato, M., Denoyer, L., Jégou, H.: Word translation without parallel data. [arXiv:1710.04087](https://arxiv.org/abs/1710.04087) (2017)
6. Dyer, C., Ballesteros, M., Ling, W., Matthews, A., Smith, N.A.: Transition-based dependency parsing with stack long short-term memory. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (2015)
7. Filippova, K., Alfonseca, E., Colmenares, C.A., Kaiser, L., Vinyals, O.: Sentence compression by deletion with LSTMs. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing (2015)
8. Filippova, K., Altun, Y.: Overcoming the lack of parallel data in sentence compression. In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pp. 1481–1491 (2013)
9. Filippova, K., Strube, M.: Dependency tree based sentence compression. In: Proceedings of the Fifth International Natural Language Generation Conference, pp. 25–32. Association for Computational Linguistics (2008)
10. Honnibal, M., Johnson, M.: An improved non-monotonic transition system for dependency parsing. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp. 1373–1378. Association for Computational Linguistics (2015)
11. Huang, Z., Xu, W., Yu, K.: Bidirectional LSTM-CRF models for sequence tagging. [arXiv:1508.01991](https://arxiv.org/abs/1508.01991) (2015)
12. Ive, J., Yvon, F.: Parallel sentence compression. In: Proceedings of COLING 2016: Technical Papers, pp. 1503–1513 (2016)
13. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. In: Proceedings of the International Conference on Learning Representations (2015)
14. Knight, K., Marcu, D.: Statistics-based summarization - step one: sentence compression. In: Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence, pp. 703–710. Association for the Advancement of Artificial Intelligence Press (2000)
15. Ma, X., Hovy, E.: End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In: Proceedings of the 54th Annual Meeting of the ACL, pp. 1064–1074 (2016)
16. McDonald, R.: Discriminative sentence compression with soft syntactic evidence. In: 11th Conference of the European Chapter of the ACL (2006)
17. Mikolov, T., Grave, E., Bojanowski, P., Puhrsch, C., Joulin, A.: Advances in pre-training distributed word representations. [arXiv:1712.09405](https://arxiv.org/abs/1712.09405) (2017)
18. Mikolov, T., Grave, E., Bojanowski, P., Puhrsch, C., Joulin, A.: Advances in pre-training distributed word representations. In: Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018) (2018)

19. Nóbrega, F.A.A., Pardo, T.A.S.: Investigating machine learning approaches for sentence compression in different application contexts for Portuguese. In: Silva, J., Ribeiro, R., Quaresma, P., Adami, A., Branco, A. (eds.) PROPOR 2016. LNCS (LNAI), vol. 9727, pp. 245–250. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-41552-9\\_25](https://doi.org/10.1007/978-3-319-41552-9_25)
20. Reimers, N., Gurevych, I.: Optimal hyperparameters for deep LSTM-networks for sequence labeling tasks. [arXiv:1707.06799](https://arxiv.org/abs/1707.06799) (2017)
21. Ruder, S., Vulicm, I., Søgaard, A.: A survey of cross-lingual embedding models. [arXiv:1706.04902](https://arxiv.org/abs/1706.04902) (2017)
22. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **15**, 1929–1958 (2014)
23. Wang, L., Jiang, J., Chieu, H.L., Ong, C.H., Song, D., Liao, L.: Can syntax help? Improving an LSTM-based sentence compression model for new domains. In: Proceedings of the 55th Annual Meeting of the ACL (2017)



# Towards Constructing a Corpus for Studying the Effects of Treatments and Substances Reported in PubMed Abstracts

Evgeni Stefchov<sup>1</sup>, Galia Angelova<sup>2(✉)</sup>, and Preslav Nakov<sup>3</sup>

<sup>1</sup> Faculty of Mathematics and Informatics,  
Sofia University “St. Kliment Ohridski”,  
5 James Bourchier Blvd., 1164 Sofia, Bulgaria  
evgenistefchov@abv.bg

<sup>2</sup> Institute for Information and Communication Technologies,  
Bulgarian Academy of Sciences,  
25A Acad. G. Bonchev Str., 1113 Sofia, Bulgaria  
galia@lml.bas.bg

<sup>3</sup> Qatar Computing Research Institute, HBKU, Doha, Qatar  
pnakov@qf.org.qa

**Abstract.** We present the construction of an annotated corpus of PubMed abstracts reporting about positive, negative or neutral effects of treatments or substances. Our ultimate goal is to annotate one sentence (rationale) for each abstract and to use this resource as a training set for text classification of effects discussed in PubMed abstracts. Currently, the corpus consists of 750 abstracts. We describe the automatic processing that supports the corpus construction, the manual annotation activities and some features of the medical language in the abstracts selected for the annotated corpus. It turns out that recognizing the terminology and the abbreviations is key for determining the rationale sentence. The corpus will be applied to improve our classifier, which currently has accuracy of 78.80% achieved with normalization of the abstract terms based on UMLS concepts from specific semantic groups and an SVM with a linear kernel. Finally, we discuss some other possible applications of this corpus.

**Keywords:** Annotated rationales · Corpus construction  
Automatic discovery of effects · PubMed abstracts  
Terminology identification · Text classification

## 1 Introduction

PubMed is a large-scale database consisting of more than 28 million references and abstracts of biomedical publications. Using it as a knowledge discovery resource is tempting but challenging due to the highly specialized biomedical language with rich terminology and numerous abbreviations. Yet, PubMed abstracts are freely available, which is in contrast to other narratives in medical AI, e.g., clinical notes, which are inaccessible according to privacy regulations and requirements for in-domain knowledge. Other attractive features of PubMed as a text corpus include the fact that

biomedical publications are written with grammatically correct sentences and more or less have a predefined discourse structure. This facilitates the application of Natural Language Processing (NLP) tools and supports various research tasks that rely on automatic identification of terminology and its patterns of use.

Nowadays, text mining of PubMed abstracts is often the first step in the creation of annotated corpora as language resources supporting further information extraction tasks. Ambitious projects such as the development of databases with structured biomedical data and discovery of valuable associations between concepts depend on the correct automatic identification of the key semantic “carriers”: *terms*, which also need normalization, *values* of essential attributes, and *sentences* that summarize the findings and the conclusions. The relevant literature shows that annotated corpora are usually prepared using a combination of automatic text processing and manual annotation and refinement.

In our task, we develop a manually annotated corpus in order to improve text classification. We want to identify one most important key sentence in every PubMed abstract that reports about negative, positive or neutral effects of some potential catalyst  $X$  on some medical condition  $Y$ . During the annotation process, we try to identify patterns about how these effects are verbalized in the texts and what the relation between the terms and abbreviations in the title and the rationale sentence is. Obviously, developing a large-scale corpus of this type is too expensive, slow and almost impossible, and thus we attempt to address this bottleneck primarily by pre-selecting PubMed abstracts that contain explicitly the key phrases *negative effect*, *positive effect* or *no effect* in their titles. In this way, we incorporate the authors’ judgment about the abstract content and continue with the manual annotation of rationale sentences.

This paper is structured as follows. Section 2 lists some related work on corpus construction and text classification using PubMed abstracts; only few articles are mentioned among the numerous research papers dealing with PubMed texts. Section 3 explains specific aspects in the development of our manually annotated corpus using automatically selected, structured abstracts; various linguistic modalities encountered in these texts are considered as well as frequent patterns for the verbalization of negative, positive and neutral effects. Section 4 presents the application scenarios we intend to explore. Section 5 contains the conclusion and outlines plans for future work.

## 2 Related Work

We consider some research work dealing with corpus development. Information extraction was used on a corpus of PubMed abstracts to automatically identify and categorize biologically relevant entities and predicative relations by Zaremba et al. [1]. The relations include the following: Genes; Gene Products and their Roles; Gene Mutations and the resulting Phenotypes; and Organisms and their associated Pathogenicity. A total of 465 abstracts were collected and randomly split into a training set (327 abstracts) and a test set (138 abstracts). The training set was used for developing a lexicon and extraction rules. Specific annotation guidelines for entities and relations were elaborated by two molecular biologists and a computational linguist. Manual mark-up was performed by one biologist and reviewed by the other. Evaluations have

shown very good accuracy, esp. given the relatively small training set: F-measure higher than 90% for entities (genes, operons, etc.) and over 70% for relations (gene/gene product to role, etc.).

Doğan et al. [2] present the disease corpus developed in the US National Center for Biotechnology Information (NCBI), which contains disease name and concept annotations in a collection of 793 PubMed abstracts. Each abstract was manually annotated by two annotators with disease mentions and their corresponding concepts in the medical vocabulary MeSH and the online catalog OMIM. Much attention was paid to achieving high inter-annotator agreement. The public release of this corpus contains 6,892 disease mentions mapped to 790 unique disease concepts. The corpus was used as a means for improving the recognition of disease names in real texts (so-called disease normalization, which is as a very difficult task). Using the corpus, three different disease normalization methods were compared, achieving an F-measure of 63.7%. The authors concluded that “these results show that the NCBI disease corpus has the potential to significantly improve the state-of-the-art in disease name recognition ... by providing a high-quality gold standard thus enabling the development of machine-learning based approaches for such tasks”.

Another dataset is PubMed 200k RCT, which contains about 200,000 abstracts of randomized controlled trials (RCT), and a total of 2.3 million sentences [3]. Each sentence is labelled with its role in the corresponding abstract: *background*, *objective*, *method*, *result*, or *conclusion*. Only PubMed articles with MeSH index D016449, corresponding to RCTs, were included in the dataset. In addition, these abstracts were explicitly structured into 3–9 sections and contained no sections labelled “None”, “Unassigned”, or “empty”. When the labels of each section were originally given by the abstracts’ authors, PubMed mapped them into a smaller set of standardized labels: “background”, “objective”, “methods”, “results”, “conclusions”, “None”, “Unassigned”, or “” (an empty string). According to [3], more than half of the RCT abstracts in PubMed were unstructured. The expectation was that the public release of this dataset would accelerate the development of algorithms for sequential sentence classification; such tools in turn would facilitate sentence label prediction and hence, efficient browsing of literature because readers would be able to access selected sections only.

Concerning the amount of annotation samples necessary for a successful classifier training, Zaidan et al. [4] proposed to reduce the number of training examples needed, but to ask human annotators to provide hints to a machine learner by highlighting contextual “rationales” for each of their annotations. In this way, the annotator can identify features of the document that are particularly relevant and mark related portions of the example. Annotating rationales does not require the annotator to think about the classification feature space, nor even to know anything about it. So they demonstrated a method eliciting extra knowledge from naïve annotators, in the form of “rationales” for their annotations, which has significantly better performance than two strong baseline classifiers. It is interesting that their approach worked for positive and negative movie reviews across four annotators who had different rationale-marking styles.

Going deeper into the idea of using rationales (short and coherent pieces of input text, sufficient to provide motivations), we mention the work of Lei et al. [5] who have

shown how to design a generator (for automatic generation of rationales) and an encoder (for predicting justification). The generated rationales are subsets of the words from the input sentences with two key properties: first, they represent short and coherent pieces of text (e.g., phrases) and, second, the selected words must alone suffice for prediction as a substitute of the original text. Here, we are interested in their manually annotated test corpus, which provided the evaluation of multi-aspect sentiment analysis on beer reviews. The evaluation was done on sentence-level annotations on around 1,000 beer reviews with multiple sentences describing five features: the appearance, smell (aroma), palate, taste and overall impression of a beer. The annotation of each sentence indicates what aspect this sentence covers. In addition to the written text, the reviewers provided the ratings for each aspect (originally on a scale from 0 to 5 stars) as well as an overall rating. However, we notice that the generator learns phrases as rationales, not full sentences. Moreover, there can be several phrases generated as rationales for the same input text and the same aspect. Here, we work with several rationales per abstract as well.

Text classification is a common approach for clinical text mining, as a single technique or as a component in more complex NLP environments. Usually, such systems are focused on specific diseases, drugs or facts. For instance, simple text classification helps to detect misdiagnoses of epilepsy West syndrome in pediatric hospital narratives [6]. A retrospective analysis was conducted on 27,524 patient records with diagnoses and a corpus of 3,744 records was constructed (for 144 patients as positive examples and 3,600 randomly selected ones as negative examples). Without any additional annotation, multinomial Naïve Bayes and Support Vector Machine (SVM) classifiers were run and compared, with SVM achieving precision 76.8%, recall 66.7% and F-measure of 71.4% when evaluated with 10-fold cross-validation. The authors note that “the use of domain knowledge is not a necessary requirement to achieve reasonable results” [6].

A more sophisticated example is a hybrid pipeline for heart disease risk factor identification that analyzes clinical texts and recognizes diseases, associated risk factors, associated medications, and the time they are presented [7]. This pipeline integrated rule-based processing and classification and achieved an F-measure of 92.68% at Track 2 of the 2014 i2b2 clinical NLP challenge. After preprocessing, this system extracted phrase-based, logical and discourse tags. Time attribute identification was interpreted as a classification task, which was solved using an SVM. The system generated candidate sentences, containing risk factors, and passed them to classifiers. The authors concluded that a possible improvement could be to generate less negative samples by limiting the candidates to those that contain medical concepts in UMLS.

Finally, we consider a more general approach for classifying patient portal messages [8]. This task is important because patient portals are increasingly adopted as communication means: patients express various needs and medical experts deliver recommendations as well as informal opinions. The main categories are informational, medical, logistical, social, and other communications, with subcategories including prescriptions, appointments, problems, tests, follow-up, contact information, and acknowledgements. Secure portal messages might contain more than one type of communication. The performance of the classifiers was evaluated using a gold corpus of 3,253 manually annotated portal messages.



### 3 Corpus Construction and Annotation

#### 3.1 Selection of PubMed Abstracts

We want to find PubMed abstracts discussing the effect of some potential catalyst  $X$  on some  $Y$  under conditions  $Z$ . With the intension to collect the corpus faster, our first idea was to filter out PubMed abstracts that contain in their titles explicit statements about effect type: *negative*, *positive* and *neutral*. A sample title with this preferred structure is “*Positive effect of direct current on cytotoxicity of human lymphocytes*”. However, even for these abstracts, the subsequent manual annotation showed that the justification of the decisions about the effect of catalysts  $X$  on certain  $Y$  is hard. There is a considerable amount of abstracts entitled “the effect of  $X1$  and  $X2$  on  $Y$ ” or “the effect of  $X$  on  $Y1$  and  $Y2$ ”, for instance the title “*The mumps and rubella vaccination: no effect of feedback of vaccination scores in general practice*” and the title “*Positive effect of treatment with synthetic steroid hormone tibolon on intimal hyperplasia and restenosis after experimental endothelial injury of rabbit carotid artery*”. The variety of patterns and potentially misleading constructions make the titles difficult to interpret and annotate:

- “*Negative effect of age, but not of latent cytomegalovirus infection on the antibody response to a novel Influenza vaccine strain in healthy adults*”
- “*Calcium influx inhibition: possible mechanism of the negative effect of tetrahydropalmitine on left ventricular pressure in isolated rat heart*”
- “*Positive effect of etidronate therapy is maintained after drug is terminated in patients using corticosteroids*”.

Table 1 shows the amount of abstracts extracted initially from PubMed because their titles contained explicit description of the effect as *negative*, *positive* and *neutral*. In order to produce quickly a dataset where the annotation of the rationale will be maximally simple for naïve annotators, we removed all abstracts with “problematic” titles from the corpus. Table 2 contains the numbers of remaining abstracts after the removal of titles that contain “and”, “or”, “but”, “review”, “study”, “meta-analysis”, etc. Another problematic case are titles where the polarity is determined by a modifier preceding the statement about *negative*, *positive* and *no* effect, for instance in a title starting by “*Absence of positive effect ...*” or, e.g., in the title “*Association of cystic fibrosis transmembrane-conductance regulator gene mutation with negative outcome of intracytoplasmic sperm injection pregnancy in cases of congenital bilateral absence of vas deferens*”. In this way, we eventually ended up with 750 abstracts as shown in Table 3.

**Table 1.** Abstracts with *positive/negative/no effect* in the title.

Pattern in the title	Effect of	Impact of	Influence of	Total
Positive	242	133	52	427
Negative	247	238	50	535
No	782	84	127	993
Total	1,271	455	229	1,955

**Table 2.** Abstracts without *and/or/but/review/study/meta analysis*, etc. in the title.

Pattern in the title	Effect of	Impact of	Influence of	Total
Positive	135	74	27	236
Negative	171	140	31	342
No	406	40	82	528
Total	712	254	140	1,106

**Table 3.** Abstracts with *positive/negative/no effect* at the beginning of the title.

Pattern in the title	Effect of	Impact of	Influence of	Total
Positive	86	57	19	162
Negative	63	73	18	154
No	341	30	63	434
Total	490	160	100	750

Note that by keeping only abstracts where the phrase *positive/negative/no effect* occurs in the beginning of the title, we avoid titles containing e.g. “*dominant negative effect*”. The latter is ambiguous because “*dominant negative*” is a term in molecular biology, meaning a mutation whose gene product adversely affects the normal, wild-type gene product within the same cell. So, domain terminology is another important factor that needs to be taken into consideration when constructing the corpus. Further potentially ambiguous titles are treated properly, e.g., “*No effect of negative mood on the alcohol cue reactivity of in-patient alcoholics*” contains the word “*negative*”, but it is considered in the neutral class because “*no effect*” is the first phrase of the title.

### 3.2 Annotating the Rationale

In Sect. 2, we listed approaches to classification based on rationales (one sentence or phrases [4, 5] that represent short and coherent pieces of text that must suffice to express the polarity of the effect presented in the text). However, both approaches [4, 5] deal with product reviews: [4] works with one rationale while [5] (dealing with multi-aspect classification) allows multiple rationales per input text. Here, we apply the same idea to biomedical abstracts, which are scientific publications, i.e., their primary aim is not to assess some specific product (movie, beer, etc.). Thus, the rationale is often expressed in several sentences listing results of tests with target patients groups. We found that the summarizing concluding sentences usually generalize the findings and provide some argumentation about the polarity of the effect, but the latter is often expressed by domain-specific verbalizations. One sample abstract is shown in Fig. 1. In contrast to product reviews that contain numeric scores in addition to the text, the variability of nuances in PubMed abstracts is much higher and we need biomedical expertise to decide about the “most important, most convincing” sentence or phrases that must suffice to express alone the polarity of the effect.

Title : Negative effect of aging on psychosocial functioning of adults with congenital heart disease

ABSTRACT:

BACKGROUND: Improvements in life expectancy among adults with congenital heart disease (ACHD) provide them with unique challenges throughout their lives and age-related psychosocial tasks in this group might differ from those of healthy counterparts. This study aimed to clarify age-related differences in psychosocial functioning in ACHD patients and determine the factors influencing anxiety and depression.

METHODS AND RESULTS: A total of 133 ACHD patients (aged 20-46) and 117 reference participants (aged 20-43) were divided in 2 age groups (20 s and 30 s/40 s) and completed the Hospital Anxiety and Depression Scale, Independent-Consciousness Scale, and Problem-Solving Inventory. Only ACHD patients completed an illness perception inventory. ACHD patients over 30 showed a significantly greater percentage of probable anxiety cases than those in their 20 s and the reference group. Moreover, ACHD patients over 30 who had lower dependence on parents and friends, registered higher independence and problem-solving ability than those in their 20 s, whereas this element did not vary with age in the reference participants. Furthermore, ACHD patients may develop an increasingly negative perception of their illness as they age. The factors influencing anxiety and depression in patients were aging, independence, problem-solving ability, and NYHA functional class.

CONCLUSIONS: Although healthy people are psychosocially stable after their 20 s, ACHD patients experience major differences and face unique challenges even after entering adulthood.

**Fig. 1.** Phrases with abbreviations supporting the polarity in Results and Conclusions (<https://www.ncbi.nlm.nih.gov/pubmed/25391256>).

In Fig. 1, the rationales (short and coherent pieces of text) are underlined. We see that the automatic recognition and processing of abbreviations is very important because often the rationales contain abbreviated versions of the basic terms that are defined in the particular abstract.

We should also note the complexity of assigning a category to each biomedical abstract. Naïve annotators, who are not medical professionals, have limited capacity to comprehend and to interpret the abstract content, and thus they tend to make decisions based on general lexica, but they might be confused by the sentiment expressed in the text, as shown in the abstract on Fig. 2. The underlined sentences convince the ordinary reader that no negative effect has been observed, and in this way the title is a bit misleading and the abstract should be annotated as reporting about a “neutral effect”. Only experts might judge whether the phrases in italic (*pleural thickening, small foci of collagen*) are dangerous deviations in the “formation of 8-hydroxydeoxyguanosine in DNA”, which is the subject according to the title. The Japanese authors of this abstract perhaps intended title to be “*On the negative effect of ...*”. This example shows that all

Title : Negative effect of long-term inhalation of toner on formation of 8-hydroxydeoxyguanosine in DNA in the lungs of rats in vivo

ABSTRACT:

We assessed the effects of long-term inhalation of toner on the pathological changes and formation of 8-hydroxydeoxyguanosine (8-OH-Gua) in DNA in a rat model. Female Wistar rats (10 wk old) were divided evenly into a high concentration exposure group (H: 15.2 mg/m<sup>3</sup>), a low concentration exposure group (L: 5.5 mg/m<sup>3</sup>), and a control group. The mass median aerodynamic diameter of the toner was 4.5 microm. The rats were sacrificed at the termination of a 1-yr or 2-yr inhalation period. Pathological examination was performed on the left lung, and the level of 8-OH-Gua in DNA from the right lung was measured using a high-performance liquid chromatography (HPLC) column. The pathological findings showed that lung cancer was not observed in any of the exposed or control groups, though *pleural thickening and small foci of collagen were observed in toner-exposed rat lungs*. Inhalation of the toner for 1 and even 2 yr did not induce the formation of 8-OH-Gua in DNA in rat lungs. These data suggest that long-term inhalation of toner may not induce lung tumors.

**Fig. 2.** An abstract that might look neutral to naïve annotators (<https://www.ncbi.nlm.nih.gov/pubmed/16195210>).

texts included in the initially selected corpus of PubMed abstracts need careful manual assessment and refinement. Only few such abstracts were found and removed so far, all of them authored by non-native speakers.

### 3.3 Typical Expressions and Frequent Patters

The typical titles of the PubMed abstracts we consider have the following structure:

[the|a] negative effect of X on|in|for Y  
 [the|a] positive effect of X on|in|for Y  
 [the|a] no|neutral effect of X on|in|for Y

The grammatical correctness of the PubMed titles enables easier automatic recognition of terms and their abbreviations. While progressing with the annotation, we discover also typical patterns of phrases that signal the effect polarity, for instance:

- In the “no effect” abstracts: “Overall results disclosed no significant effect of this drug on ...”, “no significant difference was observed ...”, “there were no notable changes”, “X is unlikely to cause clinically significant interactions”, “X does not have any detrimental effect on the longevity and clinical outcomes”
- In the “positive effect” abstracts: “X can facilitate the normalization of Y”, “X may be effective in leading to Y”, “X is likely to be worthwhile and unlikely to be harmful for Y”, “X leads to improved quality of life over 96 weeks”, “X was found to increase in yeast cells”, “Patients exposed to the integrated care model exhibited significantly fewer depressive symptoms”.

- In the “*negative effect*” abstracts: “*the prolonged use of X may have a negative effect on Y*”, “*X has a relationship not only with Y, but also with a poor outcome*”, “*X is a significant prognostic factor for low overall survival*”, “*Development of X is associated with worse long-term outcomes*”.

When selecting the rationale sentences, we choose among the candidates those that contain the terminology mentioned in the abstract’s title.

## 4 Current Classification Experiments

In previous work, we studied the classification of PubMed abstracts without using any manually annotated corpus [9]. Various experiments were performed with text pre-processing options, semantic indexing using UMLS and MetaMap, and classification algorithms like Bernoulli/Multinomial Naïve Bayes classifiers and Linear/RBF SVM.

Abstracts are represented as bags of words, but they can have sections (segments) with subtitles: e.g., Background, Results, Conclusions, etc. This structure is either provided by PubMed or we induce it automatically. Segment names are not fixed and, in addition to discovering the segments, we also map their names to a predefined set – Background and Objectives, Methods, Results, Conclusions and Others. The label “Others” is assigned to text fragments without any explicitly assigned subtitle.

Different variants are proposed for preprocessing the collected abstracts, which include the use of concepts from semantic networks such as UMLS and MeSH. The semantic indexing is performed by MetaMap [10, 11] – the conceptual search engine of UMLS that provides indexing of terms from the given text by user-selected vocabularies included in UMLS. In one of the experiments, UMLS concepts were limited by semantic groups to a predefined set.

One group of tests explicates the role of the different abstract sections. We give different inputs to the classifiers – the full abstract, only the Conclusions or the Results section, or both Results and Conclusions. We specifically chose the Results and the Conclusions sections because the usefulness of the effect should be most explicit there. A second cycle of experiments used the results of the initial ones, e.g., if the best results for linear SVM are achieved using the full abstract, then the full abstract will be considered in further experiments when classifying with Linear SVM. The second type of experiments determines the importance of domain knowledge by importing names of concepts found in UMLS and MeSH, where the concepts might be limited by semantic groups. In these experiment, concepts from UMLS or the MeSH hierarchy will be added to or replace the corresponding terms in the sentences where they occur.

We also included the title in these experiments. We tried to define the sentence from the Results or the Conclusions section that has the largest word overlap (without stop words and punctuation) with the title and to classify the given example only based on this ‘best sentence’. Word overlap is defined as the overlap of raw words from the original text or as overlap of the found UMLS/MeSH concepts. Then we use the concepts from UMLS and MeSH that are found in the title; the idea is to normalize the abstract by replacing each occurrence of the UMLS/MeSH concept from the title with the tags  $X_1, X_2, \dots, X_n$  where  $n$  is the number of concepts found in the title.

We define two baselines to compare with the results of our system. The first one is a ‘majority classifier’ that always predicts the most frequent class in our dataset. Its accuracy is 57.87%. The second baseline predicts the class of the example by searching for the same signaling phrases as in the title, but this time in the article’s abstract and choosing the most frequent class as per these phrases, breaking ties using the first baseline; the accuracy for this second baseline is 61.60%.

The best results for the experiment that defines the role of the different parts of text is achieved by using the full abstract and linear or RBF-based kernel SVM, with accuracy of 76.27% and 75.47%, respectively. When we use the full abstract and we limit the concepts from UMLS by semantic type, we achieve slightly better accuracy of 76.80%. If the classification is done only by using the ‘best sentence’, the accuracy decreases to 74.50%. This can be explained by the fact that although one sentence has the largest word overlap with the title, this does not necessarily mean that it will most clearly determine the polarity of the abstract. Our best accuracy of 78.80% is achieved with UMLS normalization of the abstract, limited by semantic type, and using a linear SVM classifier. We are not aware of other experiments that consider in such details the textual structure of the abstracts, the weight of the title’ words and the role of semantic terminology processing. We expect that training on the manually annotated rationales, the accuracy will improve further.

## 5 Conclusion and Future Work

We have presented our on-going efforts towards the construction of a corpus containing PubMed abstracts annotated with rationales. Previous work has shown that using rationales provides substantial improvements for classifying product reviews [4, 5], and now we want to test this idea on PubMed abstracts. Several questions remain open at this stage of our work, e.g., whether it is better to mark one rationale per abstract or we can annotate several ones. Another open question is whether a corpus of 750 abstracts will allow us to see significant improvements on the classification task given the large medical vocabulary and the comprehensive PubMed archive.

In future work, we plan experiments with various techniques for abstract classification, e.g., using clustering [12] and deep learning [5]. We consider very important the automatic processing of abstract terminology with special focus on the abbreviations, to enable term normalization and reference to the abbreviated forms. This will help to connect the terms in the abstract titles and the abbreviations occurring in the rationale phrases and sentences. In this way, we could incorporate rule-based analysis in the preprocessing phase. Another promising research direction is on using summarization [13] techniques to find the most important part of the abstract, and then performing classification based on this abstract, or using attention mechanism to focus the model on sentences in the abstract that are most similar to the title; the latter is typically done in an unsupervised way, but we could use our manual rationales to train a supervised attention mechanism [14]. We also plan to experiment using various word representations, which have been shown useful for biomedical text classification [15].

**Acknowledgements.** The authors are grateful to the anonymous reviewers for their valuable comments.

## References

1. Zaremba, S., et al.: Text-mining of PubMed abstracts by natural language processing to create a public knowledge base on molecular mechanisms of bacterial enteropathogens. *BMC Bioinf.* **10**, 177 (2009)
2. Doğan, R., Leaman, R., Lu, Z.: NCBI disease corpus: a resource for disease name recognition and concept normalization. *J. Biomed. Inf.* **47**, 1–10 (2014)
3. Dernoncourt, F., Lee, J.: PubMed 200 k RCT: a dataset for sequential sentence classification in medical abstracts. In: *Proceedings of the 8th International Joint Conference on Natural Language Processing (IJCNLP 2017)*, Taipei, Taiwan, pp. 308–313 (2017)
4. Zaidan, O., Eisner, J.: Modeling annotators: a generative approach to learning from annotator rationales. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2008)*, Honolulu, Hawaii, USA, pp. 31–40 (2008)
5. Lei, T., Barzilay, R., Jaakkola, T.: Rationalizing neural predictions. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2016)*, Austin, Texas, USA, pp. 107–117 (2016)
6. Sullivan, R., Yao, R., Jarrar, R., Buchhalter, J., Gonzalez, G.: Text classification towards detecting misdiagnosis of an epilepsy syndrome in a pediatric population. In: *Proceedings of the AMIA Annual Symposium*, Washington, DC, USA, pp. 1082–1087 (2014)
7. Chen, Q., Li, H., Tang, B., Wang, X., Liu, X., Liu, Z., Liu, S., Wang, W., Deng, Q., Zhu, S., Chen, Y., Wang, J.: An automatic system to identify heart disease risk factors in clinical texts over time. *J. Biomed. Inf.* **58**, S158–S163 (2015)
8. Cronin, R., Fabbri, D., Denny, J., Rosenbloom, S., Jackson, G.: A comparison of rule-based and machine learning approaches for classifying patient portal messages. *Int. J. Biomed. Inf.* **105**, 110–120 (2017)
9. Stefchov, E.: Analysis of biomedical abstracts to determine the effect of treatments and procedures. M.Sc Thesis, Sofia University “St. Kl. Ohridski”, July 2018, in Bulgarian (2018)
10. Aronson, A.: Effective mapping of biomedical text to the UMLS metathesaurus: the MetaMap program. In: *Proceedings of the AMIA Symposium*, Washington, DC, USA, pp. 17–21 (2001)
11. Aronson, A., Lang, F.-M.: An overview of MetaMap: historical perspective and recent advances. *J. AMIA* **17**(3), 229–236 (2010)
12. Malmasi, S., Hassanzadeh, H., Dras, M.: Clinical information extraction using word representations. In: *Proceedings of Australasian Language Technology Association Workshop (ALTA 2015)*, Parramatta, Australia, pp. 66 – 74 (2015)
13. Rush, A., Chopra, S., Weston, J.: A neural attention model for abstractive sentence summarization. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*, Lisbon, Portugal, pp. 378–389 (2015)
14. Mi, H., Wang, Z., Ittycheriah, A.: Supervised attentions for neural machine translation. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2016)*, Austin, Texas, USA, pp. 2283–2288 (2016)
15. Malmasi, S., Hassanzadeh, H., Dras, M.: Clinical information extraction using word representations. In: *Proceedings of Australasian Language Technology Association Workshop (ALTA 2015)*, Parramatta, Australia, pp. 66 – 74 (2015)



# Recursive Style Breach Detection with Multifaceted Ensemble Learning

Daniel Kopev<sup>1</sup>, Dimitrina Zlatkova<sup>1</sup>✉, Kristiyan Mitov<sup>1</sup>, Atanas Atanasov<sup>1</sup>,  
Momchil Hardalov<sup>1</sup>, Ivan Koychev<sup>1</sup>, and Preslav Nakov<sup>2</sup>

<sup>1</sup> FMI, Sofia University “St. Kliment Ohridski”, Sofia, Bulgaria  
{dkopev,dvzlatkova,kmmitov,amitkov}@uni-sofia.bg,  
{hardalov,koychev}@fmi.uni-sofia.bg

<sup>2</sup> Qatar Computing Research Institute, HBKU, Doha, Qatar  
pnakov@qf.org.qa

**Abstract.** We present a supervised approach for style change detection, which aims at predicting whether there are changes in the style in a given text document, as well as at finding the exact positions where such changes occur. In particular, we combine a TF.IDF representation of the document with features specifically engineered for the task, and we make predictions via an ensemble of diverse classifiers including SVM, Random Forest, AdaBoost, MLP, and LightGBM. Whenever the model detects that style change is present, we apply it recursively, looking to find the specific positions of the change. Our approach powered the winning system for the PAN@CLEF 2018 task on Style Change Detection.

**Keywords:** Multi-authorship · Stylometry · Style change detection  
Style breach detection · Stacking ensemble  
Natural language processing · Gradient boosting machines

## 1 Introduction

There are numerous tasks related to authorship attribution, but most of the research has been concentrated on large documents. An interesting problem to tackle for smaller texts is the one of style change detection: given a text document, identify whether style change occurs anywhere in it. This formulation usually entails a uniform distribution of text segments from multiple authors. A version of it is the intrinsic plagiarism detection problem, in which it is considered that there is a dominant author of the document being examined. Another variation is the task of detecting style change positions: determine the exact location of style breaches in the text. Historically, this has proven to be a difficult but interesting task, and performance in terms of accuracy has been low, leaving a lot of room for potential improvements over the state-of-the-art. Here, we target

---

D. Kopev, D. Zlatkova, K. Mitov and A. Atanasov—Equal Contribution.



two tasks: (i) predicting whether style change occurs (Style Change Detection<sup>1</sup>), and (ii) finding the exact position of the change (Style Breach Detection<sup>2</sup>).

## 2 Related Work

*Authorship Attribution.* Previous work on authorship attribution and related problems (e.g., author obfuscation [3, 4, 12, 15]) used primarily term frequencies [5, 9] and features from stylometry [8, 9, 18]. We borrowed ideas for traditional features from [2, 13], but we also designed some new ones, related to tautology, grammar contractions, quote use discrepancies, beginning and ending author statement words, and named entity spellings (see Sect. 3.4).

*Style Breach Detection.* See [19] for a summary of previous work on style breach detection and related tasks. Here we outline some of the most relevant work. Karaš et al. [5] used TF.IDF, POS tags, stop words and punctuation to represent paragraphs in the text, and applied a Wilcoxon Signed Rank test to check whether two segments are likely to come from the same distribution. They moved a sliding window over the sentences, computing similarity statistics and using dictionaries with common English words and sentiment. Then, they used a pre-defined threshold to determine whether a style breach between two sentences was likely. Safin and Kuznetsova [16] explored techniques typically used for intrinsic plagiarism detection. They vectorized sentences using pre-trained skip-thought models and looked for outliers using cosine-based distance between vectors.

## 3 Style Change Detection

Here, we describe our approach, which powered the winning system [20] for the PAN@CLEF 2018 task on Style Change Detection [7].

### 3.1 Data

We used data provided by the organizers of the CLEF-2018 PAN task on Style Change Detection [7], which was based on user posts from StackExchange covering different topics with 300–1,000 tokens per document. It included a training set of 3,000 documents and a validation set of 1,500 documents.

### 3.2 Preprocessing

We pre-process the data in two phases. The first phase is applied before any feature extraction has taken place, and it replaces URLs and long numbers with specific tokens. The second phase is applied during feature computation. It filters the stream of words and replaces file paths, long character sequences and very

<sup>1</sup> <http://pan.webis.de/clef18/pan18-web/author-identification.html>.

<sup>2</sup> <http://pan.webis.de/clef17/pan17-web/author-identification.html>.

long words with special tokens. Additionally, an attempt is made to split long hyphenated words (with three or more parts) by checking whether most of the sub-words are present in a dictionary of common words (from the NLTK words corpus [11]). The objective of all these steps is to reduce the impact of long words, which could adversely affect features that take word length into account. Such features are those from the lexical group and preprocessing is applied to them only, since it might have undesirable effect on the rest of the features.

### 3.3 Text Segmentation

Style changes in text documents entail that parts of the text would differ in some way. In an attempt to spot such differences, we split the document into four segments of roughly equal length (measured in terms of word tokens), we calculated the feature vectors for each of the segments, and we found the maximum difference between the values for each feature for any pair of segments. We chose the number of segments (namely, four) based on the distribution of the number of style changes across the entire training dataset. In order to obtain more potential data points, we applied a sliding window across each document with an overlap of one third of the segment size (see Fig. 1). We applied this segmentation procedure for four of the feature groups, three of which used a sliding window. See Sect. 3.4 for more details.

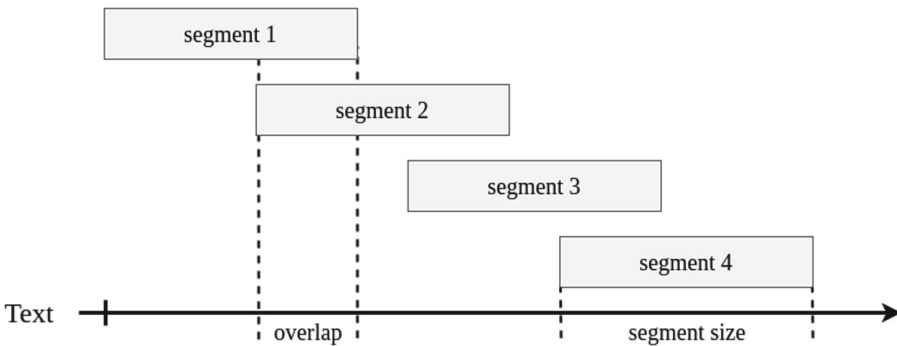


Fig. 1. Applying a sliding window on the text.

### 3.4 Text Representation

Below, we describe the features we engineered specifically for the task of discovering style changes. The dimensionality of each of them is shown in Table 1.

**Tautology:** At a grammatical and, one might say, psychological level, writers attempt to avoid using repetitive statements. We account for this by looking at the average number of occurrences of each one to five word-grams in the entire document, and we use a vector of size five with the respective averages for text representation.

**Table 1.** Dimensionality of our features.

Features	Dimensionality
Tautology	5
Grammar contractions	29
Beginning and ending of author statements	1
Quotation marks	1
Readability	9
Frequent words	415
Lexical	34
Vocabulary richness	2
Named entity spellings	2

**Grammar Contractions:** Another viewpoint we look at is based on the discrepancies in words, which have contracted forms or shortened version of words such as *I will (I'll)*, *are not (aren't)*, and *they are (they're)*. Because most people favor one or the other, contraction apostrophes are suitable discriminative features (even more so in formal contexts) for identifying whether a piece of text is likely to be single- or multi-authored.

**Frequent Words:** Frequent words include stop words (taken from NLTK [11]) and function words (compiled from three separate lists<sup>3,4,5</sup>). Each frequent word is counted per text segment.

**Lexical:** The lexical features are computed as ratios per text segment using a sliding window and can be divided into the following three types:

*Character-Based:* Spaces, digits, commas, colons, semicolons, apostrophes, quotes, parenthesis, number of paragraphs, and punctuation in general.

*Word-Based:* We POS-tag the segment using NLTK, and we extract features such as ratios of pronouns, prepositions, coordinating conjunctions, adjectives, adverbs, determiners, interjections, modals, nouns, personal pronouns and verbs. Other word-based features include words with 2 or 3 characters, words with over 6 characters, average word length, all-caps words, and capitalized words.

*Sentence-Based:* Those include ratios of question, period, exclamation sentences, short and long sentences, and average sentence length.

**Quotation Marks:** Normally used in pairs, different people might consistently prefer using either single or double quotation marks. We first exclude every shortened word with apostrophe (from a grammar contraction words dictionary), and then we use the variance in the number of single and double quotes as a single-feature representation of the documents.

<sup>3</sup> <http://semanticsimilarity.files.wordpress.com/2013/08/jim-oshea-fwlist-277.pdf>.

<sup>4</sup> <http://www.sequencepublishing.com/1/academic.html>.

<sup>5</sup> <http://www.edu.uwo.ca/faculty-profiles/docs/other/webb/essential-word-list.pdf>.

**Vocabulary Richness:** Similarly to [2], vocabulary richness is represented by averaged word frequency class. Using the Google Books common words list,<sup>6</sup> the frequency class of a word  $x$  is computed as  $\log_2 \frac{f(X)}{f(x)}$ , where  $f$  is the frequency function and  $X$  is the most frequent word in the corpus, in our case *the*. Two features are extracted per segment: the average frequency class of all words in it, and the ratio of unknown words (words not present in the common words list).

**Table 2.** The 12 most frequent beginning and ending words in author statements (after stopword removal).

(a) beginning		(b) end	
Word	Rescaled counts	Word	Rescaled counts
however	1.00	etc	1.00
one	0.96	time	0.95
note	0.92	well	0.76
edit	0.87	question	0.68
first	0.69	way	0.61
yes	0.63	god	0.58
also	0.60	p	0.45
another	0.50	example	0.45
finally	0.40	work	0.39
since	0.37	war	0.39
update	0.35	though	0.37
let	0.31	answer	0.37

**Readability:** The following readability features are computed per text segment via the *Textstat*<sup>7</sup> Python package: Flesch reading ease, SMOG grade, Flesch-Kincaid grade, Coleman-Liau index, automated readability index, Dale-Chall readability score, difficult words, Linsear write formula, and Gunning fog.

**Beginning and Ending of Author Statements:** As can be seen in Table 2, author statements begin and end with very different types of words. This can be used to locate points in documents where word clusters of small size contain high amount of these terms. We tried two approaches, applied after stopword removal, and we experimented with word phrases of sizes 1, 2 and 3, with single-terms yielding the best results. The first approach assigns scores to words to be rescaled (min-max normalized): it counts the number of times the target word is at a beginning or at an ending position relative to the author statement. A bit more sophisticated approach scores words based on how close they are to such a position. Each word is processed using a very steep half-sigmoid function (Eq. 1, with  $k$  denoting the steepness), taking its relative position and rewarding

<sup>6</sup> <http://norvig.com/google-books-common-words.txt>.

<sup>7</sup> <http://github.com/shivam5992/textstat>.

those that are extremely close to a beginning or to an ending. Then, each word list of position scores is averaged across all documents.

$$x = \frac{\left| \frac{\text{statementLength}}{2} - (\text{position} + 1) \right|}{\frac{\text{statementLength}}{2}} \quad (1)$$

$$\text{Score}(\text{position}_{\text{statement}}) = \frac{(0 + k) * x}{(1 + k) - x}$$

Finally, the document feature vector is represented by looking at local document clusters of three words, containing multiple high-scored words, indicating there may be an end of author statement immediately followed by a beginning of a new one. This document representation was not added as part of the stacking classifier (Sect. 3.5.2), but nevertheless has a strong performance on its own, yielding 65% accuracy.

**Named Entity Spellings:** Different named entity spellings can reflect personal preferences, rather than cultural ones. We use Damerau-Levenshtein string-edit distance [1, 10] to find inconsistencies in the wording of the same named entities within an edit distance of 1. The feature vector consists of the minimum counts between the different spellings for each found named entity.

## 3.5 Classification

### 3.5.1 LightGBM

Our gradient boosting approach combines LightGBM [6] with Logistic Regression and TF.IDF vector representation. Note that we use the test data when calculating the IDF statistics. This is not cheating as we do not use the labels for the examples, we only calculate word frequencies. Then, we use feature importance weights with a Logistic Regression estimator to select the best TF.IDF features; moreover, we only select features with weight greater than 0.1. We tune the Logistic Regression hyper-parameters using cross-validation. The best results are achieved with Stochastic Average Gradient descent, and inverse of the regularization strength C of 2. We trained using bagging with five folds. A simple LightGBM baseline achieved 73% accuracy on the validation set. Tuning the LightGBM hyper-parameters increased the accuracy to 86%, supported by a CV score of 85%. These parameters can be seen in Table 4.

### 3.5.2 Stacking

The basic idea behind our Stacking Ensemble classifier was to take into account different independent points of view in the context of distinguishing multi-authored documents and to learn dependencies between them. At the bottom level, we train four different non-linear classifiers—SVM, random forest, AdaBoost trees, multi-layer perceptron (described in Table 4)—for each feature vector derived from the representations in Sect. 3.4 on 75% of the training data.

Then, each one of them predicts on the remaining 25% and is assigned a weight, based on its accuracy, relative to the remaining classifiers with the same input feature vectors. Those groups form a single vector each with prediction class probabilities, based on the weights and the outputs of its classifiers. These vectors, together with the predictions of the LightGBM classifier (see Sect. 3.5.1), serve as an input to a simple linear Logistic Regression meta-learner. The process of training is visualized in Fig. 2.

Before predicting, each classifier is trained again on the whole dataset (except for LightGBM, which is not weighted across groups). For each new sample, the zero-level classifiers use the same weights learned in training to transform the given sample as an input vector of probabilities for the meta-learner. This yields accuracy of 87%. The coefficients learned by the meta-model for each text representation and their standalone accuracies can be seen in Table 3.

**Table 3.** Style Change Detection: model coefficients and accuracy for different feature representations (in isolation).

Representation	Coefficient	Accuracy
Tautology	1.50	67.4
Grammar Contractions	1.25	61.0
Quotation Marks	0.05	55.8
Readability	-0.25	61.0
Frequent Words	0.81	63.3
Lexical	0.27	64.9
Vocabulary Richness	-1.46	51.0
LightGBM with TF-IDF	5.01	88.5

## 4 Style Breach Detection

In this section, we describe our approach to the more complex task of finding the positions where style breach occurs, which we address using the supervised model from the previous Sect. 3.

### 4.1 Data

We use the dataset from the PAN-2017 competition<sup>8</sup> [19], which consists of 187 documents each containing 1,000–2,400 word tokens. About 20% of the texts have no style changes and the rest have between 1 and 8 changes. Switches of authorships<sup>9</sup> may only occur at the end of sentences, not within. The exact positions of the style changes in the multi-authored documents are provided as part of the dataset, but we did not use them for training.

<sup>8</sup> <http://pan.webis.de/clef17/pan17-web/author-identification.html>.

<sup>9</sup> In this dataset, style change also means switch of authorship.

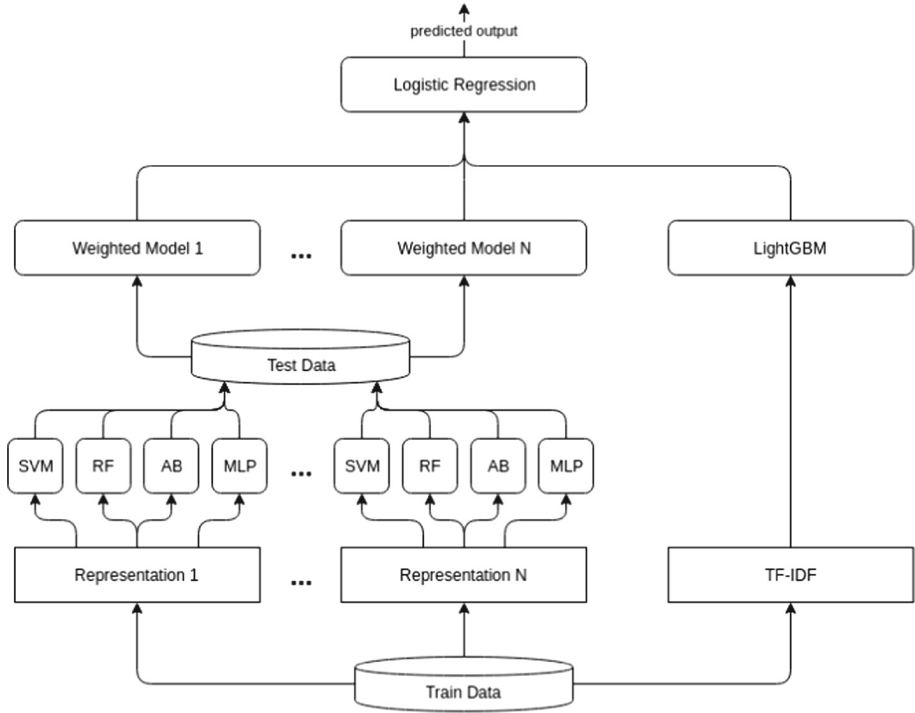


Fig. 2. Training the stacking classifier.

This dataset is hard due to its small size and to class imbalance. Applying our model from the previous section on it poses further challenges as we have originally developed the model to identify the presence of changes in shorter texts (300–1,000 tokens) and with fewer style breaches (up to 3).

## 4.2 Method

Given a document to analyze, we first apply our model from the previous Sect. 3 to predict whether there is style change in it. If style change is detected, we split the document in two: each half containing the same number of sentences. Then, we perform the same check for style change on each of the two parts, and if the results for both parts are negative then the exact position of the breach would be where the text was split in half. We repeat this procedure of splitting and searching for changes recursively until the length of the text fragment becomes less than 20 sentences, in which case, we return the middle point and we perform no more checks on the respective fragment.

Note that at training time, the model checks for the presence of changes on the full text, while at testing time it is applied to fragments of various sizes: starting with documents that are larger than those seen in training and going down to fragments whose size decreases exponentially. This discrepancy in the

**Table 4.** Meta and zero-level hyper-parameters for the classifiers in Fig. 2.

Classifier	Hyper-parameter	Value
Support Vector Machine	Kernel	Radial basis function
	Penalty $C$	1.0
	Tolerance	0.001
	Class weight	Balanced
Random Forest	Estimators	300
	With replacement	Yes
AdaBoost Trees	Base estimator	Decision tree
	Estimators	300
Multi-layer Perceptron	Layers	1
	Layer size	100
	Activation	ReLU
	Optimization	Adam
	Regularization	L2
	Regularization term	0.0001
	Learning rate	0.001
	Mini-batch size	200
	Maximum iterations	10000
LightGBM	Learning rate	0.1
	Number of leaves	31
	Feature fraction	0.6
	L1 regularization term	1.0
	L2 regularization term	1.0
	Minimum data in leaf	20
<b>Logistic Regression (meta-classifier)</b>	Optimization	Liblinear
	Regularization	L2
	Penalty $C$	1.0
	Tolerance	0.0001
	Maximum iterations	100

fragment sizes at training and testing makes the model’s task harder. In order to alleviate the problem, we chose a relatively large minimum size for the text fragments of 20 sentences, assuming that shorter texts would not be easily handled by the model; we later confirmed this suspicion experimentally.

The next issue is that due to class imbalance, the model is much more likely to predict the positive class, which results in a lot of false positive, i.e., many non-existent style breaches being predicted in a document. On many occasions, the recursive procedure predicted an unreasonable number of breaches in a single



document in the range of 20 or even 30, considering that the maximum number of style changes in a document in the training data was only 8. This is not completely unexpected though, as during training the model was never told the exact number of changes, just that there should exist at least one. Our strategy to cope with this was to increase the threshold for predicting that there is a style breach from 50% to 75%. This resulted in a significant drop in the average number of breaches predicted by the model from over 10 to 3.265.

### 4.3 Results

We evaluated the performance of our model for predicting the location of the style breaches using the following two evaluation measures:

- *WindowDiff* [14], which is standard in general text segmentation evaluation, and returns an error rate between 0 and 1 (0 indicating perfect prediction) for predicting the exact location of the breaches by penalizing near-misses less severely compared to other/complete misses or to predicting more style breach locations than there are to be found.
- *WinPR* [17], which computes common information retrieval measures, precision (WinP) and recall (WinR), and thus makes a more detailed, qualitative statement about the model performance.

We assessed our results by comparing them to the two baselines from [19]:

1. BASELINE-rnd randomly places between 0 and 10 borders at arbitrary positions inside a document.
2. BASELINE-eq also decides on a random basis how many borders should be placed (again 0–10), but then places these borders uniformly, i.e., so that all resulting segments are of equal size with respect to the tokens contained.

Table 5 shows the average results we achieved by applying our model using 5-fold cross-validation as well as the scores for the two baselines above. We can see that our stacking approach managed to outperform the two baselines on both evaluation measures. Our results are also close to the ones achieved at the PAN-2017 edition [5], although they cannot be compared directly, as the systems that participated in the PAN competition were evaluated on a different test dataset.

**Table 5.** Style Breach Detection: results for predicting the location of the breach.

	windowDiff	winP	winR	winF
BASELINE-rnd	0.6088	0.2779	0.5477	0.2366
BASELINE-eq	0.6345	0.3326	<b>0.6368</b>	0.2907
Stacking	<b>0.5719</b>	<b>0.3395</b>	0.6132	<b>0.3302</b>

## 5 Conclusion and Future Work

Detecting style change in texts is a difficult task for humans: we, ourselves, found it hard to discern between texts with and without style change while exploring the dataset. Nonetheless, our experiments have shown that it is possible for machine learning algorithms to achieve good performance for this problem.

The idea of applying a model recursively to find the exact style breach positions came to us as a natural experiment after tackling the simplified binary task of style change detection, for which we achieved an accuracy of 89%. Our results for the more complex style breach position task are very close to the current state-of-the-art without the need for much adaptation of the original solution.

We believe that the results can be improved further if training is done on text pieces of different lengths, given that during validation, the recursive procedure has to be applied to smaller and smaller fragments of the original document. Tuning the model to work better for the case of imbalanced classes can also be a source of improvement.

**Acknowledgements.** This work was supported by the Bulgarian National Scientific Fund within the project no. DN 12/9, and by the Scientific Fund of the Sofia University within project no. 80-10-162/25.04.2018.

## References

1. Damerau, F.J.: A technique for computer detection and correction of spelling errors. *Commun. ACM* **7**(3), 171–176 (1964)
2. Meyer zu Eissen, S., Stein, B., Kulig, M.: Plagiarism detection without reference collections. In: Decker, R., Lenz, H.J. (eds.) *Advances in Data Analysis*, pp. 359–366. Springer, Heidelberg (2007). [https://doi.org/10.1007/978-3-540-70981-7\\_40](https://doi.org/10.1007/978-3-540-70981-7_40)
3. Hagen, M., Potthast, M., Stein, B.: Overview of the author obfuscation task at PAN 2017: safety evaluation revisited. In: *Working Notes Papers of the CLEF 2017 Evaluation Labs*, CLEF 2017, vol. 1866 (2017)
4. Karadzhov, G., Mihaylova, T., Kiprova, Y., Georgiev, G., Koychev, I., Nakov, P.: The case for being average: a mediocrity approach to style masking and author obfuscation. In: Jones, G.J.F., et al. (eds.) *CLEF 2017. LNCS*, vol. 10456, pp. 173–185. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-65813-1\\_18](https://doi.org/10.1007/978-3-319-65813-1_18)
5. Karaś, D., Śpiewak, M., Sobiecki, P.: OPI-JSA at CLEF 2017: author clustering and style breach detection-notebook for PAN at CLEF 2017. In: *CLEF 2017 Evaluation Labs and Workshop - Working Notes Papers*, CLEF 2017, Dublin, Ireland (2017)
6. Ke, G., et al: LightGBM: a highly efficient gradient boosting decision tree. In: *Proceedings of the 30th Annual Conference on Neural Information Processing Systems*, NIPS 2017, Long Beach, California, pp. 3146–3154 (2017)
7. Kestemont, M., et al.: Overview of the author identification task at PAN-2018: cross-domain authorship attribution and style change detection. In: *Working Notes of CLEF 2018 - Conference and Labs of the Evaluation Forum*, CLEF 2018, Avignon, France (2018)
8. Khan, J.: Style breach detection: an unsupervised detection model-notebook for PAN at CLEF 2017. In: *CLEF 2017 Evaluation Labs and Workshop - Working Notes Papers*, CLEF 2017, Dublin, Ireland (2017)

9. Kuznetsov, M., Motrenko, A., Kuznetsova, R., Strijov, V.: Methods for intrinsic plagiarism detection and author diarization—notebook for PAN at CLEF 2016. In: CLEF 2016 Evaluation Labs and Workshop - Working Notes Papers, CLEF 2016, Évora, Portugal (2016)
10. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions and reversals. *Sov. Phys. Dokl.* **10**, 707 (1966)
11. Loper, E., Bird, S.: NLTK: the natural language toolkit. In: Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics, ETMTNLP 2002, Philadelphia, Pennsylvania, pp. 63–70 (2002)
12. Mihaylova, T., Karadjov, G., Kiprova, Y., Georgiev, G., Koychev, I., Nakov, P.: SU@PAN'2016: author obfuscation. In: Working Notes of CLEF 2016 - Conference and Labs of the Evaluation Forum, CLEF 2016, Évora, Portugal, pp. 956–969 (2016)
13. Pervaz, I., Ameer, I., Sittar, A., Nawab, R.: Identification of author personality traits using stylistic features—notebook for PAN at CLEF 2015. In: CLEF 2015 Evaluation Labs and Workshop - Working Notes Papers, CLEF 2015, Toulouse, France (2015)
14. Pevzner, L., Hearst, M.A.: A critique and improvement of an evaluation metric for text segmentation. *Comput. Linguist.* **28**(1), 19–36 (2002)
15. Potthast, M., Hagen, M., Stein, B.: Author obfuscation: attacking the state of the art in authorship verification. In: Working Notes Papers of the CLEF 2016 Evaluation Labs, CLEF 2016, Évora, Portugal (2016)
16. Safin, K., Kuznetsova, R.: Style breach detection with neural sentence embeddings—notebook for PAN at CLEF 2017. In: CLEF 2017 Evaluation Labs and Workshop - Working Notes Papers, CLEF 2017, Dublin, Ireland (2017)
17. Scaiano, M., Inkpen, D.: Getting more from segmentation evaluation. In: Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2012, Montreal, Canada, pp. 362–366 (2012)
18. Sittar, A., Iqbal, H., Nawab, R.: Author diarization using cluster-distance approach—notebook for PAN at CLEF 2016. In: CLEF 2016 Evaluation Labs and Workshop - Working Notes Papers, CLEF 2016, Évora, Portugal (2016)
19. Tschuggnall, M., et al.: Overview of the author identification task at PAN-2017: style breach detection and author clustering. In: Working Notes Papers of the CLEF 2017 Evaluation Labs, CLEF 2017, Dublin, Ireland (2017)
20. Zlatkova, D., et al.: An ensemble-rich multi-aspect approach towards robust style change detection: notebook for PAN at CLEF 2018. In: Working Notes of CLEF 2018 - Conference and Labs of the Evaluation Forum, CLEF 2018, Avignon, France (2018)

# **Machine Learning and Data Mining Applications**



# Improving Machine Learning Prediction Performance for Premature Termination of Psychotherapy

Martin Bohus<sup>2</sup>, Stephan Gimbel<sup>1</sup>, Nora Goerg<sup>3</sup>,  
Bernhard G. Humm<sup>1</sup> (✉), Martin Schüller<sup>4</sup>, Marc Steffens<sup>1</sup>,  
and Ruben Vonderlin<sup>1,2</sup>

<sup>1</sup> Hochschule Darmstadt - University of Applied Sciences,  
Haardtring 100, 64295 Darmstadt, Germany

{stephan.gimbel, bernhard.humm,

ruben.vonderlin}@h-da.de, marc.steffens@mail.de

<sup>2</sup> Central Institute of Mental Health, Institute for Psychiatric and Psychosomatic  
Psychotherapy, J5, 68159 Mannheim, Germany

{martin.bohus, ruben.vonderlin}@zi-mannheim.de

<sup>3</sup> Department for Clinical Psychology and Psychotherapy Varrentrappstr,  
Goethe University Frankfurt, 40-42, 60486 Frankfurt, Germany

goerg@psych.uni-frankfurt.de

<sup>4</sup> Deuschel & Schüller GbR, Sudetenstrasse 6, 64853 Otzberg, Germany

martin@deuschel-schueller.de

**Abstract.** Premature termination of psychosocial treatments is one of the major challenges in psychotherapy, with negative consequences for the patient, the therapist and the healthcare system as a whole. The aim of this pilot study is to use machine learning approaches to identify such parameters of dropout prediction based on either symptom questionnaires or open-ended diary questions in a specific population with borderline personality disorder (BPD). For this purpose, the Borderline Symptom Checklist (BSL-23) was used to create machine learning models to predict therapy dropout. We discovered that data from BSL-23 does not contain relevant predictors. Furthermore we present a concept of a private digital therapy diary (PDTD), which we use to investigate and predict dropout parameters in order to alert therapists on the danger of premature termination of individual patients.

**Keywords:** Machine learning · Psychotherapy  
Borderline personality disorder · Healthcare

---

This project (HA project no. 517/16-29) is funded in the framework of Hessen ModellProjekte, financed with funds of LOEWE – Landes-Offensive zur Entwicklung Wissenschaftlich-ökonomischer Exzellenz, Förderlinie 3: KMU-Verbundvorhaben (State Offensive for the Development of Scientific and Economic Excellence). Trial registration: World Health Organization International Clinical Trials Registry Platform DRKS00011555. <http://apps.who.int/trialsearch/default.aspx>

## 1 Introduction

Premature termination of psychosocial treatments is a general challenge in psychotherapy which affects up to 30% of all outpatients. The problem is even higher when it comes to non-specific treatments of patients with severe personality disorders, where dropout rates rise up to 67%. But also special treatments such as Dialectical Behavior Therapy (DBT) [1–4] reveal average attrition rates of 27% [5]. Premature termination of evidence based treatments mostly has negative consequences on different levels: The patient receives only parts of evidence-based treatments and thus benefits less. Both, the patient and the therapist might be demoralized and the resources have not been used efficiently, resulting also in negative financial consequences for the healthcare system and longer waiting times for other patients in need [6]. Research in the last decades has revealed some predictors of premature termination depending on characteristics of the patient (e.g. type of diagnosis or pattern of comorbidity), the treatment used (e.g. format of treatment delivery and treatment setting), the therapist, or interactions among these (e.g. therapeutic alliance) [7, 8]. However, previous research on the identification of these predictors has some limitations that make it difficult to apply these findings in psychotherapeutic practice: First, these predictors were identified post-hoc and contain only mean-based group calculations, which do not allow a conclusive assessment of premature termination risk for an individual patient. Second, the assessment of these variables does not consider dynamic processes (e.g. mood states), which are likely to fluctuate over time during the therapy process (especially in BPD-patients [9]) and might play a key role in the continuation or termination of therapy. Consequently, these findings give only indirect hints for the dropout risk of an individual patient and can therefore not be used by the therapist during a specific patient treatment.

To address this issue, we have introduced an IT-based approach to assist therapists in the early detection of an increased dropout risk of a single patient during therapy [10]. The concept includes multiple data collections with each patient during ongoing therapy. Based on these data, the therapist will be alerted if a patient is at risk for premature treatment termination.

This paper presents a detailed analysis based on the Borderline Symptom List-23 (BSL 23) [11], a disorder-specific sensitive questionnaire for patients with borderline personality disorder (BPD). BPD is a severe and common psychiatric disorder, which affects about 1% of the adult German population. The goal was to identify relevant features for a possible dropout prediction within this group of patients with sufficient accuracy. As a second step, we describe the development and feasibility of a new questionnaire form, called Personal Digital Therapy Diary (PDTD), and provide first data. Furthermore we describe the process of alerting the therapist on the risk of premature termination of therapy of an individual patient.

## 2 Related Work

A systematic literature research indicates that reasons for a premature termination of treatment are still mostly unknown and corresponding literature are partly contradicting. Anderson [12] indicates that a socioeconomic deprivation is no indication for a premature termination, while Zimmermann [13] concluded that mostly male patients with a low educational level or high levels of histrionic features are in danger of dropping out. These two studies alone show that dropouts can hardly be divided into homogenous groups, which in turn makes classification difficult.

Some additional potential weak features could be the patient's age and therapists' experience. Anderson [12] indicated that patients under the age of 25 and those treated by inexperienced therapists have a higher risk of early termination. However, a strong indication could also be identified in the form of substance abuse or drug addiction. Anderson [12] and Gene [14] have introduced three categories for early termination of treatment:

1. Environmental obstacles, which can be logistic problems, e.g. no child care, financial problems, etc.
2. Problem improvement, when patients already feel better before completing the therapy and therefore think they do not need any further treatment.
3. Dissatisfaction with service, when patients are not satisfied with the advances in therapy and cannot make a sense of it anymore.

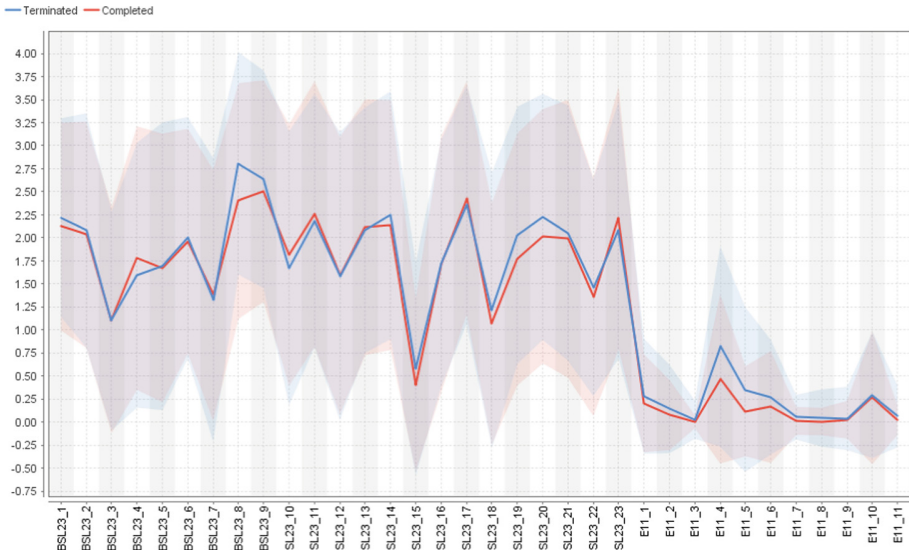
In summary, there is no clear statement for which features could predict a premature termination of therapy.

## 3 Predicting Premature Termination from Therapy Questionnaire Data

Our research focuses on the question whether premature termination of an established residential psychosocial treatment program (DBT) [1] in patients with BPD can be predicted from periodically collected questionnaire data. To concretize, we focus on the symptom questionnaires BSL-23 [11] including 11 items on problematic behavior (E11; e.g. self-injury) for BPD-patients receiving DBT [15]. We measure prediction performance by F1-measure [16] of the class treatment termination, based on a cross validation approach [17]. We consider prediction performance to be sufficient if F1-measure of the class termination is 0.8 or better.

The baseline for the analysis were 1159 BSL-23 and E11 questionnaires. These questionnaires originated from a total of 137 patients of which 92 completed the residential treatment program and 45 terminated treatment prematurely.

Figure 1 shows an overview of the average answer values of all BSL23 and E11 questions. Values are indicated as lines, the shaded areas show the standard deviation for each question. The graph indicates no clear distinction in the questionnaire answering behaviour of terminators and completers. However, it can be seen that some



**Fig. 1.** Average answer values to BSL23 and E11 questions by therapy terminators (blue) and completers (red), as well as the corresponding standard deviation for each question. (Color figure online)

questions show a higher distinction than others, e.g., BSL23-Q8 and E11-Q4 (i.e. binge-eating).

We employed the following state-of-the-art machine learning approaches for classifying patients as terminators or completers according to their questionnaire answering behaviour:

1. Logistic Regression classifier: A statistical model with one or more independent variables, with the goal to find the best fitting model to describe the relationship between the input and the outcome [18].
2. Linear Discriminant Analysis: A technique to reduce the dimensionality of a dataset from  $n$  to  $k$  ( $k \leq n-1$ ), while retaining the class discriminatory information [17].
3. Decision Tree: A method which learns simple decision rules inferred from data features in order to predict the value of a target variable [19].
4. AdaBoost classifier: A meta classifier, which starts with a single classifier, where each subsequent classifier's weights of incorrectly classified instances are modified; these classifiers focus on cases which are more difficult to classify [20].
5. Random Forest: An algorithm used to build multiple decision trees and merging them for a more accurate prediction [21].
6. Gaussian Naive Bayes: A probabilistic classifier based on Bayes' theorem with the assumption of independence between the input features [22].
7. C-Support Vector classification: A discriminative classifier using hyper-planes to categorize new samples and the option to minimize the misclassification for each training sample by an additional parameter  $C$  [23].



8. Multi-Layer Perceptron classifier: A classifier based on feedforward artificial neural networks with multiple layers of nodes with weights and bias, applying an activation function and employing backpropagation for the learning model [24–27].

Application of the above algorithms to the complete dataset with all features gave a maximum F-Score of 0.50 for the Logistic Regression algorithm. Since some of the questions show a higher distinction between completers and terminators, we performed a dimensionality reduction in order to identify the most significant features and apply the algorithms only to a selection of features available. Using Analysis of Variance [27], the ten most significant features were identified. In order of their significance (most to least) these were: E11-Q5, E11-Q4, BSL23-Q19, BSL23-Q8, E11-Q6, BSL23-Q23, BSL23-Q10, E11-Q7, E11-Q8 and BSL23-Q20.

Table 1 shows the resulting prediction performance following a cross validation, where 727 questionnaires labeled as completed and 179 questionnaires labeled as terminated were used for training. The remaining 157 questionnaires (completed) and 96 (terminated) were used for validation. Grouping the dataset by patient id and splitting it into a training and validation set was done via SciKit-Learn’s GroupShuffleSplit algorithm [28]. In order to compensate the majority of cases for the label completed, the decision tree, C-support vector and logistic regression classifier were weighted in two different ways: a weight of 1.0 on the class completed and a weight of 10 on the class terminated. The custom weight was computed by  $n\_samples/(n\_classes * np.bincount(y))$ .

The Logistic Regression classifier with fixed weight exhibits the best prediction performance (0.56) but is by far worse than the aspired F1 measure of 0.8 or better.

**Table 1.** Performance of evaluated classifiers with selected significant features.

Classifier	Class	Precision	Recall	F-Score
Logistic Regression Classifier	Terminated	0.17	0.01	0.02
C-Support Vector classification	Terminated	0.50	0.01	0.02
Linear Discriminant Analysis	Terminated	0.50	0.02	0.04
Decision Tree (Fixed Weight)	Terminated	0.27	0.09	0.14
Multi-Layer Perceptron classifier	Terminated	0.80	0.08	0.15
Random Forest	Terminated	0.52	0.12	0.20
K-Nearest Neighbor	Terminated	0.45	0.14	0.21
AdaBoost Classifier	Terminated	0.45	0.19	0.26
Decision Tree (Custom Weight)	Terminated	0.40	0.20	0.27
Gaussian Naive Bayes	Terminated	0.55	0.19	0.28
C-Support Vector classification (Custom Weight)	Terminated	0.43	0.30	0.36
Decision Tree	Terminated	0.57	0.28	0.38
Logistic Regression Classifier (Custom Weight)	Terminated	0.43	0.50	0.46
C-Support Vector classification (Fixed Weight)	Terminated	0.36	0.46	0.40
Logistic Regression Classifier (Fixed Weight)	Terminated	0.40	0.92	0.56

## 4 Private Digital Therapy Diary

### 4.1 Concept

In this section, we introduce an open-ended questionnaire which we call *Private Digital Therapy Diary (PDTD)*; For an overview on the concept of a diary used in psychotherapy see Thiele et al. [29]. Based on the method of ecological momentary assessment (EMA) [30], the aim of the PDTD is to assess the dynamic processes of a patient's subjective perception of their treatment in real time between therapy sessions [31]. Van de Leemput et al. [32] have recently shown that the pattern of EMA-recorded mood dynamics predicted the onset and termination of a depressive episode of patients and suggest the possibility to use such information within early warning systems for psychiatric disorders. In line with this research, the aim of our approach is to use the PDTD to identify the risk of premature termination of treatment and to create an application platform to realize and test an early warning system for therapists.

Patients are asked to fill in a smartphone-based electronic diary on a daily basis, as part of their patient application (see [10]). The use of the diary is voluntary and in the patient's own discretion. However, patients get reminded once a day by receiving a push notification. The content is private and is not communicated to the therapist. Therefore we assume that the patients will fill out the PDTD truthfully. The diary is structured in the following five questions, where any question may be skipped:

1. What would I tell the therapist team? (free text input)
2. What was special about today? (free text input)
3. I would like to continue/terminate the therapy (slider values 0..100)
4. What's my current mood? (slider values 0..100)
5. Is there anything that went well today? (free text input)

As diaries are not only relevant for data-assessment but might also support psychotherapy, patients can create and read an archive of their own diary entries [29].

Patients are asked to sign an informed consent form which allows further analysis of the anonymized data for research purposes and use of the data by the machine learning component of the software system in order to predict an imminent premature termination and potentially alert the therapist.

### 4.2 Research Questions

In order to figure out if the PDTD is a valid data source for predicting premature termination, we performed a detailed analysis of patients' answers with in respect to the following research questions:

1. Acceptance: Is the PDTD accepted by patients and is it filled out regularly, particularly PDTD-Q3 (continue/terminate therapy)?
2. Truthfulness: Are questions answered truthfully, particularly PDTD-Q3 (continue/terminate therapy)?
3. Prediction: Can premature termination of the therapy be predicted from PDTD data with sufficient prediction performance?

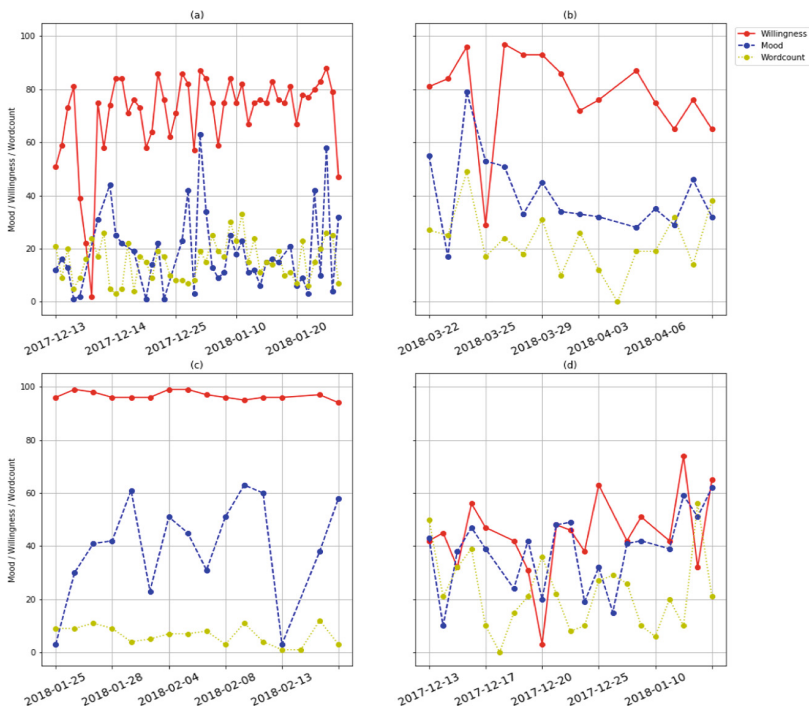
As a preliminary proof of concept we have performed the analysis on 14 patients with a total of 218 PDTD questionnaire entries. All patients have completed the therapy, but one patient has terminated the study, e.g. the use of the PDTD. Given the size of the collected dataset, no final assessment can be made at this stage. However, we have made several observations.

### 4.3 RQ1: Acceptance

We were pleased by the acceptance of the PDTD by patients. One patient filled out three questionnaires in one day. The number of average entries per week is 4.9. The maximum number of entries per week is 9. Individual questions were skipped rarely. In particular, PDTD-Q3 (continue/terminate therapy) was answered in 207 out of 218 diary entries. We conclude, the PDTD is accepted and used regularly by patients.

### 4.4 RQ2: Truthfulness

Figure 2a–d shows the sequence of PDTD questionnaire values of four anonymized example patients during therapy, in particular answers to PDTD-Q3 (red solid line) and PDTD-Q4 (blue dashed line) as well as the total word count in PDTD-Q1, 2, and 5 (yellow dotted line). In terms of truthfulness, we observed that patients who completed the therapy successfully indicated this particularly in PDTD-Q3 with high values,



**Fig. 2.** Examples for values of PDTD questionnaires during therapy.

usually above 80% (see Fig. 2c). Patients who are more impulsive and change their opinion about treatment in short time intervals, mostly still have values above 60% with an occasional dip below (see Fig. 2a, b, d). During the whole therapy we rarely observed dips that go below 30%, but if they do, normally below 10% (See Fig. 2a, d). These are usually one-time events where the patients might have had a particularly frustrating experience.

The current mood of the patient (PDTD-Q4) is mostly independent of the willingness to continue the therapy (PDTD-Q3), see Fig. 2c. This is expected, as the patient’s mood usually fluctuates, sometimes within a few hours.

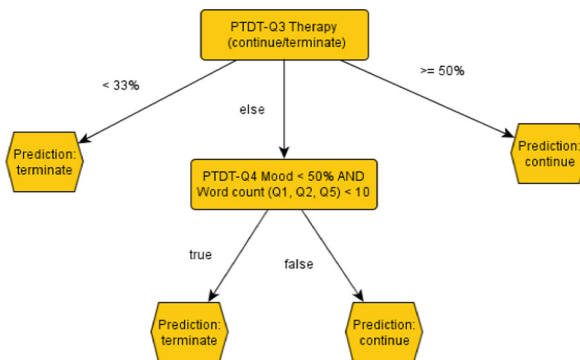
A sentiment analysis of the free text questions did not indicate any relation to PDTD-Q3 (continue/terminate). For example, one patient stated “demanding and overcharging; no perspective for change” and “I’m not going to make it” but answered continuously high values in PDTD-Q3 and eventually completed the therapy successfully.

The average word count is 14.2 words per diary questionnaire. We noticed that patients with a low value on PDTD-Q3 write no text or only short texts in PDTD-Q1, 2, and 5. This can be expected since a patient with low motivation for continuing the therapy will also have a low motivation in investing effort in diary writing.

From the data at hand we have the impression that patients answer the PDTD truthfully, which is expected due to the private nature of PDTD.

### 4.5 RQ3: Prediction

Based on our observations and analyses described above, we propose a simple *decision tree classifier* based on the PDTD data for predicting premature termination of therapy (see Fig. 3).



**Fig. 3.** Decision tree for predicting a possible premature termination.

The decision tree primarily takes PDTD-Q3 (continue/terminate) into account. If the value is  $< 33\%$  then a termination is predicted. If the value is  $\geq 50\%$ , continuation is predicted. If the value is in between, the value of PDTD-Q4 (Mood  $< 50\%$ ) and the word count of PDTD-Q1, Q2 and Q5 ( $< 10$ ) are additionally taken into account.

As the analysis has shown, sudden drops in values to PDTD-Q3 are common with a quick recovery to a high value. For this reason, we propose a *waiting period* of, e.g., 12 h. Whenever a patient enters a PDTD diary questionnaire which is classified “terminate” by the decision tree logic, then the waiting period is started. If at the end of the waiting period no additional diary questionnaire has been entered by this patient which has been classified “continue”, then the therapist is alerted on the imminent termination of this patient via an email message.

Applying the decision tree from Fig. 3 and a waiting period of 12 h to the 218 PDTD questionnaires collected to date, only one false alert would have been issued to the therapist. This is the patient shown in Fig. 2b who showed a drop in PDTD-Q3 values below 30 for two PDTD questionnaires  $\sim 25$  h apart from each other.

While this is a promising result, far more patient data needs to be acquired in order to fully answer RQ3 “Prediction”. This is subject to future work.

## 5 Conclusion and Future Work

We have analyzed a total of 1159 questionnaires of BSL-23 and 11 additional items on problematic behavior (E11) from 137 patients for possible features to predict premature termination of therapy. Using 15 state-of-the-art machine learning classifier approaches, a maximum F-score of only 0.56 could be achieved. This is far from sufficient and we conclude that premature treatment termination cannot be predicted from the amount of questionnaire data at hand such as BSL-23 and E11.

Based on those results, we introduced the concept of a private digital patient diary (PDTD), in which patients, among other questions, are explicitly asked about their willingness to continue the therapy. A preliminary analysis of 218 diary questionnaires from 14 patients indicate that patients accept the diary and fill it out truthfully. Based on this analysis we proposed a decision tree classifier for predicting premature treatment termination based on PDTD questionnaires. When combining the decision tree classifier with a waiting period, one false alert out of 218 PDTD questionnaires would have been issued to the therapist.

Since data from 14 patients is far too small to make sound statements of the prediction accuracy of our approach, we plan to collect large amounts of PDTD questionnaires in future research.

## References

1. Bohus, M., Haaf, B., Simms, T., Limberger, M., Schmahl, C., Unckel, C., et al.: Effectiveness of inpatient dialectical behavioral therapy for borderline personality disorder: a controlled trial. *Behav. Res. Ther.* **42**(5), 487–499 (2004)
2. Kröger, C., Röepke, S., Kliem, S.: Reasons for premature termination of dialectical behavior therapy for inpatients with borderline personality disorder. *Behav. Res. Ther.* **60**, 46–52 (2014)
3. Linehan, M.: *Cognitive-Behavioral Treatment of Borderline Personality Disorder*. Guilford press, New York (1993)

4. Rüschi, N., Schiel, S., Corrigan, P.W., Leihener, F., Jacob, G.A., Olschewski, M., et al.: Predictors of dropout from inpatient dialectical behavior therapy among women with borderline personality disorder. *J. Behav. Ther. Exp. Psychiatry* **39**(4), 497–503 (2008)
5. Lieb, K., Zanarini, M.C., Schmahl, C., Linehan, M.M., Bohus, M.: Borderline personality disorder. *Lancet* **364**(9432), 453–461 (2004)
6. Ogrodniczuk, J.S., Joyce, A.S., Piper, W.E.: Strategies for reducing patient-initiated premature termination of psychotherapy. *Harvard Rev. Psychiatry* **13**(2), 57–70 (2005)
7. Fernandez, E., Salem, D., Swift, J.K., Ramtahal, N.: Meta-analysis of dropout from cognitive behavioral therapy: magnitude, timing, and moderators. *J. Consult. Clin. Psychol.* **83**(6), 1108 (2015)
8. Sharf, J., Primavera, L.H., Diener, M.J.: Dropout and therapeutic alliance: a meta-analysis of adult individual psychotherapy. *Psychother. Theor. Res. Pract. Training* **47**(4), 637 (2010)
9. Ebner-Priemer, U.W., et al.: State affective instability in borderline personality disorder assessed by ambulatory monitoring. *Psychol. Med.* **37**(7), 961–970 (2007)
10. Bohus, M., et al.: Predicting premature termination of treatment in psychotherapy for borderline personality disorder. In: Bleimann, U., Humm, B., Loew, R., Regier, S., Stengel, I., Walsh, P. (eds.) *Proceedings of the Collaborative European Research Conference (CERC 2017)*, Karlsruhe, Germany, 22–23 September 2017, pp 168–177. ISSN: 2220-4164
11. Bohus, M., Kleindienst, N., Limberger, M.F., Stieglitz, R.D., Domsalla, M., Chapman, A.L., Wolf, M.: The short version of the Borderline Symptom List (BSL-23): development and initial data on psychometric properties. *Psychopathology* **42**(1), 32–39 (2009)
12. Anderson, K.N.: *Premature Termination of Outpatient Psychotherapy: Predictors, Reasons, and Outcomes*. Doctoral dissertation. University of Nebraska (2015)
13. Zimmermann, D., et al.: Therapist effects on and predictors of non-consensual dropout in psychotherapy. *Clin. Psychol. Psychother.* **24**(2), 312–321 (2017). ISSN: 1099-0879, <https://doi.org/10.1002/cpp.2022>
14. Gene, P.: Relationship of clients' reasons for dropping out of treatment to outcome and satisfaction. *J. Clin. Psychol.* **48**(1), 91–98 (1992). ISSN: 00219762. <https://doi.org/10.1002/1097-4679>
15. Linehan, M.M.: *DBT Skills Training Manual*, 2nd edn. Guilford Publications, New York (2015)
16. Chinchor, N.: MUC-4 evaluation metrics. In: *Proceedings of the Fourth Message Understanding Conference*, pp. 22–29 (1992)
17. Hastie, T., Tibshirani, R., Friedman, J.: *The Elements of Statistical Learning*. SSS, pp. 106–119. Springer, New York (2009). <https://doi.org/10.1007/978-0-387-84858-7>
18. Bishop, C.: *Pattern Recognition and Machine Learning*. Springer, New York (2006). ISBN: 978-0-387-31073-2
19. Breiman, L., Friedman, J., Olshen, R., Stone, C.: *Classification and Regression*. Taylor & Francis, Boca Raton (1984)
20. Zhu, J., Zou, H., Rosset, S., Hastie, T.: Multi-class AdaBoost. *Stat. Interface* **2**, 349–360 (2018)
21. Breiman, L.: *Machine Learning*, vol. 45, p. 5 (2001). <https://doi.org/10.1023/A:1010933404324>
22. Mitchell, T.: *Machine Learning*. McGraw-Hill Education, New York (1997). ISBN-13: 978-0071154673
23. Chang, C.-C., Lin, C.-J.: LIBSVM: a library for support vector machines. *ACM Trans. Intell. Syst. Technol.* **2**(3), 27 (2011). Article 27. <http://dx.doi.org/10.1145/1961189.1961199>
24. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: *International Conference on Artificial Intelligence and Statistics* (2010)

25. He, K., et al.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. arXiv preprint [arXiv:1502.01852](https://arxiv.org/abs/1502.01852) (2015)
26. Hinton, G.E.: Connectionist learning procedures. *Artif. Intell.* **40**(1), 185–234 (1989)
27. Kingma, D., Ba, J.: Adam: A method for stochastic optimization, arXiv preprint [arXiv:1412.6980](https://arxiv.org/abs/1412.6980) (2014)
28. Scikit-Learn, GroupShuffleSplit. [http://scikit-learn.org/stable/modules/generated/sklearn.model\\_selection.GroupShuffleSplit.html](http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GroupShuffleSplit.html). Accessed 14 May 2018
29. Thiele, C., Laireiter, A.R., Baumann, U.: Diaries in clinical psychology and psychotherapy: a selective review. *Clin. Psychol. Psychother.* **9**(1), 1–37 (2002)
30. Stone, A.A., Shiffman, S.: Ecological momentary assessment (EMA) in behavioral medicine. *Ann. Behav. Med.* **16**(3), 199–202 (1994)
31. Shiffman, S., Stone, A.A., Hufford, M.R.: Ecological momentary assessment. *Annu. Rev. Clin. Psychol.* **4**, 1–32 (2008)
32. van de Leemput, I.A., et al.: Critical slowing down as early warning for the onset and termination of depression. *Proc. Nat. Acad. Sci.* **111**(1), 87–92 (2014)



# Exploring the Usability of the Dice CAPTCHA by Advanced Statistical Analysis

Darko Brodić<sup>1</sup>, Alessia Amelio<sup>2</sup>, Ivo R. Draganov<sup>3(✉)</sup>, and Radmila Janković<sup>4</sup>

<sup>1</sup> Technical Faculty in Bor, University of Belgrade, Bor, Serbia  
dbrodic@tfbor.bg.ac.rs

<sup>2</sup> DIMES, University of Calabria, Rende, CS, Italy  
aamelio@dimes.unical.it

<sup>3</sup> Technical University of Sofia, Sofia, Bulgaria  
idraganov@tu-sofia.bg

<sup>4</sup> Mathematical Institute of the S.A.S.A., Belgrade, Serbia  
rjankovic@mi.sanu.ac.rs

**Abstract.** This paper introduces a new study of the Dice CAPTCHA usability based on advanced statistical analysis. An experiment is performed on a population of 197 Internet users, characterised by age and Internet experiences, to which the solution to the Dice CAPTCHA is required on a laptop or tablet computer. The response time, which is the solution time to successfully solve the CAPTCHA, together with the number of tries are registered for each user. Then, the collected data are subjected to association rule mining for analysing the dependence of the response time to solve the CAPTCHA in a given number of tries on the co-occurrence of the user's features. This analysis is very useful to understand the co-occurrence of factors influencing the solution to the CAPTCHA, and accordingly, to realise which CAPTCHA is closer to the "ideal" CAPTCHA.

**Keywords:** Dice CAPTCHA · Usability · Association rules  
Statistics analysis · FP-Growth

## 1 Introduction

Completely Automated Public Turing test to tell Computers and Humans Apart (CAPTCHA) is a program-based puzzle which is proposed to be a test that should be accurately solved by Internet users (humans) and hardly solved by computer programs. It is based on the "standard" Turing test, which inspects the ability of a machine (computer) to simulate the human behaviour. In this test, the questions are asked for the machine and human. The answers are evaluated by the judge, which is a human. If a machine can answer the questions like a human, it is said that it has an intelligence like a human, commonly called Artificial Intelligence (AI). Unlike the Turing test, the judge that evaluates the



correctness of the answers is exchanged in the CAPTCHA from the human to the machine (computer). Hence, CAPTCHA is commonly called “reverse” Turing test.

Many different types of CAPTCHA have been developed. However, many CAPTCHA types are lifted because of the low level of security in the practical work. The programs that simulate the human behaviour are called robot programs or bots. They integrate and incorporate different algorithms which can simulate the human behaviour during the CAPTCHA test. The successful algorithms that are integrated into the bots are Optical Character Recognition (OCR) algorithms, speech processing algorithms, etc. Hence, a text-based CAPTCHA and audio-based CAPTCHA proved to be unsecured.

Accordingly, the development of secured CAPTCHAs has moved into the areas where the computer algorithms are less effective than the humans. These areas are: (i) images, (ii) videos and (iii) puzzles. Among these areas, the most promising is the puzzle area. In recent times, different puzzle CAPTCHAs have been developed. Many of these puzzle CAPTCHAs integrate the image-based CAPTCHA in a special manner like an image puzzle. However, the real puzzle CAPTCHA does not integrate any image element. As every puzzle, the task to solve this CAPTCHA is not easy for the users. Hence, it takes more time to be solved. Still, it is an almost impossible task to be solved by the bots.

In this paper, we analyse one of the special puzzle-based CAPTCHAs called Dice CAPTCHA. It consists of two different Dice CAPTCHAs, called Dice CAPTCHA 1 and Dice CAPTCHA 2 [8]. Previous research proved good characteristics of the Dice CAPTCHAs using statistical analysis [4]. In particular, it explored the dependence of the Dice CAPTCHAs’ response time from single demographic factors of the users. Still, many of their characteristics are unexplored. To research these characteristics, we perform an experiment on 197 Internet users with different factors of age and Internet experience. Then, we measure the response time to successfully solve the two Dice CAPTCHAs in a given number of tries on laptop or tablet computer. This work extends the previous research given in [4] by analysing the dependence of the response time to correctly solve the Dice CAPTCHAs in a given number of tries on the co-occurrence of different factors of the users by employing the association rules. It is very useful to uncover the co-occurrence of factors influencing the solution to the CAPTCHAs, and accordingly, to find the CAPTCHA which is closer to the “ideal” CAPTCHA. The postulate of “ideal” CAPTCHA is to be solved in reasonable time (less than 30 s), and the response time should not depend on different personal factors [4].

The paper is organised as follows. Section 2 presents further previous works. Section 3 describes the Dice CAPTCHA. Section 4 presents the experimental part and association rule mining. Section 5 gives the results and discusses them. At the end, Sect. 6 draws the conclusions and outlines future work directions.

## 2 Related Work

In recent times, different works have been introduced in the literature for the analysis of the CAPTCHA types.

Wilkins [14] investigated some of the most common weaknesses of different CAPTCHAs, and proposed rotation and warping of individual characters, a proper selection of a font when designing strong puzzles, as well as avoiding using dictionary words, public puzzle sources and strong sources of random numbers. The study by Burzstein et al. [7] also proposed a set of design principles against automated solutions which included randomisation of the length of the CAPTCHA, character size, and waving of the CAPTCHA. The human accuracy was reduced by using complex charsets without a significant security enhancement, while in the case of breaking the CAPTCHA, it was advised to switch to another CAPTCHA scheme. Singh and Pal [10] classified the CAPTCHA in 5 categories and examined some drawbacks about the ability of the users to solve the CAPTCHA. Text-based CAPTCHAs are harder to be solved because some of the characters or whole words may be unreadable, while with the image-based CAPTCHAs, the blurring makes the images hard to identify. Audio-based CAPTCHAs are limited to the English language and may induce errors caused by similar sounding characters, while video CAPTCHAs take too much time to be solved. Lastly, the puzzle-based CAPTCHAs are harder to be solved because of the users' personal skills. Stark et al. [11] used a deep Convolutional Neural Network (CNN) and a custom generated database of test images from Cool PHP CAPTCHA to test the security of the CAPTCHA. They achieved an accuracy of 9.6% after an initial training with 10 000 images, and over 80% of accuracy after an additional training with 50 000 of correct and uncertain samples for  $10^6$  iterations. This study suggests that using only text in these systems may prove insufficient. Fidas et al. [9] investigated the necessity of user-friendly CAPTCHAs, and concluded that the currently operated CAPTCHAs are difficult to be solved. They revealed that the security is the most important factor for the respondents, while the character distortion is the main obstacle when solving the CAPTCHA, followed by the background patterns. An evaluation of different types of CAPTCHA from an aspect of effectiveness, applicability and limitations was given in [1] by Abdalla et al. They applied and compared two attack algorithms, probability pattern framework and divide and conquer the CAPTCHA, on text-based systems, with the highest success rate of 48%. Also, a previous study on the performance and accessibility of the CAPTCHA introduced a new method for the automatic prediction of the response time to solve the CAPTCHA, using a regression tree strategy [3]. In [5], image and text-based systems were investigated and seven hypotheses were analysed for the evaluation of the response time as a function of personal features. A similar analysis was performed in [6] with a primary focus on using tablets and online business systems, including e-banking.

### 3 The Dice CAPTCHA

The Dice CAPTCHA belongs to the area of the puzzle-based CAPTCHA. It requires a puzzle to be solved where the dice is visualised as the main element of the scene [8]. Accordingly, a puzzle related to the dice is asked to be solved in order to recognise if the user is a human or a bot. Only if the puzzle is correctly solved, the user is considered as a human.

Two specific types of Dice CAPTCHA have been proposed for protecting the websites from the bot's attacks: (i) Homo-sapiens Dice CAPTCHA (Dice 1), and (ii) All-the-rest Dice CAPTCHA (Dice 2) [8].

Dice 1 CAPTCHA requires to roll the dice and enter the sum of the digits which are visualised on the faces of the dice. Figure 1(a) shows a sample of Dice 1 CAPTCHA.

Dice 2 CAPTCHA consists in rolling the dice and enter the digits which are depicted on the faces of the dice as they are [8]. Figure 1(b) shows a sample of Dice 2 CAPTCHA.



**Fig. 1.** A sample of (a) Dice 1 CAPTCHA, and (b) Dice 2 CAPTCHA

### 4 The Proposed Study

The proposed study analyses the usability of solving the Dice 1 and Dice 2 CAPTCHAs on laptop or tablet computer from a population of Internet users. It is accomplished by investigating the dependence of the response time to successfully solve both CAPTCHAs in a given number of tries on the co-occurrence of some features of the users which solve the CAPTCHAs. This dependence is modelled by the association rules.

#### 4.1 Data Gathering

A population of 197 users is tested in real-life contexts, which require the solution of Dice 1 and Dice 2 CAPTCHAs on a laptop or tablet computer. The response time to correctly solve the CAPTCHAs (from the beginning until the end of the task), together with the number of tries, are registered for each user. This time is given in seconds.

Users take part voluntarily to this study. In particular, they have been informed that their data would be used anonymously for study purposes of the

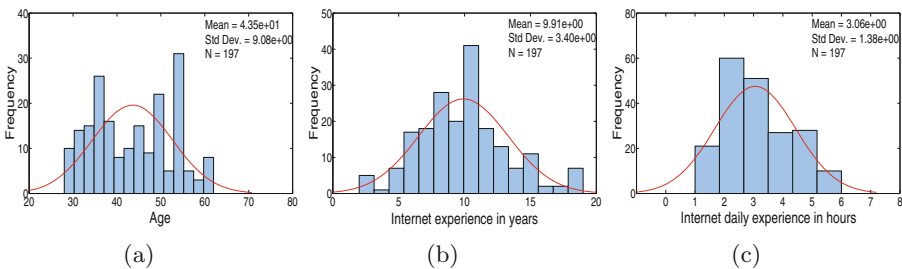
research team. Accordingly, they have agreed to an online consent form before starting the experiment. However, they did not know the aim of the study, neither the registered data, in order to avoid bias effects.

Users' data are stored into a database of 197 instances (tested users), each with the following 6 variables: (i) age, (ii) Internet experience per years of Internet use, (iii) Internet experience per daily hours of Internet use, (iv) type of device where the CAPTCHA is solved (tablet or laptop), (v) number of tries, and (vi) response time. One database is available for each of the two Dice CAPTCHAs. Data have been processed by a statistical tool which confirmed their statistical significance.

## 4.2 Data Analysis

The users adopted different devices to solve Dice 1 CAPTCHA and Dice 2 CAPTCHA. Accordingly, 100 out of 197 users employed a tablet device to solve the CAPTCHAs, while the rest of 97 users employed a laptop computer to solve the CAPTCHAs. Also, the users needed to successfully solve Dice 1 CAPTCHA and Dice 2 CAPTCHA. They had 3 tries to solve these CAPTCHAs. Dice 1 CAPTCHA is solved by 163 users in the first try, by 26 users in the second try, and by 8 users in the third try. By contrast, Dice 2 CAPTCHA is solved by 182 users in the first try, by 10 users in the second try, and by 5 users in the third try.

The total number of male users is 122 or 61.93%, while the rest of the 75 users are women or 38.07%. The age of the users is between 28 and 62 years. Their Internet experience per years of Internet use is in the range from 1 to 19. Their Internet experience per daily hours of Internet use is between 1 and 6.



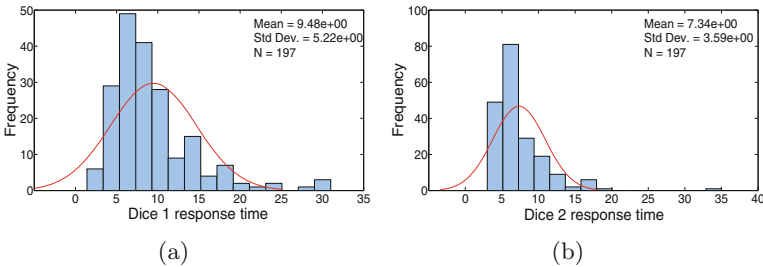
**Fig. 2.** Frequency histogram (a) of the users' age, (b) of the users and their Internet experience per years, and (c) of the users and their daily Internet use

The frequency histogram of the users' age is shown in Fig. 2(a). The frequency histogram of the users and their Internet experience per years is given in Fig. 2(b). It is worth noting that the distribution of the Internet experience in years of Internet use has a Gaussian like shape. Figure 2(c) illustrates the frequency histogram of the users and their daily hours of Internet use. It can be

noticed that the histogram has a shape which is slightly deviant from the ideal normal distribution.

The response time for Dice 1 CAPTCHA varies from 1.4 to 31 s. Hence, it is within the range of 29.60 s. Its distribution is given in Fig. 3(a). The highest number of users, i.e. 49 solve Dice 1 CAPTCHA in 8.00 s. Hence, the median is equal to 8.00 s, while the mean value is 9.48 s. Furthermore, Dice 1 CAPTCHA is typically solved by the users in 12.09 s on tablet, and in 6.78 s on laptop computers.

The response time for Dice 2 CAPTCHA varies from 3 to 35 s. Hence, it is within the range of 32 s. Its distribution is given in Fig. 3 (b). The highest number of users, i.e. 80 solve Dice 2 CAPTCHA in 6.00 s. Hence, the median is equal to 6.00 s, while the mean value is 7.34 s. Furthermore, Dice 2 CAPTCHA is typically solved by the users in 8.59 s on tablet, and in 6.04 s on laptop computers.



**Fig. 3.** Frequency histogram of the response time distribution for: (a) Dice 1 CAPTCHA, and (b) Dice 2 CAPTCHA

According to the results from the distributions, Dice 2 CAPTCHA has a lower response time than Dice 1 CAPTCHA, with most of the values <30 s.

### 4.3 Data Discretisation and Association Rules Extraction

In order to apply the association rule mining, the database variables have been discretised in ranges. In particular, the age is divided into typical intervals below 35 and above 35 years. The Internet experience per years of Internet use is split into four intervals: (i)  $(-\infty, 5]$  (low experience), (ii)  $(5-10]$  (middle experience), (iii)  $(10-15]$  (high experience), and (iv)  $(15, +\infty)$  (very high experience). The Internet experience per daily hours of Internet use is divided into three intervals: (i)  $(-\infty, 2]$  (low daily use), (ii)  $(2, 4]$  (middle daily use), and (iii)  $(4, +\infty)$  (high daily use). Finally, the response time of both Dice CAPTCHAs has the following five intervals: (i)  $(-\infty, 5.8]$  (very low), (ii)  $(5.8, 8.2]$  (low), (iii)  $(8.2, 13]$  (middle), (iv)  $(13, 22]$  (high), and (v)  $(22, +\infty)$  (very high).

The discretisation of the Internet experience is performed by the Equal-Width binning method [12], which naturally splits the data into intervals of

the same width. In order to discretise the response time, the K-Medians clustering method is used [2], which proved the best performances according to the extracted association rules. In terms of number of intervals, a test has been conducted by varying this number multiple times for the Internet experience and response time. At the end, the best number of intervals was selected which received the highest variability of response time and tries in the association rules.

The discretised database is subjected to association rule mining for exploring the dependence of the response time to correctly solve the Dice 1 and Dice 2 CAPTCHAs in a given number of tries on the co-occurrence of: (i) age, (ii) type of device, (iii) Internet experience per years of Internet use, and (iv) Internet experience per daily hours of Internet use. Each instance of the database is a transaction characterised of 6 items. Each item corresponds to the value of a distinct variable. From the transactions, the Association Rules (ARs) with support and confidence higher than or equal to an established threshold are extracted using the FP-Growth algorithm [13]. An AR is an implication  $W \rightarrow Z$ , representing the dependence of the consequent itemset ( $Z$ ) from the antecedent itemset ( $W$ ). The strength of the ARs is measured by: (i) support, (ii) confidence, and (iii) lift. Support represents the statistical significance of an AR. It is the ratio between the number of transactions including  $W \cup Z$  and the total number of transactions. Confidence measures the conditional probability of  $W$  given  $Z$ . It is the ratio between  $W \cup Z$  and the number of transactions only including  $W$ . Lift quantifies the predictability of  $Z$  given  $W$ .

## 5 Results and Discussion

The experimentation has been performed in Matlab R2016b on a laptop computer Quad-Core 2.2 GHz, with 16 GB RAM and UNIX operating system. A trial and error procedure is conducted by extracting the ARs with different values of support and confidence thresholds from 5% to 90%. The values of threshold are selected according to: (i) number of extracted ARs, (ii) variability of the response time and number of tries in the consequent of the ARs, and (iii) variability of the other features in the antecedent of the ARs. Accordingly, a combination of the parameters corresponding to a low number of ARs with high variability in the antecedent and consequent of the ARs is preferred, in order to capture the most meaningful patterns. In particular, the thresholds of minimum support and confidence are respectively 5% and 40%. At the end, the ARs are filtered in order to only keep those having in the consequent the response time and number of tries.

Tables 1 and 2 show the extracted ARs for Dice 1 and Dice 2 CAPTCHA in terms of antecedent and consequent. Also, the support, confidence and lift values are reported for each AR.

We can observe that Dice 2 CAPTCHA is more easily solved than Dice 1 CAPTCHA. In fact, many ARs of Dice 2 CAPTCHA are characterised by a very low response time in their consequent (see Table 2). On the contrary, most of the ARs of Dice 1 CAPTCHA include a low response time in their consequent

**Table 1.** Association rules for Dice 1 CAPTCHA

ID	Antecedent	Consequent	Supp	Conf	Lift
1	Above 35, Tablet, High Int. experience	1 t, Middle	0.06	0.44	1.99
2	Middle Int. experience	1 t, Low	0.24	0.45	1.34
3	Middle Int. experience, Middle Int. daily use	1 t, Low	0.13	0.52	1.55
4	Below 35	1 t, Low	0.11	0.44	1.31
5	Middle Int. experience, below 35	1 t, Low	0.08	0.53	1.60
6	Tablet, below 35	1 t, Low	0.08	0.42	1.26
7	Above 35, Middle Int. experience	1 t, Low	0.17	0.42	1.25
8	Middle Int. experience, Low Int. daily use	1 t, Low	0.09	0.45	1.34
9	Above 35, Middle Int. experience, Low Int. daily use	1 t, Low	0.06	0.43	1.29
10	Above 35, Middle Int. experience, Middle Int. daily use	1 t, Low	0.10	0.49	1.45
11	Laptop	1 t, Low	0.20	0.40	1.20
12	Middle Int. experience, Laptop	1 t, Low	0.17	0.51	1.54
13	Above 35, Middle Int. experience, Laptop	1 t, Low	0.14	0.48	1.44
14	Middle Int. experience,Laptop, Low Int. daily use	1 t, Low	0.06	0.50	1.49
15	Above 35, Middle Int. experience, Laptop,Low Int. daily use	1 t, Low	0.05	0.46	1.37
16	Above 35, Laptop	1 t, Very low	0.18	0.42	1.94
17	Above 35, Laptop,Low Int. daily use	1 t, Very low	0.08	0.40	1.83
18	Above 35, Middle Int. experience,Laptop,Low Int. daily use	1 t, Very low	0.05	0.42	1.91
19	Laptop, Middle Int. daily use	1 t, Low	0.10	0.45	1.36
20	Middle Int. experience,Laptop, Middle Int. daily use	1 t, Low	0.10	0.58	1.72
21	Laptop, High Int. experience	1 t, Very low	0.08	0.64	2.93
22	Above 35, Laptop,High Int. experience	1 t, Very low	0.07	0.67	3.05
23	Above 35, Laptop,Middle Int. daily use	1 t, Low	0.09	0.47	1.41
24	Above 35, Middle Int. experience, Laptop, Middle Int. daily use	1 t, Low	0.09	0.57	1.69

(see Table 1). Accordingly, it is easier to enter the digits which are shown on the faces of the dice instead of entering their sum.

It is worth noting that Dice 1 CAPTCHA can be more easily solved on the laptop computer than on the tablet computer. In fact, although both cases can be successfully solved in one try, they have a difference in the response time. In the first case, the ARs including the laptop are characterised by a low or very low response time.

In the second case, the ARs including the tablet have a middle-low response time (see Table 1). A similar condition can be observed for Dice 2 CAPTCHA, where users operating on the tablet take a low response time, while users working on the laptop take a very low response time (see Table 2). It clearly proves the difficulties of the users in solving the CAPTCHA on the tablet computer. An explanation for the difference in response time between tablet and laptop use can be found in the: (i) touchscreen, and (ii) smaller size of the screen in the tablet. In fact, the use of the touchscreen in the tablet can make more difficult the typing of the digits on the virtual keyboard for some categories of Internet users. Also, a screen of smaller size in the tablet can reduce the ability to recognise the numbers appearing on the dice.

We can notice that there is no clear statistically significant difference of the age groups in the response time to Dice CAPTCHA in a given number of tries. In fact, for Dice 1 CAPTCHA, there is one AR (rule 4) with age group below 35

years, while there is no rule with an age above 35 years. Hence, nothing can be deduced about the age difference. For Dice 2 CAPTCHA, the rules 4 and 7 show a difference in terms of response time between users below and above 35 years. However, their lift value is between 1.16 and 1.19. Hence, the predictability of the response time from the age is not high.

On the contrary, a difference can be noticed when the age is combined with other features, including the type of device and the Internet experience. In Dice 1 CAPTCHA, we can observe the rules 1 and 6, where the age difference on the tablet computer determines a difference in response time (users above 35 years take a middle time in solving the CAPTCHA) with a lift value up to 2.

**Table 2.** Association rules for Dice 2 CAPTCHA

ID	Antecedent	Consequent	Supp	Conf	Lift
1	Middle Int. experience	1 t, Low	0.22	0.41	1.11
2	Middle Int. experience,Tablet	1 t, Low	0.09	0.44	1.18
3	Tablet, Middle Int. daily use	1 t, Low	0.07	0.41	1.11
4	Below 35	1 t, Low	0.11	0.44	1.19
5	Middle Int. experience, below 35	1 t, Low	0.06	0.46	1.25
6	Tablet, below 35	1 t, Low	0.08	0.42	1.14
7	Above 35	1 t, Very low	0.32	0.43	1.16
8	Above 35, Low Int. daily use	1 t, Very low	0.15	0.45	1.21
9	Above 35, High Int. experience	1 t, Very low	0.10	0.42	1.11
10	Low Int. daily use,High Int. experience	1 t, Very low	0.05	0.40	1.06
11	Above 35, Middle Int. experience, Middle Int. daily use	1 t, Low	0.08	0.41	1.11
12	Above 35, Tablet, Middle Int. daily use	1 t, Low	0.05	0.42	1.12
13	Middle Int. experience	1 t, Very low	0.22	0.41	1.09
14	Middle Int. daily use	1 t, Very low	0.16	0.41	1.09
15	Middle Int. experience,Middle Int. daily use	1 t, Very low	0.11	0.44	1.16
16	Above 35, Middle Int. experience	1 t, Very low	0.20	0.51	1.35
17	Middle Int. experience,Low Int. daily use	1 t, Very low	0.10	0.47	1.26
18	Above 35, Middle Int. experience, Low Int. daily use	1 t, Very low	0.10	0.63	1.69
19	High Int. daily use	1 t, Low	0.08	0.42	1.14
20	Middle Int. experience,High Int. daily use	1 t, Low	0.05	0.58	1.56
21	Tablet, High Int. daily use	1 t, Low	0.06	0.43	1.16
22	Above 35, Middle Int. daily use	1 t, Very low	0.14	0.45	1.20
23	Above 35, Middle Int. experience, Middle Int. daily use	1 t, Very low	0.09	0.46	1.23
24	Laptop	1 t, Very low	0.27	0.55	1.45
25	Middle Int. experience,Laptop	1 t, Very low	0.18	0.54	1.45
26	Above 35, Laptop	1 t, Very low	0.26	0.60	1.60
27	Above 35, Middle Int. experience,Laptop	1 t, Very low	0.17	0.59	1.56
28	Laptop, Low Int. daily use	1 t, Very low	0.12	0.53	1.42
29	Above 35, Laptop,Low Int. daily use	1 t, Very low	0.12	0.57	1.53
30	Middle Int. experience,Laptop,Low Int. daily use	1 t, Very low	0.08	0.58	1.53
31	Above 35, Middle Int. experience,Laptop,Low Int. daily use	1 t, Very low	0.08	0.62	1.66
32	Laptop, Middle Int. daily use	1 t, Very low	0.13	0.57	1.51
33	Above 35, Laptop, Middle Int. daily use	1 t, Very low	0.12	0.63	1.68
34	Middle Int. experience, Laptop, Middle Int. daily use	1 t, Very low	0.09	0.54	1.45
35	Above 35, Middle Int. experience, Laptop, Middle Int. daily use	1 t, Very low	0.09	0.57	1.51
36	Laptop, High Int. experience	1 t, Very low	0.08	0.60	1.60
37	Above 35, Laptop, High Int. experience	1 t, Very low	0.08	0.71	1.90



A difference can also be observed within the same age group above 35 years, where a high number of years in Internet use determines a very low response time, while a middle number of years determines a low response time in solving the CAPTCHA on the laptop computer (see rules 13 and 22 with high lift value up to 3 and confidence up to 0.67). It is worth noting that a long daily Internet use does not support the users above 35 years in quickly solving the CAPTCHA in one try on the laptop computer (see rules 18 and 24 where a low Internet daily use determines a very low response time, while a middle Internet daily use determines a low response time). In Dice 2 CAPTCHA, users above 35 years operating on the laptop computer are not influenced by the Internet experience in years or in daily hours. It can be observed from the rules 27, 29, 33, and 37 where the difference in Internet experience determines a very low response time in all cases. Hence, in Dice 1 CAPTCHA, the long term Internet experience (years of use) instead of the short term one (daily use) has an influence in quickly solving the CAPTCHA. On the contrary, in Dice 2 CAPTCHA, both types of experience have a low influence on the response time.

From the extracted ARs, we can summarise as follows: (1) Dice 2 CAPTCHA is more easily solved than Dice 1 CAPTCHA, (2) both Dice 1 and 2 CAPTCHAs are more easily solved on the laptop than on the tablet computer, (3) there is no statistically significant difference of the age groups in the response time to Dice CAPTCHA in a given number of tries, (4) age groups operating on the tablet computer show a statistically significant difference in response time for Dice 1 CAPTCHA, (5) a long Internet experience in years of use reduces the response time to solve the Dice 1 CAPTCHA on the laptop computer, while it has no influence on the response time to solve the Dice 2 CAPTCHA for the age group above 35 years, (6) a long daily Internet use does not support the users above 35 years in quickly solving the Dice 1 and 2 CAPTCHAs in one try on the laptop computer.

## 6 Conclusions

According to this analysis, we can conclude that Dice 2 CAPTCHA is solved in less time <30 s than Dice 1 CAPTCHA. Also, Dice 2 CAPTCHA is less influenced by age and Internet experiences than Dice 1 CAPTCHA. Hence, Dice 2 CAPTCHA is closer to the postulate of “ideal” CAPTCHA than Dice 1 CAPTCHA. Still, effort is needed for designing a Dice CAPTCHA which could be more independent from the users’ abilities. This analysis proved that Dice CAPTCHA is not still easily solved on the tablet computer. Also, this study showed the main feature combinations which influence the response time to successfully solve the Dice CAPTCHA in a given number of tries. It was accomplished by the association rule mining, which exploited the dependence of successfully solving the Dice 1 and 2 CAPTCHAs in a given number of tries from the co-occurrence of age, type of device and Internet experiences.

Future work will create an artificial neural network model for predicting the response time to successfully solve the Dice CAPTCHA from input factors

of the users. Since the Dice 2 CAPTCHA is less influenced by personal and demographic factors of the users, we expect to obtain a lower predictability of its response time than the Dice 1 CAPTCHA.

**Acknowledgments.** The authors are fully grateful to the voluntary participants for anonymously providing their data.

This work was supported by the Ministry of Education, Science and Technological Development of the Republic of Serbia (Project TR33-037) and through Mathematical Institute of the Serbian Academy of Sciences and Arts (Project III44006).

This work is dedicated to Professor Darko Brodić with full gratitude.

## References

1. Abdalla, K., Kaya, M.: An evaluation of different types of CAPTCHA: effectiveness, user-friendliness, and limitations. *Int. J. Sci. Res. Inf. Syst. Eng.* **2**(3) (2017)
2. Bradley, P. S., Mangasarian, O. L., Street, W. N.: Clustering via concave minimization. In: *Advances in Neural Information Processing Systems*, vol. 9, pp. 368–374. MIT Press (1997)
3. Brodić, D., Amelio, A., Ahmad, N., Shahzad, S.K.: Usability analysis of the image and interactive CAPTCHA via prediction of the response time. In: Phon-Amnuaisuk, S., Ang, S.-P., Lee, S.-Y. (eds.) *MIWAI 2017. LNCS (LNAI)*, vol. 10607, pp. 252–265. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-69456-6\\_21](https://doi.org/10.1007/978-3-319-69456-6_21)
4. Brodić, D., Amelio, A., Draganov, I.R.: Statistical Analysis of Dice CAPTCHA Usability. *CoRR abs/1706.10177* (2017)
5. Brodić, D., Amelio, A., Janković, R.: Exploring the influence of CAPTCHA types to the users response time by statistical analysis. *Multimedia Tools Appl.* **77**(10), 12293–12329 (2017)
6. Brodić, D., Janković, R.: Usability analysis of the specific CAPTCHA types. In: *International Scientific Conference UNITECH*, 18–19 November, Gabrovo, Bulgaria, pp. II-272–II-277 (2016)
7. Bursztein, E., Martin, M., Mitchel, J.: Text-based CAPTCHA strengths and weaknesses. In: *18th ACM Conference on Computer and Communications Security*, 17–21 October, Chicago, IL, USA, pp. 125–138. ACM (2011)
8. Dice CAPTCHA. <http://dice-captcha.com>
9. Fidas, C. A., Voyiatzis, A. G., Avouris, N. M.: On the necessity of user-friendly CAPTCHA. In: *SIGCHI Conference on Human Factors in Computing Systems*, 7–12 May, Vancouver, BC, Canada, pp. 2623–2626. ACM (2011)
10. Singh, V.P., Pal, P.: Survey of different types of CAPTCHA. *Int. J. Comput. Sci. Inf. Technol.* **5**(2), 2242–2245 (2014)
11. Stark, F., et al.: CAPTCHA recognition with active deep learning. In: *GCPR Workshop on New Challenges in Neural Computation*, 10 October, Aachen, Germany. LNCS, vol. 9358. Springer (2015)
12. Sullivan, D.G.: Data Mining V: Preparing the Data. [http://cs-people.bu.edu/dgs/courses/cs105/lectures/data\\_mining\\_preparation.pdf](http://cs-people.bu.edu/dgs/courses/cs105/lectures/data_mining_preparation.pdf)
13. Tan, P.-N., Steinbach, M., Kumar, V.: *Introduction to Data Mining*, 1st edn. Addison-Wesley Longman Publishing Co., Boston (2005)
14. Wilkins, J.: Strong CAPTCHA guidelines v1.2, p. 8 (2009). Accessed Nov 2010. <http://www.123seminaronly.com/Seminar-Reports/008/47584359-captcha.pdf>



# Time Series Analysis for Sales Prediction

Costin-Gabriel Chiru<sup>1,2(✉)</sup> and Vlad-Valentin Posea<sup>1</sup>

<sup>1</sup> University Politehnica Bucharest,  
Splaiul Independenței 313, Bucharest, Romania  
{costin.chiru, vlad.posea}@cs.pub.ro  
<sup>2</sup> Vivre Deco S.A., Bulevardul Tudor Vladimirescu nr  
22 Green Gate Office, etaj 7, Bucharest, Romania  
costin.chiru@vivre.eu

**Abstract.** In this paper, we present an approach to forecasting the number of paintings that will be sold daily by Vivre Deco S.A. Vivre is an online retailer for Home and Lifestyle in Central and Eastern Europe. One of its concerns is related to the stocks that it needs to make at its own warehouse (considering its limited available space) to ensure a good product flow that would maximize both the company profit and the users' satisfaction. Since stocks are directly connected to sales, the purpose is to predict the amount of sales from each category of products, given the selling history of these products. Thus, we have chosen a category of products (paintings) and used ARIMA for obtaining the predictions. We present different considerations regarding how we chose the model, along with the solver and the optimization method for fitting ARIMA. We also discuss the influence of the differencing on the obtained results, along with information about the runtime of different models.

**Keywords:** Time series analysis · Sales prediction · Auto-regression  
Moving-average · Differencing · Lags · ARIMA · FBProphet

## 1 Problem Description

Vivre is a leading online retailer for Home and Lifestyle in Central and Eastern Europe, currently activating in 8 countries from this area. They are offering their customers both limited-time discounts (“flash sales”) called “campaigns” in which are grouped products from similar fields (such as “entrance and bath mats”, “sun glasses” or “LEGO accessories”) and long-term available products, which are grouped in the “product catalog”.

Since 2012, when it was launched, Vivre accumulated an important quantity of historical data regarding sales, providers and customers, information that could be used to improve the company's activity.

The problem described in this paper derived as an attempt to forecast the daily quantity from each product to be sold by Vivre. These numbers could be very valuable for the company, as they would help maintain in the warehouse only the right stock from each product, under the constraints imposed by the restricted storage area. Another benefit would be that of allowing faster products delivery towards clients, (as the products are readily available in the warehouse and thus one does not have to also

wait for their arrival from the providers,) which can be translated in increasing the customer satisfaction and a better positioning of the brand on the market. Finally, good estimation of the daily sold quantities of each product could help improve the flow of the incoming and outgoing deliveries for the products, thus optimizing the usage of the warehouse and eliminating the time lost when trailers are available but there is no storage in the warehouse or there are not enough products to be shipped. This may finally translate in larger profit for the company.

However, even though these numbers are extremely important, it is not trivial to derive them even when enough historical data is available, due to several factors. First, there are over 1 million different product ids in the database, making the task of predicting the daily sales for all of them time and resource consuming, especially considering this a recurrent (daily) task. Secondly, at any moment there are not more than about 50–60.000 products available on the website and thus, predicting the sale of any of the other products would be meaningless, as they cannot be seen by the user (and therefore cannot be bought by them). Thirdly, among all the products, there are numerous that are/were available on the website only for a very limited amount of time (during a campaign, for less than 30 days for a history spreading for more than 6 years), thus not having enough information for predicting their sales. Fourthly, there are products that may be replaced one for the other (e.g. products having different colors or sizes, but serving to the same purpose) and thus their sales are tightly connected. Finally, the sale of a product is highly influenced by the seasonality, marketing budget, existing campaigns on the website, providers of the goods that are available for buying and many other factors, some of them being logged in the system, while for others not having any information at all.

Considering all the above problems, a design decision has been made with the purpose to alleviate some of the issues: instead of trying to forecast the daily quantity to be sold for each product from the database, first a grouping over the similar products was made and then the forecast of the daily quantity to be sold was generated for each such group (called “generic name”). This decision was intended to solve the first four issues, as it seriously reduced the number of (group of) products for which predictions should be provided (from over a million to around 27,000). Moreover, it solved the problem of products not being available on the website, since all the obtained groups had at least one representative on the site. Finally, since similar products were grouped in the same generic name, the problem with rarely available products and with the replaceable products was also solved as they were simply part of the same larger distribution. Another effect of this decision was to diminish the scarcity and variability in the data, which might lead in the end to better predictions. On the other hand, there was also a drawback since, by joining the information related to multiple products in the same generic name, after prediction, the obtained data should be disentangled to obtain the quantities for the initial products.

Thus, the task described in this paper is to forecast the daily quantity to be sold for each generic name that was identified starting from the product ids from the database. Considering the historical data that was aggregated for each such generic name, we used time series analysis (that is described in the next section) for obtaining the predictions. Section 3 presents a use case scenario for one of the generic names that was used, along with the problems and decisions that were made. The obtained results

are presented in Sect. 4, while the paper concludes with our observation regarding the used methodology and with some proposed directions for extending this research.

## 2 Time Series Analysis and Some of Its Applications

A time series is “a set of well-defined data items collected at successive points at uniform time intervals” [1]. Time series analysis represents a class of methods for processing these items to determine the main patterns of the dataset, which then may help predicting future values based on the identified patterns. One of the simplest method from this class is called autoregressive (AR) model and “specifies that the output variable depends linearly on its own previous values” [2]. The notation AR(p) – autoregressive model of order p – means that the current value depends on the previous p values of the time series. The expression of AR(p) is given by (1), where  $\gamma_1, \gamma_2, \dots, \gamma_p$  are the parameters of the model, c is a constant, and  $\varepsilon_1, \varepsilon_2, \dots, \varepsilon_t$  are white noise error terms.

$$X_t = c + \sum_{i=1}^p \gamma_i X_{t-i} + \varepsilon_t \quad (1)$$

In 1951, the AR model was extended by Whittle [3] into the autoregressive-moving-average (ARMA) model, which had two parts: an autoregressive (AR) part, consisting on the regression of the variable on its own past values, and a moving average (MA) part, modelling the error term as a linear combination of the current error term, along with some of the previous ones. The new model, depicted as ARMA(p, q) represents a model with p autoregressive terms and q moving-average terms, given by (2), where  $\gamma_1, \gamma_2, \dots, \gamma_p$  are the AR model parameters,  $\theta_1, \theta_2, \dots, \theta_q$  are the MA model parameters, c is a constant, and  $\varepsilon_1, \varepsilon_2, \dots, \varepsilon_t$  are white noise error terms:

$$X_t = c + \varepsilon_t + \sum_{i=1}^p \gamma_i X_{t-i} + \sum_{i=1}^q \theta_i \varepsilon_{t-i} \quad (2)$$

This new model was further improved by Box and Jenkins [4] to obtain the Autoregressive Integrated Moving Average (ARIMA) model. The difference between ARMA and ARIMA is that the latter’s first step is to convert a non-stationary data to a stationary one by replacing the actual data values with the difference between these values and previous ones (process called “differencing”, that may be performed multiple times). The ARIMA model has three parameters ARIMA(p, d, q), where p is the autoregressive order, d is the degree of differencing (the number of times the data have had past values subtracted) and q is the order of the moving-average model. Its formula is the same as in (2), with the only difference that instead of having the actual values  $X_i$ , we work with the difference between  $X_i$  and past values. One observation that should be made is that ARIMA(p, 0, q) represents in fact the ARMA(p, q) model, while ARIMA(p, 0, 0) represents the AR(p) model.

Being given the 3 parameters ( $p$ ,  $d$ ,  $q$ ) and the actual data  $X$ , the ARIMA model uses the Box-Jenkins method [4] to find the best fit of a time-series model to past values of a time series. Later on, the parameters identified during fitting may be used to generate predictions on the future behavior of the time series. However, choosing the best values of  $p$  and  $q$  is not easy. Several options were proposed by the researchers. One option was proposed by Brockwell and Davis [5] who state that “our prime criterion for model selection will be the AICc”, which stands for the Akaike information criterion with correction [6]. Another option was given by Hyndman & Athanasopoulos [7] who suggest how to automatically determine both the values of  $p$  and  $q$  using ACF (autocorrelations function) and PACF (partial autocorrelations function) plots.

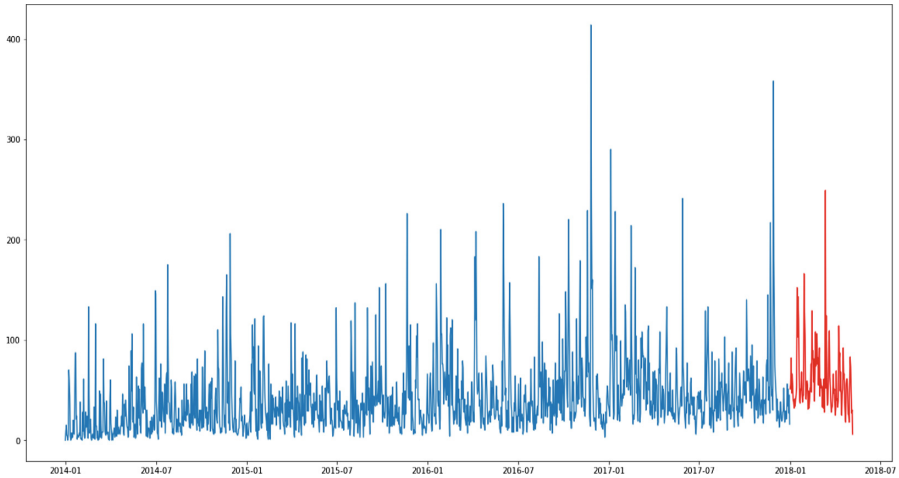
Since their inception, the ARIMA models were used to make predictions in various fields: from estimating the monthly catches of pilchard from Greek waters [8], to forecasting the Irish inflation [9], to predicting next-day electricity prices in Spain and Californian markets [10], to estimating the incidence of hemorrhagic fever with renal Syndrome (HFRS) in China during 1986–2009 [11], to foretelling the sugarcane production in India [12], and finally to prognosticate the energy consumption and greenhouse emission of a pig iron manufacturing organization [13]. However, many of the ARIMA uses were in the field of stock forecasting [14, 15], where they were trying to find the best parameters for estimating the stock prices of a particular stock.

In the following section, we will present a case study of applying different ARIMA models for predicting not the stock prices, but the quantities to be sold from different groups of products by Vivre. We could have tried to estimate the amount of sales, but since the product prices vary a lot in time, we decided to have a more stable estimation and opted out for the product quantities.

### 3 Case Study and Obtained Results

In this section, we will present the experiments that we undertook using the above-mentioned models with the purpose to predict the daily quantity to be sold for one of the generic names that were identified starting from the product ids from the database. Thus, we have chosen the “painting” generic name, having the distribution of daily quantities sold presented in Fig. 1. We chose this generic name because of two reasons: there was enough data available to enable predictions (only few days had zero-counts) and the distribution has some seasonality, but in the same time features some spikes that could be interpreted as outliers. Given this distribution, our task was to use the historical data from January 1<sup>st</sup>, 2014 to December 31<sup>st</sup>, 2017 (1461 training samples) for training the model, and the rest of the data (from January 1<sup>st</sup> to May 7<sup>th</sup>, 2018 – 127 samples) for testing the quality of the forecasting. The quality of the trained models was tested using 3 different values: RMS (root mean square error) for both the training and testing sets and MAPE (mean absolute percentage error).

Therefore, we started building several ARIMA models, with different parameters, and tested the forecasting accuracy on each of them. To start with, we used AR with different orders ( $p$ ) to generate some initial predictions. The values of  $p$  that were used in these tests were influenced by some basic assumptions regarding the data: we



**Fig. 1.** The distribution of the daily quantity sold for the generic name “painting” by Vivre Deco. The values from Jan 2014 to Jan 2018 were used for training, while the ones from Jan 2018 to May 2018 were used for testing.

assumed that the current data might be influenced by the value from the previous day (AR(1)), by the ones from the previous week (AR(7)), month (AR(31)), or year (AR(365)). The next step was to use various differencing to see if the data seasonality improves the results or not. However, since the results of AR(365) were very poor, we stopped using it for tests. Thus, we tested monthly (differencing of 31), weekly (differencing of 7) and daily seasonality (differencing of 1) combined with the remaining AR orders. We should mention that in all our experiments, by differencing of  $d$ , we understand single differencing (not stacked differencing), with lags =  $d$  (instead of  $X_i - X_{i-1}$ , we used  $X_i - X_{i-d}$ ). The obtained results are reported in Table 1.

**Table 1.** Results obtained using the AR model. The first value represents the order of the AR model, while the second represents the differencing that was used (e.g. 31, 7 means AR(31), applied not on the real values, but on the difference  $X_i - X_{i-7}$ ). A value of 0 means that no differencing was used (the real values were used to train the AR model).

Error	365, 0	31, 0	7, 0	1, 0	31, 31	31, 7	31, 1	7, 7	7, 1	1, 1
RMS	68.3	38.77	40.82	43.33	58.3	57.74	83.24	39.56	54.37	<b>37.13</b>
MAPE	109.7	42.06	<b>40.89</b>	45.26	79.3	72.04	169.2	56.44	88.73	41.55

Afterwards, we moved on to ARMA and tested models with different  $p$  (2–7) and  $q$  (1, 2, 3). To estimate the best values of  $p$  and  $q$ , we used the Hyndman & Athanapoulos [7] methodology based on ACF and PACF (see Table 2). Since ARMA could use during training multiple solvers (lbfgs – limited memory Broyden-Fletcher-Goldfarb-Shanno; newton – Newton-Raphson; nn – Nelder-Mead; cg – conjugate gradient; ncg – non-conjugate gradient; and powell) along with 3 different methods

**Table 2.** Results obtained using the ARMA model. Except for the last column, the results were obtained using an ARIMA(p, 0, q) model, where p = the first value (AR order), q = the second value (MA order). The last represents the differencing that was used. (e.g. (3, 2), 365 means ARIMA(3, 0, 2), applied not on the real values, but on the difference  $X_i - X_{i-365}$ ). A value of 0 means that no differencing was used (the real values were used to train the AR model). The last column corresponds to the results obtained using ARIMA(5, 1, 1).

Error	(7, 2), 0	(6, 2), 0	(5, 2), 0	(4, 2), 0	(3, 2), 0	(7, 1), 0	(5, 1), 0	(3, 2), 365	(4, 3), 7	(5, 1, 1)
RMS Train	<b>33.53</b>	33.68	33.66	33.97	33.98	33.57	33.83	46.49	48.1	65.3
RMS 1 step	38.52	38.48	36.1	38.46	38.47	38.57	35.96	57.76	51.77	<b>34.17</b>
MAPE 1 step	38.73	38.69	<b>31.5</b>	38.76	38.79	38.75	39.21	73.55	59.48	43.87
RMS iter	31.94	32.32	<b>26.7</b>	32.65	32.66	32.02	31.58	74.02	77.27	31.58
MAPE iter	32.86	33.41	<b>24.22</b>	33.8	34.77	33.23	39.82	97.22	144.66	39.82
Converged	yes	yes	<i>no</i>	yes	yes	yes	yes	yes	yes	yes
Converged iter	yes	<i>no</i>	<i>no</i>	<i>no</i>	yes	<i>no</i>	<i>no</i>	<i>no</i>	<i>no</i>	<i>no</i>
ACF	5	5	5	5	5	5	5	3	4	5
PACF	2	2	2	2	2	2	2	2	3	2
lags	31	31	31	31	31	31	31	365	7	31
Predictions #	127	127	<u>27</u>	127	127	127	127	<u>114</u>	127	127

(css – maximize the conditional sum of squares likelihood; mle – maximize the exact likelihood via the Kalman Filter; and css-mle – maximize the conditional sum of squares likelihood and then these values are used as starting values for the computation of the exact likelihood via the Kalman filter) for determining the model parameters, we used a grid search to find the best combination of the solver and the method for fitting the most promising ARMA model (ARMA(7, 0, 2)). The results are reported in Table 3.

Finally, we used the Augmented Dickey–Fuller test to verify if our time series was stationary, to find out if differencing could improve the results. Even though the test showed that the series was stationary, meaning that integration will not help, we still verified a couple of integrated models (with  $d = 1$ ) to see the seasonality influence on the reference values that were used. The results are presented in Table 4. In most of these tests the differencing was made explicitly before running ARIMA, and thus the value of  $d = 0$ . However, we also made a test on a random combination of p and q (5, 1) to see if the implicit integration of ARIMA(5, 1, 1) would yield different results. These are reported in Table 2.

For each of the above models, we used the ARIMA method from the statsmodels.tsa.arima\_model.ARIMA package from python. This package offered two types of predictions, and we used both in our experiments: forecast, that could predict the values for each day from the testing set in a single run based on the parameters that were obtained after training; and predict, that was predicting only the next value in the time series and was run iteratively for each day from the testing set, interleaved with



**Table 3.** Solver and method influence on the results obtained using the ARIMA(7, 0, 2) model.

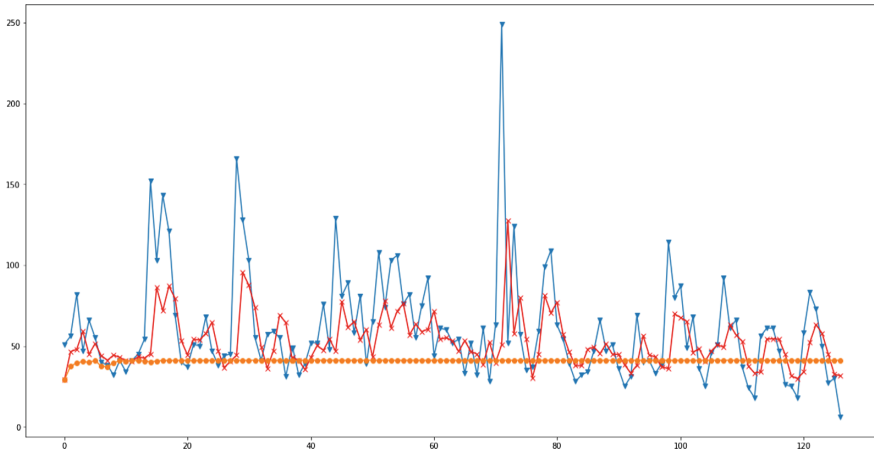
Solver	Method	Time	RMS	MAPE	Converged	Converged iter	Predictions
lbfgs	css-mle	377.23	31.94	32.86	yes	yes	127
	mle	365.52	31.94	32.86	yes	yes	127
	css	45.5	<b>31.91</b>	32.88	yes	yes	127
newton	css-mle	633.2	31.94	32.86	yes	yes	127
	mle	961.88	31.94	32.86	yes	yes	127
	css	78.65	<b>31.91</b>	32.88	yes	yes	127
nm	css-mle	200.06	31.95	<b>32.81</b>	<i>no</i>	<i>no</i>	127
	mle	131.72	31.97	32.94	<i>no</i>	<i>no</i>	127
	css	30.83	31.91	32.93	<i>no</i>	<i>no</i>	127
cg	css-mle	1197.41	31.94	32.88	<i>no</i>	<i>no</i>	127
	mle	923.67	31.94	32.87	<i>no</i>	<i>mostly</i>	127
	css	96.39	31.92	32.85	<i>no</i>	<i>no</i>	127
ncg	css-mle	1099.61	31.94	32.86	yes	yes	127
	mle	1168.33	31.94	32.86	yes	yes	127
	css	171.38	<b>31.91</b>	32.88	yes	yes	127
powell	css-mle	208.18	31.93	32.83	yes	yes	127
	mle	139.3	31.97	32.96	yes	yes	127
	css	31.61	31.94	33.01	yes	yes	127

**Table 4.** Lag influence the results obtained using the ARIMA(7, 0, 2) model.

Lags	0	365	31	7	1
RMS Train	33.57	45.74	48.77	39.54	33.36
RMS 1 step	38.57	57.78	52.56	71.66	68.87/72.88
MAPE 1 step	38.75	73.62	61.64	106.4	100
RMS iter	32.02	54.62	64.74	45.65	62.15
MAPE iter	33.23	68.98	107.91	80.63	99.2
Converged	yes	yes	yes	yes	yes
Converged iter	no	yes	yes	yes	no
ACF	5	3	4	4	1
PACF	2	2	2	3	1
lags	31	365	31	7	1
Predictions #	127	127	127	127	<u>38</u>

re-fitting the model using the true value that was observed for the previous day. The iterative method always yielded better results than the 1-step forecasting, as it used more information. The results of the best model (ARIMA(7, 0, 2)) using powell solver and css-mle optimizing method are presented in Fig. 2.

Since some of the trained models did not converge (such as ARIMA(5, 0, 2)), we also report this fact, along with the phase when they failed to converge (during training or iterative testing). If the model did not converge during the iterative testing, it was



**Fig. 2.** The best results obtained by ARIMA(7, 0, 2). The triangles represent the original values, the circles depict the 1-step forecasting and the ‘x’-s show the iterative predictions.

unable to generate the predictions for all the 127 samples from the testing set and thus we also report the number of generated predictions.

Finally, as the computation that is done for this generic name (“painting”) should be done for all the generic names, it means that the whole process should be repeated a couple thousand times. Thus, another important element is the time needed to obtain the predictions and therefore, for the most promising runs, this information is also presented.

Besides the ARIMA models, we also tested the FBProphet [16], which is a tool “for producing high quality forecasts for time series data that has multiple seasonality with linear or non-linear growth”. Prophet may be run with or without daily seasonality. Both methods generated very similar predictions (RMS 1 step 33.18/33.21; MAPE 1 step 37.85/37.77; RMS iterative 32.26/32.25; MAPE iterative 41.29/41.35; runtime 1668 s/1380 s), which were poorer than the ones obtained using ARIMA(7, 0, 2). We also tried to model the time series spikes using a different distribution with the help of the holiday option from FBProphet, but the results didn’t improve much (RMS 1 step 33.04; MAPE 1 step 37.43), remaining worse than the ones of ARIMA.

## 4 Discussion and Conclusion

The AR results showed that the best model was the one involving the previous 7 days. They also revealed that, except for AR(1) with daily seasonality, including differencing according to different seasonality worsen the results.

The table presenting the ARMA scores shows that the methodology based on ACF and PACF does not work in our case, the model with  $p$  and  $q$  generated by the ACF and PACF being the only one that did not converge during training (ARIMA(5, 0, 2)). Moreover, all the models with  $p$  and  $q$  chosen this way did not converge during testing.

Although ARIMA(5, 0, 2) seems to have the best results (MAPE 24.22), it should be noticed that the model managed to provide only 27 predictions out of 127 required. The best model was ARIMA(7, 0, 2) which converged during both training and testing.

The investigation of different solvers and optimization methods showed that they have a very small influence on the obtained results. However, since some of the methods did not converge, for further experiments was chosen the combination leading to the second-best result (powell solver, css-mle method). This combination also had a reduced running time, which counts when the process has to be repeated 27,000 times.

Finally, choosing different lags only worsen the results, showing that the best solution is to work directly with the data, without differencing.

Even though the obtained results are promising, some of them being even better than FBProphet's ones, we believe that there are still ways to improve them. They were obtained using only historical information about the daily sold quantities of a single generic names. Still, similar information is available for the other generic names, and could be used to improve the predictions, as the sale of some products also influence the sale of others. However, to use this additional information, the ARIMA model must be changed for one that allows multi-variate dependencies. If such a change happens, other additional information may also be used: marketing budget, sales events, number and type of products on sale in each sale event, number of page views, products availability, etc. In the future, we intend to advance the work presented here by inspecting several such models (logistic regression, random forest, neural nets and deep learning) and including some of the supplementary information. Another possibility to improve the results is by creating an ensemble from different models, built using the above-mentioned techniques, and thus to benefit from the fact that they generate the predictions in different ways, which might help eliminating some of the prediction errors.

**Acknowledgement.** This work was partly supported through the GEX contract *20/25.09.2017 funded by the University Politehnica of Bucharest*. We would also like to thank Vivre Deco for providing us the data that made this study possible.

## References

1. Mondal, P., Shit, L., Goswami, S.: Study of effectiveness of time series modeling (ARIMA) in forecasting stock prices. *Int. J. Comput. Sci. Eng. Appl.* **4**(2), 13 (2014)
2. Parthiban, K.T., Sekar, I., Umarani, R., Kanna, S.U., Durairasu, P., Rajendran, P.: *Industrial Agroforestry Perspectives and Prospectives*. Scientific Publishers, Jodhpur (2014)
3. Whittle, P.: *Hypothesis Testing in Time Series Analysis*, vol. 4. Almqvist & Wiksells, Uppsala (1951)
4. Box, G., Jenkins, G.: *Time Series Analysis: Forecasting and Control*. Holden-Day, San Francisco (1970)
5. Brockwell, P.J., Davis, R.A.: *Time Series: Theory and Methods*, p. 273. Springer, New York (1991). <https://doi.org/10.1007/978-1-4419-0320-4>
6. Akaike, H.: A new look at the statistical model identification. *IEEE Trans. Autom. Control* **19**(6), 716–723 (1974). <https://doi.org/10.1109/tac.1974.1100705>

7. Hyndman, R.J., Athanasopoulos, G.: Forecasting: principles and practice, OTexts. <https://www.otexts.org/fpp/8/5>. Accessed 20 May 2018
8. Stergiou, K.I.: Modeling and forecasting the fishery for pilchard (*Sardina pilchardus*) in Greek waters using ARIMA time-series models. *ICES J. Mar. Sci.* **46**(1), 16–23 (1989)
9. Meyler, A., Kenny, G., Quinn, T.: Forecasting Irish inflation using ARIMA models. Central Bank and Financial Services Authority of Ireland Technical Paper Series, vol. 1998, no. 3/RT/98, pp. 1–48, December 1998
10. Contreras, J., Espinola, R., Nogales, F.J., Conejo, A.J.: ARIMA models to predict nextday electricity prices. *IEEE Trans. Power Syst.* **18**(3), 1014–1020 (2003)
11. Li, Q., Guo, N.N., Han, Z.Y., Zhang, Y.B., Qi, S.X., Xu, Y.G., Wei, Y.M., Han, X., Liu, Y. Y.: Application of an autoregressive integrated moving average model for predicting the incidence of hemorrhagic fever with renal syndrome. *Am. J. Trop. Med. Hyg.* **87**(2), 364–370 (2012)
12. Kumar, M., Anand, M.: An application of time series ARIMA forecasting model for predicting sugarcane production in India. *Stud. Bus. Econ.* **9**(1), 81–94 (2014)
13. Sen, P., Roy, M., Pal, P.: Application of ARIMA for forecasting energy consumption and GHG emission: a case study of an Indian pig iron manufacturing organization. *Energy* **116**, 1031–1038 (2016)
14. Jarrett, J.E., Kyper, E.: ARIMA modeling with intervention to forecast and analyze chinese stock prices. *Int. J. Eng. Bus. Manag.* **3**(3), 53–58 (2011)
15. Devi, B.U., Sundar, D., Alli, P.: An effective time series analysis for stock trend prediction using ARIMA model for nifty midcap-50. *Int. J. Data Min. Knowl. Manag. Process* **3**(1), 65 (2013)
16. Taylor, S.J., Letham, B.: Forecasting at scale. *PeerJ Preprints* 5:e3190v2 (2017). <https://doi.org/10.7287/peerj.preprints.3190v2>



# Machine Learning-Driven Noise Separation in High Variation Genomics Sequencing Datasets

Milko Krachunov<sup>1</sup>, Maria Nisheva<sup>1,2</sup>(✉), and Dimitar Vassilev<sup>1</sup>

<sup>1</sup> Faculty of Mathematics and Informatics, Sofia University,  
5 James Bourchier Blvd, 1164 Sofia, Bulgaria

{milkok, marian, dimitar.vassilev}@fmi.uni-sofia.bg  
<sup>2</sup> Institute of Mathematics and Informatics, Bulgarian Academy of Sciences,  
Acad. G. Bonchev Str., Block 8, 1113 Sofia, Bulgaria

**Abstract.** Genomics studies have increasingly had to deal with datasets containing high variation between the sequenced nucleotide chains. This is most common in metagenomics studies and polyploid studies, where the biological nature of studied samples requires analysis of multiple variants of nearly identical sequences. The high variation makes it more difficult to determine the correct nucleotide sequences, as well as to distinguish signal from noise, producing digital results with higher error rates than the ones that can be achieved in samples with low variation. This paper presents an original pure machine learning-based approach for detecting and potentially correcting those errors. It uses a generic machine learning-based model that can be applied to different types of sequencing data with minor modifications. As presented in a separate part of this work, these models can be combined with data-specific error candidate selection to apply the models on, for a refined error discovery, but as shown here, can also be used independently.

**Keywords:** Machine learning · Neural network · NGS errors · Metagenomics  
Polyploid genomes

## 1 Introduction

There are two types of sequencing datasets that deal with high variation in the data among the individual genetic sequences. One of them is metagenomics, which studies entire communities of micro-organisms, such as the studies of the human microbiome [19], viral evolution [12], or the microbiome of urban environments [12]; the other is the study of polyploid genomes, such as the wheat genome [3], which contain multiple sets of similar but different chromosomes inherited from multiple ancestor organisms [17].

From a scientific perspective, metagenomics studies may be essential for medicine [1], nutrition studies [8], human space flight [22], agriculture [15], and understanding disease outbreaks [6]. At the same time, wheat is a staple food more widely grown than any other crop in the world [24], with the importance of the study of its genome proportional to its importance as food.

From a computational perspective, however, both metagenomics and polyploid studies are severely impeded by the presence of variation, and the resulting errors that are unaccounted for. These input errors inevitably lead to inaccurate final results [13, 25], and are often resolved by discarding reads suspected to contain any errors [5].

The lack of good computational tools to filter these errors out makes the use of artificial intelligence and, in particular, machine learning (ML) a suitable tool to attack this problem. ML-based models, like artificial neural networks (ANN) [21] and random forests [2], can be trained to classify examples based on numerical and other features, even when the relationship between the input and the result is unknown, or is not well studied. Random forests, in particular, are growing in popularity for approaching various Bioinformatics problems [20], where there is a rapidly increasing amount of raw data that is poorly studied.

## 2 Input Data and Problem Statement

After being digitalised using the process of genome sequencing (and especially the so-called next generation sequencing, NGS), the data in a metagenomics or polyploid sample is comprised of a set of sequences, or character strings, of a four-letter alphabet (A, C, G and T), representing the four nucleotide bases in the physical DNA or RNA chains. In raw datasets those sequences may be arbitrary fragments from the different loci in nucleotide chains, but after the use of specialised sequencing procedures and after data preprocessing, the working datasets will be comprised of aligned DNA or RNA sequences projected to be part of specific genes or regions.

For our metagenomics datasets, the preprocessing involved rough data clustering using the CD-HIT [16] software package and multiple sequence alignment using the MAFFT [7] software package. The resulting aligned clusters contain strings representing the genetic sequences from the same region of the genome of *various* micro-organisms, with the meaningful parts of the sequences aligned into the same string positions (columns) by the use of gap characters ('-'). The data was sequenced using 454. The resulting raw data is comprised of 30000 to 50000 character sequences with lengths of 300 to 500 characters, and the working test clusters are comprised of between 1000 to 6000 such sequences; all sequences starting from the same position in the micro-organism's genome.

For our hexaploid wheat NGS datasets, the preprocessing involved preliminary genome assembly [18], which attempts to cluster the genetic fragments into genes, providing a potential alignment to a representative sequence of that gene. The data was sequenced using Illumina, and thus the sequence lengths are shorter, up to 100 nucleotide bases or characters, start at various positions in the gene with varying overlaps. Up to 30–40 sequences cover the same region of the gene, and may belong to up to *three* distinct subgenomes that have similar but different genetic code.

In both instances, the character sequences contain sequencing errors. The errors may be substitutions, insertions or deletions – the strings may contain letters that differ from the real nucleotide base in the given position, or may have extra or missing letters. The main goal of this paper is to offer a way to detect these errors by distinguishing

them from the real biological variation, caused by the presence of multiple biological species or subgenomes.

This is approached as a classification problem, which separates the signal – variation present in the real genetic chains, from the noise – variation introduced in error by the sequencing equipment. Nucleotide bases, treated as regular characters, are classified individually, and erroneous insertions and deletions are dealt with by treating the gap characters ('-') as a fifth letter in the allowed alphabet, which is evaluated like the remaining four.

### 3 Using a Machine Learning Model

The presence of complex unknown structure inside the genetic chains of the studied species suggested the use of machine learning-based models. Since ML models like artificial neural networks are a very powerful tool to detect unknown and unstudied relations between the input variables, they make a perfect candidate to attempt classification of the data, especially after the analytical ways to approach the problem didn't seem to yield sufficiently good results [10], as noted later in Sect. 4.5.

To select the most appropriate ML-based model, during the study the same input was provided to different models – an artificial neural network, a random forest [2] model, as well as various other decision tree and classification models.

#### 3.1 Machine Learning Input

The learning input was composed based on two presumptions:

1. The frequency of a base in a given column is the most significant factor in whether it might be erroneous, as errors would be among the rarest bases in their position/column.
2. The frequency of the base among locally similar sequences is more important than the frequency among locally dissimilar sequences, making the factor of similarity a good criterion to split the sequences into multiple bins to produce the input values.

For each nucleotide base in each sequence, represented by a single character in the sequence's string, a learning input example will be constructed. For each such base, the nucleotide *sequences* from the dataset will be placed in multiple bins depending on how similar they are to the evaluated sequence. A triplet of such bins will be created for a pair of equidistant bases neighbouring the evaluated one. These auxiliary pairs of bases will serve as the criteria of local similarity between the sequence with the evaluated base, and the remaining sequences. For each such pair, the set of sequences can be split in three groups – the sequences that concur with both bases in both positions, the sequences that differ from the bases in both positions, and the sequences that concur with only one of them.

Around each evaluated character  $r_i$  in position  $i$ , a window of radius  $w$  will be constructed inside the evaluated sequence  $r$ , containing the neighbouring characters of  $r_i$ . Then, for each offset inside the window,  $j = 1, 2, \dots, w$ , the pair of characters  $r_{i \pm j}$  will be selected, and three bins will be constructed – the subset of character sequences

$p$  for which  $r_{i \pm j} = p_{i \pm j}$ , the subset of character sequences for  $r_{i \pm j} \neq p_{i \pm j}$ , and all the remaining sequences, which would differ in either position  $i + j$  or  $i - j$ , but not both.

The frequency of base  $r_i$  at position  $i$  will be recorded for each of these three subsets. This frequency is the count of characters  $p_i$  for which  $p_i = r_i$ , among the reads  $p$ , where  $p$  is not  $r$ . The computed three frequencies will be added to the input for the ML-based models, which will be comprised of a series of such triplets, creating an input vector of length  $3 - w$ .

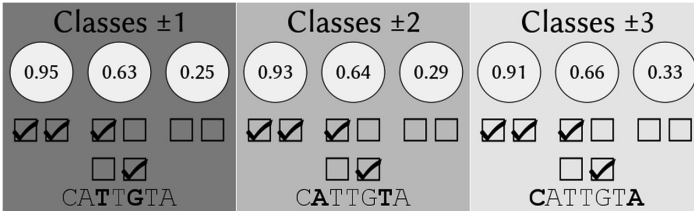


Fig. 1. Example learning input.

Figure 1 shows an example of a learning input to evaluate the middle base T, using a window  $w$  of three bases. For the closest pair of neighbouring bases T and G, the entire dataset has been split into three. Among the sequences that contain T and G in both of those positions, 95% confirm the middle base T; among the sequences that contain only one of them in their position, 63% confirm the middle base T; and among the sequences that contain neither, only 25% confirm it. These three values are used as the first three values inside the input vector that will be classified by the ML-based models. This is repeated for the next pair of characters A and T, and then for the most distant one – C and A.

Let’s assume that there are  $n$  sequences of average length  $m$ , and we need to compute the input vectors for all approximately  $m$  characters inside all  $n$  sequences. Without complex, potentially inexact, optimisations this will have a time complexity of  $O(mnw)$  and space complexity of  $O(mw)$ . The time complexity is reasonable, since  $O(mnw)$  would be the time to loop over the learning inputs once.

Because speed wasn’t the main aim of this paper, during the experiments a slower algorithm with time complexity of  $O(mn^2w)$  and space complexity of  $O(w)$  was used, as it was more straightforward to verify that it constructs the examples exactly as defined here.

### 3.2 Training with Virtual Errors

The genomic data from metagenomics and wheat presents two challenges that prevent using real errors to construct training examples – on one hand, the errors that are to be evaluated are unknown, and on the other hand, the error rate is low enough that the real errors will not provide enough positive training examples [4] for errors. For that reason, instead of training on examples of real errors, the models were trained to recognise an artificially introduced error at each position.



In other words, the dataset was unchanged and no errors were simulated at large. Instead, only the individual base used to construct the error example was substituted with an erroneous one, and then reverted back for the construction of the remaining examples. This was referred to as a ‘virtual’ error by the research team, as no permanent errors were added to the dataset for the training.

This technique is easier to implement than simulation of errors, allows for the construction of a very high number of error examples, ensures that the trained models are not overfitted to the patterns of simulated errors, although they may be overfitted to other features of the dataset.

## 4 Results

To empirically test the chosen way to construct the input, as well as the way to construct training examples, models constructed using them were tested on metagenomics data, and then in a later experiment, on wheat data.

### 4.1 Artificial Neural Networks on Metagenomics Data

Two neural networks were trained using two sets of metagenomics input data using the proposed ‘virtual’ errors. The only preprocessing applied on the input data was clustering and multiple sequence alignment. To measure the performance of the neural network, three different approaches were taken – splitting of the training set into training and testing set with 2:1 ratio; constructing a testing set from a subset of the same sequences used to build the training set, but corrected using a rudimentary analytical error detection [11]; and constructing two testing sets on a completely different dataset of genomic sequences – one on uncorrected, and one on sequences corrected by the aforementioned rudimentary approach. The results of these experiments are shown in Table 1.

**Table 1.** Experiments using artificial neural networks.

Training set	Testing set	Noise Errors	Detected Missed Identified as		Sensitivity	Signal Correct	Detected Missed Identified as		Specificity
			‘error’	‘correct’			‘correct’	‘error’	
C1	C1 2:1 split	19215	19154	61	99.682%	23063	22980	83	99.640%
C1	C1 corrected	56458	55729	729	98.708%	68296	67962	334	99.510%
C1	C3 original	50698	50323	375	99.260%	58688	58516	172	99.706%
C1	C3 corrected	51131	50853	278	99.456%	59373	59145	228	99.615%
C3	C3 2:1 split	17191	17162	29	99.831%	20000	19952	48	99.760%
C3	C3 corrected	51131	50857	274	99.464%	59373	59186	187	99.685%
C3	C1 original	56321	55940	381	99.323%	68025	67583	442	99.350%
C3	C1 corrected	56458	56036	422	99.252%	68296	67879	417	99.389%

It is evident that the ANN models trained in this manner have a satisfactory sensitivity, as it correctly classifies the examples corresponding to the artificial errors with a very high accuracy. This is even true when the testing set was constructed using a completely different set of biological data (referred to as C1 and C3 in the table). Unfortunately, the specificity – while still high – is not satisfactory. Even with high error rates, the errors are still found in disproportionately fewer numbers than the correct nucleotide bases, starting with an already high signal-to-noise ratio.

For the ANN model to be usable for correction purposes, the specificity needs to be at least higher than the frequency of correct bases. This prompted earlier work focused on pre-filtering the classification examples using analytical approaches [10], but also leaves the ANN model applicable for data digitalised using sequencing technologies that have much lower signal-to-noise ratio, such as Oxford Nanopores [14]. In addition, when the same models were trained on wheat datasets, a much higher specificity was observed, justifying the direct use of this model, as is shown in Sect. 4.4.

The result also shows a significant drop in the specificity once the model is applied to a biological data set different from the one it was trained on. Given the proposed method of training, it should be possible to re-train the model for each dataset and overlook that difference. The virtual errors introduced in the dataset are random and statistically independent from the actual errors. The models are trained to recognise those virtual amidst a small sample of the bases in the dataset, making such dataset-targeted training a potentially valid approach, allowing us to take advantage of a specificity that's almost twice as good.

It should be also noted that neither the training, nor the testing data was perfect, as it already contained errors before the virtual errors had been introduced. This would lead to some minimal bias during training, but it would also lead to underestimation of the specificity of the models – since there are already errors in the data, and since the models would detect those errors, we would incorrectly count them as falsely identified. This would be most significant for the random forest results in Sect. 4.3, and the wheat results in Sect. 4.4.

## 4.2 Varying the ANN Model Parameters

The results in Table 1 were achieved after an initial selection of the ANN model parameters – 30 input neurons (corresponding to 10 pairs of nucleotide bases neighbouring the evaluated one) and 16 hidden neurons. This seems to be a reasonable choice – a radius of 10 bases gives enough information about the local variation to the ANN model, which will allow the neural network to determine which part of the datasets that corroborates the evaluated nucleotide is made of relevant sequences. A number of hidden neurons that's approximately half the number of input neurons is a common initial choice as well. However, ANN models are first and foremost a heuristic and empirical tool, where the results are dependent on experiments, not on some hard set rules, so some experiments supporting the choice of parameters or exploring other potential values have to be conducted.

First, we varied the radius of the window around the evaluated base to determine if 10 pairs of nucleotide bases give enough local context to the ANN model, thus varying the number of input neurons away from the initial of 30. Table 2 shows the results

**Table 2.** Input neuron variation.

Radius	Test on split dataset				Test on different bio. data			
	Errors		Correct		Errors		Correct	
	Missed	Sensitivity	Missed	Specificity	Missed	Sensitivity	Missed	Specificity
10 (chosen)	84	99.56%	68	99.70%	456	99.10%	190	99.67%
6	64	99.67%	77	99.67%	390	99.23%	237	99.59%
5	74	99.61%	91	99.61%	403	99.20%	233	99.60%
4	75	99.61%	85	99.63%	408	99.19%	141	99.76%
3	76	99.60%	93	99.60%	239	99.53%	246	99.58%
2	99	99.48%	78	99.66%	341	99.32%	219	99.62%
1	186	99.03%	99	99.57%	480	99.05%	214	99.63%
Outer 6	111	99.42%	153	99.34%	599	98.81%	381	99.35%
Outer 4	166	99.14%	139	99.40%	790	98.43%	457	99.22%
Outer 2	285	98.52%	151	99.34%	911	98.19%	348	99.40%

Split: 19201 errors, 23049 correct bases  
Diff. data: 50396 errors, 58372 correct bases

**Table 3.** Varying the number of hidden neurons.

Hidden neurons	Errors	Identified	Missed	Sensitivity	Correct	Identified	Missed	Specificity
<i>Test on split dataset</i>								
16 (chosen)	19201	19117	84	99.56%	23049	22981	68	99.70%
17	19201	19116	85	99.56%	23049	22974	75	99.67%
15	19201	19124	77	99.60%	23049	22975	74	99.68%
<i>Test on different bio. data</i>								
16 (chosen)	50396	49940	456	99.10%	58372	58182	190	99.67%
17	50396	49938	458	99.09%	58372	58172	200	99.66%
15	50396	49946	450	99.11%	58372	58160	212	99.64%

when radii of 1 to 6 (3 to 18 input neurons) were used. In addition, tests were made with only the outer 2, 4 and 6 pairs of bases inside a 10-base window to test if the closest bases provide the most relevant context.

It is evident that increasing the number of input neurons improves the detection accuracy, particularly the sensitivity. The specificity is also affected. With a radius of 1 base, or only three input neurons corresponding to the two adjacent bases, the accuracy is substantially weakened; the sensitivity doesn't become reasonable until at least 2 base radius, where 6 input neurons summarise information about the 4 neighbouring nucleotide bases. Beyond 6 bases, there is saturation of accuracy as adding more bases doesn't lead to a dramatic improvement, demonstrating that input 18 neurons are enough.

This is only valid when the dataset-targeted training would be used, as the results on a different biological dataset are highly inconsistent. One can presume the possibility of overfitting the model towards the biological features in the training set – a problem that seems less pronounced when a dataset-targeted training is to be used.

It should also be noted that using the data for only distant bases leads to a significant and consistent decrease in the accuracy, both on a split of the training set and on a different biological dataset. This confirms the expectations that the nearby bases are the most important, and also suggests that the nearest 4 bases may be enough, but are also necessary to make an accurate determination using this ANN model.

Table 3 shows the effect of varying the number of neurons on the *hidden layer*. For this experiment, the input neurons were fixed to 30, and the hidden neurons were increased and decreased from 16 to see if that would affect the result positively or negatively.

It was observed that 16 is close to an optimum number of hidden neurons, as varying it up and down leads to less accuracy, and in 7 out of 8 cases 16 hidden neurons are better than 15 or 17 hidden neurons. This is not a definite guarantee that 16 is the best choice, but it supported the use of 16 hidden neurons enough to postpone modifications of the hidden layer as means to improve the results.

### 4.3 Application of Other ML-Based Models

During the development of a threshold-based analytical approach that was presented in [11], it was noted that thresholds may be a good way to separate the signal (correct bases) from the noise (errors). Since decision trees models over numerical data rely on the use of empirically-determined thresholds, this inspired us to test a decision tree algorithm, and then subsequently test 20 different ML-based models, focusing on the algorithms that use trees.

**Table 4.** Experiments with different ML-based models.

Model	Test on split data				Test on different bio. data			
	Errors		Correct		Errors		Correct	
	Missed	Sensitivity	Missed	Specificity	Missed	Sensitivity	Missed	Specificity
Alternating Decision Tree	110	99.43%	82	99.64%	140	99.72%	609	98.96%
Decision Stump	180	99.06%	107	99.54%	344	99.32%	334	99.43%
Logistic Model Tree	52	99.73%	48	99.79%	522	98.96%	329	99.44%
Naïve Bayes Tree	190	99.01%	69	99.70%	407	99.19%	325	99.44%
Ripple-Down Rule	74	99.62%	44	99.81%	351	99.30%	313	99.46%
LogitBoost Alt. Dec. Tree	72	99.63%	63	99.73%	238	99.53%	260	99.56%
Decision Tree + Naïve Bayes	99	99.48%	55	99.76%	351	99.30%	244	99.58%
Reduced-Error Pruning Tree	62	99.68%	55	99.76%	333	99.34%	242	99.59%
Support Vector Machine	78	99.59%	117	99.49%	245	99.51%	239	99.59%
Random Tree	60	99.69%	55	99.76%	555	98.90%	226	99.61%

(continued)

**Table 4.** (continued)

Model	Test on split data				Test on different bio. data			
	Errors		Correct		Errors		Correct	
	Missed	Sensitivity	Missed	Specificity	Missed	Sensitivity	Missed	Specificity
Neural network	84	99.56%	68	99.71%	456	99.10%	190	99.68%
Best-First Tree	56	99.71%	52	99.77%	358	99.29%	150	99.74%
Functional Tree	66	99.66%	59	99.74%	455	99.10%	142	99.76%
Partial C4.5	54	99.72%	53	99.77%	477	99.05%	130	99.78%
Decision Table	111	99.42%	33	99.86%	732	98.55%	124	99.79%
Simple Cart	56	99.71%	51	99.78%	476	99.06%	105	99.82%
C4.5	62	99.68%	42	99.82%	499	99.01%	103	99.82%
Grafting C4.5	57	99.70%	44	99.81%	374	99.26%	89	99.85%
RIPPER Rule Learner	55	99.71%	58	99.75%	344	99.32%	74	99.87%
Random forest (10 trees)	51	99.73%	23	99.90%	452	99.10%	34	99.94%
Random forest (60 trees)	46	99.76%	18	99.92%	393	99.22%	29	99.95%
Split: 19201 errors, 23049 correct bases								
Diff. biodata: 50396 errors, 58372 correct bases								

Table 4 shows the results with all the tree-based models that were selected in the Weka software [26], as well as other classification models like RIPPER. The table is sorted by *specificity* on a different biological dataset, in an effort to select the model that best suited to be used directly to detect the errors, without the need to preselect error candidates, by finding the highest possible specificity.

The observed results confirm the expectations that decision trees are also a good tool to approach this problem. Half of the decision tree algorithms produced specificity higher than the ANN when used on a different set of biological data.

The random forest model achieved the highest accuracy overall, with a specificity of over 99.9%. This means less than one falsely selected error for every 1000 bases. This suggests random forests are the best model to be used directly to detect errors in the biological data, without use of further tools to improve the accuracy. On the contrary, in a separate study, we discovered that the random forest model becomes an ill-suited choice when it is combined with error candidate pre-selection [10].

#### 4.4 Application of the ML-Based Models on Wheat

The wheat genome, consisting of three close but different subgenomes, also poses a challenge similar to that of metagenomics, but because the subgenomes are only three, the problem is less pronounced as the number of variants is much more limited. Given that the specificity of the proposed model seemed to come short on metagenomics data, except for when random forests were used, we decided to attempt to apply the same model on wheat to test if the limited number of variants leads to substantially better results.

It should be noted that while the training and testing scheme are identical, the experiment ended up being conducted with a different set of ML-based models. The experiment was repeated with artificial neural networks, random forests and naive Bayes, but three SVM models were added.

**Table 5.** Experiments with different ML-based models.

Model	Accuracy	Misclassified	
		Non-errors	Errors
Artificial neural network	99.89%	91	0
Random forests of 60 trees	99.91%	70	1
Random forests of 100 trees	99.91%	70	1
SVM with Gaussian kernel	99.57%	325	0
SVM with sigmoid kernel	99.58%	341	0
SVM with linear kernel	99.74%	214	0
Naive Bayes classifier	99.37%	125	389

Table 5 shows the performance of the selected models on wheat. The models were trained using 5 pairs of neighbouring bases, or 15 input values for the models. The testing was performed using cross-validation within the same gene. A more detailed description of both the study and the results was published in [9].

In all of the cases, all models except for the naïve Bayes classifier achieved sensitivity close to 100%, and error specificity over 99.5%. The neural network and the random forests were the only two models that got specificity close to 99.9%, with the random forest model still outperforming the neural network model.

One thing that was noted during this experiment is the near-100% sensitivity of several of the models. This suggested that the decreased number of variants has positively impacted the general accuracy of the models. Since in this particular experiment, the number of actual errors was unknown, it more strongly suggested the possibility that a high number of the misclassified non-errors are actually accounted for by unknown errors in the data, as hypothesized in Sect. 4.1. It is already known that in all of these tests, the specificity is underestimated, but with the near-100% sensitivity and the increased specificity of the neural network, the underestimate may be significant. Should this reduction be measured to be high enough, the direct application of this model on wheat datasets without any additional filtering would be justified.

#### 4.5 Viability of Other Approaches

Earlier attempts to solve the problem of error detection inside a high-variation dataset using analytical tools proved unfruitful. We attempted to isolate the gene variants by modelling the local similarity in the same window  $w$  as in the ML-based model presented here, and find the rarest nucleotide bases after that isolation. That led to 18% less error predictions compared to using an approach intended for dataset with no variants. However, when the ANN model was applied on those predictions, it discarded

51.5% of them as non-errors, on average; it wasn't discarding almost any simulated<sup>1</sup> errors [10]. This means that more than half of the analytically predicted errors were actually correct bases.

The same analytical similarity-based isolation was compared to SHREC [23], which was the only viable error detection tool that seemed to account for potential variation at the time of the testing. SHREC seemed to make between 6.5% more incorrect error predictions, without detecting more errors. Once the analytical isolation was improved, SHREC's incorrect predictions became 13.7% more on average, and the addition of a ML-based model increased the difference to 27.1%, with the ML-based model detecting 12.7% more of the simulated errors<sup>2</sup>.

## 5 Conclusions

This paper presents a ML-based model for detection of noise inside genomic datasets with high variation. The model was tested on both metagenomics datasets with a high number of variants and wheat datasets with limited number of variants, where its performance was measured. Artificial neural networks and random forests were tested, along with a variety of other models.

The error detection specificity of all the models was very high, suggesting that they detect almost all of the errors present in the data. The specificity, although numerically high, was close to the ratio of correct bases to all bases, thereby risking more error false positives than errors. While in different portion of the study [10] that problem was remedied with use of additional candidate filtering, here we observe that the random forest models on metagenomics, and both random forests and neural networks on wheat might have a high-enough specificity to be used without such filtering.

The observed high accuracy of the ML models proved them to be better a tool to detect the errors than any of the other approaches attempted or tried by the authors, including the author's own analytical approach and third party software tools. However, even the ML models were impeded by the high disparity between correct bases and incorrect bases.

**Acknowledgements.** The presented work has been funded by the Bulgarian NSF within the "GloBIG: A Model of Integration of Cloud Framework for Hybrid Massive Parallelism and its Application for Analysis and Automated Semantic Enhancement of Big Heterogeneous Data Collections" project, Contract DN02/9 of 17.12.2016, and by the Sofia University SRF within the "Models for semantic integration of biomedical data" project, Contract 80-10-207 of 26.04.2018.

---

<sup>1</sup> The study used a probabilistic error simulation based on estimated error rates.

<sup>2</sup> Since SHREC has no parameters to tune, our models were tuned to be close to – or above – SHREC's detection rates to compare, so each experiment used different tunings and the figures aren't directly comparable.

## References

1. Allen-Vercoe, E., Petrof, E.O.: The microbiome: what it means for medicine. *Br. J. Gen. Pract.* **64**(620), 118–119 (2014)
2. Breiman, L.: Random forests. *Mach. Learn.* **45**(1), 5–32 (2001)
3. Brenchley, R., et al.: Analysis of the bread wheat genome using whole-genome shotgun sequencing. *Nature* **491**(7426), 705–710 (2012)
4. Gilles, A., Megléc, E., Pech, N., Ferreira, S., Malausa, T., Martin, J.F.: Accuracy and quality assessment of 454 gs-flx titanium pyrosequencing. *BMC Genomics* **12**, 245 (2011)
5. Huse, S., Huber, J., Morrison, H., Sogin, M., Welch, D.: Accuracy and quality of massively parallel dna pyrosequencing. *Genome Biol.* **8**(7), R143 (2007)
6. Karlsson, O.E., Hansen, T., Knutsson, R., Löfström, C., Granberg, F., Berg, M.: Metagenomic detection methods in biopreparedness outbreak scenarios. *Biosecurity Bioterrorism Biodefense Strategy Pract. Sci.* **11**(S1), S146–S157 (2013)
7. Katoh, K., Kuma, K., Toh, H., Miyata, T.: MAFFT version 5: improvement in accuracy of multiple sequence alignment. *Nucleic Acid Res.* **33**(2), 511–518 (2005)
8. Kau, A.L., et al.: Human nutrition, the gut microbiome, and immune system: envisioning the future. *Nature* **474**(7351), 327–336 (2011)
9. Kirov, K., Krachunov, M., Kulev, O., Nisheva, M., Vassilev, D.: Reducing false negatives for errors in snp detection using a machine learning approach. *Comptes rendus de l'Académie bulgare des Sciences* **69**(2), 155–160 (2016)
10. Krachunov, M., Nisheva, M., Vassilev, D.: Machine learning models in error and variant detection high-variation high-throughput sequencing datasets. *Procedia Comput. Sci.* **108C**, 1145–1154 (2017)
11. Krachunov, M., Vassilev, D.: An approach to a metagenomic data processing workflow. *J. Comput. Sci.* **5**, 357–362 (2014)
12. Kristensen, D., Mushegian, A., Dolja, V., Koonin, E.: New dimensions of the virus world discovered through metagenomics. *Trends Microbiol.* **18**(1), 11–19 (2010)
13. Kunin, V., Engelbrekton, A., Ochman, H., Hugenholtz, P.: Wrinkles in the rare biosphere: pyrosequencing errors can lead to artificial inflation of diversity estimates. *Environ. Microbiol.* **12**(1), 118–123 (2010)
14. Laver, T., et al.: Assessing the performance of the Oxford Nanopore Technologies MinION. *Biomol. Detect. Quantification* **3**, 1–8 (2015)
15. Li, R.W. (ed.): *Metagenomics and its Applications in Agriculture, Biomedicine and Environmental Studies*. Nova Science Pub Inc. (2010)
16. Li, W., Godzik, A.: Cd-Hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics* **22**(13), 1658–1659 (2006)
17. Marcussen, T., et al.: Ancient hybridizations among the ancestral genomes of bread wheat. *Science* **345**(6194), 286–291 (2014)
18. Miller, J.R., Koren, S., Sutton, G.: Assembly algorithms for Next-Generation Sequencing data. *Genomics* **95**(6), 315–327 (2010)
19. Nelson, K., White, B.: Metagenomics and its applications to the study of the human microbiome. In: *Metagenomics: Theory, Methods and Applications*, pp. 171–182 (2010)
20. Qi, Y.: Random forest for bioinformatics. In: Zhang, C., Ma, Y. (eds.) *Ensemble Machine Learning*, pp. 307–323. Springer, Boston (2012). [https://doi.org/10.1007/978-1-4419-9326-7\\_11](https://doi.org/10.1007/978-1-4419-9326-7_11)
21. Rojas, R.: *Neural Networks: A Systematic Introduction*. Springer, Heidelberg (1996). <https://doi.org/10.1007/978-3-642-61068-4>
22. Saei, A.A., Barzegari, A.: The microbiome: the forgotten organ of the astronaut's body—probiotics beyond terrestrial limits. *Future Microbiol.* **7**(9), 1037–1046 (2012)



23. Schröder, J., Schröder, H., Puglisi, S.J., Sinha, R., Schmidt, B.: SHREC: a short-read error correction method. *Bioinformatics* **25**(17), 2157–2163 (2009)
24. United Nations, Food and Agriculture Organization, S.D.F. Crops/World total/Wheat/Area harvested (2014). <https://web.archive.org/web/20150906230329/>, <http://faostat.fao.org/site/567/DesktopDefault.aspx?PageID=567>. Accessed 25 June 2018
25. Valverde, J., Mellado, R.: Analysis of metagenomic data containing high biodiversity levels. *PLoS ONE* **8**(3) (2013). Article no. e58118
26. Witten, I.H., Frank, E., Hal, M.A.: *Data Mining: Practical Machine Learning Tools and Techniques*, 3rd edn. Morgan Kaufmann Publishers, San Francisco (2011)



# Machine Learning Techniques for Survival Time Prediction in Breast Cancer

Iliyan Mihaylov, Maria Nisheva, and Dimitar Vassilev<sup>(✉)</sup>

Faculty of Mathematics and Informatics, Sofia University “St Kliment Ohridski”,  
5 James Bourchier, Blvd, 1164 Sofia, Bulgaria  
{mihaylov, marian, dimitar.vassilev}@fmi.uni-sofia.bg

**Abstract.** The use of machine learning in disease prediction and prognosis is part of a growing trend of personalized and predictive medicine. Cancer studies are domain of active machine learning implementation in particular in sense of accuracy of cancer prognosis and prediction. The accuracy of survival time prediction in breast cancer is the main object of the study. Two major features for survival time prediction, based on clinical data are used: the created in the study tumor integrated clinical feature and Nottingham prognostic index. The applied machine learning methods aside with data normalisation and classification provide promising results for accuracy of survival time prediction. Results showed prepotency of the support vector regression modles - linear and decision tree regression models, for more accurate prediction of the survival time in breast cancer. Cross-validation, based on four parameters for error evaluation, confirms the results of the model performance concerning the accuracy of survival time prediction in breast cancer.

**Keywords:** Bioinformatics · Machine learning · Classification analysis  
Breast cancer · Survival time prediction

## 1 Introduction

Breast cancer is a cancer manifesting in women mostly (more than 99%) and concerns approximately one in eight women over their lifetime, according to American Cancer Network [1]. The same source reported that the average 10-year survival rate is 83%. If the cancer is located only in the breast, the 5-year relative survival rate of people with breast cancer is 99%. Sixty-two percent (62%) of cases are diagnosed at this stage.

In direct medical sense - breast cancer can be diagnosed by classifying tumors. Usually there are two classes of tumors: malignant and benign tumors. Medical doctors and oncologists in particular need a reliable diagnosis procedure to distinguish between these two classes of tumors. Data and new modeling approaches in cancer studies grows in recent years because of the massive use of laboratory high-throughput technologies - generating big amount of data [2, 3]. In fact, it is still very difficult to distinguish tumors even by the experts. The obvious rapid development of new knowledge-driven diagnostics for tumor detection is based on bioinformatics, statistics and computer science. Aside with that many of these methods are difficult to be integrated and combined in a meaningful workflow. With the advent of a larger application of machine learning

(ML) methods, cancer studies became more divergent, with improved accuracy and validation and based on discovery of new enriched knowledge about the origin, classification, prognosis and therapy. There are many algorithms for classification and prediction of breast cancer outcomes. Our work comprises machine learning methodologies for survival prediction rate in breast cancer. We apply and compare the performance of four ML methods for prediction of survival time and several methods for cross-validation of the applied machine learning models.

The objective of this study is to assess the efficiency and accuracy of the used machine learning models for survival time prediction in breast cancer.

## 2 Problem Description

A more accurate prediction of survival rate in patients with breast cancer remains a real challenge due to the increasing complexity of cancer, treatment protocols and various patient population samples. Reliable and well validated predictions could assist in a better way personalised care and treatment, and improve the control over the cancer development. Usually in good clinical practices, clinicians use data collected from different sources as medical records, clinical laboratory tests and studies aiming at a more precise diagnostic, therapy and disease development prognosis.

There is a definite increase in the use of classification based approaches in contemporary medical diagnostics [3, 4]. Cancer studies are the major target in using contemporary bioinformatics, statistics and machine learning techniques for the purposes of more accurate and rapid diagnostics. In the scope of constantly growing significance of predictive and personalized medicine there is a rapidly growing demand to apply machine learning-driven models to make predictions and prognosis in cancer studies [5, 6].

At first sight all these classification approaches based on various and heterogeneous medical data can inflate the quality of diagnostics. On the contrary numerous recent developments in computer science, data science and machine learning driven approaches definitely assist in decrease of errors in overall diagnostics. The use of artificial intelligence techniques for classification in cancer studies provide the more informative and knowledge based background for prediction and prognosis of cancer to be tested more meticulously and in shorter time (rapidly) [5].

Prediction and prognosis of cancer development are focused in three major domains: risk assessment or prediction of cancer susceptibility, prediction of cancer relapse, and prediction of cancer survival rate. The first domain comprises prediction of the probability of developing certain cancer disease prior to the patient diagnostics. The second issue is related to prediction of cancer recurrence in terms of diagnostics and treatment and the third case is aiming in prediction of several possible parameters characterizing cancer development and treatment after the diagnosis of the disease: survival time, life expectancy, progression, drug sensitivity, etc. The survivability rate and the cancer relapse are dependent very much on the medical treatment and the quality of the diagnosis [6].

A major line of machine learning studies on breast cancer development is focused on predicting patient survivability. There is a variety of machine learning based approaches

used on different datasets. The obvious trend is that all the major studies with clinical data use mostly models related to artificial neural networks (ANN), support vector machines (SVM) and using statistical methods for validation [3, 7]. In this way some problems with classification and validation have been overcome. Although there is an obvious demand in improving the machine learning impact in survival time prediction studies in breast cancer in the scope of generality, better accuracy and validation. These challenges are also in the scope of our work.

### 3 Related Work

Artificial intelligence and in particular machine learning models have a visible history in cancer research and practical implementation [4]. Artificial neural networks (ANNs) [11] and decision trees (DTs) [9] have been used in cancer detection and diagnosis for nearly 30 years. Different models based on Support Vector Machine (SVM) [10] applied to cancer prognosis issues are used from about couple of decades. Other models for prediction of cancer development also have been used in a number of studies [4]. Today the role of machine learning grows in applied data science and bioinformatics methodologies, used from diagnostics to prediction and prognosis in cancer [3, 13]. All these research studies are concerned with using machine learning methods to identify, classify, detect, or distinguish tumors and other malignancies as well as to predict cancer development.

The breast cancer survival time prediction studies based on machine learning models occupied a significant part of the contemporary research in this area [2, 3]. There is a number of studies consider the effect of ensemble of machine learning techniques to predict the survival time in breast cancer. These techniques show improved accuracy to a certain extent comparing to previous results [3, 8]. A number of papers concerns different problems in applying machine learning algorithms for breast cancer prediction. Authors experimented on breast cancer data using C5 algorithm with bagging [9] to predict breast cancer survivability. Another authors gain 93% accuracy of survivability in breast cancer prediction [10]. Some of the studies was focused on a comparative analysis of the performance of the applied supervised learning classifiers such as Naïve Bayes, SVM-RBF kernel, RBF neural networks, Decision trees (J48) and simple CART; to find the best classifier in breast cancer datasets [11].

Many problems with the use of machine learning in breast cancer predictions studies are related to the lack of optimal and precise validation. We could admit that the use of ML models can improve the accuracy of survival prediction - but the choice of proper validation approach is of a great value in particular for studying the breast cancer time of survivability. Among the most common methods for evaluation the performance of the applied model is the cross-validation method. Cross validation is very suitable for machine learning based modeling and is used for training and for testing [12].

An obvious trend in the proposed works includes also integration of mixed data of clinical and lab origin. This makes possible to use also data science technologies and models for data integration and subsequent normalisation and classification for the purposes of a predictive study [13]. A reasonable semantic data integration can provide

better quality of the input data sets for using the machine learning for prediction of survival time of breast cancer.

## 4 Data Description

Data used in the study is based on clinical records, including patient age, stage of tumor development, tumor size and its living status. There is also data concerning different types of therapies and surgery intervention. There are many parameters related to disease development - tumor size, age at diagnosis, tumor stage, information about applied surgeries and applied treatments like chemotherapy, hormone therapy, etc. The data is heterogeneous and many records are not complete. Only for patients with breast cancer after surgery is available Nottingham prognostic index (NPI) – a feature, used to determine prognosis following surgery for breast cancer. Its value is calculated using three pathological criteria: the size of the lesion; the number of involved lymph nodes; and the grade of the tumor. In a particularly related study [14] the NPI can be used to predict survival of 80%, 42% and 13% in three groups of patients according to the NPI score. The same study reported also that NPI has been refined and patients were segregated into four groups, and afterwards was used to predict five-year survival rate (in accordance with the more commonly used time scales for survival of other types of cancers). The recorded parameters used in our study is shown on Table 1. All these parameters are integrated in a schema-less NoSQL database for easy access and management.

**Table 1.** Studied set of features from breast cancer patients’ clinical data records (total number of patients is 2301).

Name	Value	Description
HER2_STATUS (1220 patients)	-/empty/”	HER2 proteins are receptors on breast cells. Normally, HER2 receptors help control how a healthy breast cell grows, divides, and repairs itself.
HER2_SNP6 (1224 patients)	String	HER2 with SNP (Single nucleotide polymorphism)
AGE_AT_DIAGNOSIS (1223 patients)	Float	Age at diagnosis
NPI (595 patients)	Integer	Nottingham prognostic index
CELLULARITY (1224 patients)	String high/low/empty/-	The number and type of cells in a given tissue with issues
HORMONE_THERAPY (1224 patients)	Yes/No	

(continued)

**Table 1.** (continued)

Name	Value	Description
TUMOR_SIZE (1220 patients)	Float	
SAMPLE_TYPE (1220 patients)	Float	
OS_MONTHS (1178 patients)	Float	Months after diagnosis
OS_STATUS (1224 patients)	String	life status
BREAST_SURGERY (1224 patients)	String	Type of SURGERY
CHEMOTHERAPY (1224 patients)	Yes/No	
GRADE (1220 patients)	Integer	Cancer specific grade

## 5 Suggested Methodology

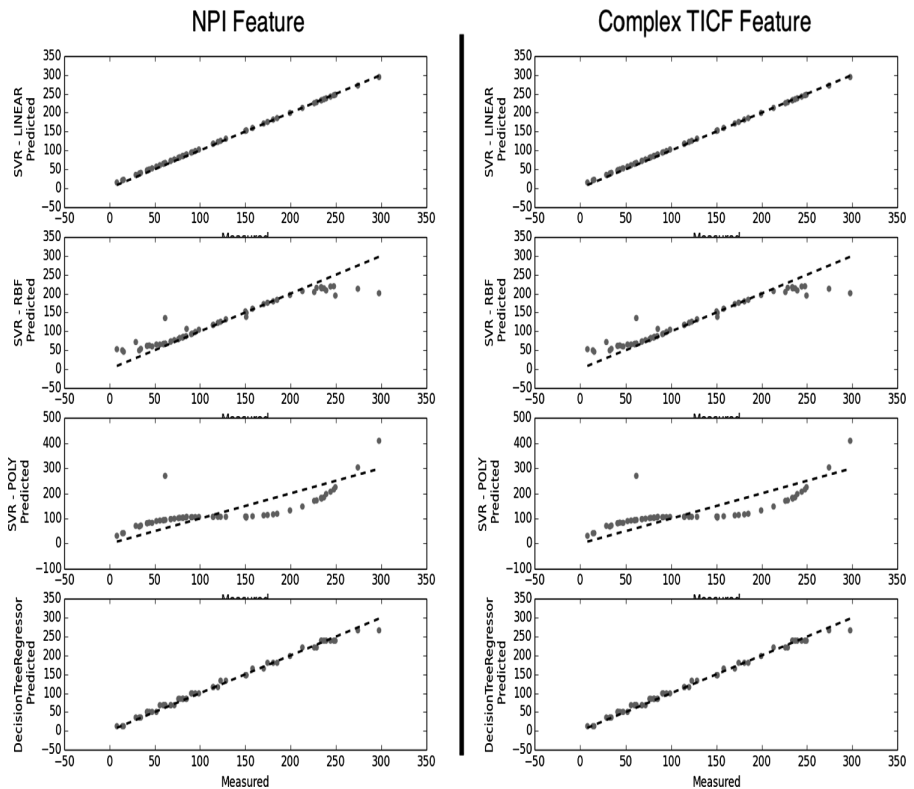
We develop a machine learning-driven approach for survival time prediction in breast cancer based on three parameters: tumor size, tumor stage, and age at diagnosis. We combine these three features in a novel one. This new feature is built by a numerical concatenation of the tumor stage, tumor size and age at diagnosis. The order of concatenation in this tumor integrated clinical feature (TICF) is important because of ranking clinical information for tumor development and its relevance to the patient survival rate. For example, a studied patient has tumor in stage 1 with size 10 mm and the patient is 45 years old; the new concatenated value of this TICF record is 11045. We provide in this manner a bigger distance between the patients with different tumor stages, tumor sizes and age at diagnosis. Such a distance is important for the subsequent machine learning approaches applied for survival time prediction. The studied data contain also the Nottingham Prognostic Index (NPI). We normalize both data sets based on two features (TICF and NPI) by removing the mean and scaling them to unit variance. Mean and standard deviation are stored and used in subsequent data analysis using the transform method. Standardization of a dataset is a common requirement for many machine learning classifiers: they might behave badly if the individual feature does not more or less look like standard normally distributed data (with 0 mean and unit variance).

Next stage in our methodology is to apply machine learning models to predict the survival time and to validate them. The used machine learning models are Support Vector Machine - Regression (SVR) with different kernels: radial basis function (RBF), Linear and Poly as well as a Stochastic Gradient Descent model (SGD). We choose these models because they have shown good results for survival time prediction [13]. We validate the results by using randomly smaller subsets of both raw and integrated data.

We used a cross-validation approach for to evaluate the performance of the applied machine learning models. This is a k-fold cross-validation, where the original sample is randomly partitioned into k equal sized subsamples. Of the k subsamples, a single subsample is retained as the validation data set for testing the model, and the remaining k - 1 subsamples are used as training data sets. The cross-validation process is then repeated k times (the folds), with each of the k subsamples used exactly once as the validation data. The k results from the folds can then be averaged to produce a single estimation. The advantage of this method over repeated random sub-sampling is that all observations are used for both training and validation, and each observation is used for validation exactly once. 10-fold cross-validation is commonly used, but in general k remains an unfixed parameter. This validation model can be used to estimate any quantitative measure of fit that is appropriate for certain data and model.

We compare the results provided by the cross validation of two approaches with different features to find the optimal model for survival time prediction.

All scripts are developed by us in Python with *sklearn* library.



**Fig. 1.** Survival time prediction success rate by NPI and TICF feature within different ML models

## 6 Results and Discussion

The suggested methodology for survival time prediction in breast cancer uses two features (NPI and TICF) for classification of the data based on *k-neighbours*. The NPI and TICF groups are unbalanced and they contain *k-neighbours* subgroups with small number of patient records which is an obvious obstacle to predict accurately the survival time rate. In this line the test data is clustered in 5 subgroups by the use of *k-fold* algorithm.

After the dataset is normalised and classified we applied several machine learning models for survival time prediction. Among the applied ML models shown on Fig. 1, all the SVR based models (RBF, Linear and Poly) as well as the SGD model have superior success rate (percentage of survival time accuracy). The potential of these models is in improving the accuracy of prediction by better training of the dataset, which undoubtedly depends on the data by itself.

On Fig. 1 is shown the performance in terms of accuracy of the four used machine learning methods. For all applied ML models, results based on our newly introduced

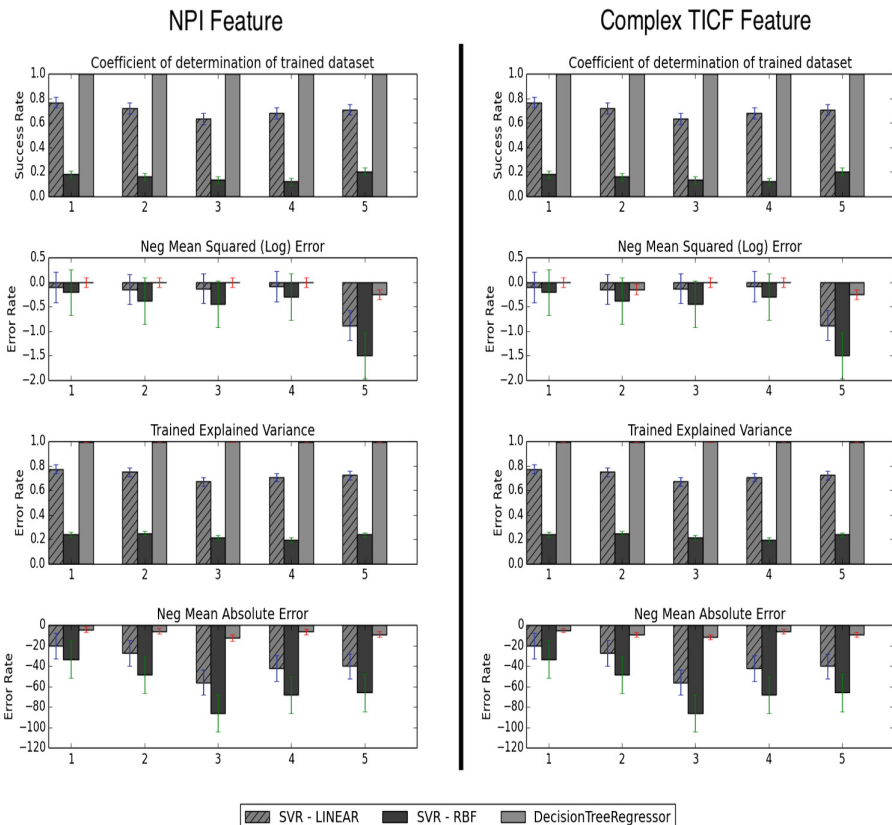


Fig. 2. Accuracy of used machine learning models for average patients' survival time prediction



TICF feature are superior to the results based on the NPI feature in accuracy for prediction of survival time in breast cancer.

Among the models for survival time prediction of breast cancer in our study the SVR linear model has best performance in accuracy. After that is DTR model and the SVR-RBF and SVR-Poly have inferior performance. The SVR-Poly model is excluded from the analysis because of bad performance as compared to the other used models.

For validation of the models or which one of them has less noise level the SVR-RBF and the DTR modules are most accurate (Fig. 2). For this result we applied cross-validation approach based on four parameters for error evaluation: trained  $R^2$  (coefficient of determination), negative mean square log error, explained variance, negative mean absolute error.

The  $R^2$  and the explained variance are related to the accuracy of the used model, while the two others are related to the noise (error) level. The results for accuracy shown on Fig. 2 again underline that the TICF feature have better performance in accuracy. As an outcome of the study we can claim that the SVR-linear and DTR models are more suitable for accurate survival prediction in breast cancer for the studied case.

## 7 Conclusions

We develop a machine learning based approach for survival time prediction in breast cancer. We create a new TICF feature for classification and analysis, which showed best results compared to the existing NPI feature. We proved that by using four machine learning models based on two studied features and compared them. The cross-validation for accuracy of the used models showed better performance for SVR-Linear and decision regression tree with TICF feature.

**Acknowledgements.** The presented work has been funded by the Bulgarian NSF within the “GloBIG: A Model of Integration of Cloud Framework for Hybrid Massive Parallelism and its Application for Analysis and Automated Semantic Enhancement of Big Heterogeneous Data Collections” project, Contract DN02/9 of 17.12.2016, and by the Sofia University SRF within the “Models for semantic integration of biomedical data” project, Contract 80-10-207/26.04.2018.

## References

1. American Cancer Society: Cancer Statistics Center. <http://cancerstatisticscenter.cancer.org>. Accessed 25 May 2018
2. Ivanova, D.: Big data analytics for early detection of breast cancer based on machine learning. AIP Conf. Proc. **1910**(1), 060016 (2017). <https://doi.org/10.1063/1.5014010>
3. Luo, J., Wu, M., Gopukumar, D., Zhao, Y.: Big data application in biomedical research and health care: a literature review. Biomed. Inform. Insights **8**, 1–10 (2016). <https://doi.org/10.4137/BII.S31559>
4. Cruz, J.A., Wishart, D.S.: Applications of machine learning in cancer prediction and prognosis. Cancer Inform. **2**, 59–77 (2006)

5. Weston, A.D., Hood, L.: Systems biology, proteomics, and the future of health care: toward predictive, preventative, and personalized medicine. *J. Proteome Res.* **3**, 179–196 (2004)
6. Hagerty, R.G., Butow, P.N., et al.: Communicating prognosis in cancer care: a systematic review of the literature. *Ann. Oncol.* **16**(7), 1005–1053 (2005). <https://doi.org/10.1093/annonc/mdi211>
7. Futschik, M., Michael, S.: Prediction of clinical behaviour and treatment for cancers. *OMJ Appl. Bioinform.* **2**, 53–58 (2003)
8. Djebbari, A., Liu, Z., Phan, S., Famili, F.: International journal of computational biology and drug design (IJCBDD). In: 21st Annual Conference on Neural Information Processing Systems (2008)
9. Liu, Y.-Q., Wang, C., Zhang, L.: Decision tree based predictive models for breast cancer survivability on imbalanced data. In: 3rd International Conference on Bioinformatics and Biomedical Engineering, pp. 1–4 (2009). <https://doi.org/10.1109/icbbe.2009.5162571>
10. Delen, D., et al.: Predicting breast cancer survivability: a comparison of three data mining methods. *Artif. Intell. Med.* **34**(2), 113–127 (2005). <https://doi.org/10.1016/j.artmed.2004.07.002>
11. Lisboa, H.W., Harris, P., et al.: A Bayesian neural network approach for modelling censored data with an application to prognosis after surgery for breast cancer. *Artif. Intell. Med.* **28**(1), 1–25 (2003). [https://doi.org/10.1016/S0933-3657\(03\)00033-2](https://doi.org/10.1016/S0933-3657(03)00033-2)
12. Seker, H., et al.: Assessment of nodal involvement and survival analysis in breast cancer patients using image cytometric data: statistical, neural network and fuzzy approaches. *Anticancer Res. Int. J. Cancer Res. Treat.* **22**(1), 433–438 (2002)
13. Zhang, H., Guo, Y., Li, Q., George, T.J., Shenkman, E.A., Bian, J.: Data integration through ontology-based data access to support integrative data analysis: a case study of cancer survival. In: 2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), Kansas City, MO, pp. 1300–1303 (2017). <https://doi.org/10.1109/bibm.2017.8217849>
14. Rakha, E.A., Soria, D., Green, A.R., et al.: Nottingham Prognostic Index Plus (NPI+): a modern clinical decision making tool in breast cancer. *Br. J. Cancer* **110**(7), 1688–1697 (2014). <https://doi.org/10.1038/bjc.2014.120>

# **Knowledge Representation, Reasoning and Search**



# Tractable Classes in Exactly-One-SAT

Yazid Boumarafi<sup>(✉)</sup> and Yakoub Salhi

CRIL - CNRS, Artois University, Lens, France  
{boumarafi,salhi}@cril.fr

**Abstract.** In this paper, we aim at proposing a new approach for defining tractable classes for Exactly-One-SAT problem (in short EO-SAT). EO-SAT is the problem of deciding whether a given CNF formula has a model so that each clause has exactly one true literal. Our first tractable class is defined by using a simple property that has to be satisfied by every three clauses sharing at least one literal. In a similar way, our second tractable class is obtained from a property that has to be satisfied by particular sequences of clauses. The proposed tractable classes can, in a sense, be seen as natural counterparts of tractable classes of the maximum independent set problem.

**Keywords:** The EO-SAT problem · Graph theory · Tractable classes

## 1 Introduction

Developing efficient algorithms for solving the satisfiability problem (in short SAT) is an important challenge in computer science in general, and artificial intelligence in particular. In both practical and theoretical points of view, SAT is widely used in different domains like planning and formal verification (see e.g. [14, 20]). SAT consists in checking whether a propositional formula in the conjunctive normal form (in short CNF) admits a satisfying assignment. It was identified as the first NP-complete problem [5], however, it has several tractable classes such as Horn [8], Renamable Horn [16], Q-Horn [1], Krom [15].

Defining tractable classes for NP-complete problems through cross-fertilization between different problems is a central research topic. Indeed, poly-time reducibility is one of the most fundamental tools in complexity theory. Using this tool, new tractable classes can be defined by finding in the source language counterparts of tractable classes in the target language. For instance, in [19], the authors have shown that using the order encoding some tractable CSP instances result in tractable SAT instances. Furthermore, in graph theory, many problems are tractable for graphs belonging to special classes, such as claw-free graphs [9, 17], convex and chordal graphs [7], AT-free graphs [21] and perfect graphs [11] (see [3] for a survey). Tractable classes in graph theory have been used for characterizing tractable classes in CSP [6], however, We can also mention the tractable classes highlighted by different authors in constraint

satisfaction problems (CSP) [6], few works investigated the existence of their counterparts in SAT [2] and its related problems.

In this article, we aim at defining tractable classes for the Exactly-One-SAT problem (in short EO-SAT). This problem consists in deciding whether a given CNF formula has a model so that each clause has exactly one true literal. It is worth mentioning that defining tractable classes of EO-SAT have not received much attention in the literature. Moreover, as shown in this article, even the restriction of EO-SAT to monotone formulæ remains NP-complete, which means that the restriction of EO-SAT to Horn formulæ is also NP-complete. In a sense, this shows the difficulty and the interest of defining tractable classes for EO-SAT compared to SAT. It shows also the need of new approaches in characterizing tractable classes.

In order to define our tractable classes, we establish a simple relationship between the NP-hard problem of maximum independent set and EO-SAT. Indeed, we show that an instance of EO-SAT has a solution if and only if the size of the largest independent set of a particular associated graph is equal to the number of clauses. Thus, thanks to tractability results obtained for the maximum independent set problem (see e.g. [17, 18]), we define two tractable classes for EO-SAT that can be seen as natural counterparts of tractable classes for the maximum independent set problem. In this work, our approach presents tractability results, but most importantly proposes a new way to deal with tractability in EO-SAT.

## 2 Preliminaries

### 2.1 Propositional Satisfiability and EO-SAT Problem

Let us first define the syntax and the semantics of propositional logic. We use the letters  $p, q, r, \dots$  to range over an infinite set of propositional variables, we use  $\mathcal{V}$  to denote that set. Also, we use the Greek letters  $\phi, \psi$  to represent formulæ, the set of *propositional formulæ* is defined inductively started from  $\mathcal{V}$ , the constant  $\perp$  denoting false, the constant  $\top$  denoting true and using the logical connectives  $\neg, \wedge, \vee, \rightarrow$ . Formally, the language of propositional logic is defined by the following:  $\phi ::= p \mid \perp \mid \top \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \phi \rightarrow \phi$ .

Given a formula  $\phi$ , we use  $\mathcal{V}(\phi)$  to denote the set of propositional variables appearing in  $\phi$ . A *Boolean interpretation*  $\mathcal{I}$  of a formula  $\phi$  is defined as a function from  $\mathcal{V}(\phi)$  to  $\{0, 1\}$  (0 corresponds to *false* and 1 to *true*). It is inductively extended to propositional formulæ as usual:  $\mathcal{I}(\perp) = 0$ ,  $\mathcal{I}(\top) = 1$ ,  $\mathcal{I}(\phi \wedge \psi) = \min(\mathcal{I}(\phi), \mathcal{I}(\psi))$ ,  $\mathcal{I}(\neg\phi) = 1 - \mathcal{I}(\phi)$ ,  $\mathcal{I}(\phi \vee \psi) = \max(\mathcal{I}(\phi), \mathcal{I}(\psi))$ ,  $\mathcal{I}(\phi \rightarrow \psi) = \max(1 - \mathcal{I}(\phi), \mathcal{I}(\psi))$ .

A *model* of a formula  $\phi$  is a boolean interpretation  $\mathcal{I}$  that satisfies  $\phi$ , i.e.  $\mathcal{I}(\phi) = 1$ . It is worth noticing that we can restrict the language to the connectives  $\neg$  and  $\wedge$ , since we have the following equivalences:  $p \vee q \equiv \neg(\neg p \wedge \neg q)$  and  $p \rightarrow q \equiv \neg p \vee q$ .

Let us now define the *conjunctive normal form* (CNF) representation of the propositional formulæ. A propositional formula in conjunctive normal form is a conjunction ( $\wedge$ ) of clauses, where a *clause* is a disjunction ( $\vee$ ) of literals. A *literal* is a propositional variable ( $p$ ), called positive literal, or a negated propositional variable ( $\neg p$ ), called negative literal, we use  $\mathcal{L}$  to denote the set of literals appearing in a CNF formula. A CNF formula can also be seen as a set of clauses, and a clause as a set of literals. Every propositional formula can be translated to CNF using Tseitin's linear encoding [22].

Let us mention some types of clauses. A clause is called *unit* if it contains exactly one literal whereas *binary* ones contain two literals. A clause is called *positive* (resp. *negative*) if it contains only positive (resp. negative) literals.

The unit propagation method (in short UP). UP is a linear procedure that recursively simplifies a CNF formula by propagating the literal in the unit clauses. Given a unit clause, let  $l$  be its literal, the UP procedure consists in removing every clause containing the literal  $l$  and deleting  $\neg l$  from the remaining clauses in the formula. The application of UP method leads to a new set of clauses that is equivalent to the original one. Obviously, a CNF formula is satisfiable if and only if the CNF formula obtained from it by applying the UP mechanism is satisfiable. From now on, we only consider formulæ with no unit clauses.

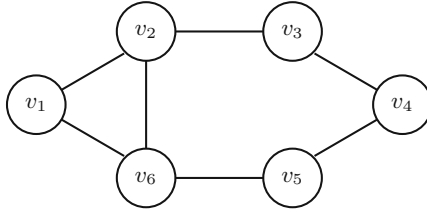
The SAT problem consists in checking whether a given CNF formula has a model, in other words, checking if there exists a boolean interpretation that satisfies all the clauses in the CNF formula or not.

SAT is an NP-complete problem [5], however, some fragments exhibit a polynomial-time algorithms for SAT. Among them, let us mention the Horn fragment, which is made of Horn clauses only. A Horn [8] (resp. reverse Horn) clause contains at most one positive (resp. negative) literal. Renamable Horn clauses also form polynomial fragments [16]: renamable Horn clauses are clauses that can be transformed into Horn ones by systematically replacing some negative literals by new boolean variables, Krom: restriction to binary clauses [15].

We consider in this work the problem of Exactly-One-SAT (in short EO-SAT) which is a generalization of 1-in-3 SAT. It consists in deciding whether a given CNF formula has a model such that each clause has exactly one true literal. In the same way as SAT, EO-SAT is NP-Complete.

## 2.2 Maximum Independent Set Problem

Given an undirected graph  $G = (V, E)$ , an *independent set* of  $G$  is a subset of vertices  $S \subseteq V$  such that no two vertices in  $S$  are adjacent, i.e., for all  $v, v' \in S$  with  $v \neq v'$ ,  $\{v, v'\} \notin E$ . An independent set is called *maximal* if it is not a subset of any larger independent set. A *maximum independent set* is an independent set of largest size. We here use  $\alpha(G)$  to denote the value of the maximum size of the independent sets.



**Fig. 1.** An undirected graph

*Example 1.* Let us consider the undirected graph given in Fig. 1:

The set of vertices  $S = \{v_1, v_3, v_5\}$  is a maximum independent set for the undirected graph since the three vertices are not adjacent and we also have  $\alpha(G) = 3$ .

The problem of finding a maximum independent set for a graph is an NP-Hard problem. However, there are several classes of graphs for which the problem can be solved in polynomial-time, such as claw-free graphs [17]. Let us recall that a graph is claw-free if no vertex has three pairwise nonadjacent neighbors, i.e., for all  $v \in V$ , if there exist three distinct vertices  $v_1, v_2, v_3 \in V$  s.t.  $\{\{v, v_1\}, \{v, v_2\}, \{v, v_3\}\} \subseteq E$ , then at least one of the edges  $\{v_1, v_2\}$ ,  $\{v_2, v_3\}$  and  $\{v_1, v_3\}$  is in  $E$ .

The maximum independent set is also tractable for other classes, such as Perfect graphs [12], in particular, chordal graphs [4]. A *chordal graph* is a graph in which every cycle of four or more vertices has an edge connecting two vertices of the cycle but that is not part of the cycle, this edge is called a *chord*.

### 3 Motivations

A monotone formula is a CNF formula such that all its literals are either positive or negative. Finding a model of a monotone formula is clearly a linear-time task. Indeed, it suffices to set all the literals to either true or false to obtain a model. However, EO-SAT restricted to monotone formulæ is not tractable. In a sense, this shows the difficulty and the interest of defining tractable classes for EO-SAT compared to SAT. It shows also the need of new approaches in characterizing tractable classes.

**Theorem 1.** *The restriction of EO-SAT to the monotone formulæ is NP-complete.*

*Proof.* Clearly EO-SAT is in NP, since one can check whether a Boolean interpretation is a solution or not in linear time.

To show NP-hardness, we provide a reduction of the three-coloring problem. Given an undirected graph  $G = (V, E)$ , a 3-coloring of  $G$  corresponds to a

partition of  $V$  into 3 sets  $P = \{V_{c_1}, V_{c_2}, V_{c_3}\}$  so that no two vertices in a same subset are adjacent, and  $V_{c_1}$ ,  $V_{c_2}$  and  $V_{c_3}$  refer to three different colors  $c_1$ ,  $c_2$  and  $c_3$  respectively.

In order to encode the 3-coloring problem into EO-SAT restricted to monotone formulæ, we associate to each vertex  $v \in V$  and each color  $c_i$  a distinct propositional variable denoted  $p_v^{c_i}$  (the variable  $p_v^{c_i}$  is set to true iff  $v$  is colored with  $c_i$ ). Further, we associate to each edge  $e \in E$ , three distinct variables denoted  $q_1^e, q_2^e, q_3^e$ . Then, the encoding is defined as follows:

$$\bigwedge_{v \in V} p_v^{c_1} \vee p_v^{c_2} \vee p_v^{c_3} \quad (1)$$

$$\bigwedge_{\substack{i=1..3 \\ e=\{v,v'\} \in E}} (p_v^{c_i} \vee q_1^e \vee q_2^e) \wedge (q_2^e \vee q_3^e) \wedge (p_{v'}^{c_i} \vee q_3^e) \quad (2)$$

It is easy to see that the formula (1) represents the fact that each vertex is colored with exactly one color. The formula (2) represents the fact that each adjacent vertices cannot be colored with the same color. Indeed, assume that there exists a solution s.t. there exists  $e = \{v, v'\} \in E$  and  $i \in 1..3$  s.t.  $p_v^{c_i}$  and  $p_{v'}^{c_i}$  are assigned to true. Thus,  $q_1^e$  and  $q_2^e$  are both assigned to false and, as a consequence,  $q_3^e$  is assigned to true. The latter property means that  $p_{v'}^{c_i}$  is assigned to false and we get a contradiction. Furthermore, one can easily build a solution of the encoding from any 3-coloring.  $\square$

Knowing that the class of monotone negative formulæ is a sub-class of that of Horn formulæ, we obtain from Theorem 1 the following corollary.

**Corollary 1.** *The restriction of EO-SAT to the Horn formulæ is NP-complete.*

It is worth noticing that the class of binary CNF formulæ remains tractable in EO-SAT. Indeed, given a binary CNF formula  $\phi \equiv (l_1 \vee l'_1) \wedge \dots \wedge (l_n \vee l'_n)$ ,  $\phi$  has a solution in EO-SAT if and only if the binary formula  $(l_1 \vee l'_1) \wedge \dots \wedge (l_n \vee l'_n) \wedge (\bar{l}_1 \vee \bar{l}'_1) \wedge \dots \wedge (\bar{l}_n \vee \bar{l}'_n)$  is satisfiable, and the latter can be decided in polynomial time.

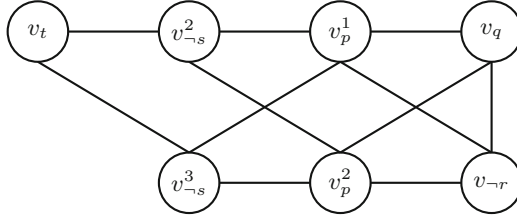
## 4 EO-SAT and the Maximum Independent Set Problem

We provide here an approach for solving EO-SAT by using the maximum independent set problem. It consists simply in associating to every EO-SAT instance an undirected graph so that the considered EO-SAT instance has a solution if and only if the size of the largest independent set is equal to the number of clauses.

Given a CNF formula  $\phi$  and a literal  $l$  occurring in  $\phi$ , we associate to  $l$  a set of  $n$  distinct elements  $S_l = \{v_l^1, \dots, v_l^n\}$  where  $n$  is the number of clauses containing  $l$ . Further, we use  $G_\phi = (V, E)$  to denote its corresponding undirected graph where  $V = \bigcup_{l \in \mathcal{L}(\phi)} S_l$ , and for all  $v_l^i, v_{l'}^j \in V$ ,  $\{v_l^i, v_{l'}^j\} \in E$  if and only if there is a clause in  $\phi$  containing both  $l$  and  $l'$ .



*Example 2.* The graph described in Fig. 2 corresponds to the following EO-SAT instance  $\phi \equiv (p \vee q \vee \neg r) \wedge (p \vee \neg s) \wedge (\neg s \vee t)$ . Assigning the variables  $p$  and  $t$  to *true* allows us to get a solution of  $\phi$ .



**Fig. 2.** The graph associated to the formula in Example 2

**Proposition 1.** *Let  $\phi$  be an EO-SAT instance and  $M$  a maximum independent set of the graph  $G_\phi = (V, E)$ . Then,  $S_l \cap M \neq \emptyset$  iff  $S_l \subseteq M$  for every  $l \in \mathcal{L}(\phi)$ .*

*Proof.* This property is a direct consequence of the fact that the vertices in  $S_l$  have all the same adjacent vertices. □

The following theorem allows us to establish a direct relationship between EO-SAT and the maximum independent set problem.

**Theorem 2.** *Let  $\phi$  be an EO-SAT instance. Then,  $\phi$  has a solution iff  $\alpha(G_\phi) = n$  where  $n$  is the number of clauses occurring in  $\phi$ .*

*Proof.* Regarding the *if part*, the base idea consists in assigning to true only the literals that are associated with the vertices occurring in a maximum independent set. More precisely, given a maximum independent set  $M$  of  $G_\phi$ , we define the Boolean interpretation  $\mathcal{I}_M$  using  $M$  as follows:  $\mathcal{I}_M(p) = 1$  iff  $S_p \subseteq M$ . Clearly, for every two distinct literals  $l$  and  $l'$  occurring in the same clause, we get  $S_l \not\subseteq M$  or  $S_{l'} \not\subseteq M$ . As a consequence, the size of  $M$  corresponds to the number of satisfied clauses by  $\mathcal{I}_M$  in  $\phi$ . Thus,  $\mathcal{I}_M$  is a solution of  $\phi$  since  $\alpha(G_\phi) = n$ .

To prove the *only if part*, we only need to build a maximum independent set from every solution of  $\phi$ . Indeed, given a solution  $\mathcal{I}$  of  $\phi$ , the set  $M_{\mathcal{I}} = \bigcup_{l \in \mathcal{L}(\phi), \mathcal{I}(l)=1} S_l$  is a maximum independent set of  $\phi$ . Moreover, the size of  $M_{\mathcal{I}}$  is clearly equal to the number of satisfied clauses  $n$ . Knowing that the size of the independent sets is bounded by  $n$ ,  $\alpha(G_\phi) = n$  holds. □

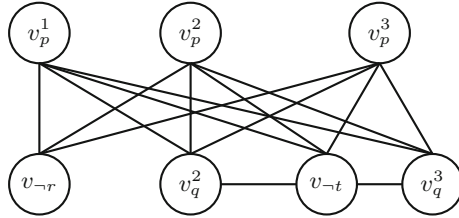
## 5 Tractable Classes

In this section, we introduce new tractable classes for EO-SAT. The base idea consists in using the relationship described in Theorem 2 between EO-SAT and the maximum independent set problem. Indeed, our tractable classes for EO-SAT can be seen as natural counterparts of tractable classes for the maximum independent set problem.

### 5.1 Class of CF-formulae

We here describe a tractable class that we obtain from the class of claw-free graphs for the maximum independent set problem.

**Definition 1 (CF-formula).** A CF-formula  $\phi$  is a CNF formula where for every three distinct clauses  $c_1, c_2$  and  $c_3$  sharing at least one literal  $l$  ( $l \in c_1 \cap c_2 \cap c_3$ ), there are three distinct integers  $i, j, k \in 1..3$  s.t.  $c_i \subseteq c_j \cup c_k$ .



**Fig. 3.** The graph  $G_\phi$  associated to  $\phi$

Consider for instance the CNF formula  $\phi \equiv (p \vee \neg r) \wedge (p \vee q) \wedge (p \vee q \vee \neg t)$ . It is a CF-formula since the literals in the clause  $(p \vee q)$  belong to the set of the literals occurring in the clauses  $(p \vee \neg r)$  and  $(p \vee q \vee \neg t)$ .

**Theorem 3.** *The restriction of EO-SAT to the CF-formulae is tractable.*

*Proof.* Theorem 2 shows how we can solve EO-SAT by finding a maximum independent set in a particular graph. Further, we know that the maximum independent set problem is tractable in the class of claw-free graphs.

*Claim:* For all CF-formula  $\phi$ ,  $\mathcal{G}_\phi$  is a claw-free graph.

Let  $\phi$  be a CF-formula. Assume that the graph associated to  $\phi$  contains a claw as an induced subgraph. Then, there are three clauses of the form  $c_1 = l \vee l_1 \vee c'_1$ ,  $c_2 = l \vee l_2 \vee c'_2$  and  $c_3 = l \vee l_3 \vee c'_3$  s.t. there is a clause in  $\phi$  containing at least two literals in  $\{l_1, l_2, l_3\}$ . The vertices involving in the considered claw are associated to the literals  $l, l_1, l_2$  and  $l_3$ . Using the definition of CF-formula, we consider w.l.o.g.  $c_1 \subseteq c_2 \cup c_3$ . Thus, there exists a clause  $c \in \{c_2, c_3\}$  s.t. at least two literals in  $\{l_1, l_2, l_3\}$  occur in  $c$  and, as a consequence, we get a contradiction.  $\square$

Let us consider again the formula  $\phi$  in the previous example. Its associated graph  $G_\phi$  is described in Fig. 3. One can easily see that this graph is claw-free. Further, the set of vertices  $M = \{v_p^1, v_p^2, v_p^3\}$  is a maximum independent set for  $G_\phi$ . Using Theorem 2, we know that  $\phi$  is satisfiable since it contains three clauses and  $\alpha(G_\phi) = 3$ . Hence, the solution of  $\phi$  built from the maximum independent set  $M$  is  $\{p\}$ .

It is worth mentioning that the problem of checking whether an EO-SAT instance is a CF-formula or not is tractable. Indeed, given an EO-SAT instance with  $n$  clauses, it suffices to check the condition in Definition 1 for every distinct three clauses ( $\mathcal{O}(n^3)$  times).

### 5.2 Class of LC-formula

We here describe our second tractable class of EO-SAT. In the same way as the class of CF-formulae, this class can be seen as a counterpart of the class of chordal graphs, which is tractable for the maximum independent set problem.

**Definition 2 ( $k$ -Literal-Cycle).** Let  $\phi$  be a CNF formula. A  $k$ -literal-cycle of  $\phi$  is a sequence of  $k$  pairs of literals  $\mathcal{C} = s_1 = \{l_1, l_2\}, s_2 = \{l_2, l_3\}, \dots, s_{k-1} = \{l_{k-1}, l_k\}, s_k = \{l_k, l_1\}$  such that, for all  $i \in 1..k$ , there exists a clause  $c \in \phi$  s.t.  $s_i \subseteq c$ .

For instance, consider the CNF formula  $\phi \equiv (l_1 \vee l_2) \wedge (l_2 \vee l_3 \vee l_4 \vee l_5) \wedge (l_5 \vee l_1) \wedge (l_1 \vee l_4)$ . The three first clauses form a clausal-cycle, and this allows us to get the 3-literal-cycle  $\mathcal{C}_1 = \{l_1, l_2\}, \{l_2, l_5\}, \{l_5, l_1\}$ . The other literal-cycles of  $\phi$  modulo permutations are:

- $\mathcal{C}_2 = \{l_1, l_2\}, \{l_2, l_3\}, \{l_3, l_4\}, \{l_4, l_5\}, \{l_5, l_1\}$ ;
- $\mathcal{C}_3 = \{l_1, l_2\}, \{l_2, l_3\}, \{l_3, l_4\}, \{l_4, l_1\}$ ;
- $\mathcal{C}_4 = \{l_1, l_2\}, \{l_2, l_4\}, \{l_4, l_1\}$ .

It is worth mentioning that the clause  $(l_2 \vee l_3 \vee l_4 \vee l_5)$  is involved three times in the literal-cycle  $\mathcal{C}_2$ .

**Definition 3 (LC-formula).** An LC-formula  $\phi$  is a CNF formula where for every  $k$ -literal-cycle  $\mathcal{C} = s_1, s_2, \dots, s_k$  with  $k > 3$ , there exists a clause  $c \in \phi$  s.t.  $\{l, l'\} \subseteq c$ , and  $\exists s, s' \in \mathcal{C}$  s.t.  $l \in s$  and  $l' \in s'$ , and  $\forall s \in \mathcal{C}, \{l, l'\} \neq s$ .

Consider again the CNF formula  $\phi$  described in the previous example. There are two  $k$ -literal-cycles modulo permutations such that  $k > 3$ :  $\mathcal{C}_2$  and  $\mathcal{C}_3$ . The fact that there is a clause in  $\phi$  containing the literals  $l_2$  and  $l_4$  allows us to deduce that  $\phi$  is an LC-formula.

**Theorem 4.** *The restriction of EO-SAT to the LC-formulae is tractable.*

*Proof.* Using Theorem 2 and knowing that the maximum independent set problem is tractable in the case of a chordal graphs, we only need to show that  $G_\phi$  is a chordal graph for every LC-formula  $\phi$ . Assume that there exists a  $k$ -cycle  $\mathcal{C}$  in  $G_\phi$  where  $k > 3$ . Then, there exists a  $k$ -literal-cycle  $\mathcal{C}' = s_1, s_2, \dots, s_k$  in  $\phi$ . Using Definition 3, we know that there exists a clause  $c \in \phi$  s.t.  $\{l, l'\} \subseteq c$ , and  $\exists s, s' \in \mathcal{C}'$  s.t.  $l \in s$  and  $l' \in s'$ , and  $\forall s \in \mathcal{C}', \{l, l'\} \neq s$ . As a consequence, the  $k$ -cycle  $\mathcal{C}$  admits a chord. □

Regarding the recognition problem of LC-formulae, it suffices to check whether the graph associated to a CNF formula is chordal or not, which is a tractable task [4].

## 6 Related Works

Several results are obtained thanks to polynomial reductions between different NP-Complete or NP-Hard problems (e.g. [13]). Let us mention the work of Fischer et al. [10], where they studied algorithms for #SAT and its generalized version #GENSAT (the problem of computing the number of satisfying assignments of a set of (generalized) propositional clauses). It is well known, that, given a graph of tree-width  $k$ , a  $k$ -tree decomposition can be found in polynomial time. The authors considered the clauses given by their incidence graph, signed bipartite graph and derived graphs, they presented an algorithm for #GENSAT for formulæ of bounded tree-width  $k$ . Their results are essentially for #SAT, and hence also for SAT.

In [2], the authors proposed new tractable classes for SAT by considering propositional formulæ in conjunctive normal form (CNF) made of a set of Positive clauses and a set of Binary Negative clauses (PBN). The PBN form is suitable for establishing connections between SAT and problems in graph theory and also constraint satisfaction problem (CSP). In the same way as in this article, tractable classes of formulæ in PBN form are defined using tractable classes of the maximum independent set problem.

## 7 Conclusion and Perspectives

In this paper, we mainly proposed an approach for characterizing tractable classes in EO-SAT. We first showed that the restriction of EO-SAT to monotone formulæ remain NP-complete. Then, we provided a reduction of EO-SAT in the maximum independent set problem. Then, we proposed two tractable classes of EO-SAT that can be seen as natural counterparts of tractable classes of the maximum independent set problem.

As a future work, we intend to define other tractable classes for EO-SAT using tractability results in graph theory.

## References

1. Boros, E., Hammer, P.L., Sun, X.: Recognition of q-Horn formulæ in linear time. *Ann. Math. Artif. Intell.* **1**, 21–32 (1990)
2. Boumarafi, Y., Sais, L., Salhi, Y.: From SAT to maximum independent set: a new approach to characterize tractable classes. In: *LPAR-21 21st International Conference on Logic for Programming, Artificial Intelligence and Reasoning*, Maun, Botswana, 7–12 May 2017, pp. 286–299 (2017)
3. Brandstädt, A., Le, V.B., Spinrad, J.P.: *Graph Classes: A Survey*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA (1999)
4. Chandrasekharan, N., Iyengar, S.S.: NC algorithms for recognizing chordal graphs and  $k$  trees. *IEEE Trans. Comput.* **37**(10), 1178–1183 (1988)
5. Cook, S.A.: The complexity of theorem-proving procedures. In: *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, pp. 151–158 (1971)

6. Cooper, M.C., Zivny, S.: A new hybrid tractable class of soft constraint problems. In: Principles and Practice of Constraint Programming - CP 2010 –16th International Conference, CP 2010, St. Andrews, Scotland, UK, 6–10 September 2010, Proceedings, pp. 152–166 (2010)
7. Damaschke, P., Müller, H., Kratsch, D.: Domination in convex and chordal bipartite graphs. *Inf. Process. Lett.* **36**(5), 231–236 (1990)
8. Dowling, W.F., Gallier, J.H.: Linear-time algorithms for testing satisfiability of propositional Horn formulae. *J. Logic Program.* **1**(3), 267–284 (1984)
9. Faudree, R., Flandrin, E., Ryjck, Z.: Claw-free graphs a survey. *Discrete Math.* **164**(13), 87–147 (1997)
10. Fischer, E., Makowsky, J.A., Ravve, E.V.: Counting truth assignments of formulas of bounded tree-width or clique-width. *Discrete Appl. Math.* **156**(4), 511–529 (2008)
11. Golubic, M.C. (ed.): Algorithmic Graph Theory and Perfect Graphs. *Annals of Discrete Mathematics* (2004)
12. Grötschel, M., Lovász, L., Schrijver, A.: The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica* **1**(2), 169–197 (1981)
13. Karp, R.M.: Reducibility among combinatorial problems. In: Miller, R.E., Thatcher, J.W., Bohlinger, J.D. (eds.) *Complexity of Computer Computations*, pp. 85–103. Springer, Boston (1972). [https://doi.org/10.1007/978-1-4684-2001-2\\_9](https://doi.org/10.1007/978-1-4684-2001-2_9)
14. Kautz, H.A., Selman, B.: Planning as satisfiability. In: ECAI, pp. 359–363 (1992)
15. Krom, M.R.: The decision problem for a class of first-order formulas in which all disjunctions are binary. *Math. Logic Q.* **13**(1–2), 15–20 (1967)
16. Lewis, H.R.: Renaming a set of clauses as a Horn set. *J. ACM* **25**(1), 134–135 (1978)
17. Minty, G.J.: On maximal independent sets of vertices in claw-free graphs. *J. Comb. Theory, Ser. B* **28**(3), 284–304 (1980)
18. Okamoto, Y., Uno, T., Uehara, R.: Counting the number of independent sets in chordal graphs. *J. Discrete Algorithms* **6**(2), 229–242 (2008)
19. Petke, J., Jeavons, P.: The order encoding: from tractable CSP to tractable SAT. In: Sakallah, K.A., Simon, L. (eds.) SAT 2011. LNCS, vol. 6695, pp. 371–372. Springer, Heidelberg (2011). [https://doi.org/10.1007/978-3-642-21581-0\\_34](https://doi.org/10.1007/978-3-642-21581-0_34)
20. Prasad, M.R., Biere, A., Gupta, A.: A survey of recent advances in sat-based formal verification. *STTT* **7**(2), 156–173 (2005)
21. Stacho, J.: 3-colouring AT-free graphs in polynomial time. In: Cheong, O., Chwa, K.-Y., Park, K. (eds.) ISAAC 2010. LNCS, vol. 6507, pp. 144–155. Springer, Heidelberg (2010). [https://doi.org/10.1007/978-3-642-17514-5\\_13](https://doi.org/10.1007/978-3-642-17514-5_13)
22. Tseitin, G.S.: On the complexity of derivations in the propositional calculus. In: Slesenko, H.A.O. (ed.) *Structures in Constructives Mathematics and Mathematical Logic, Part II*, pp. 115–125 (1968)



# Semantic Meta-search Using Cohesion Network Analysis

Ionut Daniel Chelcioiu<sup>1</sup>, Dragos Corlatescu<sup>1</sup>,  
Ionut Cristian Paraschiv<sup>1,2</sup>, Mihai Dascalu<sup>1,2,3</sup>✉,  
and Stefan Trausan-Matu<sup>1,2,3</sup>

<sup>1</sup> Department of Computer Science,  
University Politehnica of Bucharest, 060042 Bucharest, Romania  
daniel.chelcioiu@yahoo.com,  
dragos.corlatescu@cti.pub.ro, {ionut.paraschiv,  
mihai.dascalu, stefan.trausan}@cs.pub.ro  
<sup>2</sup> Research Technology S.R.L., Sos. Virtutii, nr. 19D, Bucharest, Romania  
<sup>3</sup> Academy of Romanian Scientists,  
Splaiul Independenței 54, 050094 Bucharest, Romania

**Abstract.** Online searching is one of the most frequently performed actions and search engines need to provide relevant results, while maintaining scalability. In this paper we introduce a novel approach grounded in Cohesion Network Analysis in the form of a semantic search engine incorporated in our *Hub-Tech* platform. Our aim is to help researchers and people unfamiliar with a domain find meaningful articles online, relevant for their project scope. In addition, we integrate state-of-the-art technologies to ensure scalability and low response time, namely SOLR – for data storage and full-text search functionalities – and Akka – for parallel and distributed processing. Preliminary validations denote promising search results, the software being capable to suggest articles in approximately the same way as humans consider them most appropriate – 75% are close results and top 20% are identical to user recommendations. Moreover, *Hub-Tech* recommended more suitable articles than Google Scholar for our specific task of searching for articles related to a detailed description given as input query (50 + words).

**Keywords:** Meta-search engine · Semantic search · Parallel processing  
Akka · SOLR

## 1 Introduction

Relevant online information is becoming more and more difficult to retrieve as the amount of data stored online and the number of available articles is growing at an exponential rate. Many companies have thrived because they managed to create platforms that enable indexing and searching through unstructured information found online (e.g., Google, Microsoft – Bing, Yahoo – Yahoo Search), and most of the largest software companies have built their own search engines.

Researchers and people unfamiliar with a domain are investing precious time in searching for relevant scientific papers in a specific field of study. More and more

papers are published on diverse topics and can be consulted for gathering information. However, with the high diversity of available information, finding relevant papers on a given subject has become a cumbersome task. Traditional keyword-based approaches for finding papers are not very efficient since people usually search for specific information that cannot be discovered only by providing a limited list of concepts. Moreover, the probability of finding relevant papers given a list of keywords decreases when additional words are given. Surely, information can be found by performing subsequent searches with different combinations of keywords that the user considers suitable; results are filtered, and a set of relevant articles is eventually determined. Nevertheless, semantic searches are more accurate when more words are given, increasing the overall semantic context, and eventually determining more relevant results.

Our current work relies on open-source Web 3.0 technologies and is aimed at suggesting relevant scientific papers starting from detailed user queries (e.g., summary of project objectives) written as unstructured natural language texts. The user queries are modeled using the Cohesion Network Analysis [1] and represented into various semantic models which enable in-depth comparisons to other semantic resources.

This article describes our software platform *Hub-Tech* employed for searching scientific papers, which leverages semantic search instead of keywords matching. The second section presents key points for developing search and meta-search engines. Afterwards, the third section introduces the processing architecture that ensures our semantic search goals. Preliminary validation experiments with corresponding statistics about the relevance of the semantic search findings are presented in the fourth section, while section five concludes the paper.

## 2 State of the Art

Information Retrieval (IR) focuses on finding relevant information from various data sources, generally unstructured, starting from a given query [2]. When focusing on the IR sub-domain of searching for online documents or articles, the processing pipeline usually considers the following stages: data is crawled (and web crawling is most frequently used), indexed for rapid access, followed by query tokenization and search in the indexed resources. Another approach is employed by meta-search engines which do not index themselves online data, but rely on a combination of search results from other engines. The main difference between the two types of search engines is that the general-purpose search engines are mostly looking for syntactic similarities, while meta-search engines are more focused on finding similarities at the semantic level.

Several meta-search engines were proposed in time. For example, Helios [3] is one of the first engines that relied on the data gathered from other 18 search engines, among which the most important ones were Google News, Google Scholar, Yahoo! and AOL Search. In their research, the authors found that search engines provided different results for the same query, claiming that Google and Yahoo, two of the most used search engines at that time, shared only a small portion of their results. Thus, aggregating results from multiple search engines represented a suitable approach. Helios provided a web interface in which the user inserts a query that is subsequently processed by a Local

Query Parser, and an Emitter is responsible for converting it into a format that was understood by other search engines. After the search engines were queried, their results were combined and returned to the user. Helios ensured high-performance by using asynchronous I/O and parallel TCP connections, as well as high extensibility because the integration of new remote engines was quite straightforward. Nowadays, things drastically changed as Google does not allow meta-search on top of their engine. A comparison between modern search and meta-search engines has been made by Mohammedsmeil and Naraghian [4] who performed a side-by-side evaluation of the relevance corresponding to the retrieved documents. The queries included keywords from the dentistry domain and the results showed that, although the meta-search engines are newer, they can be used by both amateurs and experts with satisfactory results for both user categories.

In addition to the previous categories of search engines, semantic search engines that consider the words' semantics were developed for various fields. Mangold [5] performed a first thorough analysis of 22 different semantic search models while considering architecture, coupling, transparency, user context, query modifications, ontology structure, and technology. The results of his findings are still relevant as his analysis was meant to discover which semantic approaches deserve further attention, as well as to create a list of important topics to be covered in the future development of semantic search. In our opinion, some of the most important findings from his study are presented below:

- *meta-semantic search*: little attention has been directed towards building a semantic search approach based on other semantic search engines, although many semantic search engines emerged.
- *user acceptance*: the number of relevant results increases with the rate of interaction with the user, but a tradeoff should be made since the user most likely does not want to spend too much time interacting with the search engine.
- *ranking*: although ranking is one of the most important tasks in a search engine, little attention has been directed into ranking the documents from specific ontologies when assessing the relevance of a result.
- *integration with Data/Content Management Systems*: from his survey, neither of the approaches makes use of the benefits of integrating with a data or a content management system, the author stating that the information stored by these systems could provide important data for follow-up semantic analyses.
- *performance*: the usage of semantic analysis requires much more processing time and resources, thus resulting a time penalty for the user.

As an example, Cohen, Mamou, Kanza and Sagiv [6] introduced XSearch, a semantic search engine capable of returning fragments from related documents using a simple query language. First, the authors emphasize two drawbacks of traditional search engines, namely their impossibility to take advantage of the meta-tags and the fact that results are links and not actual documents, meaning that users cannot search inside the document for required information. Afterwards, the authors propose a query syntax to easily highlight if certain keywords need to exist within the results. A query example would be “+title: *Odyssey*, +Tyson” in which the form “+label: word” suggests that relevant documents must have in their title the word *Odyssey* and other fields



must also contain the word *Tyson*. The syntax of the query is highly extensible since labels are optional and additional ones can be defined. XESearch [6] ranks results by assigning weights to every word using a modified *tf-idf* (term frequency, inverse document frequency). Another metric taken into account when ranking answers is the weight associated to every label (if the labels are present in the query). Since a query retrieves multiple results, the XESearch engine reorders the results by leveraging the documents' structures and their content.

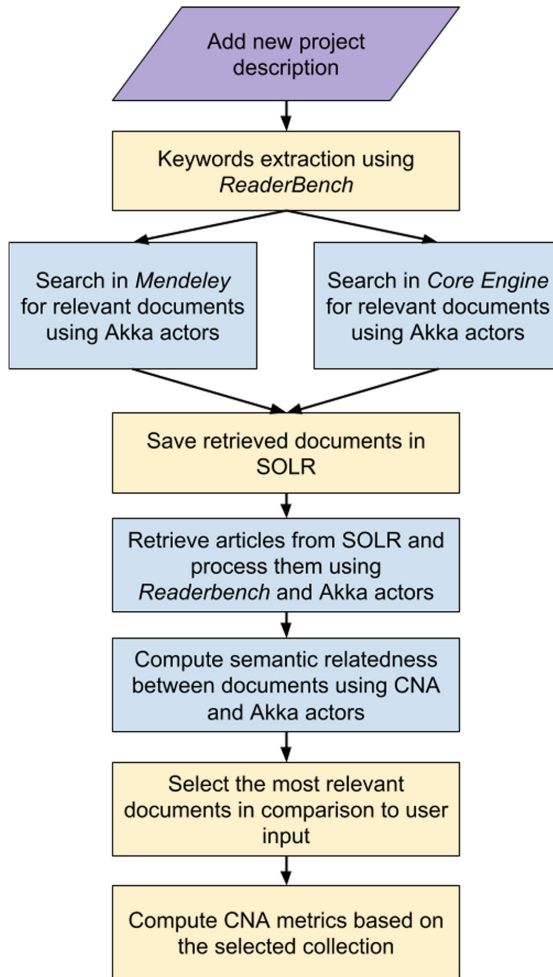
Another example is SICH (Semantic Illegal Content Hunter) [7], which can be used to detect illegal content on the Internet. Besides the proposed architecture for building a generic semantic search engine, several search methods are also considered, such as: linguistic search, spatial search (using geolocation), events-based search and searches based on a corpora selection.

A combination of a semantic search engine and a meta-search engine was proposed by Mukhopadhyay, Sharma, Joshi, Pagare and Palwe [8]. Their search engine (SemanTelli) relies on three other engines: DuckDuckGo, Hakia and SenseBot. Each search engine has an initial weight which is taken into account when aggregating the results. The user inserts a composed query, which is split into combinations of meaningful keywords and sent to the aforementioned search engines. This split is performed after removing stopwords (e.g., “the”, “or”, “and”) and applying suffix stripping (ex. “usage” will be stripped and will result “use”). Afterwards, the ranking algorithm counts how many of the keywords are present in each result in order to assess importance. The sequence of keywords is taken into account if the keywords' count is equal for two or more resources.

Our *Hub-Tech* platform [9] integrates the functionalities provided by the *ReaderBench* framework [10, 11] in term of text analysis. *ReaderBench* makes use of Cohesion Network Analysis (CNA) [1] which provides an in-depth discourse structure centered on the cohesive links between text segments and spanning across the text. *ReaderBench* integrates several advanced Natural Language Processing (NLP) techniques and is used for extracting keywords and for computing the semantic relatedness between two documents. In terms of semantic models, *ReaderBench* incorporates Latent Semantic Analysis [12], Latent Dirichlet Allocation [13], word2vec [14], alongside semantic distances between words within lexicalized ontologies such as WordNet [15].

### 3 The *Hub-Tech* Semantic Processing Architecture

Our semantic meta-search functionality is integrated within the *Hub-Tech* platform [9] and it enables users to enter a project description and receive recommendations regarding current state-of-the-art approaches. The recommendations represent articles retrieved from various online databases (namely *Mendeley* – <http://dev.mendeley.com/methods> – and *Core Engine* – <https://core.ac.us/docs/> – in this release) that are semantically related to the user's input. Figure 1 presents the application's workflow. Users submit an online form in which they input the description corresponding to a new idea or research project. The first step consists of extracting the most relevant keywords (only content words) from the user's text using the *ReaderBench* framework; the result



**Fig. 1.** *Hub-Tech* processing pipeline.

is a word list trimmed to a preset number between 5 and 10. This range of keywords turned out to be most relevant for the text, it does not request extensive computational power for follow-up processes, and we observed that no more than 10 keywords are required to describe the overview of a paper or of an abstract. The following step represents the semantic meta-search on *Mendeley* [16] and *Core Engine* [17] using the previous keywords and Akka actors (<https://doc.akka.io/docs/akka/2.5/actors.html>), as well as saving the articles with their corresponding metadata in the SOLR database (<http://lucene.apache.org/solr/>). Afterwards, articles are processed in a concurrent manner using Akka actors for generating *ReaderBench* documents that have undergone the NLP analyses. Then, documents are ranked in descending order of semantic relatedness between the input query and the converted documents. As this is a time-consuming process, Akka actors were employed for the third time to parallelize the

evaluation. In the end, the most relevant documents are returned to the user, alongside their semantic distances, and are used to create rich user interfaces to visualize the results corresponding to the 2-mode CNA graph that depicts the interactions between the most relevant authors and their corresponding articles [18].

Both *Mendeley* and *Core Engine* provide REST APIs for retrieving articles. To ensure that these services are not cluttered with requests, a small timeout (200 ms) was added between REST calls. *Mendeley* handles the management of scientific articles and its service requires OAuth2 [19] authentication with the following request parameters: keyword(s), number of pages to be retrieved, and number of documents per page. The last two parameters are useful because they enable results pagination. The results returned by *Mendeley* contain the following information: title, authors, abstract, publication date, publisher, keywords, reader count, publisher and others which are not used in our platform. Similar to *Mendeley*, the *Core Engine* web service gathers multiple scientific articles from different sources and makes them freely available. The *Core Engine* service is accessible by making HTTP POST requests, using a list of keywords and pagination as parameters. The returned information contains the title, authors, content, keywords, abstract or description, and full content (an advantage over the Core Engine service).

Searching for articles is a time-consuming task; thus, we opted to create a parallel and distributed processing method using the Akka framework (<http://doc.akka.io/docs/2.5/java/guide>) that relies on the Actor Model [20]. In Akka, an actor is similar to a processing thread that runs in a thread-pool. The Akka framework has several benefits, among which the most important are: thread management and locking mechanisms, asynchronous processing of messages, immutability and modular architecture. Parallel computing was implemented via the master-slave paradigm bundled with the Akka actor framework as presented in Fig. 2. For implementing this paradigm, two types of actors were created: a master and slaves (gathering actor) responsible for gathering the articles from the external services. The master actor has only one instance, whereas the slave actor has 2 instances, one for *Mendeley* and one for *Core Engine* service, which

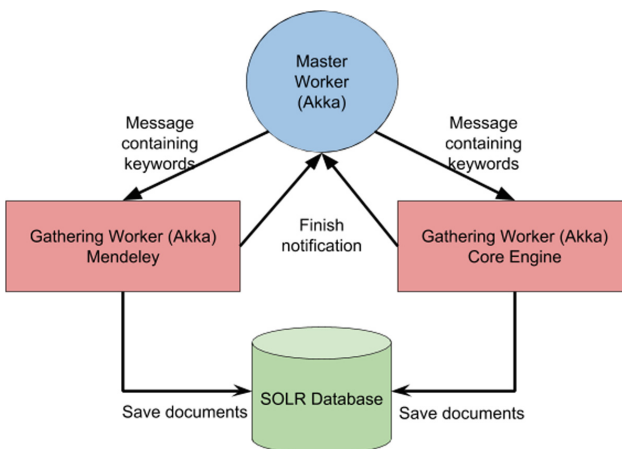


Fig. 2. Architecture for searching online articles.

can be scaled depending on the architecture. When making a search request, a message is sent to the master actor which extracts the keywords and sends 2 simultaneous messages to the gathering actors to extract articles from the external services; afterwards, the retrieved documents are indexed in the SOLR database.

## 4 Experiments and Results

Our first preliminary validation experiment was conducted on a dataset containing 1,044 documents extracted using the semantic search algorithm, having as main focus different sub-domains of artificial intelligence: neural networks, classification and regression algorithms, robotics, and others. The aim of this experiment is to provide an in-depth analysis with regards to the relevance of the retrieved documents, while assessing article rankings. The input used for testing was an abstract from a paper by Leung and Joseph [21] that aimed to predict sport results using a data mining approach. The text was given as an input to the *Hub-Tech* processing pipeline, and our model generated 10 texts/articles with the highest semantic relevance. Subsequently, ten users aged from 21 to 30, both experts and unfamiliar with the domain, were asked to rank the selected articles from their perspective with unique numbers from 1 to 10, where 10 is the most relevant article that can help them fulfill their goals, and 1 the least relevant article.

**Table 1.** Comparison between automated and human rankings.

	Automated ranking	Human ranking	Human Average score	Standard deviation of human scores
Article 1	9	6	4.91	2.57
Article 2	8	3	6.16	2.51
Article 3	10	10	2.08	1.24
Article 4	5-6	4	5.58	2.10
Article 5	7	9	2.91	1.44
Article 6	1	1	8.83	1.64
Article 7	2	2	7.58	2.10
Article 8	3-4	8	4.16	2.58
Article 9	3-4	5	5.41	2.46
Article 10	5-6	7	4.58	2.27

The results are shown in Table 1 in which similar scores for both automated and human evaluations are marked as pairs of ranks (e.g., “5–6” and “7–8”). Rows in green denote similar rankings and an important observation is that the first and the second most relevant and highly ranked articles fall in this category. Rows in yellow suggest a close match in ratings (i.e., a maximum difference of two points in rankings). The red color represents rows with a high disagreement between automated and human rankings. Overall, we can observe that only two articles were highly rated (average scores

above 7.5), two articles were lowly ranked (average scores below 3), while the remaining ones are rather close to each other in terms of relevance. Moreover, the articles that had the highest differences between the automated and manual rankings also exhibited a higher standard deviation, thus denoting that even the human evaluators were far from reaching a consensus.

Our second preliminary validation experiment is centered on evaluating the accuracy of our results by performing a side-by-side comparison between our *Hub-Tech* search engine and Google Scholar (<https://scholar.google.com>). The used metrics to are specific to the informational retrieval domain, namely: precision (i.e., ratio between the number of relevant articles and the number of the retrieved documents) and precision at  $K$  (similar to precision, but the domain of articles in cut or expanded to a cardinal of  $K$ ). In order to emphasize the benefits of performing semantic searches, we opted to consider only long queries (50 + words) simulating, for example a problem definition, a project synopsis or the description of new idea. The input consisted of abstracts from eight Artificial Intelligence articles that were fed to both platforms; top 10 retrieved articles from each engine were saved for follow-up analyses. Five participants performed a binary assessment of the relevance of each of the 132 documents extracted from both platforms. A document was considered relevant based on majority voting from the raters.

**Table 2.** Side-by-side comparison between Hub-Tech platform and Google Scholar

	Hub-Tech precision	Hub-Tech precision at 10	Google scholar precision	Google scholar precision at 10
Article 1	6/9	6/10	2/10	2/10
Article 2	5/10	5/10	7/8	7/10
Article 3	3/10	3/10	1/2	1/10
Article 4	4/10	4/10	0	0
Article 5	2/10	2/10	9/10	9/10
Article 6	3/10	3/10	2/3	2/10
Article 7	5/10	5/10	1/10	1/10
Article 8	4/10	4/10	4/10	4/10

Table 2 introduces the comparative results for our specific queries that contain detailed descriptions. *Hub-Tech* platform recommended relevant articles with a precision of 33%, while Google Scholar only achieved 26%. The main problem that we noticed about Google Scholar in this experiment is that the search engine does not return enough articles if the query is not clear enough or too long (e.g., Article 4 has no results, Article 3 has only 2 results, Article 6 has 3).

Moreover, we observed additional features and limitations for both engines:

- The *Hub-Tech* database contains only 1044 documents related to the artificial intelligence field, while Google Scholar has access to at least 100 million articles with no particular restriction regarding the domain of focus [22].
- Specific abstracts having keywords from one domain, rather than complex multi-disciplinary descriptions, are performing better on Google Scholar because it is easier for the search engine to retrieve documents based on keywords.
- The underlying semantic models from *Hub-Tech* may affect the overall results if the training set does not contain the keywords of a text.
- The average execution time for the *Hub-Tech* platform is 9 min, while Google Scholar search engine provides responses in 0.5 s for complex queries as the ones given as input in our experiments.
- Google Scholar allows only inputs with a maximum of 256 characters. If the search input is longer, it gets trimmed to 256 characters.

Overall, *Hub-Tech* is performing better than Google Scholar for long, specific queries and is exhibiting consistency reflected in the number of retrieved results. When complex, cross or multi-domain, texts are given as input, *Hub-Tech* is performing better; however, when specific texts are provided, Google Scholar presents more relevant results while relying on the keywords extracted from the query.

## 5 Conclusions

This paper introduced a novel semantic meta-search engine built on top of Cohesion Network Analysis that makes use of the Akka Actor system and SOLR to increase its performance and to ensure scalability. The initial results are encouraging, the system being capable to recommend relevant articles based on a provided user description. The generated recommendations were confirmed by humans with a high precision, but more in-depth validations need to be performed as subsequent analyses. A comparison with Google Scholar was performed to evaluate the system's performance in recommending relevant articles, given a description of a project or a paper abstract. Both platforms have their limitations, but *Hub-Tech* is performing overall better at this task due to the semantic contextualization. Moreover, we believe that accuracy can improve by adding more articles to our semantic database, as well as by training specific semantic models for certain sub-domain.

Based on the existing solutions, meta-search and semantic search engines provide valuable methods and results when retrieving relevant resources. The *Hub-Tech* platform tackles this requirement and tries to help users to jumpstart their project with less time invested in searching online for connected articles, and with a more relevant collection of reference documents as a starting point.

**Acknowledgements.** The work presented in this paper was partially funded by the European Funds of Regional Development with the Operation Productivity Program 2014-2020 Priority Axe 1, Action 1.2.1 D-2015, "Innovative Technology Hub based on Semantic Models and High Performance Computing" Contract no. 6/1 09/2016.

## References

1. Dascalu, M., McNamara, D.S., Trausan-Matu, S., Allen, L.K.: Cohesion network analysis of CSCL participation. *Behav. Res. Meth.* **50**(2), 604–619 (2018)
2. Manning, C.D., Raghavan, P., Schütze, H.: *Introduction to Information Retrieval*, vol. 1. Cambridge University Press, Cambridge (2008)
3. Gulli, A., Signorini, A.: The indexable web is more than 11.5 billion pages. In: *Special Interest Tracks and Posters of the 14th International Conference on World Wide Web*, pp. 902–903. ACM (2005)
4. Mohammadesmeil, S., Naraghian, N.: Comparing search engines and meta search engines in dentistry information retrieval. *J. Res. Dental Sci.* **14**(2), 1–1 (2017)
5. Mangold, C.: A survey and classification of semantic search approaches. *Int. J. Metadata Semant. Ontol.* **2**(1), 23–34 (2007)
6. Cohen, S., Mamou, J., Kanza, Y., Sagiv, Y.: XSEarch: a semantic search engine for XML. In: *Proceedings of the 29th International Conference on Very Large Data Bases*, vol. 29, pp. 45–56. VLDB Endowment (2003)
7. Laura, L., Me, G.: Searching the Web for illegal content: the anatomy of a semantic search engine. *Soft. Comput.* **21**(5), 1245–1252 (2017)
8. Mukhopadhyay, D., Sharma, M., Joshi, G., Pagare, T., Palwe, A.: Experience of developing a meta-semantic search engine. In: *2013 International Conference on Cloud & Ubiquitous Computing & Emerging Technologies (CUBE)*, pp. 167–171. IEEE (2013)
9. Paraschiv, I.C., Dascalu, M., Banica, C.K., Trausan-Matu, S.: Designing a scalable technology hub for researchers. In: *5th International Workshop on Semantic and Collaborative Technologies for the Web, in Conjunction with the 13th International Conference on eLearning and Software for Education (eLSE 2017)*, vol. 3, pp. 13–18. Advanced Distributed Learning Association, Bucharest, Romania (2017)
10. Dascalu, M., Stavarahe, L.L., Dessus, P., Trausan-Matu, S., McNamara, D.S., Bianco, M.: ReaderBench: the learning companion. In: *17th International Conference on Artificial Intelligence in Education (AIED 2015)*. LNCS, vol. 9112, pp. 915–916. Springer, Madrid (2015)
11. Dascalu, M., Dessus, P., Trausan-Matu, Ș., Bianco, M., Nardy, A.: *ReaderBench* an environment for analyzing text complexity and reading strategies. In: Lane, H.C., Yacef, K., Mostow, J., Pavlik, P. (eds.) *AIED 2013*. LNCS (LNCS), vol. 7926, pp. 379–388. Springer, Heidelberg (2013). [https://doi.org/10.1007/978-3-642-39112-5\\_39](https://doi.org/10.1007/978-3-642-39112-5_39)
12. Dumais, S.T.: Latent semantic analysis. *Ann. Rev. Inf. Sci. Technol.* **38**(1), 188–230 (2004)
13. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent Dirichlet allocation. *J. Mach. Learn. Res.* **3**(4–5), 993–1022 (2003)
14. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representation in vector space. In: *Workshop at ICLR, Scottsdale, AZ* (2013)
15. Miller, G.A.: WordNet: a lexical database for English. *Commun. ACM* **38**(11), 39–41 (1995)
16. Zaugg, H., West, R.E., Tateishi, I., Randall, D.L.: Mendeley: creating communities of scholarly inquiry through research collaboration. *TechTrends* **55**(1), 32–36 (2011)
17. Knoth, P., Zdrahal, Z.: CORE: three access levels to underpin open access. *D-Lib Mag.* **18**(11/12) (2012). <http://www.dlib.org/dlib/november12/knoth/11knoth.print.html>
18. Paraschiv, I.C., Dascalu, M., McNamara, D.S., Trausan-Matu, S.: Finding the needle in a haystack: who are the most central authors within a domain? In: Verbert, K., Sharples, M., Klobučar, T. (eds.) *EC-TEL 2016*. LNCS, vol. 9891, pp. 632–635. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-45153-4\\_79](https://doi.org/10.1007/978-3-319-45153-4_79)

19. Jones, M., Hardt, D.: The OAuth 2.0 authorization framework: Bearer token usage (2012)
20. Agha, G.A.: Actors: a model of concurrent computation in distributed systems. Massachusetts Institute of Technology Cambridge Artificial Intelligence Laboratory (1985)
21. Leung, C.K., Joseph, K.W.: Sports data mining: predicting results for the college football games. *Procedia Comput. Sci.* **35**, 710–719 (2014)
22. You, J.: Just how big is Google Scholar? Ummm ..., (2014). <http://www.sciencemag.org/news/2014/09/just-how-big-google-scholar-ummm>. Accessed 28 June 2018





# A Query Language for Cognitive Maps

Adrian Robert, David Genest<sup>(✉)</sup>, and Stéphane Loiseau

LERIA, Université d'Angers, 49100 Angers, France  
{adrian.robert,david.genest,stephane.loiseau}@univ-angers.fr

**Abstract.** A *Cognitive map* is a graphical semantic model that represents influences between concepts. A cognitive map is easy to design and use, but the only query a user can make on it is to infer the propagated influence from a concept to another. This paper (This work is made in the context of “Analyse Cognitive de Savoirs”, a computer science project, granted by French region Pays de la Loire.) proposes CMQL, a general query language for cognitive maps. This language provides a way to query the different items of the model and not only the propagated influence. The language is declarative and is inspired of the domain relational calculus.

**Keywords:** Cognitive map · Visual knowledge representation  
Query language · Analysis · Influence · Graphical semantic model

## 1 Introduction

A *Cognitive map* [2] is a semantic model originally coming from cognitive psychology. A cognitive map is an efficient tool to model strategies, more generally influence systems. Cognitive maps are used in domains like social sciences [2], biology [14] or geography [3]. A cognitive map is a graph whose nodes are *concepts* and edges are *influences*. The influences are labelled with an *influence value* that belongs to a predefined value set that quantifies it. This set can contain symbolic values such as  $\{-, +\}$  [14] or  $\{\text{none, some, much, a lot}\}$  used in fuzzy cognitive maps [7], or numerical values such as  $[-1; 1]$  [7]. A sequence of influences from a concept to another constitutes an influence *path*. Using those paths, this model provides a way to infer the global influence value that any concept can have on any other one, such an inference is called the *propagated influence value*.

Cognitive maps are used in two ways. On the one hand, they are used as a knowledge representation model for influence systems. This makes them a convenient visual support for brainstorming [10] or discussions [5]. On the other hand, they are used not only for knowledge representation but also for analytical inference [6] and different analysis of the cognitive map to make a decision.

This paper addresses difficulties of this second way. The only mean a user has to make an analysis of the map is to query the propagated influence value from a concept on another. This is not enough, users should be able to make

many more interesting queries such as getting all paths between two concepts, getting all concepts that influence positively another specific concept etc. This lack of solutions has been felt in applied research projects [12]. Indeed, in the context of the KIFANLO<sup>1</sup> project, geographers for whom a cognitive map tool [1, 9] has been developed, asked us to think about other solutions to query the cognitive maps model.

The contribution of this article is to propose a general language called *CMQL*, to query cognitive maps. CMQL is based on two original ideas: the language is *declarative* and its basis is a set of *primitives* that gives an access to path and values. First, this language is declarative so that it is easy to formulate the desired query. The syntax of CMQL, uses the well-known form of SQL [4] (**SELECT variables/FROM map/WHERE condition**), which makes it intuitive. Its semantics is close to domain relational calculus [8] which combines formulae with first order logic operators and whose *variables* denotes items of the model. Second, *primitive formulae* are used as predicates that provide meaning to variables in the condition clause of queries. They allow a direct access to the different items of the cognitive map model and are used syntactically as an equivalent of relations in query languages for the relational data model.

This article is composed of three parts. First, the cognitive map model is recalled in a formal way. Second, the primitives are presented. Third, CMQL is presented.

## 2 The Cognitive Map Model

A cognitive map is a graph defined on a concept set and a value set. The nodes of the graph are the concepts and the directed edges are the influences. An influence of a concept on another one is labelled with an influence value. The value set can vary as it depends on the semantics of the cognitive map.

**Definition 1 (Cognitive map).** *Let  $C$  be a concept set. Let  $I$  be a value set. A cognitive map defined on  $I$  is an oriented labelled graph  $CM = (C, A, label)$ :*

- *The concepts of  $C$  are the nodes of the graph*
- *$A \subseteq C \times C$  are the edges of the graph called influences*
- *label:  $A \rightarrow I$  is a label function that associates to each influence an influence value.*

*Example 1.* CM1 is the cognitive map of Fig. 1. This map is used for all examples in the paper. It is defined on the value set  $I = [-1; 1]$  (where 1 is a great influence and 0.2 a low influence). CM1 represents the fishing strategy of a fisherman of the french Atlantic coast. We notice that the fisherman's goal is the concept of Rentability and other concepts influence it. For instance the concept BadWeather influences NoFishing and this influence is labelled 0.5. NoFishing influences Rentability and this influence is labelled negatively  $-1$ .

<sup>1</sup> KIFANLO is a 2013–2017 project financed by the *Fondation de France* to help geographers analyse fishing strategies of fishermen in the French Atlantic coast. In this project, fishing strategies were modelled using cognitive maps.

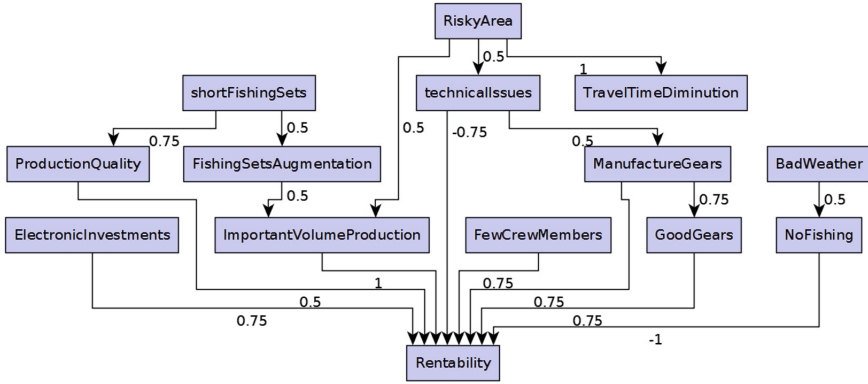


Fig. 1. CM1

A path is a sequence of influence which represents a direct or indirect way a concept influences another. A path is said minimal if it does not contain any cycle. Notice that between two paths, there can be many minimal paths.

**Definition 2 (Path).** Let  $CM = (C, A, label)$  be a cognitive map. Let  $c_1, c_2 \in C$  be two concepts of  $CM$ .

- A path  $P$  from  $c_1$  to  $c_2$  is a sequence of length  $length_P \geq 1$  of influences  $(u_i, u_{i+1}) \in A$  (with  $i \in [0; length_P - 1]$ ) such that  $u_0 = c_1$  is the source of  $P$  noted  $source_P$  and  $u_{length_P} = c_2$  is the destination of  $P$  noted  $dest_P$ . This path is denoted by  $c_1 \rightarrow u_1 \rightarrow \dots \rightarrow c_2$ .
- $P$  is said minimal if  $\forall i, j \in [0; length_P], i \neq j \Rightarrow u_i \neq u_j$ .
- The set of all minimal paths on  $CM$  is denoted by  $Paths_{CM}$

*Example 2.*  $ShortFishingSets \rightarrow ProductionQuality \rightarrow Rentability$  is a minimal path of length = 2, from the source concept  $ShortFishingSets$  to the destination concept  $Rentability$ . This path is a sequence of 2 influences.

One of the main features of cognitive maps is its ability to infer the propagated influence from any concept to any other one. To do that, every way one concept influences another has to be considered, in other words every path from one concept to the other is involved.

The propagated influence from a concept to another can be calculated differently depending on the map’s semantics and on the value set on which it is defined. In all cases, the computation of the propagated influence first assigns a propagated influence for each path with an operator before aggregating those values with a second operator.

**Definition 3 (Propagated Influence).** Let  $CM$  be a cognitive map defined on  $I$ .

- The path value is a function  $PV_{path} : Paths_{CM} \rightarrow I$  which infers the propagated influence of a path.

- The propagated influence value is a function  $PV: C \times C \rightarrow I$  which infers the propagated influence from a concept to another one.

In this paper, to illustrate with a specific example, we will use the value set  $I = [-1; 1]$  and define functions for the propagated influence according to this set [7].

**Definition 4 (Propagated Influence for  $I = [-1; 1]$ ).** Let  $CM$  be a cognitive map defined on  $I = [-1; 1]$ . Let  $p_1$  be a path. Let  $c_1, c_2 \in C$  be two concepts of  $C$ . Let  $Paths_{c_1, c_2}$  be the set of all minimal paths between  $c_1$  and  $c_2$ .

$$\begin{aligned}
 - PV_{path}(p_1) &= \prod_{i=0}^{length_{p_1}-1} label((u_i, u_{i+1})). \\
 - PV(c_1, c_2) &= \frac{\sum_{p \in Paths_{c_1, c_2}} PV_{path}(p)}{|Paths_{c_1, c_2}|}.
 \end{aligned}$$

*Example 3.* Lets infer the propagated influence value between `ShortFishingSets` and `Rentability`. The set of all minimal paths between those two concepts is  $Paths_{ShortFishingSets, Rentability}$ , it contains two paths;

- $p_1 = \text{ShortFishingSets} \rightarrow \text{ProductionQuality} \rightarrow \text{Rentability}$ .
- $p_2 = \text{ShortFishingSets} \rightarrow \text{FishingSetsAugmentation} \rightarrow \text{ImportantValueProduction} \rightarrow \text{Rentability}$ .

To infer the propagated influence value between `ShortFishingSets` and `Rentability` we need  $PV_{Path}(p_1)$  and  $PV_{Path}(p_2)$ . From the definition above, we have  $PV_{Path}(p_1) = 0.75 \times 0.5 = 0.37$  and  $PV_{Path}(p_2) = 0.5 \times 0.5 \times 1 = 0.25$ . Then, aggregating the path values,  $PV(\text{ShortFishingSets}, \text{Rentability}) = \frac{(0.37+0.25)}{2} = 0.31$ . So the propagated influence value from `ShortFishingSets` to `Rentability` is 0.31.

### 3 The Primitives

Primitives are predicates that will be used in the condition part of a query. They are useful to describe and obtain characteristics on paths and values.

In order to introduce the primitives, the definition of a relation has to be recalled, it comes from the domain relational calculus [11]. The structure of a relation  $R$  is a mapping  $R(A_1 : D_1, \dots, A_n : D_n)$  from the attributes  $A = A_1, \dots, A_n$  of the relation to their domain  $D = D_1, \dots, D_n$ . Its value is a subset of the indexed cartesian product<sup>2</sup> of the attribute/domain pairs. Primitive formulae are syntactic components that access primitives, they have arguments that can be either constants or variables. Variables are written like in SPARQL [13] with a question mark as prefix. The meaning of such a formula is a relation whose attributes are the variables of the formula.

<sup>2</sup>  $\prod(A: D) = \{t \text{ is a total function} : A \rightarrow D_1 \cup \dots \cup D_n / \forall i \in [1, n], t(A_i) \in D_i\}$ .

### 3.1 Primitives of Paths

The Path primitive links a path with its source concept and destination concept, it provides an access to paths and concepts.

**Definition 5 (Path).** Let  $CM = (C, A, label)$  be a cognitive map.

$Path(s:C,d:C,p:Paths_{CM})$  is a relation whose value is  $\{(s, d, p) / p \in Paths_{CM} \wedge source_p = s \wedge dest_p = d\}$ .

*Example 4.* Applied to CM1 - Fig. 1.

- $Path(RiskyArea, Rentability, ?p)$  is a primitive formula. Its meaning is a unary relation whose value is the set of tuples ( $?p$ ):

$?p$
$RiskyArea \rightarrow ImportantVolumeProduction \rightarrow Rentability$
$RiskyArea \rightarrow TechnicalIssues \rightarrow Rentability$
$RiskyArea \rightarrow TechnicalIssues \rightarrow ManufactureGears \rightarrow Rentability$
$RiskyArea \rightarrow TechnicalIssues \rightarrow ManufactureGears \rightarrow GoodGears \rightarrow Rentability$

- $Path(BadWeather, ?c2, ?p)$  is primitive formula. Its meaning is a relation whose value is the set of tuples ( $?c2, ?p$ ):

$?c2$	$?p$
$NoFishing$	$BadWeather \rightarrow NoFishing$
$Rentability$	$BadWeather \rightarrow NoFishing \rightarrow Rentability$

The primitive Length is a relation between a path and its length.

**Definition 6 (Length).** Let  $CM = (C, A, label)$  be a cognitive map.

$Length(p:Paths_{CM}, l:\mathbb{N}^*)$  is a relation whose value is  $\{(p, l) / p \in Paths_{CM} \wedge l = length_p\}$ .

*Example 5.* Applied to CM1 - Fig. 1.

$Length(?p, 4)$  is a primitive formula. Its meaning is a unary relation whose value is the set of tuples ( $?p$ ):

$?p$
$RiskyArea \rightarrow TechnicalIssues \rightarrow ManufactureGears \rightarrow GoodGears \rightarrow Rentability.$

The primitives Contains and ContainsPath provide an access to the content of paths. Contains is a relation between a path and a concept contained by this path while ContainsPath is a relation between two paths where the first contains the second. Primitives Contains and ContainsPath(not defined here) have a ‘complement’ primitive respectively NotContains and NotContainsPath with the same respective arguments.

**Definition 7 (Contains).** Let  $CM = (C, A, label)$  be a cognitive map.

$Contains(p:Paths_{CM}, c:C)$  is a relation whose value is  $\{(p, c) / p \in Paths_{CM} \wedge \exists (u_n, u_{n+1}) \in p, c = u_n \vee c = u_{n+1}\}$ .

Example 6. Applied to CM1 - Fig. 1.

$\text{Contains}(\text{?p}, \text{ProductionQuality})$  is a primitive formula. Its meaning is a unary relation whose value is the set of tuples ( $\text{?p}$ ):

$\text{?p}$
$\text{ShortFishingSets} \rightarrow \text{ProductionQuality}$
$\text{ProductionQuality} \rightarrow \text{Rentability}$
$\text{ShortFishingSets} \rightarrow \text{ProductionQuality} \rightarrow \text{Rentability}$

**Definition 8 (ContainsPath).** Let  $CM = (C, A, \text{label})$  be a cognitive map.

$\text{ContainsPath}(p1:\text{Paths}_{CM}, p2:\text{Paths}_{CM})$  is a relation whose value is  $\{(p1, p2) / p1, p2 \in \text{Paths}_{CM} \wedge p2 \text{ is a subsequence of } p1\}$ .

Example 7. Applied to CM1 - Fig. 1.

$\text{ContainsPath}(\text{?p}, \text{ProductionQuality} \rightarrow \text{Rentability})$  is a primitive formula. Its meaning is a unary relation whose value is the set of tuples ( $\text{?p}$ ):

$\text{?p}$
$\text{ProductionQuality} \rightarrow \text{Rentability}$
$\text{ShortFishingSets} \rightarrow \text{ProductionQuality} \rightarrow \text{Rentability}$

### 3.2 Primitives of Values

The primitive PathValue links a path with its value.

**Definition 9 (PathValue).** Let  $CM = (C, A, \text{label})$  be a cognitive map defined on the value set  $I$ .  $\text{PathValue}(p:\text{Paths}_{CM}, i:I)$  is a relation whose value is  $\{(p, i) / p \in \text{Paths}_{CM} \wedge i = PV_{\text{Path}}(p)\}$ .

Example 8. Applied to CM1 - Fig. 1.

$\text{PathValue}(\text{RiskyArea} \rightarrow \text{TechnicalIssues} \rightarrow \text{Rentability}, \text{?i})$  is a primitive formula. Its meaning is a unary relation whose value is the set of tuples ( $\text{?i}$ ):

$\text{?i}$
-0.38

The primitive Value links two concepts with the propagated influence value from the first to the second.

**Definition 10 (Value).** Let  $CM = (C, A, \text{label})$  be a cognitive map defined on the value set  $I$ .  $\text{Value}(c1:C, c2:C, i:I)$  is a relation whose value is  $\{(c1, c2, i) / c1, c2 \in C \wedge i = PV(c1, c2)\}$ .

Example 9. Applied to CM1 - Fig. 1.

$\text{Value}(\text{?c}, \text{ImportantVolumeProduction}, \text{?i})$  is a primitive formula. Its meaning is a relation whose value is the set of tuples ( $\text{?c}, \text{?i}$ ):

$\text{?c}$	$\text{?i}$
$\text{FishingSetsAugmentation}$	0.5
$\text{ShortFishingSets}$	0.25
$\text{RiskyArea}$	0.5

## 4 The Query Language CMQL

Primitive formulae can be combined with first order logic operators to build more complex queries.

This section gives an intuitive overview of its capabilities and its design through a first detailed example of query. Other examples will just be introduced to give a broader idea of the language. Due to lack of space in the paper, CMQL syntax and semantics are succinctly introduced afterwards.

### 4.1 Detailed Example

This example describes a query and its results. The goal of the query is to find the concepts that are influenced at the same time by `shortFishingSets` and `RiskyArea`, the following query can be made:

```
SELECT ?c1 FROM CM1 WHERE{
  Path(shortFishingSets,?c1,?p1)
  AND Path(RiskyArea,?c1,?p2)}
```

This query selects from `CM1`, the concept `?c1` where there exist a path from `shortFishingSets` to `?c1` and another path from `RiskyArea` to `?c1`.

The formula is composed of two conditions combined with a conjunction. The first one describes `?c1` as the destination of a path `?p1` whose source is the concept `shortFishingSets`. The second one describes `?c1` as the destination of a path `?p2` whose source is `RiskyArea`. The meanings of those two conditions are unary relations whose values are the following sets of tuples:

<i>?c1</i>	<i>?c1</i>
<i>ProductionQuality</i>	<i>TechnicalIssues</i>
<i>Rentability</i>	<i>TravelTimeDiminution</i>
<i>FishingSetsAugmentation</i>	<i>ManufactureGears</i>
<i>ImportantVolumeProduction</i>	<i>GoodGears</i>
	<i>ImportantVolumeProduction</i>
	<i>Rentability</i>

The meaning of the conjunction of those two condition is a relation whose value is the intersection of the two previous sets of tuples. It is also the result of the query:

<i>?c1</i>
<i>Rentability</i>
<i>ImportantVolumeProduction</i>

### 4.2 Other Examples

The following examples are given to illustrate CMQL.

- All concepts of the map with their propagated influence on `Rentability` sorted:

```
SELECT ?c,?i FROM CM1 WHERE{ Value(?c,Rentability,?i)}
```

ORDER BY ?i

?c	?i
<i>ImportantVolumeProduction</i>	1
<i>ElectronicInvestments</i>	0.75
...	...

- The concepts that do not influence any other concepts:  
SELECT ?c1 FROM CM1 WHERE{ NOT Path(?c1,?c2,?p)}

?c1
<i>TravelTimeDiminution</i>
<i>Rentability</i>

- The paths from RiskyArea to Rentability that do not contain TechnicalIssues:  
SELECT ?p FROM CM1 WHERE{ Path(RiskyArea,Rentability,?p)  
AND NotContainsPath(?p,TechnicalIssues)}

?p
<i>RiskyArea</i> → <i>ImportantVolumeProduction</i> → <i>Rentability</i>

- The concepts that influence Rentability positively and negatively at the same time:

SELECT ?c FROM CM1 WHERE{  
Path(?c,Rentability,?p1) AND PathValue(?p1,?i1) AND ?i1<0  
AND  
Path(?c,Rentability,?p2) AND PathValue(?p2,?i2) AND ?i2>0}

?c
<i>RiskyArea</i>
<i>TechnicalIssues</i>

### 4.3 Syntax and Semantics

The syntax presented below is abstract, and aims only to introduce the general syntactic structure and components for the definition of the semantics.

The syntactic components are: Q (Query), V (Variable), M (Cognitive Map name), F (Formula), P(Primitive), E (Expression like  $x < 3, y = z$  etc.), C (Constant) and OP (OPerators like =, >, <...).

$\langle Q \rangle ::= \text{'SELECT' } \langle V \rangle + \text{'FROM' } \langle M \rangle \text{'WHERE' } \langle F \rangle$

$\langle F \rangle ::= \text{'NOT' } \langle F \rangle \mid \text{'(' } \langle F \rangle \text{' )' } \mid \text{'ALL' } \langle V \rangle + \text{'(' } \langle F \rangle \text{' )' }$   
 $\mid \langle F \rangle \text{'AND' } \langle F \rangle \mid \langle F \rangle \text{'OR' } \langle F \rangle \mid \langle P \rangle \mid \langle E \rangle$

$\langle E \rangle ::= \langle V \rangle \langle OP \rangle \langle V \rangle \mid \langle V \rangle \langle OP \rangle \langle C \rangle$

This syntax being abstract, a well formed query needs to respect the four following syntactic rules. First, in the first syntactic rule, variables V are the free variables of the query, free variables must all be present in the formula and distinct. At least one variable must appear. Free variables cannot be quantified. Second, in the formula rule 'ALL', variables V cannot appear outside of the formula. Third, for the E (expression) rule, variables V and constants C must



be in the same domain, the operator must be defined for the domains of V and C. Fourth, at least one primitive formula must appear in the query formula. Variables in E (expressions) must also appear in a primitive formula.

The principal design of this semantics definition is that each formula denotes a relation whose attributes are the free variables of the formula. The most general meaning of a query is as follows:

Let  $S_{CM}$  be the set of possible cognitive map instances. The meaning of a query Q is a function from a cognitive map instance to a relation whose attributes are the free variables of the query:  $mng_Q : S_{CM} \rightarrow 2^{\prod(V:D)}$  where V is the set of free variables with D the set of their corresponding domain, and  $2^{\prod(V:D)}$  the power set of  $\prod(V:D)$  which is the set of all the subsets of  $\prod(V:D)$ .

In a general way, the meaning of a formula is a relation whose attributes are the free variables of the formula:  $mng_F : F \rightarrow \prod(V:D)$ . The different formula rules of the abstract syntax are listed below with their meaning. Let  $fr(F)$  be the set of variables occurring free in the formula F.

- $mng_F(NOT F_1) = \prod(fr(F_1) : D) - mng_F(F_1)$ .
- $mng_F(F_1 AND F_2) =$   
 $mng_F(F_1) \times \prod(fr(F) - fr(F_1) : D) \cap mng_F(F_2) \times \prod(fr(F) - fr(F_2) : D)$
- $mng_F(F_1 OR F_2) =$   
 $mng_F(F_1) \times \prod(fr(F) - fr(F_1) : D) \cup mng_F(F_2) \times \prod(fr(F) - fr(F_2) : D)$
- $mng_F(All V_1..V_n F_1) =$   
 $\{t \in \prod(fr(V) - \{V_1, \dots, V_n\} : D) / \{t\} \times \prod(\{V_1, \dots, V_n\} : D) \subseteq mng_F(F_1)\}$
- $mng(P(A_1 : T_1, \dots, A_n : T_n)) = \{t \in \prod(V : D) / \exists t_1 \in P, \forall i \in [1, n],$   
 $t_1(A_i) = T_i \text{ if } T_i \text{ is a constant, } t_1(A_i) = t(T_i) \text{ if } T_i \text{ is a varibale } \}$
- $mng_F(V_1 OP C) = \{t \in \prod(V : D) / t(V) OP C\}$
- $mng_F(V_1 OP V_2) = \{t \in \prod(\{V_1, V_2\} : D) / t(V_1) OP t(V_2)\}$

CMQL also provides a set of post-query processing that allow the result set of tuples to be rearranged and modified. Those functions are similar to the ones of SQL and SPARQL. They can apply to some attributes like COUNT, AVG, ABS, SUM, or they can apply to the whole tuples with functions like ORDERBY, GROUPBY, LIMIT and OFFSET. Their semantic is not described as they are considered trivial.

## 5 Conclusion

The query language for cognitive maps CMQL has been introduced. It turned out that query languages for similar models are not appropriate for cognitive maps, as the models' semantics are still very distant. For instance RDF's SPARQL [13] and Neo4j's Cypher [15] are mainly oriented towards scheme graph projection. Their exploitation of nodes and edges properties do not provide a way to take into account this idea of paths as proposed in CMQL. Some of those languages like SPARQL provide regular path expressions, but CMQL's primitives are more adequate to query cognitive maps as they can manipulate paths and their length as variable.

Indeed, what is needed is a way to access nodes that are influenced or not, get all paths between two concepts and get the inferred propagated influence. To do so, CMQL provides a way to query paths with their properties in an original way, using primitives. Other features like the propagated influence also are queried using primitives which provide coherence for the language. To our knowledge no other work proposes a query language for Cognitive maps.

CMQL has been implemented and integrated into the software VSPCC [1], which has been developed in our lab and provides tools to use cognitive maps. Researchers in geography began to use it and this software is useful. The features of CMQL also let us think that it could be interesting to use it in a broader context and not limit its expressiveness to cognitive maps. Indeed, CMQL could be fit to query other graph based models that require to work with paths.

## References

1. <http://www.info.univ-angers.fr/pub/ledorze/vspcc/index.html>
2. Axelrod, R.M.: *Structure of Decision: The Cognitive Maps of Political Elites*. Princeton University Press, Princeton (1976)
3. Çelik, F.D., Ozesmi, U., Akdogan, A.: Participatory ecosystem management planning at tuzla lake (Turkey) using fuzzy cognitive mapping. arXiv preprint q-bio/0510015 (2005)
4. Date, C.J., Darwen, H.: *A Guide to the SQL Standard: A User's Guide to the Standard Relational Language SQL*. Addison-Wesley, Reading (1989)
5. Eden, C.: On the nature of cognitive maps. *J. Manag. Stud.* **29**(3), 261–265 (1992)
6. Eden, C., Ackermann, F., Cropper, S.: The analysis of cause maps. *J. Manag. Stud.* **29**(3), 309–324 (1992)
7. Kosko, B.: Fuzzy cognitive maps. *Int. J. Man-Mach. Stud.* **24**, 65–75 (1986)
8. Lacroix, M., Pirotte, A.: Domain-oriented relational languages. In: *Proceedings of the Third International Conference on Very Large Data Bases-Volume 3*, pp. 370–378. VLDB Endowment (1977)
9. Le Dorze, A., Garcia, L., Genest, D., Loiseau, S.: Synthesis of cognitive maps and applications. In: *2014 IEEE 26th International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 291–298. IEEE (2014)
10. Lee, K.C., Kwon, S.: A cognitive map-driven avatar design recommendation dss and its empirical validity. *Decis. Support Syst.* **45**(3), 461–472 (2008)
11. Louis, G., Pirotte, A.: A denotational definition of the semantics of DRC, a domain relational calculus. In: *VLDB*, pp. 348–356 (1982)
12. Papageorgiou, E.I., Salmeron, J.L.: A review of fuzzy cognitive maps research during the last decade. *IEEE Trans. Fuzzy Syst.* **21**(1), 66–79 (2013)
13. Prud, E., Seaborne, A., et al.: *SPARQL query language for RDF* (2006)
14. Tolman, E.C.: Cognitive maps in rats and men. *Psychol. Rev.* **55**(4), 189–208 (1948)
15. Webber, J.: A programmatic introduction to Neo4j. In: *Proceedings of the 3rd Annual Conference on Systems, Programming, and Applications: Software for Humanity*, pp. 217–218. ACM (2012)



# Approaches for Enumerating All the Essential Prime Implicants

Yakoub Salhi<sup>(✉)</sup>

CRIL-CNRS, Université d'Artois, 62307 Lens Cedex, France  
salhi@cril.fr

**Abstract.** The aim of this paper is to study the problem of enumerating all the essential prime implicants (EPIes) of a CNF formula. We first provide some interesting computational complexity results. We show in particular that the problem of checking whether a prime implicant of a CNF formula is essential is NP-complete. Then, we propose a simple characterization of the e-models of a CNF formula. An e-model is a model covered by a unique prime implicant, which is necessarily essential. Our characterization is then used to define a linear-time algorithm for checking whether a model of CNF formula is an e-model or not. Finally, using our characterization of the e-models, we propose two approaches for enumerating all the EPIes of a CNF formula.

**Keywords:** Essential prime implicant · Propositional logic  
Solution enumeration

## 1 Introduction

The problem generating all the prime implicants (PIes) is a central task in different areas in computer science. One of its important applications is in the problem of Boolean function minimization, which consists in reducing a Boolean function to a smaller and equivalent function [12, 14, 15]. Boolean function minimization problem has several applications in computer science, such as reducing digital circuit size which is a way for enhancing computation speed [21]. The problem of enumerating all the PIes finds other applications in other areas in computer science in general, and artificial intelligence in particular, including databases [8], model based diagnosis [6], knowledge compilation [5] and multi-agent systems [19].

A prime implicant is essential if it covers at least one model which is not covered by any other prime implicant. In the same way as the number of prime implicants, the number of essential prime implicants is possibly exponential [5]. The problem of generating the EPIes is crucial for the problem of Boolean function minimization (see e.g. the minimum cover problem [17]). Indeed, from the previous definition of the EPIes, one can see that every subset of prime implicants of a Boolean function that cover all its models contains all the EPIes of this function [12, 14], which shows in particular the interest of EPIes in converting

propositional formulas to Disjunctive Normal Form (DNF). In other words, the EPIs of a formula are included in any set of its prime implicants covering it. Our aim in this work is to study the problem of enumerating all the EPIs of a propositional formula in Conjunctive Normal Form (CNF).

There are many approaches in the literature for enumerating all or a part of the PIs of a CNF formula. One can particularly cite DPLL-like procedures [2, 16, 18], integer linear programming based encodings [11, 13] and SAT-based encoding [10] (SAT stands for satisfiability checking problem in propositional logic). However, there are few works in the literature considering the problem of enumerating all the EPIs. The authors of [3, 4] have proposed a method to compute implicitly the set of EPIs based on the use of Binary Decision Diagrams (BDDs). Similarly to several methods in the literature, the authors of [9] have provided a method for computing the EPIs from the set of all the PIs. Other methods are based on constructing (possibly) exponential size structures (see e.g. [1]) or using formula representations different from CNF (see e.g. [20]). To the best of our knowledge, there is no work on generating explicitly all the EPIs dedicated to the CNF formulas and benefiting from their structure without generating all its prime implicants.

The aim of this work is to propose methods for enumerating explicitly all the EPIs in the case of CNF formulas. We first provide some interesting computational complexity results. We show in particular that the problem of checking whether a prime implicant is essential is NP-complete. Then, we provide a simple characterization of the e-models in the case of CNF formulas. An e-model of a formula is a model covered by a unique prime implicant, which is necessarily essential. This characterization is used in particular for defining an algorithm for checking in linear time whether a model of a CNF formula is an e-model or not. Our characterization of the e-models is also used for defining two methods for enumerating all the EPIs. The first method is based on using a decision procedure (an oracle) for checking whether a model is an e-model or not and generating the EPIs from the found e-models. The second method corresponds to a SAT-based encoding. The base idea of this encoding consists in using additional propositional variables for representing the elements of every EPI and representing our characterization of the e-models by a propositional formula.

## 2 Background

We are here interested in the conjunctive normal form (CNF) representation for propositional formulas. A CNF formula is a conjunction of clauses where a *clause* is a disjunction of literals. A *literal* is a positive or negated propositional variable. It is well-known that every propositional formula can be translated to CNF w.r.t. the satisfiability problem (equisatisfiability) using Tseitin's linear encoding. Propositional variables are denoted by the lowercase letters  $p, q, r$ , and propositional formulas by the Greek letters  $\phi, \psi, \chi$ . Given a propositional formula  $\phi$ , we use  $Var(\phi)$  (resp.  $Lit(\phi)$ ) to denote the set of propositional variables (resp. literals) occurring in  $\phi$ . Further, given a literal  $l$ , we use  $\bar{l}$  to denote its complementary literal. Moreover, given a set of literals  $L$ , we use  $\bar{L}$  to denote the clause  $\bigvee_{l \in L} \bar{l}$ .

A CNF formula can also be seen as a set of clauses and a clause as a set of literals. Thus, a CNF formula of the form  $(l_1^1 \vee \dots \vee l_{k_1}^1) \wedge \dots \wedge (l_1^n \vee \dots \vee l_{k_n}^n)$  can be also represented in this work by the set of clauses  $\{(l_1^1 \vee \dots \vee l_{k_1}^1), \dots, (l_1^n \vee \dots \vee l_{k_n}^n)\}$  and the set of sets of literals  $\{\{l_1^1, \dots, l_{k_1}^1\}, \dots, \{l_1^n, \dots, l_{k_n}^n\}\}$ .

A tautological clause is a clause containing two complementary literals (e.g.  $p \vee \neg p \vee q \vee r$ ). One can easily see that the tautological clauses can be removed from a CNF formula without changing its truth value. From now on, we only consider the CNF formulas that does not contain any tautological clause.

A *Boolean interpretation* of a propositional formula  $\phi$  is an assignment that associates truth values in  $\{0, 1\}$  to propositional variables in  $Var(\phi)$ , where 0 stands for false and 1 stands for true. It is extended to propositional formulas as usual. A Boolean interpretation of  $\phi$  is *complete* if it gives a truth value to each variable in  $Var(\phi)$ , otherwise it is said to be *partial*. A *model* of a propositional formula is a complete Boolean interpretation satisfying this formula. The problem of determining if there exists a model that satisfies a given propositional formula, abbreviated as SAT, is one of the most studied NP-complete problems.

For convenience purposes, we represent from now on the Boolean interpretations as sets of literals. More precisely, the set of literals  $\{p_1, \dots, p_m, \neg q_1, \dots, \neg q_n\}$  represents the Boolean interpretation that associates 1 to the variables  $p_1, \dots, p_m$  and 0 to  $q_1, \dots, q_n$ . Note that to be a Boolean interpretation a set of literals does not have to contain a literal and its negation.

### 3 Essential Prime Implicants

In this section, we first describe the notions of prime implicant, essential prime implicant and e-model. We also provide some important computational complexity results. Further, we propose an algorithm for solving the problem of checking whether a model of a CNF formula is one of its e-models.

A partial Boolean interpretation that satisfies a propositional formula is called an *implicant* of this formula. Moreover, an interpretation  $\mathcal{I}$  of a formula  $\phi$  is called a *prime implicant* (in short PI) of this formula if for all literal  $l \in \mathcal{I}$ , the Boolean interpretation  $\mathcal{I} \setminus \{l\}$  does not satisfy  $\phi$ . In other words, a prime implicant is an implicant which is not covered by a more general implicant.

In Algorithm 1, we describe a simple method for computing a prime implicant from a model. In the for-loop, a literal is removed every time it is not necessary to get satisfiability. Algorithm 1 and other more efficient algorithms for computing a prime implicant from a model are proposed in [2, 7].

**Definition 1 (Essential Prime Implicant (EPI)).** *Given a propositional formula  $\phi$ , a prime implicant  $\mathcal{I}$  of  $\phi$  is essential if there exists a complete model  $\mathcal{M}$  of  $\phi$  such that  $\mathcal{I} \subseteq \mathcal{M}$  (we say  $\mathcal{I}$  covers  $\mathcal{M}$ ) and  $\mathcal{I}' \not\subseteq \mathcal{M}$  for every prime implicant  $\mathcal{I}'$  of  $\phi$  different from  $\mathcal{I}$ .*

In other words, an essential prime implicant is a prime implicant that covers a complete model that no other implicant is able to cover. From this definition, one can see that the essential prime implicants of a formula are included in

---

**Algorithm 1.** The algorithm  $Prime(\phi, \mathcal{M})$  for computing a prime implicant from a given model

---

**Data:** A CNF formula  $\phi$  and a complete model  $\mathcal{M}$  of  $\phi$ .

**Result:** A prime implicant of  $\phi$ .

```

1  $\mathcal{I} \leftarrow \mathcal{M}$ ;
2 for  $l \in \mathcal{M}$  do
3   | if  $\mathcal{I} \setminus \{l\}$  satisfies  $\phi$  then  $\mathcal{I} \leftarrow \mathcal{I} \setminus \{l\}$  ;
4 end
5 return  $\mathcal{I}$ ;

```

---

every set of its prime implicants that covers all its models. This shows why the problem of generating the EPIs is important for the problem of Boolean function minimization [17].

Let us for instance consider the CNF formula  $\phi = (p \vee q) \wedge (p \vee r)$ . This formula admits two prime implicants, namely  $\{p\}$  and  $\{q, r\}$ . The complete model  $\{p, q, r\}$  is covered by the two prime implicants of  $\phi$ . However, every other complete model is covered by only a unique prime implicant of  $\phi$ . For instance,  $\{p, \neg q, \neg r\}$  is only covered by  $\{p\}$  and  $\{\neg p, q, r\}$  is only covered by  $\{q, r\}$ . As a consequence,  $\{p\}$  and  $\{q, r\}$  are both EPIs. In this context, we call the particular complete models  $\{p, \neg q, \neg r\}$  and  $\{\neg p, q, r\}$  e-models.

**Definition 2 (E-Model).** *Given a propositional formula  $\phi$ , a complete model  $\mathcal{M}$  of  $\phi$  is an e-model if there exists one and only one prime implicant  $\mathcal{I}$  of  $\phi$  such that  $\mathcal{I} \subseteq \mathcal{M}$ . The prime implicant  $\mathcal{I}$  is necessarily essential.*

In Algorithm 2, we propose a simple method for checking whether a model is an e-model or not. Indeed,  $IsEModel(\phi, \mathcal{M})$  uses first  $Prime(\phi, \mathcal{M})$  to compute an arbitrary prime implicant  $\mathcal{I}$  of  $\phi$  that covers  $\mathcal{M}$  (see Line 1). In the while-loop, our algorithm checks whether there is no prime implicant other than the computed one covering the model  $\mathcal{M}$ . More precisely,  $IsEModel(\phi, \mathcal{M})$  checks whether there exists a literal in  $\mathcal{I}$  such that  $\mathcal{M} \setminus \{l\}$  satisfies  $\phi$ . If such a literal exists, then  $IsEModel(\phi, \mathcal{M})$  returns false (Line 4), otherwise it returns true (Line 7). The soundness of this algorithm comes from the fact that if there exists a literal  $l$  in  $\mathcal{I}$  such that  $\mathcal{M} \setminus \{l\}$  satisfies  $\phi$ , then there exists a prime implicant that covers  $\mathcal{M}$  and different from  $\mathcal{I}$ , which means that  $\mathcal{M}$  is not an e-model since it is covered by at least two prime implicants.

**Theorem 1.** *The problem of checking whether a model of a CNF formula is an e-model is in  $\mathcal{P}$ .*

The previous theorem is clearly a direct consequence of Algorithm 2.

**Theorem 2.** *The problem of checking whether a prime implicant of a CNF formula is essential is  $\mathcal{NP}$ -complete.*

*Proof.* One can easily see that this problem is in  $\mathcal{NP}$ . Indeed, a prime implicant  $\mathcal{I}$  of a formula  $\phi$  is an EPI if there exists an e-model  $\mathcal{M}$  s.t.  $\mathcal{I} \subseteq \mathcal{M}$ , and using

---

**Algorithm 2.** The algorithm  $IsEModel(\phi, \mathcal{M})$  for checking whether a model is an e-model or not

---

**Data:** A CNF formula  $\phi$ , a complete model  $\mathcal{M}$  of  $\phi$ .

**Result:** true or false according to whether or not  $\mathcal{M}$  is an e-model.

```

1  $\mathcal{I} \leftarrow Prime(\phi, \mathcal{M});$ 
2 for  $l \in \mathcal{I}$  do
3   | if  $\mathcal{M} \setminus \{l\}$  satisfies  $\phi$  then
4   |   | return false ;
5   | end
6 end
7 return true ;

```

---

Theorem 1, we know that the problem of checking whether a model of a given formula is an e-model is in  $\mathcal{P}$ . Let us now show that this problem is  $\mathcal{NP}$ -hard. To this end, we use the NP-complete problem SAT. Let  $\phi$  be a CNF formula. We associate to each clause  $c \equiv l_1 \vee \dots \vee l_k$  in  $\phi$  a fresh propositional variable  $p_c$  and the set of binary clauses  $E_c = \{p_c \vee \bar{l}_1, \dots, p_c \vee \bar{l}_k\}$ . In this context, it is easy to show that  $\mathcal{I}_\phi = \{p_c \mid c \in \phi\}$  is an EPI of  $\psi \equiv \bigcup_{c \in \phi} E_c$  if and only if the formula  $\phi$  is satisfiable. We first show that  $\mathcal{I}_\phi$  is a prime implicant of  $\psi$ . Assume that there exists a clause  $c \equiv l_1 \vee \dots \vee l_k$  s.t.  $\mathcal{I}_\phi \setminus \{p_c\}$  satisfies  $\psi$ . However,  $\mathcal{I}_\phi \setminus \{p_c\} \cup \{\neg p_c, l_1, \dots, l_k\}$  is a countermodel of  $\psi$  and we get a contradiction. As a consequence,  $\mathcal{I}_\phi$  is a prime implicant of  $\psi$ .

Assume that  $\phi$  is satisfiable and  $\mathcal{M}$  is a complete model of  $\phi$ . Clearly,  $\mathcal{I}_\phi \cup \mathcal{M}$  is a model of  $\psi$ . Moreover, for all clause  $c$  in  $\phi$ , there exists  $l \in c$  such that  $p_c \vee \neg l$  in  $\psi$  and  $l \in \mathcal{M}$ . Thus, for all  $c \in \phi$ ,  $(\mathcal{I}_\phi \setminus \{p_c\}) \cup \mathcal{M}$  does not satisfy  $\psi$ . Therefore,  $\mathcal{I}_\phi \cup \mathcal{M}$  is an e-model of  $\psi$  and  $\mathcal{I}_\phi$  is an essential prime implicant.

Assume now that  $\mathcal{I}_\phi$  is an essential prime implicant of  $\psi$ . Then, there exists a complete Boolean interpretation  $\mathcal{M}$  of  $\phi$  such that  $\mathcal{I}_\phi \cup \mathcal{M}$  is an e-model of  $\psi$ . Assume that  $\mathcal{M}$  is a countermodel of  $\phi$ . Then, there exists a clause  $c \in \phi$  such that for all  $l \in c$  we have  $\neg l \in \mathcal{M}$ . Thus,  $(\mathcal{I}_\phi \setminus \{p_c\}) \cup \mathcal{M}$  satisfies  $\psi$  and we get a contradiction since  $\mathcal{I}_\phi \cup \mathcal{M}$  is an e-model of  $\psi$  and  $\mathcal{I}$  an essential prime implicant of  $\psi$ .

## 4 A Characterization of E-Models

In this section, we provide a simple characterization of the e-models of a CNF formula. This characterization is used in particular for defining an algorithm for checking in linear time whether a model of a CNF formula is an e-model.

Given a CNF formula  $\phi$  and a complete model  $\mathcal{M}$  of  $\phi$ , we use  $U(\phi, \mathcal{M})$  to denote the subset of literals  $\{l \in \mathcal{M} \mid \exists c \in \phi, c \cap \mathcal{M} = \{l\}\}$ . In other words, a literal belongs to  $U(\phi, \mathcal{M})$  if and only if there exists a clause in  $\phi$  which is satisfied by only this literal using  $\mathcal{M}$ . From the definition of  $U(\phi, \mathcal{M})$ , we clearly get the following property.

**Proposition 1.** *Let  $\phi$  be a CNF formula and  $\mathcal{M}$  a complete model of  $\phi$ . Then, for all  $l \in U(\phi, \mathcal{M})$ ,  $\mathcal{M} \setminus \{l\}$  does not satisfy  $\phi$ .*

In the following theorem, we use  $U(\phi, \mathcal{M})$  to provide a characterization of the e-models in the case of CNF formulas.

**Theorem 3.** *Let  $\phi$  be a CNF formula and  $\mathcal{M}$  a complete model of  $\phi$ . Then,  $\mathcal{M}$  is an e-model of  $\phi$  iff for all  $l \in \mathcal{M}$  and for all clause  $c \in \phi$  with  $l \in c$ , there exists a literal  $l' \in c$ , possibly  $l$ , s.t.  $l' \in U(\phi, \mathcal{M})$ .*

*Proof. Part  $\Rightarrow$ .* We first assume that  $\mathcal{M}$  is an e-model of  $\phi$ . Then,  $\mathcal{M}$  is covered by a unique prime implicant  $\mathcal{I}$  of  $\phi$ . One can easily see that for all  $l \in \mathcal{I}$  we get  $l \in U(\phi, \mathcal{M})$  since  $\mathcal{I}$  is the unique prime implicant covering  $\mathcal{M}$ . Indeed, if there exists  $l \in \mathcal{I}$  s.t.  $l \notin U(\phi, \mathcal{M})$ , then  $\mathcal{M} \setminus \{l\}$  satisfies  $\phi$ . Using the fact that there exists a prime implicant of  $\phi$  covering  $\mathcal{M} \setminus \{l\}$  (a prime implicant that covers  $\mathcal{M}$  and different from  $\mathcal{I}$ ), we get a contradiction with the fact that  $\mathcal{M}$  is an e-model. Moreover, using the fact that  $\mathcal{I}$  satisfies  $\phi$ , we get  $c \cap \mathcal{I} \neq \emptyset$  for all clause  $c$  in  $\phi$ . Thus, for all  $l \in \mathcal{M}$  and for all clause  $c \in \phi$  with  $l \in c$ , there exists a literal  $l' \in c$  s.t.  $l' \in U(\phi, \mathcal{M})$  ( $l' \in \mathcal{I}$ ).

*Part  $\Leftarrow$ .* Assume now that for all  $l \in \mathcal{M}$  and for all clause  $c \in \phi$  with  $l \in c$ , there exists a literal  $l' \in c$  s.t.  $l' \in U(\phi, \mathcal{M})$ . Then, we have  $U(\phi, \mathcal{M})$  satisfies  $\phi$ . Moreover, using the definition of  $U(\phi, \mathcal{M})$ ,  $U(\phi, \mathcal{M})$  is an EPI of  $\phi$ . Indeed, this comes from the fact that for all  $l \in U(\phi, \mathcal{M})$ , there exists a clause  $c$  in  $\phi$  s.t.  $c \cap \mathcal{M} = \{l\}$ , i.e., for all  $l \in U(\phi, \mathcal{M})$ ,  $\mathcal{M} \setminus \{l\}$  does not satisfy  $\phi$ . Knowing that  $U(\phi, \mathcal{M})$  is an EPI of  $\phi$ ,  $\mathcal{M}$  is an e-model of  $\phi$ .

Given a CNF formula  $\phi$  and a model  $\mathcal{M}$  of  $\phi$ , the fact that for all  $l \in \mathcal{M}$  and for all clause  $c \in \phi$  with  $l \in c$ , there exists a literal  $l' \in c$  such that  $l' \in U(\phi, \mathcal{M})$ , means that  $U(\phi, \mathcal{M})$  satisfies  $\phi$ . Moreover, using the definition of  $U(\phi, \mathcal{M})$ , we obtain that  $U(\phi, \mathcal{M})$  is a prime implicant of  $\phi$ . Thus, we get the following corollary as a direct consequence of Theorem 3.

**Corollary 1.** *Given a CNF formula  $\phi$  and a complete model  $\mathcal{M}$  of  $\phi$ ,  $\mathcal{M}$  is an e-model of  $\phi$  iff  $U(\phi, \mathcal{M})$  is a prime implicant (an EPI a fortiori) of  $\phi$ .*

Let us consider again the formula  $\phi = (p \vee q) \wedge (p \vee r)$  and its models  $\mathcal{M}_1 = \{\neg p, q, r\}$  and  $\mathcal{M}_2 = \{p, \neg q, \neg r\}$ . Then, we have  $U(\phi, \mathcal{M}_1) = \{q, r\}$  and  $U(\phi, \mathcal{M}_2) = \{p\}$ . Using the fact that  $\{q, r\}$  and  $\{p\}$  are prime implicants of  $\phi$ , we obtain that  $\mathcal{M}_1$  and  $\mathcal{M}_2$  are e-models of  $\phi$ .

Using Corollary 1, to check whether or not a given complete model  $\mathcal{M}$  of a given formula  $\phi$  is an e-model of the latter, we only need to check that  $U(\phi, \mathcal{M})$  satisfies  $\phi$ . Indeed, using the definition of  $U(\phi, \mathcal{M})$ , we obtain  $U(\phi, \mathcal{M})$  is a prime implicant of  $\phi$  iff  $U(\phi, \mathcal{M})$  satisfies  $\phi$ . Moreover, we know that if  $\mathcal{M}$  is an e-model of  $\phi$ , then its associated EPI is  $U(\phi, \mathcal{M})$ .

In Algorithm 3, we propose another algorithm for solving in linear time the problem of checking whether a model is an e-model or not using the previous approach. In the first for-loop,  $IsEModel2(\phi, \mathcal{M})$  computes in linear time the



---

**Algorithm 3.** The algorithm  $IsEModel2(\phi, \mathcal{M})$  for checking if a model is an e-model

---

**Data:** A CNF formula  $\phi$ , a complete model  $\mathcal{M}$  of  $\phi$ .

**Result:** true or false according to whether or not  $\mathcal{M}$  is an e-model of  $\phi$ .

```

1  $U \leftarrow \emptyset$ ;
2  $\psi \leftarrow \emptyset$ ;
3 for  $c \in \phi$  do
4   if  $|c \cap \mathcal{M}| = 1$  then
5      $U \leftarrow U \cup (c \cap \mathcal{M})$ ;
6   else
7     if  $c \cap U = \emptyset$  then  $\psi \leftarrow \psi \cup \{c\}$ ;
8   end
9 end
10 for  $c \in \psi$  do
11   if  $c \cap U = \emptyset$  then return false;
12 end
13 return true;

```

---

value of  $U(\phi, \mathcal{M})$  which is saved in the variable  $U$ . In the second for-loop, our algorithm checks in linear time if  $U(\phi, \mathcal{M})$  satisfies  $\phi$ . The soundness of Algorithm 3 is a direct consequence of Corollary 1.

## 5 SAT-based Methods for Generating EPIEs

In this section, we propose two methods for generating all the EPIEs of a CNF formula by using our characterization of the e-models. The first method is based on using a decision procedure (an oracle) for checking whether a model is an e-model or not, while the second method uses a SAT encoding for enumerating the essential prime implicants.

### 5.1 A Method Based on an E-Model Checking Procedure

In Algorithm 4, we describe our first method for generating all the EPIEs of a CNF formula. In the following, we describe the instructions of this algorithm. In Line 3,  $SAT(\cdot)$  is a decision procedure for checking the satisfiability of a CNF formula (A SAT oracle). In Line 4,  $IsEModel - Oracle(\cdot, \cdot)$  is an e-model checking procedure. For instance, one can use the algorithm  $IsEModel2(\cdot, \cdot)$  as an e-model checking procedure.

In the while loop, if the found model  $\mathcal{M}$  is an e-model,  $U(\phi, \mathcal{M})$  is added to the set of EPIEs and the clause  $\overline{U(\phi, \mathcal{M})}$  is added to  $\psi$  to avoid finding any other e-model which is covered by  $U(\phi, \mathcal{M})$  (see Lines 4–6). Otherwise,  $\mathcal{M}$  is not an e-model and the clause  $\overline{\mathcal{M}}$  is added to  $\psi$  to find different models in the next iterations (see Line 8).

**Theorem 4.** *Algorithm 4 is sound, i.e., it terminates and enumerates all the EPIEs of the input CNF formula.*

---

**Algorithm 4.** The algorithm *ModEnum4EPIes* for generating the EPIes of a CNF formula

---

**Data:** A CNF formula  $\phi$ .  
**Result:** The EPIes of  $\phi$ .

```

1  $L \leftarrow \emptyset$ ;
2  $\psi \leftarrow \phi$ ;
3 while SAT( $\psi$ ) do
    /*  $\mathcal{M}$  is a model of  $\psi$  */
4   if IsEModel-Oracle( $\phi, \mathcal{M}$ ) then
5      $L \leftarrow L \cup \{U(\phi, \mathcal{M})\}$ ;
6      $\psi \leftarrow \psi \wedge \overline{U(\phi, \mathcal{M})}$ ;
7   else
8      $\psi \leftarrow \psi \wedge \overline{\mathcal{M}}$ ;
9   end
10 end
11 return  $L$ ;
```

---

*Proof.* Using the instructions at Line 6 and Line 8, *ModEnum4EPIes* terminates since the number of models is finite. Using Corollary 1, we get that the set returned by *ModEnum4EPIes* contains only EPIes. Indeed, for all e-model  $\mathcal{M}$  of  $\phi$ , the set of literals  $U(\phi, \mathcal{M})$  is an EPI of  $\phi$ .

Assume now that there exists an EPI  $\mathcal{I}$  of  $\phi$  that does not belong to the set returned by *ModEnum4EPIes*. Then, there exists a Boolean interpretation  $\mathcal{M}$  such that  $\mathcal{I} \subseteq \mathcal{M}$  and  $\mathcal{M}$  is an e-model of  $\phi$  and, using again Corollary 1,  $\mathcal{I} = U(\phi, \mathcal{M})$  holds. Knowing that every e-model is covered by a unique EPI, we get a contradiction since the clauses added to  $\phi$  during the computation allows only to avoid the found EPIes and the models that are not e-models.

## 5.2 A SAT-Based Encoding

We here provide our SAT-based encoding for enumerating all the EPIes. The base idea consists in using additional propositional variables for representing the elements of  $U(\phi, \mathcal{M})$  described previously and used to characterize the e-models.

Given a CNF formula  $\phi$ , we use  $\mathcal{E}nc(\phi)$  to denote the following formula:

$$\phi \wedge \left( \bigwedge_{l \in Lit(\phi)} r_l \leftrightarrow (l \wedge \bigvee_{c \in \phi, l \in c} \bigwedge_{l' \in c \setminus \{l\}} \overline{l'}) \right) \wedge \left( \bigwedge_{c \in \phi} \bigvee_{l \in c} r_l \right)$$

where the propositional variables of the form  $r_l$  are fresh propositional variables, i.e., for every literal  $l \in Lit(\phi)$ , a new propositional variable  $r_l$  is associated to  $l$ .

The two following propositions shows the soundness of our encoding in the sense that it allows to enumerate all the EPIes in the case of CNF formulas.

**Proposition 2.** *Given a CNF formula  $\phi$ , if  $\mathcal{M}$  is a model of  $\mathcal{E}nc(\phi)$  then  $\mathcal{I} = \{l \in Lit(\phi) \mid r_l \in \mathcal{M}\}$  is an EPI of  $\phi$ .*

---

**Algorithm 5.** A SAT-based Approach for Enumerating the EPIes of a CNF formula

---

```

Data: A CNF formula  $\phi$ .
Result: The EPIes of  $\phi$ .
1  $L \leftarrow \emptyset$ ;
2  $\psi \leftarrow \mathcal{Enc}(\phi)$ ;
3 while  $SAT(\psi)$  do
    | /*  $\mathcal{M}$  is a model of  $\psi$  */ */
4 |  $U \leftarrow \{l \in Lit(\phi) \mid r_l \in \mathcal{M}\}$ ;
5 |  $L \leftarrow L \cup \{U\}$ ;
6 |  $\psi \leftarrow \psi \wedge \bigvee_{l \in U} \bar{l}$ 
7 end
8 return  $L$ ;

```

---

*Proof.* Knowing that  $\mathcal{M}$  is a model of  $\mathcal{Enc}(\phi)$ , the Boolean interpretation  $\mathcal{M} \cap Lit(\phi)$  is a model of  $\phi$  since  $\phi$  is a subformula of  $\mathcal{Enc}(\phi)$ . Using the fact that  $r_l \rightarrow l$  is a logical consequence of  $\mathcal{Enc}(\phi)$  for every  $l \in Lit(\phi)$ ,  $\mathcal{M} \cap \{r_l, r_{\bar{l}}\} \leq 1$  holds for every  $l \in Lit(\phi)$ . As a consequence,  $\mathcal{I}$  does not contain any complementary literals. Using the subformula  $\bigwedge_{c \in \phi} \bigvee_{l \in c} r_l$ , we obtain that  $\mathcal{I}$  is an implicant of  $\phi$ , i.e., it satisfies  $\phi$ . Furthermore, using the subformulas of the form  $\bigvee_{c \in \phi, l \in c} \bigwedge_{l' \in c \setminus \{l\}} \bar{l}'$ , we know that, for all  $l \in \mathcal{M} \cap Lit(\phi)$ ,  $l \in \mathcal{I}$  iff there exists a clause  $c \in \phi$  such that  $c \cap \mathcal{I} = \{l\}$ . Thus, we get  $\mathcal{I} = \{l \in \mathcal{M} \cap Lit(\phi) \mid \exists c \in \phi, c \cap \mathcal{M} = \{l\}\} = U(\phi, \mathcal{M} \cap Lit(\phi))$ . Therefore, using Corollary 1,  $\mathcal{M} \cap Lit(\phi)$  is an e-model of  $\phi$  that is covered by the EPI  $\mathcal{I}$ .

**Proposition 3.** *Given a CNF formula  $\phi$ , if  $\mathcal{I}$  is an EPI of  $\phi$ , then there exists a model  $\mathcal{M}$  of  $\mathcal{Enc}(\phi)$  s.t.  $\mathcal{I} = \{l \in Lit(\phi) \mid r_l \in \mathcal{M}\}$ .*

*Proof.* Using the fact that  $\mathcal{I}$  is an EPI of  $\phi$ , there exists an e-model  $\mathcal{M}'$  of  $\phi$  such that  $\mathcal{I} \subseteq \mathcal{M}'$ . Then, using Corollary 1,  $\mathcal{I} = U(\phi, \mathcal{M}')$  holds. We know that  $R = \{r_l \mid l \in \mathcal{I}\} \cup \{\bar{r}_l \mid l \in Lit(\phi) \setminus \mathcal{I}\}$  satisfies the subformula  $\bigwedge_{c \in \phi} \bigvee_{l \in c} r_l$  since  $\mathcal{I}$  satisfies  $\phi$ . We also know that  $R \cup \mathcal{M}'$  satisfies the subformula  $\bigwedge_{l \in Lit(\phi)} r_l \leftrightarrow (l \wedge \bigvee_{c \in \phi, l \in c} \bigwedge_{l' \in c \setminus \{l\}} \bar{l}')$  since  $\mathcal{I} = U(\phi, \mathcal{M}')$ . Therefore,  $\mathcal{M} = R \cup \mathcal{M}'$  is a model of  $\mathcal{Enc}(\phi)$  where  $\mathcal{I} = \{l \in Lit(\phi) \mid r_l \in \mathcal{M}\}$ .

Algorithm 5 describes how to use our SAT-based encoding for enumerating all the EPIes of a CNF formula. The soundness of this algorithm is a direct consequence of the soundness of our encoding. Note that every time a solution  $\mathcal{M}$  is found, we add the clause  $\bigvee_{l \in U} \bar{l}$  instead of the clause  $\bigvee_{l \in \mathcal{M}} \bar{l}$  to avoid to found the same EPIes in the next iterations.

The interest of our encoding is not only for generating all the EPIes, it allows also to focus on EPIes having particular properties that can be expressed within SAT. For instance, if we want to enumerate all the EPIes that contain at least one of the literals  $l_1, \dots, l_n$ , we just need to add to our encoding the clause  $l_1 \vee \dots \vee l_n$ . Moreover, the encoding  $\mathcal{Enc}(\phi)$  can be used for solving problems

related to the notion of EPI. For example, it can be used for solving the problem of checking whether or not a CNF formula has at least one EPI. Indeed,  $\mathcal{Enc}(\phi)$  is satisfiable if and only if  $\phi$  has an EPI. Our encoding can also be used for solving the problem mentioned in Sect. 3 that consists in checking whether or not a prime implicant is essential. For this purpose, we only have to check whether a formula obtained from our encoding is satisfiable or not. More precisely, the prime implicant  $\mathcal{I}$  of  $\phi$  is essential if and only if the formula obtained from  $\mathcal{Enc}(\phi)$  by assigning the truth values in  $\mathcal{I} \cup \{r_l, \bar{r}_l \mid l \in \mathcal{I}\}$  is satisfiable.

Furthermore, our encoding can be used for reasoning about the size of EPIs, by way of illustration, for computing a minimum size essential prime implicant. As a side note, the problem of computing a minimum size prime implicant has been studied in [11] and solved through an Integer Linear Programming (ILP) model. A solution of the problem of computing a minimum size essential prime implicant can be obtained by using our encoding in a Partial Max-SAT (P-Max-SAT) model. Let us recall that P-Max-SAT consists in solving instances of the form  $(\phi_H, \phi_S)$  where  $\phi_H$ , called *hard part*, and  $\phi_S$ , called *soft part*, are sets of clauses. A solution of a P-Max-SAT instance  $(\phi_H, \phi_S)$  is a Boolean interpretation that satisfies  $\phi_H$  and a maximum number of clauses in  $\phi_S$ . Given a CNF formula  $\phi$ , we use  $MinSize(\phi)$  to denote the following P-Max-SAT model:

*Hard Part:*  $\mathcal{Enc}(\phi)$

*Soft Part:*  $\neg r_l$  for  $l \in Lit(\phi)$

Clearly, every solution of  $MinSize(\phi)$  allows us to get a minimum size essential prime implicant of  $\phi$ . Indeed, given a solution  $\mathcal{M}$  of  $MinSize(\phi)$ ,  $\{l \in Lit(\phi) \mid r_l \in \mathcal{M}\}$  is an essential prime implicant of  $\phi$  since  $\mathcal{M}$  satisfies  $\mathcal{Enc}(\phi)$ . Knowing that  $\mathcal{M}$  satisfies a maximum number of clauses in the soft part, we obtain that  $\{l \in Lit(\phi) \mid r_l \in \mathcal{M}\}$  is a minimum size essential prime implicant.

It is worth mentioning that a SAT-based encoding for enumerating all the prime implicants of a CNF formula has been proposed in [10]. It consists in encoding the input CNF formula as a new one, so that the models of the encoding represent all the prime implicants of the original formula. In the same way as our encoding, Jabbour et al's encoding uses additional propositional variables to represent the literals belonging to the prime implicants.

## 6 Conclusion and Perspectives

In this paper, we proposed methods for enumerating explicitly all the essential prime implicants in the case of CNF formulas without generating other prime implicants. These methods are based on a simple characterization of the e-models of a CNF formula. An e-model is a model covered by a unique prime implicant, which is necessarily essential. Our characterization was used for proposing a linear time algorithm for solving the problem of checking whether or not a model of a CNF formula is an e-model. Then, our first method for enumerating all the essential prime implicants was defined by generating the essential prime implicants from the e-models through the use of an oracle that solves the latter problem. Our second method corresponds to a SAT-based encoding where the

base idea consists in using additional propositional variables for getting the elements of every essential prime implicant and representing our characterization of the e-models by a propositional formula.

A future work, we intend to implement and evaluate the proposed methods for generating the essential prime implicants. We also plan to study tractable classes in the case of the problem of checking whether a prime implicant is essential.

## References

1. Caruso, G.: A local selection algorithm for switching function minimization. *IEEE Trans. Comput.* **C-33**(1), 91–97 (1984)
2. Castell, T.: Computation of prime implicates and prime implicants by a variant of the Davis and Putnam procedure. In: Eighth International Conference on Tools with Artificial Intelligence, ICTAI 1996, pp. 428–429 (1996)
3. Coudert, O., Madre, J.C.: Implicit and incremental computation of primes and essential primes of Boolean functions. In: Proceedings of the 29th Design Automation Conference, Anaheim, California, USA, pp. 36–39 (1992)
4. Coudert, O., Madre, J.C.: A new method to compute prime and essential prime implicants of Boolean functions. In: Knight, T., Savage, J. (eds.) Proceedings of the 1992 Brown/MIT Conference on Advanced Research in VLSI and Parallel Systems, pp. 113–128 (1992)
5. Darwiche, A., Marquis, P.: A knowledge compilation map. *J. Artif. Intell. Res.* **17**, 229–264 (2002)
6. de Kleer, J., Mackworth, A.K., Reiter, R.: Characterizing diagnoses. In: Proceedings of the 8th National Conference on Artificial Intelligence (AAAI 1990), pp. 324–330 (1990)
7. Déharbe, D., Fontaine, P., Le Berre, D., Mazure, B.: Computing prime implicants. In: Formal Methods in Computer-Aided Design, FMCAD 2013, Portland, OR, USA, pp. 46–52 (2013)
8. del Val, A.: Tractable databases: how to make propositional unit resolution complete through compilation. In Proceedings of the 4th International Conference on Principles of Knowledge Representation and Reasoning (KR 1994), pp. 551–561 (1994)
9. Ignatiev, A., Previti, A., Marques-Silva, J.: SAT-based formula simplification. In: Heule, M., Weaver, S. (eds.) SAT 2015. LNCS, vol. 9340, pp. 287–298. Springer, Cham (2015). [https://doi.org/10.1007/978-3-319-24318-4\\_21](https://doi.org/10.1007/978-3-319-24318-4_21)
10. Jabbour, S., Marques-Silva, J., Sais, L., Salhi, Y.: Enumerating prime implicants of propositional formulae in conjunctive normal form. In: Fermé, E., Leite, J. (eds.) JELIA 2014. LNCS (LNAI), vol. 8761, pp. 152–165. Springer, Cham (2014). [https://doi.org/10.1007/978-3-319-11558-0\\_11](https://doi.org/10.1007/978-3-319-11558-0_11)
11. Manquinho, V.M., Flores, P.F., Marques-Silva, J.P., Oliveira, A.L.: Prime implicant computation using satisfiability algorithms. In: 9th International Conference on Tools with Artificial Intelligence, ICTAI 1997, pp. 232–239 (1997)
12. McCluskey, E.J.: Minimization of Boolean functions. *Bell Syst. Tech. J.* **35**(6), 1417–1444 (1956)
13. Pizzuti, C.: Computing prime implicants by integer programming. In: Eighth International Conference on Tools with Artificial Intelligence, ICTAI 1996, pp. 332–336 (1996)

14. Quine, W.V.: The problem of simplifying truth functions. *Am. Math. Monthly* **59**(8), 521–531 (1952)
15. Quine, W.V.: On cores and prime implicants of truth functions. *Am. Math. Monthly* **66**(9), 755–760 (1959)
16. Ravi, K., Somenzi, F.: Minimal assignments for bounded model checking. In: Jensen, K., Podelski, A. (eds.) *TACAS 2004*. LNCS, vol. 2988, pp. 31–45. Springer, Heidelberg (2004). [https://doi.org/10.1007/978-3-540-24730-2\\_3](https://doi.org/10.1007/978-3-540-24730-2_3)
17. Rudell, R.L.: Multiple-valued logic minimization for PLA synthesis. Technical report, EECS Department, University of California, Berkeley (1986)
18. Schrag, R.: Compilation for critically constrained knowledge bases. In: *Proceedings of the Thirteenth National Conference on Artificial Intelligence and Eighth Innovative Applications of Artificial Intelligence Conference, AAAI 1996, IAAI 1996*, pp. 510–515 (1996)
19. Slavkovik, M., Ågotnes, T.: A judgment set similarity measure based on prime implicants. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems, AAMAS 2014*, pp. 1573–1574 (2014)
20. Strzemecki, T.: Polynomial-time algorithms for generation of prime implicants. *J. Complexity* **8**(1), 37–63 (1992)
21. Tison, P.: Generalization of consensus theory and application to the minimization of Boolean functions. *IEEE Trans. Electron. Comput.* **EC-16**(4), 446–456 (1967)



# Evolving a Team of Asymmetric Predator Agents That Do Not Compute in Predator-Prey Pursuit Problem

Ivan Tanev<sup>1</sup>(✉), Milen Georgiev<sup>1</sup>, Katsunori Shimohara<sup>1</sup>,  
and Thomas Ray<sup>2</sup>

<sup>1</sup> Doshisha University, Kyotanabe, Kyoto 610-0321, Japan  
itanev@mail.doshisha.ac.jp

<sup>2</sup> University of Oklahoma, Norman, Oklahoma 73019, OK, USA

**Abstract.** We herein revisit the predator-prey pursuit problem – using very simple predator agents. The latter – intended to model the emerging micro- and nano-robots – are morphologically simple. They feature a single line-of-sight sensor and a simple control of their two thrusters. The agents are behaviorally simple as well – their decision-making involves no computing, but rather – a direct mapping of the few perceived environmental states into the corresponding pairs of thrust values. We apply genetic algorithms to evolve such a mapping that results in the successful behavior of the team of these predator agents. To enhance the generality of the evolved behavior, we propose an asymmetric morphology of the agents – an angular offset of their sensor. Our experimental results verify that the offset of both  $20^\circ$  and  $30^\circ$  yields efficient and consistent evolution of successful behaviors of the agents in all tested initial situations.

**Keywords:** Simple agents · Micro-robots · Asymmetric morphology  
Predator-prey problem · Genetic algorithms

## 1 Introduction

Multi-agent systems (MAS) are widely applied for problem solving, software engineering, and the simulation of (human, robotic, etc.) societies. Owing to their complex, non-linear nature, MAS can often solve problems that a single agent cannot tackle. In our work, we consider MAS as a model of a society of mobile robots. We are especially interested in the emerging small-scale robots – micro- and nano-robots – that are promising candidates in future manufacturing and biomedicine [1]. However, several challenges are currently hindering the progress of the real-world applicability of these robots. Because of the physical constraints due to their small size, these robots could not be morphologically advanced – both the sensors and the actuators would have to be rather simple. Further, their behavior would be simple as well. It would not involve any computing; instead, it would feature a direct mapping of the (few) perceived environmental states into actuators commands. Additionally, such robots would most likely be unable to communicate with each other. As an example of such robots, we consider robots equipped with a single line-of-sight sensor providing only two bits of

information, and two thrusters (wheels, in two dimensions, or propellers, in three-dimensional environments) in a differential drive configuration, controlled by two motors. Specifically, we focus on the two-dimensional (2D) implementation of such robots, first modeled as agents by Gauci et al. [2]. The agents could self-organize to solve the simple robot aggregation problem. A similar framework also successfully solved the more complex object-clustering problem, in which the agents need to interact with an introduced static object [3]. The ability of such agents to conduct a social (surrounding) behavior in an environment featuring dynamic objects was demonstrated in solving the shepherding problem, where a team of simple agents guided multiple dynamic agents toward a defined goal [4].

In our current research, we considered the emergence of complex social behaviors of a team of similar and very simple agents in a different task – the well-studied, yet difficult to solve predator-prey pursuit problem (PPPP) [5–8]: eight identical, simple agents (predators) are used to capture a single dynamic agent (prey). Our *objective* is to investigate whether the PPPP is solvable by the team of such simple predator agents. Further, we investigated the feasibility of applying genetic algorithms (GA) to evolve direct mapping of the four perceived environmental states into the respective velocities of the wheels of the predators that yield the social behavior of the predators, resulting in the successful capturing of the prey.

The primary *motivation* of our work is the recognition that several real-world scenarios – such as pinpoint drug delivery, surrounding and destroying (cancer) cells or bacteria, and gathering around cells to facilitate their repair or imaging [1] – could be modeled by the our new instance of the PPPP.

The remainder of this article is organized as follows. Section 2 describes the entities in the PPPP. In Sect. 3, we elaborate the GA, adopted for the evolution of predator behaviors. In Sect. 4, we present the experimental results and introduce the proposed asymmetric morphology of predators. In the same section, we elaborate on the robustness of the evolved behavior. In Sect. 5 we discuss the effect of varying the degree of the asymmetry of the morphology on the behavior of the predators. We draw a conclusion in Sect. 6.

## 2 The Entities

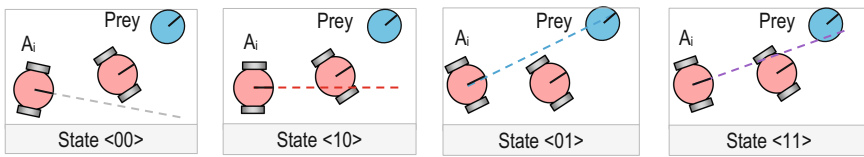
Each of the eight identical *predators* models a simple cylindrical robot with a sensor featuring a limited range, and two wheels, controlled by two motors in a differential drive configuration. The features of the predators are shown in Table 1.

The sensor provides two bits of information: each bit encodes if an entity (predator or prey) – is detected in the line of sight. The sensor, aligned with the longitudinal axis of the agent, could comprise two photodetectors, sensitive to non-overlapping wavelengths of (ultraviolet, visible, or infrared) light emitted by the predators and prey, respectively. Such sensors allow the predators to perceive only four discrete environmental states, as shown in Fig. 1. The state <11> is the most challenging one, and it could be sensed under the following two assumptions: the prey is taller than the predators, and do not obscure the shorter predators, the cross-section of the prey is either narrower than that of the predators, or (at least partially) transparent for the light



**Table 1.** Features of the predator- and prey agents

Feature	Value of the feature	
	Predators	Prey
Number of agents	8	1
Diameter (and wheel axle track), units	16	16
Max linear velocity of wheels, units/s	10	10
Max speed of agents, units/s	10	10
Type of sensor	Single line-of-sight	Omni-directional
Range of visibility of the sensor, units	200	50
Orientation of sensor	Parallel to longitudinal axis	NA



**Fig. 1.** The four possible environmental states perceived by (any) predator agent  $A_i$ .

perceived by the predators. The perceived environmental states do not provide the predators with any insight about the distance to the perceived entities, nor their total number.

The reactive behavior of the predator agents could be described as a direct mapping of each of the four perceived environmental states into a corresponding rotational speed of the wheel motors. For simplicity, hereafter, we will assume a mapping into the linear velocities of the wheels, expressed as a percentage – within the range  $[-100\% \dots +100\%]$  – of their respective maximum linear velocities. The decision-making of the predator agents could be formally expressed by the following octet  $D$ :

$$D = \{V_{00L}, V_{00R}, V_{01L}, V_{01R}, V_{10L}, V_{10R}, V_{11L}, V_{11R}\} \tag{1}$$

where  $V_{00L}$ ,  $V_{00R}$ ,  $V_{01L}$ ,  $V_{01R}$ ,  $V_{10L}$ ,  $V_{10R}$ ,  $V_{11L}$ , and  $V_{11R}$  are the linear velocities (as a percentage of the maximum linear velocity) of the left and right wheels of the predators for the perceived environmental states  $\langle 00 \rangle$ ,  $\langle 01 \rangle$ ,  $\langle 10 \rangle$ , and  $\langle 11 \rangle$ , respectively.

Our *objective* of evolving (via GA) the optimal direct mapping of the four perceived environmental states into their respective velocities of wheels could be rephrased as evolving such values of the velocities, shown in the octet in Eq. (1), resulting in an efficient capturing behavior of the team of predator agents.

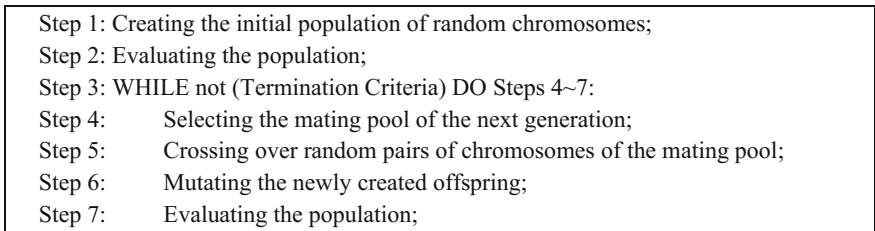
The *prey* features an omnidirectional sensor with limited visibility range. The visibility range is shorter – 50 units (e.g., nm,  $\mu\text{m}$ , mm, etc.) compared to the 200 units of predators (Table 1). The maximum speed of the prey, however, is identical to that of the predators. Such sensory- and moving abilities of the entities result in an inherently cooperative environment for the predators: it would be impossible for them to capture the prey alone, without cooperating with each other. In contrast to the predator

behaviors, the prey behavior is handcrafted. It attempts to escape from the closest predator (if any) by running at its maximum speed in the direction that is opposite to the direction of the predator, if the latter is detected [8]; otherwise, it remains still.

The *world* is simulated as an unbounded 2D area. The perceptions, decision making, and the resulting new state (e.g., location, orientation, and speed) of the agents are updated with sampling interval of 100 ms. The duration of trials is 120 s, modeled in 1200 time steps. We approximated the new state of predators in the following two stages. First, we calculated the new orientation from the current orientation, yaw rate (obtained from the difference between the linear velocities of the wheels, and the length of the axis between wheels), and sampling interval duration. Subsequently, we calculated the new position (as 2D Cartesian coordinates) as a projection (in time, equal to the sampling interval duration) of the vector of the predator's linear velocity. The vector is aligned with the newly calculated orientation, and its magnitude is equal to the average of the velocities of the wheels.

### 3 Evolving the Behavior of Predator Agents

MAS, as complex systems, feature a significant semantic gap between the hierarchically lower-level properties of the agents, and the (emergent) higher-level properties of the system as a whole. Thus, we could not analytically infer the optimal velocity values of the wheels of the agents from the desired behavior of the team of predator agents. Therefore, we applied the GA – a nature-inspired heuristic approach to gradually evolve good values of the parameters, similar to the evolution of species in nature. GA have proven to be efficient in finding the optimal solution(s) to combinatorial optimization problems featuring large search spaces [9, 10]. Thus, consonant with the concept of evolutionary robotics [11], we adopted the GA [12] to evolve good values of the eight velocities of the wheels of the predators that resulted in an efficient behavior – presumably involving exploring the environment, surrounding-, and capturing the prey – of the team of predators. The algorithmic steps of the GA are shown in Fig. 2, and its main attributes are elaborated below. The main parameters of the GA are summarized in Table 2.



**Fig. 2.** Main steps of GA

**Table 2.** Parameters of GA

Parameter	Value
Population size	400 chromosomes
Selection	Binary tournament, ratio: 10%, and elitism, ratio: 1%
Crossover	Both single- and two-point
Mutation	Random single-point (with even distribution), ratio: 5%
Fitness cases	10 initial situations
Duration of the trial	120 s per situation
Value of OF	Sum of OF values of each situation: Successful situation: time needed to capture the prey Unsuccessful situation: 10,000 + shortest distance between the prey and any predator
Termination criteria	(OF < 600) or (#Generations > 200) or (OF stagnation for 32 generations)

### 3.1 Genetic Representation

The decision-making of the predator agents is encoded genetically as a “chromosome”. The latter consist of an array of eight integer values (“alleles”) of the evolved wheel velocities of the agents, as shown in Eq. (1). These values are within the range  $[-100\% \dots +100\%]$ , and are discretized into 40 values, with an equal interval of 5% between them. This number of discrete values provides an acceptable trade-off between the resolution of the evolved velocities and the size of the search space ( $40^8$ ) of the GA. The size of the population is 400 and the breeding strategy is homogeneous – each chromosome is evaluated after being cloned to all eight predator agents.

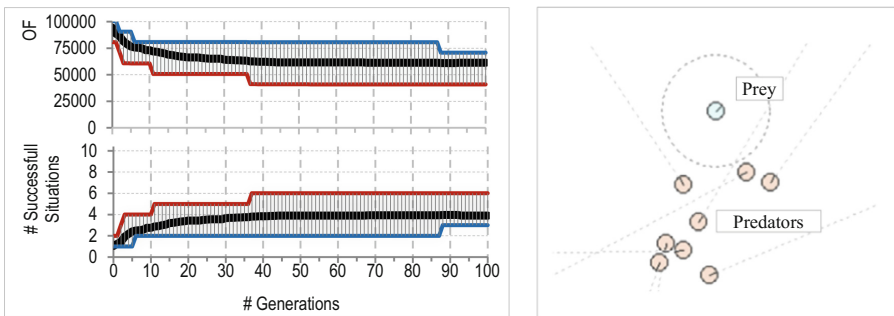
### 3.2 Genetic Operations

We employed a binary tournament selection. It is computationally efficient, and has been proven to provide a good trade-off between diversity of the population and the fitness convergence rate [10]. Further, we implemented – with equal probability – both one- and two-point crossovers. The two-point crossover results in an exchange of the values of both velocities (of the left- and right wheels, respectively) associated with a given environmental state (e.g., both  $V_{oIL}$  and  $V_{oIR}$ ). This reflects our assumption that the velocities of both wheels determine the moving behavior of the agents (for a given environmental state); therefore, they should be treated as a whole – as an evolutionary building block. The one-point crossover is applied to develop such building blocks (*exploration* of the search space), while the two-point crossover is intended to preserve them (*exploitation*).

### 3.3 Fitness Evaluation

To evolve predator behaviors that are general to several initial situations, we evaluated the objective (fitness) function (OF) of each of the evolved chromosomes on 10 different initial situations. In each of these situations, the prey is located in the center of

the world. The predators are scattered in a small cloud situated south of the prey (Fig. 3). The OF is the sum of the values, scored in each of the 10 initial situations. For a successful situation, the OF is the time needed to capture the prey (selection favoring the lowest values). For an unsuccessful situation, the OF is calculated as the sum of (i) the closest distance, registered during the trial, between the prey and any predator, and (ii) a penalty of 10,000. The former component provides evolution with a cue about the comparative quality of the different unsuccessful behaviors. We verified empirically that this heuristic quantifies the “near-misses” well, and correlates with the chances of the predators – pending small evolutionary tweaks to their genome – to successfully capture the prey in the future. The second component penalizes heavily the lack of success of the predators in any given initial situation.



**Fig. 3.** Convergence of the values of best objective function (top left) and the number of successful situations (bottom left) of 32 independent runs of GA. The bold curves correspond to the mean, while the envelope shows the minimum and maximum values in each generation. A snapshot of a sample initial situation is shown on the right.

Our PPPP is an instance of a minimization problem, as a lower OF value corresponds to a better performing team of predator agents. The evolution terminates on OF values lower than 600, which implies a successful capture of the prey in all 10 initial situations in an average time shorter than 60 s (i.e., half of the trial duration).

## 4 Experimental Results

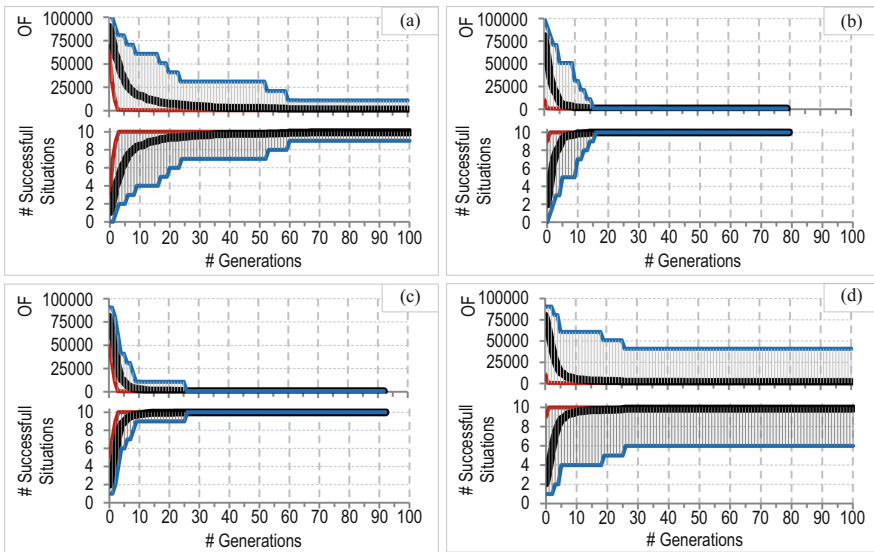
### 4.1 Evolving the Team of Straightforward Predator Agents

The experimental results of 32 independent runs of the GA evolving the predator behaviors are illustrated in Fig. 3. As Fig. 3 (top left) illustrates, the mean value of the OF slowly converges to approximately 60,000, indicating that, on average, only 4 (of 10) initial situations could be successfully resolved (Fig. 3, bottom left). The best result, achieved by the team of predators, is only 6 successful situations. These results suggest that the PPPP is, in general, intractable.

## 4.2 Enhancing the Morphology of Predators

To improve the generality of the evolved predator behaviors, we focus on modifying their morphological features. The last of the features listed in Table 1 – the orientation of the sensors – implies a straightforward implementation of the agents. This, indeed, is the common configuration of predators (e.g., [4]). We are interested in whether an a priori fixed *asymmetry* – an angular offset – would facilitate the evolution of more general behaviors of the team of predators. We speculate that a sensory offset would allow the predators to realize an equiangular (proportional) pursuit of the prey, aiming at the anticipated point of contact with the moving prey, rather than the currently perceived position of the prey.

In our experimental setup, we fixed the offset of all predators to  $10^\circ$ ,  $20^\circ$ ,  $30^\circ$ , and  $40^\circ$  counterclockwise and conducted 32 evolutionary runs of the GA for each of these 4 configurations. The results are shown in Figs. 4a, b, c, and d, respectively, and summarized in Table 3. As Fig. 4a and Table 3 illustrate, offsetting the sensors by only  $10^\circ$  significantly improves the generality of the evolved predator behaviors. They can resolve all 10 situations in 30 (93.75%) of the 32 evolutionary runs. The probability of success – the statistical estimation of the efficiency of evolution, defined for the PPPP as the probability to resolve all 10 initial situations, reaches 90% by generation #60 (Table 3). The terminal value of the OF in the worst evolutionary run is 10,987, corresponding to only one unresolved initial situation.



**Fig. 4.** Convergence of the values of best objective function (top) and the number of successful situations (bottom) of 32 runs of GA evolving predators with sensor offset of (a)  $10^\circ$ , (b)  $20^\circ$ , (c)  $30^\circ$ , and (d)  $40^\circ$ , respectively. The bold curves correspond to the mean, while the envelope illustrates the minimum and maximum values in each generation.

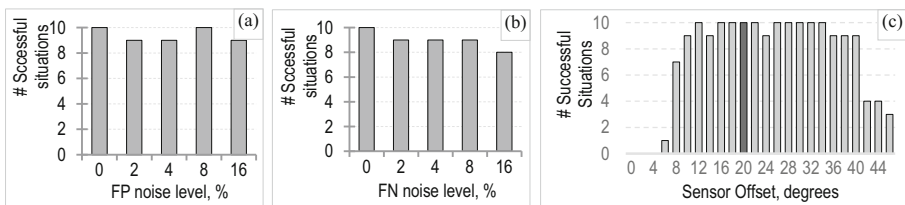
**Table 3.** Efficiency of evolution of the team of predator agents

Offset	Terminal value of objective function				Successful runs		# Generations needed to reach probability of success 90%
	Best	Worst	Mean	Standard deviation	Number	In % of 32 runs	
No offset	40,928	70,729	61,064	8,516	0	0	NA
10°	504	10,987	1,310	2,531	30	93.75	60
20°	<b>468</b>	818	588	57.2	<b>32</b>	<b>100</b>	<b>9</b>
30°	495	<b>713</b>	<b>574</b>	<b>38.5</b>	<b>32</b>	<b>100</b>	12
40°	475	40,903	1,840	7,128	31	96.875	15

More efficient evolution and more general behaviors were obtained for the sensory offsets of 20° and 30°. As Fig. 4b and Table 3 depict for 20°, the predators successfully resolved all 10 initial situations in all 32 evolutionary runs. The probability of success reaches 90% relatively quickly – by generation #9 (Table 3). Both the efficiency of evolution and the generality of the predator behaviors are similar for agents with a sensory offset of 30°, while these two characteristics deteriorate with the further increase of the offset to 40° (Figs. 4c, d, and Table 3).

### 4.3 Robustness of the Evolved Behavior to Noise

We examined the effect of a random perceptual noise on all evolved behaviors of the most general predators – those with sensory offsets of 20° and 30°. We introduced two types of noise – a false positive (FP) and false negative (FN), respectively. The FP results in either of the two bits of perception information to be occasionally (with a given probability) read as “1” regardless of whether an entity is detected by the predators. The FN results in readings of “0” even if an entity is the line of sign. The best results with the increase in the amount of noise from 0 to 16% (Fig. 5a and b) were achieved by a predator with a sensor offset of 20°, as shown in Table 4. The OF value of such predators in a noiseless environment is 552 – close to the average (588) and far from the best evolved (468). Interestingly, the same behavior, being evolved for the sensor offset of 20°, exhibits an impressive robustness to errors in the angular positioning of the sensor, as well. As shown in Fig. 5c, the predators can resolve 9 (of 10) initial situations when the sensor offset of all the agents is set to any value between 10° and 40°.



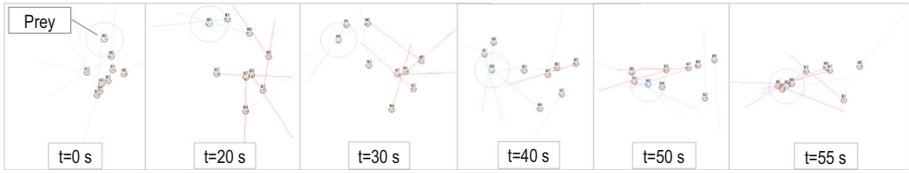
**Fig. 5.** Robustness of a sample best evolved behavior of predators with sensor offset of 20° to random false positive (FP) noise (a), false negative (FN) noise (b), and to error in angular positioning of the sensor (c).

**Table 4.** Evolved velocities of wheels of predators that result in a behavior that is most robust to noise. The sensor offset is  $20^\circ$ .

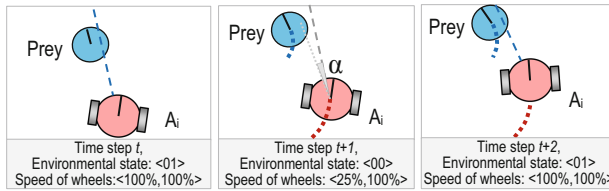
$V_{00L}$	$V_{00R}$	$V_{01L}$	$V_{01R}$	$V_{10L}$	$V_{10R}$	$V_{11L}$	$V_{11R}$
25%	100%	100%	100%	-25%	-20%	100%	100%

The team of predators exhibits three *emergent behaviors*, as illustrated in Fig. 6 (a movie is available at <http://isd-si.doshisha.ac.jp/itanev/SA/>): (i) exploring the environment by dispersing themselves into a wide area in the world ( $t = 0$  and  $t = 20$  s), (ii) shepherding ( $t = 30$  s and  $t = 40$  s), and (iii) capturing the prey ( $t = 50$  s and  $t = 55$  s), respectively. The agents commence the trial ( $t = 0$ ) with no entity in sight. Controlled by  $\langle V_{00L} = 25\%, V_{00R} = 100\% \rangle$  (Table 4) they turn to the left until either a predator (most likely) or a prey is detected. Detecting a predator activates the setup of the wheels  $\langle V_{10L} = -25\%, V_{10R} = -20\% \rangle$ , resulting in both turning slowly and moving (dispersing) away from the perceived predator. Such a dispersion widens the area of the cloud of predators and enhances their ability *to explore* the environment and to detect the prey ( $t = 0$  s and  $t = 20$  s). When the predators detect the prey, they activate the setup  $\langle V_{01L} = 100\%, V_{01R} = 100\% \rangle$ , resulting in a chase of the prey in the forward direction with maximum speed (Fig. 6,  $t = 20$  s and  $t = 30$  s). As a result of the optical parallax, during the chase, the prey might become temporarily invisible, as shown in Fig. 7 (left) and Fig. 7 (middle). When this occurs, the predator activates the setup  $\langle V_{00L} = 25\%, V_{00R} = 100\% \rangle$ , which yields a counterclockwise rotation towards the invisible prey. The predator exhibits an *embodied cognition* that the parallax is a result, in part, of its own forward motion; therefore, the new location of the prey is – due to the counterclockwise offset of the sensor, – most likely on the left of its own orientation. Therefore, the *virtue* of the sensor offset is in the more deterministic direction of the prey disappearance, which facilitates a faster rediscovery of the latter by the predator (Fig. 7 (middle) and Fig. 7 (right)). The predator could quickly rediscover the prey by turning slightly (and quickly) by only a few degrees to the left ( $\alpha$ ). Conversely, in an eventual straightforward implementation of the sensor, the predator would need to turn  $\alpha$  degrees if the direction of turning by chance coincides with the new location of the disappeared prey, or  $(360-\alpha)$  degrees otherwise. Because, from the predator viewpoint, the moving of the disappearing prey is non-deterministic, on average, the predator would have to turn  $180^\circ$  – i.e., much wider (and slower) than turning only a few degrees ( $\alpha$ ), as with an offset sensor. Moreover, such a chase, due to the sensor offset, yields a counterclockwise, circular trajectory of both the chasing predator(s) and the prey (Fig. 7 (right)), thereby resulting in *shepherding (driving)* the prey back into the (already widely dispersed) other predators. Surrounded from all sides of the world by both current and newly encountered chasing predators, the prey is finally being captured (Fig. 6,  $t = 40$  s,  $t = 50$  s and  $t = 55$  s).

In hindsight, we could also argue that the initial dispersion illustrates the emergent strategy of the predators, i.e., for a capture, only three of them (the “critical mass”) would be sufficient. By moving away from each other, most of the predators move further away from the prey as well (Fig. 6,  $t = 0$  and  $t = 20$  s), thereby compromising their chances to capture the prey. However, such an altruistic behavior results in a faster



**Fig. 6.** Phases of a sample best evolved behavior of the predators with sensor offset of  $20^\circ$ .



**Fig. 7.** Reliable tracking of the prey by chasing predator  $A_i$ .

discovery – and a faster capture of the prey by some (e.g., just three) predators, presumably, for the benefit of the whole team.

## 5 Discussion

As the experimental results indicate, the proposed asymmetric morphology implemented as an angular offset of the sensors of the predators facilitates both (i) a more effective behavior of predators, and (ii) more efficient evolution of such behavior. The offset contributes to a counterclockwise, circular trajectory of *shepherding (driving)* the prey back into the pack of the predators. The offset results in more deterministic, and therefore – faster redetection and chase of the prey during shepherding. The offset of  $20^\circ$  and  $30^\circ$  provides an optimal speed of such a chase. Lower values of the offset imply that the last detected position of the disappeared prey would be closer to the longitudinal axis of the predator, and therefore, a less deterministic – either to the left or to the right of the longitudinal axis – actual location of the prey. It is therefore less certain if the prey disappeared to the left or to the right of the longitudinal axis. This, in turn, would result in a slower rediscovery of the prey. Alternatively, higher values of the offset would imply a higher value of the tangential- and lower value of the radial (i.e., towards the prey) component of the vector of the speed of chasing predator that the chasing predator is moving tangential to the prey rather than toward the prey., and consequently, This will result in a slower overall chasing speed of the latter. In addition, this would result in a too a circular trajectory of the chased prey. It would be less likely for such a trajectory to head towards the pack of the other (dispersed) predators.

Despite the simplicity of the predator agents, the emergent shepherding behavior of the evolved team of these agents is very similar to the cooperative hunting strategy of the



bottlenose dolphins (*Tursiops truncatus*) in nature: one individual assumes the role of a ‘driver’, herding the fish in a circle towards the remaining ‘barrier’ of dolphins [13].

## 6 Conclusion

We examined a PPPP featuring very simple, non-computing predator agents, equipped with a single line-of-sight sensor and a simple control of velocities of their two wheels. We applied a GA to evolve the successful behavior of the team of predator agents. To enhance the generality of the evolved behavior, we proposed an asymmetric morphology of the predators. Offsetting their sensors angularly to 20° and 30° yielded the most efficient and consistent evolution of successful behaviors of agents.

In our future work we are planning to evolve the optimal value of the offset as an algebraic function of the most relevant parameters (e.g., speed, range of sensors, diameter, etc.) of both predator- and prey agents. Also, we will consider a more sophisticated behavior of the prey.

## References

1. Diamandis, P.H.: Nanorobots: Where We Are Today and Why Their Future Has Amazing Potential, Singularity Hub, 16 May 2016. <https://singularityhub.com/2016/05/16/nanorobots-where-we-are-today-and-why-their-future-has-amazing-potential/>
2. Gauci, M., Chen, J., Li, W., Dodd, T.J., Groß, R.: Self-organized aggregation without computation. *Int. J. Robot. Res.* **33**(8), 1145–1161 (2014)
3. Gauci, M., Chen, J., Li, W., Dodd, T.J., Groß, R.: Clustering objects with robots that do not compute. In: *Proceedings of the 2014 International Conference on Autonomous Agents and Multiagent Systems, IFAAMAS*, pp. 421–428 (2014)
4. Ozdemir, A., Gauci, M., Groß, R.: Shepherding with robots that do not compute. In: *Proceedings of the 14th European Conference on Artificial Life (ECAL)*, 8 pages (2017)
5. Benda, M., Jagannathan, B., Dodhiawala, R.: On optimal cooperation of knowledge sources. Technical report BCS-G2010-28, Boeing AI Center, Boeing Computer Services, Bellevue, WA (1986)
6. Haynes, T., Sen, S.: Evolving behavioral strategies in predators and prey. In: Weiß, G., Sen, S. (eds.) *IJCAI 1995. LNCS*, vol. 1042, pp. 113–126. Springer, Heidelberg (1996). [https://doi.org/10.1007/3-540-60923-7\\_22](https://doi.org/10.1007/3-540-60923-7_22)
7. Luke, S., Spector, L.: Evolving teamwork and coordination with genetic programming. In: *Proceedings of the 1st Annual Conference on Genetic Programming*, pp. 150–156 (1996)
8. Tanev, I., Brzozowski, M., Shimohara, K.: Evolution, generality and robustness of emerged surrounding behavior in continuous predators-prey pursuit problem. *Genet. Program. Evol. Mach.* **6**(3), 301–318 (2005)
9. Holland, J.: *Adaptation in natural and artificial systems*, The University of Michigan, 212 pages (1975)
10. Goldberg, D.: *Genetic Algorithms in Search, Optimization and Machine Learning*, 412 pages. Addison-Wesley, Reading (1989)
11. Nolfi, S., Floreano, D.: *Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-organizing Machines*. MIT Press, Cambridge (2000)

12. Tanev, I., Shimohara, K.: XML-based genetic programming framework: design philosophy, implementation, and applications. *Artif. Life Robot.* **15**(4), 376–380 (2010)
13. Gazda, S., Connor, R., Edgar, R., Cox, F.: A division of Labour with Role Specialization in Group-hunting Bottlenose Dolphins (*Tursiops truncatus*) off Cedar Key, Florida. *Proc. Biol. Sci.* **272**(1559), 135–140 (2005)

# Posters



# Semantic Graph Based Automatic Summarization of Multiple Related Work Sections of Scientific Articles

Nouf Ibrahim Altmami<sup>(✉)</sup> and Mohamed El Bachir Menai

Department of Computer Science, College of Computer and Information Sciences, King Saud University, P.O. Box 51178, Riyadh 11543, Saudi Arabia  
naltmami@su.edu.sa, menai@ksu.edu.sa

**Abstract.** The summarization of scientific articles and particularly their related work sections would support the researchers in their investigation by allowing them to summarize a large number of articles. Scientific articles differ from generic text due to their specific structure and inclusion of citation sentences. Related work sections of scientific articles generally describe the most important facts of prior related work. Automatically summarizing these sections would support research development by speeding up the research process and consequently enhancing research quality. However, these sections may overlap syntactically and semantically. This research proposes to explore the automatic summarization of multiple related work sections. More specifically, the research goals of this work are to reduce the redundancy of citation sentences and enhance the readability of the generated summary by investigating a semantic graph-based approach and cross-document structure theory. These approaches have proven successful in the field of abstractive document summarization.

**Keywords:** Automatic text summarization · Scientific article · Related work Multi-document · Semantic graph · Cross-document structure theory

## 1 Introduction

Automatic summarization of scientific articles would be useful for researchers to quickly study and evaluate the state-of-the-art. However, the most recent articles refer to the same related work and hence a large number of articles is cited in every related work section. Automatically summarizing multiple related work sections would be useful and helpful by reducing the time needed to review a large number of related work sections.

Related work sections have specific characteristics that make them unique. First, they include citation sentences. Second, these sections are short in length, which makes the problem more challenging. Hence, extractive techniques would generate a summary suffering from a lack of readability and coherence. Finally, the overlap between multiple related work sections is an important issue.

Limited research studies addressed related work summarization. Most of these studies have generated a related work section for a target paper by summarizing a set of articles [1–3]. Only one study has tackled the problem of summarization of the related

work section of a single article [4]. Automatically summarizing multiple related work sections of a set of articles on a particular topic would aid the generation of a more comprehensive summary. To the best of our knowledge, no previous research has tackled this particular problem. This research work proposes to address this problem by investigating a semantic graph-based approach and cross-document structure theory (CST).

The remaining of this paper is as follows. Section 2 examines various prior studies in the field of scientific article summarization. The proposed approach is presented in Sect. 3. Finally, Sect. 4 concluded this paper.

## 2 Related Work

Among the interesting concerns of scientific articles summarization is the generation of research article abstract. Lloret et al. [5] suggested two approaches for this task. The first one is an extractive summarization approach. The second one is based on both extractive and abstractive techniques. Saggion and Lapalme [6] proposed an approach for generating an indicative and informative abstract called Selective Analysis. This type of summarization is not an accurate scientific summary since it stated the contributions in a less focused fashion and general form.

The above-mentioned problems motivated the generation of citation based summaries. Abu-Jbara and Radev [7] tackled some issues related to the readability and coherency of this type of summaries. C-LexRank, a graph based summarizer, is also proposed by Qazvinian and Radev [8] to summarize single scientific article. Chen and Zhuge [9] made additional progress by exploiting a set of terms that co-occur in a set of citations according to the common fact phenomenon.

Related work summarization is a specific instance of scientific article summarization. Hoang and Kan [1] proposed a heuristic system called ReWoS for the automatic generation of a related work section based on a topic hierarchy tree. Chen and Zhuge [2] used citation sentences and performed a comparison of the content of the target article and the content of the citation sentences while Hu and Wan [3] considered this problem as an optimization problem. Widyantoro and Amin [4] proposed a different approach for summarizing a related work section in scientific articles. Their proposed approach consists of two main stages. First, they extracted citation sentences. Then, they categorized these citation sentences into three different classes (i.e., problem, method and conclusion).

## 3 The Proposed Approach

Our goal is to automatically summarize multiple related work sections while maximizing the readability of the generated summary and minimizing the redundancy of citation sentences. We propose to investigate a hybrid method based on both a semantic graph-based approach and CST. Moreover, different abstractive techniques will be investigated to improve the readability, including multi-sentence compression [10] and language generation [11].

Positive feedback has been obtained when using graph-based approaches in the field of (MDS) [11–13]. However, it suffers from some essential limitations. First, it depends on similarity measures without taking into consideration the semantic relationships among sentences. A second limitation is the lack of diversity of the generated summary due to the ranking algorithms. Thus, we plan to investigate the use of a semantic graph-based approach to cope with the redundancy of citation sentences. Moreover, we will investigate ranking algorithms to take into consideration the semantic similarity. In the other hand, CST has been used to analyze multi-documents to discover semantic relations among their content [14–16]. Based on the particular content of the related work section, CST could help to further reduce redundancy between citation sentences. Different content selection methods will be investigated, including a redundancy operator, general operator [17] and the method proposed by Otterbacher et al. [18]. Following is a small instance of the problem to illustrate the proposed approach.

<i>A part of the related work section of paper [1]:</i>	<i>A part of the related work section of paper [2]:</i>
“Further, Mei and Zhai (2008) and Qazvinian and Radev (2008) utilized citation information in creating summaries for a single scientific article in computational linguistics domain.”	“Based on the finding, Qazvinian and Radev employ the citations to create the summary for the scientific paper [3, 5].”

Reference Qazvinian and Radev (2008) in paper [1] is cited as [5] in paper [2] and the two text spans have the same information content. Thus, the result of the proposed approach should be similar to:

*Mei and Zhai [1] utilized citation information in creating summaries for a single scientific article in computational linguistics domain. Qazvinian and Radev [4, 6] employed the citations to create the summary for the scientific paper.*

The main steps of the proposed approach are:

- A preprocessing step to identify the same reference in each related work section and to reduce them to one format for example IEEE format.
- A Graph: to represent the relations between the references and their citation sentences.
- CST to analyze the different citation sentences of the same reference in order to discover semantic relations among their content.
- Content selection: the final step is summary extraction by transforming the graph into smaller one while preserving its properties.

The main objectives of this research are summarized in the following points:

- Summarizing multiple related work sections of scientific articles while enhancing the readability of the generated summary and minimizing the redundancy of citation sentences.
- Proposing a hybrid method based on both semantic graph based approach and CST.

- Finding the semantic relationships between different contents in order to not influence the discourse meaning.
- Examining different abstractive techniques to hopefully improve the readability.
- Building our own dataset for the summarization of related work sections. According to our first investigation, we have not found a benchmark dataset available online for the summarization of related work sections. However, we have been able to obtain the data set used in [4] to evaluate summaries of related work sections. This dataset is composed of a collection of 20 article sets, and each set contains different reference articles that need to be summarized to generate a related work section.

## 4 Conclusion

In this paper, we took the first step towards summarizing multiple related work sections of scientific articles. We outlined a hybrid approach which consists of combining semantic graphs and CST. It aims at minimizing the redundancy of citation sentences and improving the readability of the generated summary by investigating different abstractive and content selection techniques.

## References

1. Hoang, C.D.V., Kan, M.-Y.: Towards automated related work summarization. In: Proceedings of the 23rd International Conference on Computational Linguistics: Posters 2010, pp. 427–435 (2010)
2. Chen, J., Zhuge, H.: Summarization of related work through citations. In: 12th International Conference on Semantics, Knowledge and Grids (SKG) 2016, pp. 54–61. IEEE (2016)
3. Hu, Y., Wan, X.: Automatic generation of related work sections in scientific papers: an optimization approach. In: EMNLP 2014, pp. 1624–1633 (2014)
4. Widyantoro, D.H., Amin, I.: Citation sentence identification and classification for related work summarization. In: International Conference on Advanced Computer Science and Information Systems (ICACSIS) 2014, pp. 291–296 (2014)
5. Lloret, E., Romá-Ferri, M.T., Palomar, M.: COMPENDIUM: a text summarization system for generating abstracts of research papers. In: Muñoz, R., Montoyo, A., Métais, E. (eds.) NLDB 2011. LNCS, vol. 6716, pp. 3–14. Springer, Heidelberg (2011). [https://doi.org/10.1007/978-3-642-22327-3\\_2](https://doi.org/10.1007/978-3-642-22327-3_2)
6. Saggion, H., Lapalme, G.: Selective analysis for automatic abstracting: evaluating indicativeness and acceptability. In: Content-Based Multimedia Information Access-Volume 1, pp. 747–764 (2000)
7. Abu-Jbara, A., Radev, D.: Coherent citation-based summarization of scientific papers. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1, pp. 500–509 (2011)
8. Qazvinian, V., Radev, D.R.: Scientific paper summarization using citation summary networks. In: Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1, pp. 689–696 (2008)
9. Chen, J., Zhuge, H.: Summarization of scientific documents by detecting common facts in citations. *Future Gener. Comput. Syst.* **32**, 246–252 (2014)

10. Banerjee, S., Mitra, P., Sugiyama, K.: Multi-document abstractive summarization using ILP based multi-sentence compression. arXiv preprint [arXiv:1609.07034](https://arxiv.org/abs/1609.07034) (2016)
11. Atif, K., Salim, N., Kumar, Y.: Genetic semantic graph approach for multi-document abstractive summarization. In: Fifth International Conference on Digital Information Processing and Communications (ICDIPC) 2015. IEEE (2015)
12. Erkan, G., Radev, D.R.: Lexrank: graph-based lexical centrality as salience in text summarization. *J. Artif. Intell. Res.* **22**, 457–479 (2004)
13. Ganesan, K., Zhai, C., Han, J.: Opinosis: a graph-based approach to abstractive summarization of highly redundant opinions. In: Proceedings of the 23rd International Conference on Computational Linguistics 2010, pp. 340–348 (2010)
14. Zhang, Z., Otterbacher, J., Radev, D.: Learning cross-document structural relationships using boosting. In: Proceedings of the Twelfth International Conference on Information and Knowledge Management 2003, pp. 124–130 (2003)
15. del Rosario Castro Jorge, M.L., Pardo, T.A.S.: Experiments with CST-based multidocument summarization. In: Proceedings of the 2010 Workshop on Graph-based Methods for Natural Language Processing 2010, pp. 74–82 (2010)
16. Zhang, Z., Blair-Goldensohn, S., Radev, D.R.: Towards CST-enhanced summarization. In: AAAI/IAAI (2002)
17. Radev, D.R.: A common theory of information fusion from multiple text sources step one: cross-document structure. In: Proceedings of the 1st SIGdial Workshop on Discourse and Dialogue-Volume 10, pp. 74–83 (2000)
18. Otterbacher, J.C., Radev, D.R., Luo, A.: Revisions that improve cohesion in multi-document summaries: a preliminary study. In: Proceedings of the ACL-02 Workshop on Automatic Summarization-Volume 4, pp. 27–36 (2002)





# Collective Lévy Walk for Efficient Exploration in Unknown Environments

Yara Khaluf<sup>(✉)</sup>, Stef Van Havermaet, and Pieter Simoens

IDLab, Ghent University - imec, Ghent, Belgium  
yara.khaluf@ugent.be  
<http://www.yarakhaluf.net>

**Abstract.** One of the key tasks of autonomous mobile robots is to explore the unknown environment under limited energy and deadline conditions. In this paper, we focus on one of the most efficient random walks found in the natural and biological system, i.e., Lévy walk. We show how Lévy properties disappear in larger robot swarm sizes because of spatial interferences and propose a novel behavioral algorithm to preserve Lévy properties at the collective level. Our initial findings hold potential to accelerate target search processes in large unknown environments by parallelizing Lévy exploration using a group of robots.

**Keywords:** Multi-robot systems · Swarm robotics · Random walks  
Lévy walk

## 1 Introduction

Exploring unknown environments to spot targets is one of the most fundamental problems in the context of mobile robots used for search and rescue, environment mapping or agricultural applications [3]. An efficient exploring strategy that provides a maximized area coverage in a minimized time interval is the main design goal. Since there are no clues for the robot on where to explore, it must execute a random walk. Amongst the random walks that were revealed in natural collective systems, the Lévy walk (LW) is one of the most efficient patterns [4]. For sparse targets, the LW maximizes the search efficiency, i.e. the number of targets found in a specific time interval. With a LW, the step orientation is sampled from a uniform distribution, while the step lengths are sampled from a heavy-tailed (power-law) distribution:

$$p(l) \sim l^{-(\alpha+1)} \quad (1)$$

where  $l$  is the step length and the distribution exponent  $0 < \alpha < 2$ .

LWs allow the agent to execute several short steps before executing a long one. In case of sparse targets, the LW provides an optimal search behavior because the long steps sampled from the power-law distribution maximize the number of visited sites. Taking only short steps, the robot will frequently pass

sites that it has already visited [6]. In the absence of any knowledge about the target distribution, the optimal value of the exponent is given by  $\alpha = 1$  [5].

In existing works on efficient exploration by robot swarms, the trajectory followed by each robot is sampled from a Lévy distribution [1]. Despite the promising results in terms of exploration and area coverage [2], the main drawback is that the Lévy distribution and its properties are lost when the swarm size increases. As the swarm size increase, the probability that independently generated Lévy walks is intersecting also increases and already with moderate swarm sizes the collective trajectory no longer follows a Lévy distribution. Consequently, the current strategies to implement Lévy walks in robot swarms are not scalable. To the best of our knowledge, no previous works addressed the question of how to generate a *collective* Lévy walk that emerges from the different robots' trajectories and which preserves Lévy properties.

In this paper, we address this key challenge in an unbounded space by introducing an efficient algorithm which controls robot swarms to generate a Collective Lévy Walk (CLW) for large swarm sizes. We introduce our CLW algorithm in Sect. 2 and compare its performance for different swarm sizes with the baseline of independently generated Lévy walks.

## 2 Collective Lévy Walk (CLW) Algorithm

As a first step, we have launched a set of exploration experiments using swarms of different sizes to investigate the presence of Lévy properties in the trajectory obtained by summing up the individual LWs generated by the robots independently. Our results (see Sect. 3) reveal that the obtained trajectory follows a Lévy distribution for small swarm sizes only. The main reason is that long steps are interrupted (aborted) by intersecting robots. Intuitively, increasing the number of robots in the swarm leads to more spatial interferences and consequently to a decrease in the probability of obtaining the heavy tail (Eq. 1) for the step length in the combined trajectory.

Our CLW algorithm prioritizes longer steps in robot trajectories by exploiting information exchanged between robots. The CLW algorithm is described by the deterministic finite automaton that is shown in Fig. 1. Each robot starts in the “Walk state” to explore the unknown environment. Robots move with a fixed linear speed and sample the duration of their next step  $T_L$  from a Lévy distribution. When the interval  $T_L$  is over, the robot switches to the “Rotate state” and rotates at a constant angular velocity during  $T_U$ , with  $T_U$  sampled from a uniform distribution. Whenever the walking robot detects an obstacle using its proximity sensors, it leaves the “Walk state” immediately and starts executing a collision avoidance behavior. In this state, the robot rotates with an angle that is determined based on the distance of the obstacle. When all obstacles have been avoided, the robot transitions to the “Walk state” again, and samples a new time interval  $T_L$  to proceed its next step in the Lévy walk.

The key part of the CLW algorithm resides in exploiting the communication between the walking robots to generate a collective Lévy walk. Robot  $i$  broadcasts its sampled time interval  $T_L(c)$  to its local neighbors—i.e., these are the

robots within its communication range and within line-of-sight. The step is categorized as “short” or “long” based on a predefined threshold  $T_{Threshold}$ . In case of a short step of robot  $i$  and a long step of its neighbor  $j$ , robot  $i$  starts moving away from robot  $j$ , using the principles of potential field [7]. The repulsive force  $F(j)$  driving robot  $i$  away from neighbor  $j$ , which is executing a long step, is computed from the position of robot  $j$ . This position is obtained using the range-and-bearing sensors. Finally, the force  $F$  is computed for all neighbors of robot  $i$ , which are executing long steps, and these forces are averaged to generate the final repulsive force applied to robot  $i$ . In cases where all neighbors are executing short steps, robot resume their walking behavior as planned.

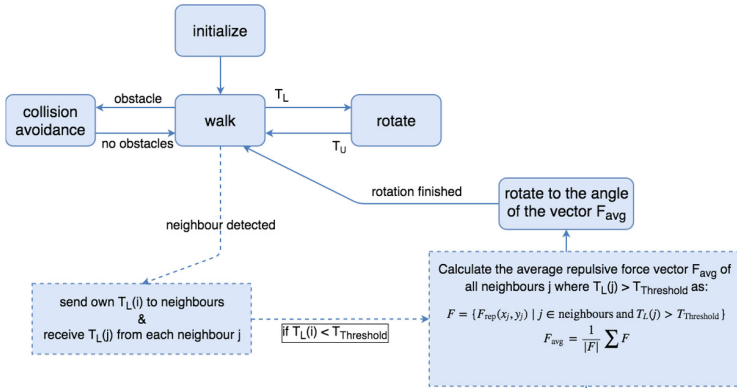


Fig. 1. Finite state diagram of the CLW algorithm.

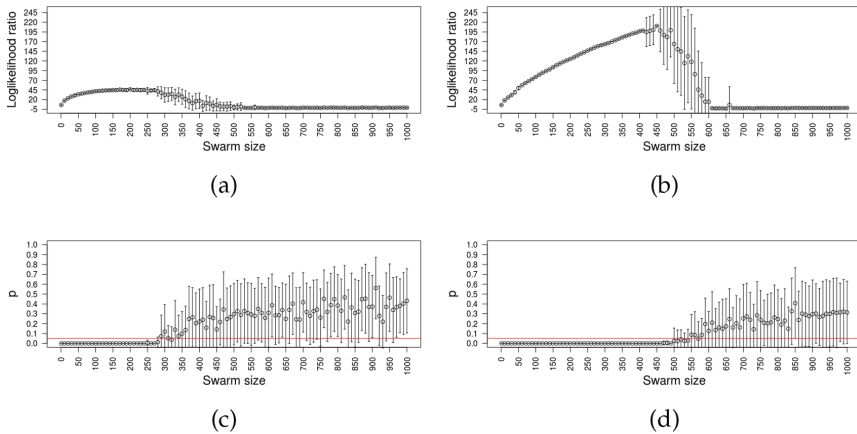
### 3 Results and Conclusion

We have performed a set of physics-based simulations using the ARGoS simulator<sup>1</sup>. An arena of  $20 \times 20 \text{ m}^2$  is implemented as an unbounded space: the robot that reaches a side of the arena will re-enter from the opposite side without interrupting its currently executed step. Simulation results are averaged over 30 runs, with each run lasting 5000 time steps. The exponent  $\alpha$  is set to 1 (see Eq.(1)), the communication range of the robot is set to 1.35 m and the linear speed to 5 m/s. The step threshold is set to  $T_{Threshold} = 9 \times 0.17 = 1.53 \text{ m}$  (0.17 the diameter of the simulated robot). We use the log-likelihood test to determine the best fitting of the obtained distribution of the collective trajectory. We rely on two outputs of the test to judge the fitting: the p-value of the test, and the log-likelihood ratio. If the log-likelihood ratio is  $> 0$ , the best fitting is a heavy-tailed Lévy distribution when the p-value is  $< 0.05$ ; when the p-value is  $> 0.05$  the best fitting is an exponential distribution. If the log-likelihood ratio  $< 0$ ,

<sup>1</sup> The ARGoS simulator allows to simulate large swarms of robots while taking the desired level of physical details into consideration.

neither the Lévy distribution nor the exponential distribution are best fittings and a transition between the two distributions is observed. Figure 2 depicts the results for both the combined trajectory of the independent implementation of the robots' Lévy walks and the trajectory generated when applying the CLW algorithm. We can notice a clear phase transition from power-law distribution to exponential distribution. Following the independent implementation, the swarm up to 300 preserves a Lévy walk, and between 300 and 500, a transition from Lévy distribution to the exponential distribution is observed. When applying the CLW algorithm, the Lévy walk is preserved collectively up to 500 robot, between 500 and 600 the transition from Lévy distribution to the exponential distribution is observed.

In this paper, we have proposed an efficient algorithm for robot swarms that exploited local communication among robots to generate collective Lévy walk. Our results show the ability of the CLW to generate such a collective trajectory also for larger swarm sizes.



**Fig. 2.** The log-likelihood ratio of the independent Lévy walk implementation and of the CLW algorithm in (a) and (b), respectively. The p-value of the independent Lévy walk implementation and of the CLW algorithm in (c) and (d), respectively.

## References

1. Beal, J.: Superdiffusive dispersion and mixing of swarms. *ACM Trans. Auton. Adapt. Syst.* **10**(2), 10:1–10:24 (2015)
2. Dimidov, C., Oriolo, G., Trianni, V.: Random walks in swarm robotics: an experiment with kilobots. *ANTS 2016. LNCS*, vol. 9882, pp. 185–196. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-44427-7\\_16](https://doi.org/10.1007/978-3-319-44427-7_16)
3. Khaluf, Y.: Edge detection in static and dynamic environments using robot swarms. In: 2017 IEEE 11th International Conference on Self-Adaptive and Self-Organizing Systems (SASO), pp. 81–90, September 2017

4. Khaluf, Y., Ferrante, E., Simoens, P., Huepe, C.: Scale invariance in natural and artificial collective systems : a review. *J. R. Soc. Interface* **14**, 20 (2017)
5. Plank, M., James, A.: Optimal foraging: Lévy pattern or process? *J. R. Soc. Interface* **5**(26), 1077–1086 (2008)
6. Raposo, E.P., Buldyrev, S.V., da Luz, M.G.E., Viswanathan, G.M., Stanley, H.E.: Lévy flights and random searches. *J. Phys. Math. Theor.* **42**(43), 1–12 (2009)
7. Sutantyo, D.K., Kernbach, S., Nepomnyashchikh, V.A., Levi, P.: Multi-robot searching algorithm using levy flight and artificial potential field. *CoRR* abs/1108.5624 (2011)



# A BLAST-Based Algorithm to Find Evenly Distributed Unique Subsequences

Maciej Kulawik and Robert M. Nowak<sup>(✉)</sup>

Institute of Computer Science, Warsaw University of Technology,  
Nowowiejska 15/19, 00-665 Warsaw, Poland  
[r.m.nowak@elka.pw.edu.pl](mailto:r.m.nowak@elka.pw.edu.pl)

**Abstract.** Genomics rearrangements detection involves processing of large amounts of DNA data and therefore efficiency of the used algorithms is crucial. We propose the algorithm based on evenly distributed unique subsequences. In this paper BLAST-based pattern matching is examined in terms of computation time and detection quality. The experiments were carried out both on real sequence with artificially introduced random rearrangements. The algorithm extension was implemented as part of genomecmp web application which provides graphical user interface for ease and convenience of use.

**Keywords:** Bioinformatics · Genomics rearrangements detection  
Computer program · Marker · BLAST

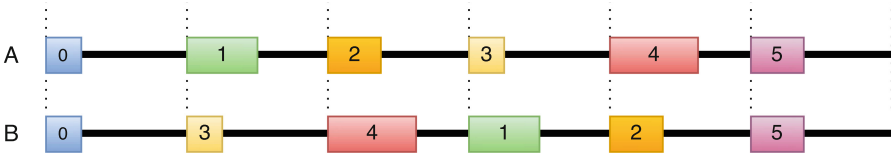
## 1 Introduction

One of topics of interest in Bioinformatics is detection of changes in DNA sequences caused by replication and repair processes. These changes are called rearrangements and their most simple types are: deletion (when part of a sequence is missing), inversion (when part of a sequence is in reverse order), transposition (when part of a sequence is moved to other place) and duplication (when part of a sequence is duplicated). The rearrangements are not so common as mutations, but could involve 10% of genome. In our previous works the algorithm based on evenly distributed unique subsequences, called markers, was proposed to enable fast rearrangement detection [4].

This algorithm, in comparison to other methods to detecting genomic structural variants [6] is very fast if the assembly of studied genomes is available. Although the assembly step is time consuming, we target our algorithm to be useful in situations when it is already done for other genome analyses and its results may be reused also for rearrangement detection. Another limitation of the algorithm we are aware of is the fact that it is not best suited for detection of short rearrangement (approx. 50 bp), but rather for longer genome changes. The shortest rearrangement length that can be detected is dependent on parameters determining number of markers to be used and the ratio of sequence length

to the number. Our algorithm has linear time complexity in terms of number of markers, what implies a trade off between time and shortest rearrangement which detection is possible. For example, for sequence of length 1 Mbp and 10000 markers, shortest rearrangement to be accurately detected ought to be longer than 100 bp, as it should consist of at least two markers. For longer sequence of length 100 Mbp and same number of markers, the shortest ought to be longer than 10 kbp.

The algorithm is depicted in Fig. 1. The algorithm consists of three phases: defining the markers using base sequence, searching for markers in compared sequence and then finding rearrangements based on markers' positions. We implemented the web application, called *genomecmp*, which is used for rearrangement searching.



**Fig. 1.** Rearrangement searching based on unique sequence positions implemented in our ‘*genomecmp*’ application. First we define unique subsequences spread uniformly called markers on base genome, then we search for such markers in compared sequence, finally we find the rearrangement based on markers' position using longest common subsequence of markers returned by Myers' algorithm [5]. The picture shows transposition of region near markers 1,2 with region near markers 3,4.

In our solution, the first and second phase of algorithm, i.e. the markers defining and markers searching are performed with help of pattern matching algorithm, we use two pattern matching algorithms: Knuth-Morris-Pratt (KMP) algorithm [2] and ChunkedXcorr [4]. KMP algorithm allowed fast rearrangement detection, but because it is an exact pattern matching algorithm it was susceptible to point mutations which may occur in a sequence. The ChunkedXcorr algorithm proved to be more immune to this kind of noise in data, but its computational complexity is too high to allow day-to-day comparisons of multiple sequences, especially for plant genomes.

In this paper we study pattern matching algorithm based on BLAST heuristics searching. Basic Local Alignment Tool (BLAST) [1] is an algorithm which approximates alignments of genomics sequences. The usage of BLAST gives both: efficiency and immunity to mutations noise. We also consider using BLAT [3], but BLAST is simpler in terms of implementation and we expect similar results.

## 2 Experiments

Each experiment was run 10 times following the procedure: (1) loading base sequence; (2) generating random rearrangements: 20 deletions, 20 insertions and

20 transpositions; (3) creating compared sequence by introducing generated rearrangements to base sequence; (4) making given number of random point mutations in compared sequence; (5) executing rearrangement detection; (6) evaluating results.

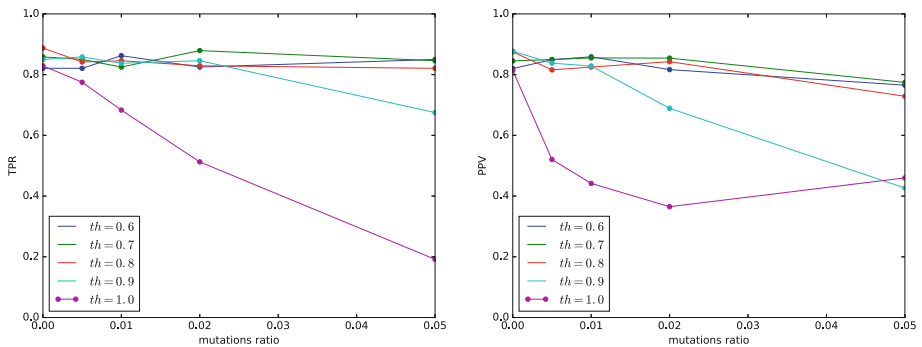
The base sequence was *Thermus thermophilus* genome from NCBI database, (NCBI ID = AP008226.1), of length 1849742 bp ( $\approx 2$  Mbp). We generate 24 rearrangement of length 40000 bp, with mutation ratio 0, 0.005, 0.01, 0.02, 0.05 respectively. We use 100 markers of minimum length 50 bp. The BLAST searching were performed for threshold 1.0, 0.9, 0.8, 0.7, 0.6 respectively, where the seed length is set to 8, score for match is 5, score for mismatch  $-4$ , maximum score difference 40.

In the evaluation step we calculate confusion matrix as well as precision (Positive Predictive Value, PPV), sensitivity (True Positive Rate, TPR) and F1 Score ( $F1score = 2 * \frac{PPV * TPR}{PPV + TPR}$ ). The execution environment was the virtual machine with 14 cores and 16 GB RAM, emulated by QEMU 2.1.2 on 64-bit Debian GNU Linux 8.5.

### 3 Results

#### 3.1 Detection Quality of BLAST-Based Solution

The experiments' results concerning BLAST-based solution detection quality for different levels of mutations and different thresholds are presented in the Fig. 2.



**Fig. 2.** BLAST-based algorithm TPR (on left) and PPV (on right) to ratio of mutations.

As expected, greater mutations ratio degrades sensitivity - less markers are correctly found in the rearranged and mutated sequence and more rearrangements are undetected. This effect is especially noticeable for the threshold equal 1.0, when almost exact matches are required.



For thresholds 0.8, 0.7 and 0.6 the algorithm copes well with increasing level of mutations noise maintaining TPR above 0.8 for mutations ratio lower or equal 0.05.

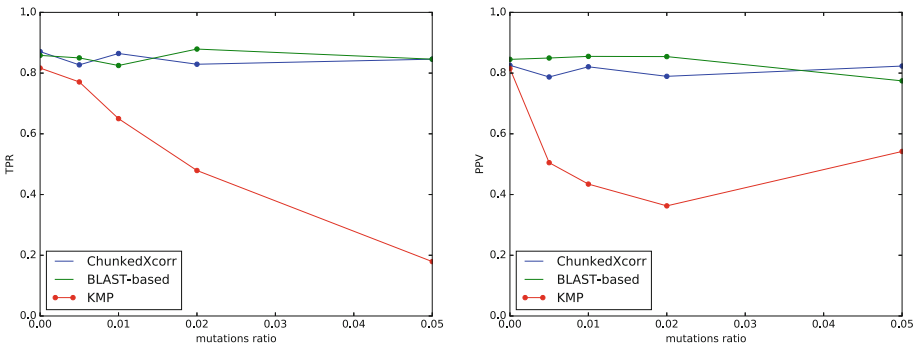
The precision falls more rapidly than sensitivity with the increase of mutations ratio. The same as for the sensitivity degradation, the reason of the fall is not finding all markers in the compared sequence because of mutations. Not found markers are recognized as deletions, so more rearrangements are incorrectly detected.

For threshold equal 1.0 small growth of PPV for higher mutations ratio is observable. It is caused by the fact that less rearrangements are detected both correctly (smaller TPR) and incorrectly and the proportion slightly changes. The algorithms maintains relatively high PPV for thresholds 0.7 and 0.8.

Using F1 score results it is possible to choose the best threshold for this range of mutations ratio, rearrangements' length and for the set of the algorithm parameters. The threshold that preserved the best quality in this experiment is equal 0.7.

### 3.2 Detection Quality Comparison

The results of detection quality comparison of BLAST-based solution and other two studied previously (KMP, ChunkedXcorr) for different levels of introduced mutations are presented in Fig. 3. The results shown for BLAST-based algorithm are for threshold set to 0.7, as that setting provided the best detection quality, shown in previous experiment. The ChunkedXcorr threshold parameter, as mentioned before, was also set to 0.7.



**Fig. 3.** Algorithms' TPR and PPV to ratio of mutations to sequence length

In terms of sensitivity, BLAST-based solution achieved similar results to the one using ChunkedXcorr. It is evidently distinguishable that KMP algorithm being an exact pattern matching is sensitive for any introduced noise.

BLAST-based solution has slightly better precision for lower mutations ratios than the one using ChunkedXcorr. KMP algorithm vulnerability to mutations is also noticeable in terms of precision.

The BLAST-based algorithm achieved similarly good F1 score results as the one using ChunkedXcorr and better than solution using KMP.

The BLAST-based algorithm had the shortest mean computation time in comparison ChunkedXcorr and KMP based algorithms.

## 4 Discussion and Conclusion

The rearrangement detection algorithm, based on unique subsequences, has been extended. The time complexity was decreased and immunity to mutations noise is achieved. The new module is a part of our web application, genomecmp, which is freely available with MIT license at <http://genomecmp.sourceforge.net>.

The proposed version of the marker definition algorithm determines uniqueness of given subsequence by searching it in the reference sequence and checking if it occurs only once. It was intended to define such heuristic that it could determine subsequence uniqueness basing only on the subsequence and possibly on some auxiliary data gathered once beforehand from the reference sequence. Both the auxiliary data gathering and its accessing during marker definition should be computationally inexpensive, so that the whole process would not require more operations than searching the reference sequence.

Promising heuristics are functions basing on information about the distribution of short subsequences in the reference sequence. The work planned is usage of function indicating that the processed subsequence is unique only when it consist of at least one short subsequence which is unique.

It could reduce the computational time significantly, because the first step, marker defining takes about 90% of time.

## References

1. Altschul, S.F., Gish, W., Miller, W., Myers, E.W., Lipman, D.J.: Basic local alignment search tool. *J. Mol. Biol.* **215**(3), 403–410 (1990). [https://doi.org/10.1016/S0022-2836\(05\)80360-2](https://doi.org/10.1016/S0022-2836(05)80360-2). <http://www.sciencedirect.com/science/article/pii/S0022283605803602>
2. Knuth, D.E., Morris, J.H., Pratt, V.R.: Fast pattern matching in strings. *SIAM J. Comput.* **6**(2), 323–350 (1977). <https://doi.org/10.1137/0206024>
3. Kent, W.J.: Blatthe blast-like alignment tool. *Genome Res.* **12**(4), 656–664 (2002)
4. Kulawik, M., Nowak, R.M.: Genomecmp: computer software to detect genomic rearrangements using markers (2017). <https://doi.org/10.1117/12.2280483>
5. Myers, E.W.: An  $O(ND)$  difference algorithm and its variations. *Algorithmica* **1**(1–4), 251–266 (1986)
6. Tattini, L., D’Aurizio, R., Magi, A.: Detection of genomic structural variants from next-generation sequencing data. *Front. Bioeng. Biotechnol.* **3**, 92 (2015)



# Implementation of Multilayer Perceptron in Graphics Processing Unit

Ventsislav Nikolov<sup>(✉)</sup>

Technical University of Varna, Varna, Bulgaria  
v.nikolov@tu-varna.bg

**Keywords:** Parallel neural network · Graphics Processing Unit  
Compute Unified Device Architecture

## 1 Introduction

Often in computer technologies methods are used based on exact calculations. For example, in searching algorithms the goal is to find exactly a given element in a given set. Searching a record in a database is performed by looking for exact value in a given field, for example the identifier of the record or the values in a group of fields.

The neural networks, in difference with the exact solutions, work with approximations [2]. This makes them convenient for solving of problems related to inexact or partial data. There are some situations for which it is difficult to find a solution, except the using of principles of the inexact solutions. Such situations often arise in tasks in which a finding of solution, even not exact one, is more important than the absolute accuracy. Inexact solutions are convenient in searching of the most similar, in some sense, pre-processed or raw data. The neural networks can generalize, that is one of the most important their characteristic [3]. They are inspired by the biological neural networks and as such they are a very simplified analogy of the natural neural networks [7].

The neural networks can be realized both hardware and software. The hardware realization often works faster because of their design to solve specific problems. However, the flexibility of the hardware realization is worse. Often the neural network architecture parameters depend on the problem that have to be solved and the software realization provides significant advantages. Despite that the software realization is performed in a general purpose machine, normally it is cheaper and significantly easier for changes, even if it is not as fast as the hardware realization. That is why here the theoretical basis will be emphasized to the software realization on general purpose Graphics Processing Unit (GPU).

## 2 GPU Versus CPU

The type of the neural network considered here is multilayer perceptron [4] that is one of the most often used neural networks in practice. There are different types and architectures of neural networks [5, 8] but most of them are well-known as models that strongly incorporate fine-grained parallelism [10] which is appropriate for the GPU,

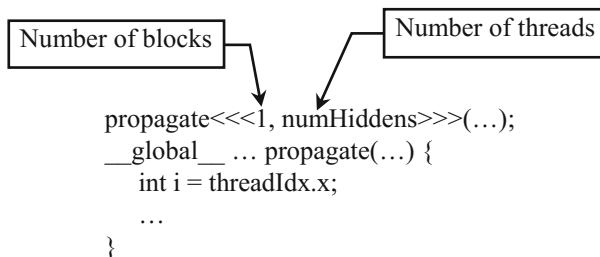
which most often contains a huge number of simple Streaming Multiprocessors (SMs). The realization presented here is based on Compute Unified Device Architecture (CUDA) [1] which is an extension of the C language and allows GPU code to be written in regular C. The written code can be executed in both host Central Processing Unit (CPU) or at the device processor GPU.

The main problem of executing program code in GPU is that the algorithms must be adapted to be with fine-grained parallelism. Here an approach is presented and used for parallel execution in both forward and backward stages on the multilayer perceptron. The independent processing units in each layer work in parallel as in the forward stage the parallel calculations are realized in the hidden and output layers and in the backward stage in the hidden and input layer. The proposed approach is described for neural networks with one hidden layer but it is applicable for arbitrary number of hidden layers as well. The experimental data used for the training are financial data of individuals where the goal is to determine their credit rating based on historical examples. The performance of the realization is shown according to the time for execution in the sequential and parallel implementations.

### 3 Specific Features of CUDA Programming

In CUDA programming, entry points are provided to the GPU by C functions called kernels. Syntactically they are invoked as normal functions with two differences.

- Memory management between CPU and GPU. The memory regions available in CPU must be copied in order to be used in GPU before invoking the kernels and after that in opposite directions to obtain the results. Finally the allocated GPU memory must be freed.
- By calling the kernel the number of grids, blocks and threads must be specified. They are 3D dimensional arrays which are physically used for execution of kernel functions. In kernel entry points grids, blocks and threads are specified that describe the level of parallelism. The kernel function is executed once for every thread, so specified number of threads determines the number of executions and in the kernel function appropriate program constructs must be used to execute independent code fragments in parallel. The indexes of blocks and threads are available by built-in variables provided by CUDA. An example of calling a kernel function and the kernel function prototype is shown below.



### 3.1 Training

In the training stage the data available is presented as pairs of input-output training vectors  $z_1 \rightarrow d_1, z_2 \rightarrow d_2, \dots, z_p \rightarrow d_p$ . Input vectors are denoted here as  $z$  and the output as  $d$ . The number of the input neurons and output neurons are determined according to the number of input and output values in provided training examples and the number of the hidden neurons is determined by different approaches [9]. For example cross-validation can be used comparing the output error and choosing the best hidden neurons numbers.

Output values of the input elements are the same as their input values  $z_1, z_2, \dots, z_p$ , but for the other layers an activation function is used of different types [9]. Thus, the data should be transformed in its definition domain ranging from minimum to maximum according to the activation function [4].

The activation function can vary in the neural network layers and processing neurons but most often only one type is chosen in the training. The main principle for choosing a function type is that it must be non-linear and its derivative must be easy to be computed.

The neural network is trained until the criterion for end of training is satisfied, which can be, for example, reaching the given number of epochs or minimal error for all training patterns [6]. In every epoch all training patterns are presented to the neural network in random or spatially ordered way and the weights of the connections between neurons are modified, iteratively improving the generated outputs to be as close as possible to the given outputs.

### 3.2 Parallel Algorithm

The parallel execution is performed in both forward and backward stages of the backpropagation training algorithm. In the forward stage the independent calculations are done for the neurons in the hidden layer and in the same way the calculations for the output neurons are performed on separate SM – Fig. 1.

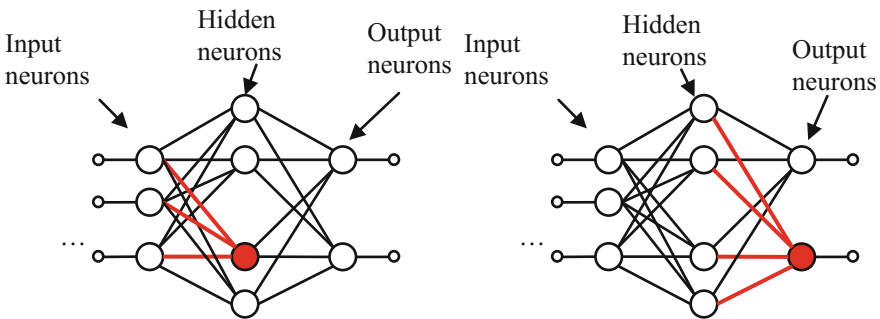
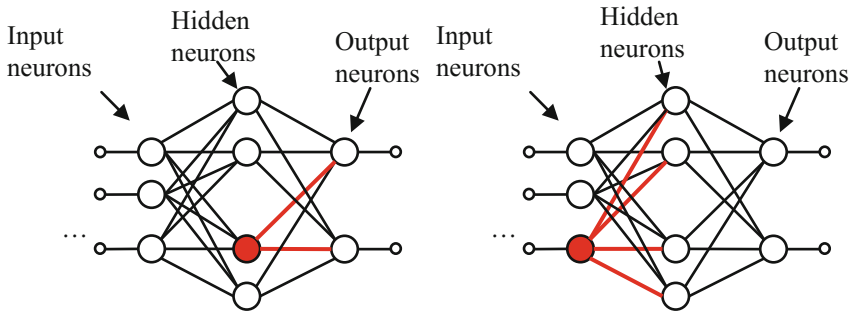


Fig. 1. Independent neuron in the hidden layer for the forward stage



**Fig. 2.** Independent neuron in the hidden layer for the backward stage

In the backward stage calculations in the neurons of the hidden layers are independent in and neurons in input layer are also performed in parallel – Fig. 2.

## 4 Results

The results shown here are obtained for 100 training examples and every training example consists of 22 real values. The training is performed in 500 epochs and neural network architecture with one hidden layer, 22 input neurons, 30 hidden neurons and one output neuron. The experiments are performed 20 times and the average time for the sequential realization in GPU is 25.89 s; the parallel GPU realization lasts 3.73 s. If the parallel calculations are performed only in the forward stage, shown in Fig. 1, the time for execution is 18.55 s.

## 5 Conclusions and Future Work

The parallel realization depends on the architecture and the algorithm of the neural network. There are different other approaches that have also to be realized and tested, as local approach, according to which the training patterns are separated in subgroups and for every subgroup a separate neural network is trained, that could be done in parallel. Also some hierarchical structures can be developed with independents sub-structures. As the results show that the investigated approach is promising, especially if the processing elements are as more as possible in the hidden and output layers, it is applicable in practical software solutions.

## References

1. Cheng, J., Grossman, M., McKercher, T.: Professional CUDA Programming. Wrox (2014)
2. Du, K.-L., Swamy, M.N.S.: Neural Networks in a Softcomputing Framework. Springer, London (2006)
3. Fausett, L.: Fundamentals of Neural Networks: Architectures, Algorithms, and Applications. Prentice-Hall, Upper Saddle River (1994). ISBN: 0-13-334186-0
4. Galushkin, A.I.: Neural Networks Theory. Springer, Heidelberg (2007). ISBN: 978-3-540-48124-9
5. Hech-Nielsen, R.: Theory of the Backpropagation Neural Network. Neural Networks for Perception, (vol. 2): Computation, Learning, Architectures, pp. 65–93. Hecourt Brace & Co. (1992). ISBN: 0-12-741252-2
6. Zurada, J.: Introduction to Artificial Neural Systems. West Publishing Co., St. Paul (1992)
7. Kasabov, N.K.: Foundations of Neural Networks, Fuzzy Systems, and Knowledge Engineering. The MIT Press, Cambridge (1998). ISBN-10: 0-262-11212-4, ISBN-13: 978-0-262-11212-3
8. Kermanshahi, B.: Recurrent neural network for forecasting next 10 years loads of nine Japanese utilities. *Neurocomputing* **23**(1), 125–133 (1998)
9. Tarassenko, L.: A Guide to Neural Computing Applications. Elsevier (2004)
10. Touretzky, D.S.: 15-486/782: Artificial Neural Networks, Lectures, Carnegie Mellon University, Fall (2006). <http://www.cs.cmu.edu/afs/cs.cmu.edu/academic/class/15782-f06/syllabus.html>



# Semantic Annotation Modelling for Protein Functions Prediction

Deyan Peychev<sup>1</sup> and Irena Avdjieva<sup>2</sup>(✉)

<sup>1</sup> AgroBioInstitute, 8 Dragan Tsankov Str., 1164 Sofia, Bulgaria  
deyan.pey@gmail.com

<sup>2</sup> Faculty of Mathematics and Informatics, Sofia University  
“St. Kliment Ohridski”, 5 James Bourchier Blvd, 1164 Sofia, Bulgaria  
i.y.avdjieva@fmi.uni-sofia.bg

**Abstract.** Functional protein annotation is a key phase in the analysis of de-novo sequenced genomes. Often the automatic annotation tools are insensitive to removing wrong annotations associated with contradictions and non-compliance in biological terms. In this study, we introduce a semantic model for representation of functional annotations based on a resource description framework standard (RDF).

We have integrated several databases with information for protein sequences and ontologies describing the functional relationships of the protein molecules. By using Web Ontology Language (OWL) axioms, RDF storage engines are able to take decisions which candidate annotations should be marked as biologically unviable and do not withstand the reality checks associated with coexistence, subcellular location and species affiliation [1]. This approach reduces the number of false positives and time spent in machine annotation’s curation process. The presented semantic data model is designed to combine the semantic representation of annotations with examples designed for machine learning.

Current work is part of a large scale project of functional annotation of plant genomes.

**Keywords:** Functional annotation · Semantic web · Data model  
Machine learning

## 1 Introduction

The genome annotation process is a limiting step in genome sequencing due to difficulties in searching and interpreting the various candidate gene reference records available in biological databases. Due to lack of standards for knowledge presentation in sequencing, the main problems are related to the semantics of annotation descriptions and quality checks. In addition, expert assessment of automatic annotations shows that most annotation-related issues originate from inconsistencies due to failure of major reality checks. Such inconsistencies increase false positives when assigning functions to sequenced regions. The assignment of gene functions appears to be one of the limitation steps for NGS data analyses, and depends on the slow process of manual curation to obtain proper knowledge of the function of newly-sequenced regions [2].



The aim of the current work is to design a semantic model for formal description of some protein sequence databases and provide an integrated repository of machine learning examples. We experiment with tracking inconsistencies between automatic annotations using the capabilities of OWL Lite and some restrictions in the Gene Ontology classes [3]. Reducing inconsistencies will reduce the effort and time spent in quality assessment of annotations.

We use RDF standard to represent referenced database records and their relationships with controlled vocabularies, families, patterns, motifs, and other objects used in annotation. The formal description of these units and relationships provides opportunities for extracting knowledge and presenting implicit machine conclusions, which can be used to automate a large number of checks performed by curators [4]. We also use web service API to easily navigate and request a semantic network based on the SPARQL protocol. With SPARQL endpoints, researchers are able to link results from different research groups together, thus improving the productivity and quality of the process of sequencing multiple genomes.

## 2 Materials and Methods

There are many public databases that offer protein information [5] but for the purpose of our study we rely on those listed below:

**UniProt** [6] is a major resource of protein sequence information that contains protein sequences, features and a lot of manually annotated protein functions. For this study we obtained manually annotated protein sequences from multiple species that are also referenced to both the PROSITE database and the Gene Ontology Consortium.

**PROSITE** [7] is a database for protein domains, families and functional motifs. PROSITE descriptions come from manually curated sources and are often included in machine predicted annotations. Extracted information, used for the study, contains patterns - regex-like representation of the rules which are applied to composing a pattern that matches a set of proteins. Every pattern is linked to set of UniProt entries.

**Gene Ontology (GO)** [8] is a major resource for protein annotation which describes proteins in terms of function, subcellular location and biological process. In the current study only terms for “molecular function” are included. The disjoint restrictions of Gene Ontology are used to define semantics of reality checks applied in a semantic model.

**InterPro** [9] is an integrated resource used for protein classification. IT is used to obtain and predict information about protein domains and motifs. It also contains machine annotations linked to Gene Ontology terms that can be used as benchmark to compare different machine learning annotation methods.

### 2.1 Data Preparation

The first task is to convert the PROSITE patterns to real regex patterns. Thus, they are capable to be executed against linked corresponding sequences and to retrieve actual matching amino acid fragments that characterize the link between the pattern and protein sequence. These amino acid fragments later can be used to train machine

learning algorithms such as Support Vector Machine [10] or J48 [11] to predict which GO term describes molecular function of de-novo sequenced proteins. PROSITE general entry information, amino acid fragments and UniProt identifiers are converted to RDF format keeping all identifiers of the original data sources intact and resolvable. Following basic schemata of OWL classes and properties, this transformation can be described in N-Triples-like syntax (Fig. 1).

```
@prefix rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#
@prefix rdfs: http://www.w3.org/2000/01/rdf-schema#
@prefix prosite: https://prosite.expasy.org/
@prefix uniprot: http://purl.uniprot.org/uniprot/
@prefix abi-model: http://abi.bg/model/

prosite:PS00023 rdf:type          prosite:Pattern
prosite:PS00023 rdfs:label       "Fibronectin type-II collagen-binding domain signature."
prosite:PS00023 prosite:pattern  "P-F-X-[FYWIV]-x(7)-C-x(8,10)-W-C"
prosite:PS00023 abi-model:fragment "PFFWV"
prosite:PS00023 abi-model:fragment "FWVW"
prosite:PS00023 abi-model:fragment "PFYVW"
prosite:PS00023 prosite:uniprotReference uniprot:Q075Z2
prosite:PS00023 prosite:uniprotReference uniprot:Q3UW26
prosite:PS00023 prosite:uniprotReference uniprot:Q9GL25
```

**Fig. 1.** Transformation syntax

UniProt and Gene Ontology already have their own RDF distributions and can be directly referenced with resolvable URIs. UniProt is huge information resource and that is the reason to use only part of the database related to protein sequences and Gene Ontology annotations. These data sets can be directly mapped to PROSITE RDF serialization without any modifications. N-Triples-like syntax (Fig. 2):

```
@prefix rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#
@prefix uniprot: http://purl.uniprot.org/uniprot/
@prefix uniprot-core: http://purl.uniprot.org/core/
@prefix uniprot-isoforms: http://purl.uniprot.org/isoforms/
@prefix gene-ontology: http://purl.obolibrary.org/obo/

uniprot:A0A0S4L2B6 uniprot-core:classifiedWith http://purl.obolibrary.org/obo/GO_0005743
uniprot:A0A0S4L2B6 uniprot-core:classifiedWith http://purl.obolibrary.org/obo/GO_0016021
uniprot:A0A0S4L2B6 uniprot-core:classifiedWith http://purl.obolibrary.org/obo/GO_0070469
uniprot:A0A0S4L2B6 uniprot-core:sequence uniprot-isoforms:A0A0S4L2B6-1
uniprot-isoforms:A0A0S4L2B6-1 rdf:value
"MVGTSLSMLIRTELSSPEKLIENDQIYNTIVTAHAFIMIFFMVMVMIGGFGNWLIPLMLGA"
```

**Fig. 2.** Serialization syntax

InterPro hierarchy of protein families is converted to RDF format from its XML distribution using `rdfs:subClassOf` property. UniProt identifiers that refer to protein families are converted to be URIs with "<http://purl.uniprot.org/uniprot/>" prefix to fit directly to the rest of the schema.

## 2.2 Annotation Schema

Data sources mentioned in previous section are transformed to RDF format and linked together with common persistent identifiers in the form of URIs. This part of the semantic model is suitable to feed instances required to train machine learning model. One more schemata is designed to describe and store predictions from machine learning model in order to provide semantic functional annotations about new proteins which are unseen by machine learning model. This schema contains amino acid sequence which is analysed, predicted Gene Ontology terms linked as labels and some metadata for details of the analysis - such as the algorithm used for classification, the name of the problem transformation method used if the classification task is considered to be multi-label task, date of the analysis and the name of the training dataset. To make the link between annotation object and particular Gene Ontology class, the build-in RDF property “rdf:type” is used.

## 2.3 Storage Engine and Inference Rules

After RDF transformations, data sets are loaded to RDF-triple storage engine “GraphDB” provided by Ontotext AD (<http://graphdb.ontotext.com>). Inference is based on one of the predefined GraphDB reasoners “OWL-Horst” and axioms defined in Gene Ontology and InterPro class hierarchy. GraphDB supports custom inference rules which generates custom extensions of the default reasoner. One custom rule is defined to follow disjointness between GO classes in order to materialize inconsistencies derived from machine annotations. The semantic model provides possibilities to improve the machine learning model prior every iteration.

This rule states the following logic - if there are two classes “a” and “b” and they are in disjoint, then for every annotation instance linked to both, an additional implicit statement will be generated indicating inconsistent link between the annotation object and one of the classes. This is a convenient way to analyse the number and the nature of inconsistencies using SPARQL queries.

Unlike OWL DL, OWL Lite does not allow the use of owl:disjointWith [12]. Using OWL Lite reasoning the usage of owl:disjointWith can be interpreted as not built-in property and consistency checks will never fail, but at the same time we can easily trigger disjointness applying custom inference rule to generate implicit statements to materialize inconsistencies as regular RDF statements.

## 3 Results and Discussion

PROSITE database is represented with 2511 unique patterns in release 2017\_11. A total number of 268 692 unique relationships can be loaded in a semantic repository for two hours without inference. After this time, the inference overloads and triples the loading time. It took 16 h to infer OWL Lite axioms over Gene Ontology classes on 8 CPU core machine with 32 GB of RAM. Overall number of statements after the inference step is scaled to 2.5 million.

To test the inference of the semantic model, one batch of 300 simulated instances of machine annotations are loaded with total number of 32 inconsistencies found and materialized as implicit RDF statements. Simulated annotations are generated with the principle to add some GO terms which are in disjoint as classes for one annotation. For example the terms GO:0003690 and GO:0003697 which are actually “double-stranded DNA binding” and “single-stranded DNA binding” terms. They are in disjoint as GO classes and didn’t pass the reality check because two terms are opposite [1]. The predicate “abi-model:inconsistentWith” is generated between annotation node and the second term linked to that node (in this case GO:0003697).

The question is which GO term do not pass the reality check. Actually both classes have explicit disjointWith statements for each other and just the order of the appearance is necessary but not sufficient criteria to choose one of them. One probable solution is to keep the term with higher probability score generated by machine learning algorithm at prediction time. This is possible if the classification task is defined as multi-label task and probabilities distribution of labels is returned for every prediction. Descending order of Gene Ontology terms thus will provide possibility first to add the term with highest score into semantic repository, next is the second one etc. If some terms are in disjoint with other terms from label collection which are already added to the repository, they will be marked as inconsistent.

## 4 Conclusion

Despite the quality of the predictions made by machines, quality check by curators is a mandatory task in tracking and discarding logical inconsistencies. Semantic networks are capable to deal with human knowledge in machine understandable way providing great possibilities for researchers to encode domain specific knowledge about properties of objects of interest like genes and proteins.

Using knowledge stored in ontologies, semantic networks can reduce probabilities of false positive annotations due to inconsistencies between proposed concepts and conclusions of learned models. This study provides not only an alternative to existing methods for prediction annotations, but also a way to detect false or contradicting automatic annotations. Our future work in the field is to apply these machine learning annotations from use case scenario and evaluate them through manually annotated data.

**Acknowledgements.** The presented work has been funded by the Bulgarian NSF within the “GloBIG: A Model of Integration of Cloud Framework for Hybrid Massive Parallelism and its Application for Analysis and Automated Semantic Enhancement of Big Heterogeneous Data Collections” project, Contract DN02/9 of 17.12.2016, and by Sofia University SRF within the “Models for semantic integration of biomedical data” project, Contract 80-10-207/26.04.2018

## References

1. Koonin, V., Galperin, Y.: Computational Approaches in Comparative Genomics. Sequence - Evolution – Function (2003)
2. Poux, S., Arighi, N., et al.: Expert curation in UniProtKB: a case study on dealing with conflicting and erroneous data. *Database* **2014**(1) (2014). <https://doi.org/10.1093/database/bau016>
3. The Gene Ontology Consortium: Gene ontology annotations and resources. *Nucleic Acids Res.* **41**(D1), 530–535 (2013). <https://doi.org/10.1093/nar/gks1050>
4. Claude, P., Fabrice, G., et al.: THEA: Ontology driven analysis of microarray data. *Bioinformatics* (2007). <https://doi.org/10.1093/bioinformatics/bth295>
5. Xu, D.: Protein databases on the internet. *Curr. Protoc. Mol. Biol.*, Chap. Unit–19.4 (2004). <https://doi.org/10.1002/0471142727.mb1904s68>
6. Pundir, S., Martin, J., et al.: UniProt protein knowledgebase. In: Wu, C., Arighi, C., Ross, K. (eds.) *Protein Bioinformatics* (2017). *Methods Mol. Biol.* **1558**
7. Sigrist, A., Cerutti, L., et al.: PROSITE, a protein domain database for functional characterization and annotation. *Nucleic Acids Res.* **38**(Database issue), 161–166 (2010). <https://doi.org/10.1093/nar/gkp885>
8. Gene Ontology Consortium: The Gene Ontology (GO) database and informatics resource. *Nucleic Acids Res.* **32**(suppl. 1), D258–D261 (2004)
9. Apweiler, K., Attwood, A., et al.: The InterPro database, an integrated documentation resource for protein families, domains and functional sites. *Nucleic Acids Res.* **29**(1), 37–40 (2001)
10. Vapnik, V.: *The Nature of Statistical Learning Theory*. Springer, Heidelberg (2013)
11. Kaushik, S., Chouhan, U., Dwivedi, A.: Prediction of protein subcellular localization of human protein using j48, random forest and best first tree techniques. *J. Adv. Appl. Sci. Res.* **1**(12) (2017)
12. W3C Recommendation: OWL Web Ontology Language Reference, 10 February 2004



# *ReadME* – Enhancing Automated Writing Evaluation

Maria-Dorinela Sirbu<sup>1</sup>, Robert-Mihai Botarleanu<sup>1</sup>,  
Mihai Dascalu<sup>1,2,3(✉)</sup>, Scott A. Crossley<sup>4</sup>,  
and Stefan Trausan-Matu<sup>1,2,3</sup>

<sup>1</sup> University Politehnica of Bucharest,  
313 Splaiul Independentei, 060042 Bucharest, Romania  
maria.sirbu@cti.pub.ro,  
robert.botarleanu@stud.acs.pub.ro,  
{mihai.dascalu, stefan.trausan}@cs.pub.ro

<sup>2</sup> Academy of Romanian Scientists,  
54 Splaiul Independenței, 050094 Bucharest, Romania

<sup>3</sup> Cognos Business Consulting S.R.L.,  
32 Bd. Regina Maria, Bucharest, Romania

<sup>4</sup> Department of Applied Linguistics/ESL,  
Georgia State University, Atlanta, GA 30303, USA  
scrossley@gsu.edu

**Abstract.** Writing is a central skill needed for learning that is tightly linked to text comprehension. Good writing skills are gained through practice and are characterized by clear and organized language, accurate grammar usage, strong text cohesion, and sophisticated wording. Providing constructive feedback can help learners improve their writing; however, providing feedback is a time-consuming process. The aim of this paper is to present an updated version of the tool *ReadME*, which generates automated and personalized feedback designed to help learners improve the quality of their writing. Sampling a corpus of over 15,000 essays, we used the *ReaderBench* framework to generate more than 1,200 textual complexity indices. These indices were then grouped into six writing components using a Principal Component Analysis. Based on the components generated by the PCA, as well as individual index values, we created an extensible rule-based engine to provide personalized feedback at four granularity levels: document, paragraph, sentence, and word levels. The *ReadME* tool consists of a multi-layered, interactive visualization interface capable of providing feedback to writers by highlighting sections of texts that may benefit from revision.

**Keywords:** Textual complexity · Feedback generation and visualization  
Automated writing evaluation · *ReaderBench* framework  
Natural language processing · Rule-based engine

## 1 Introduction

Many students are interested in improving writing skills because writing quality is an important aspect in defining intellectual capabilities at almost all academic levels. Providing personalized feedback to students about their writing ability is a fundamental component in the learning process. However, providing feedback is a time-consuming and complex process, and many teachers do not have the time to provide feedback in an iterative manner that best supports student learning. As a result, many computer-based systems have been developed to support students in the writing process. These systems are generally referred to automated writing evaluation systems (AWE).

The aim of this paper is to introduce a new version (1.1) of the *ReadME* AWE system based on the *ReaderBench* framework [1]. *ReaderBench* is a multi-lingual framework which integrates advanced Natural Language Processing techniques, and multiple complexity indices: (a) surface level features including word count, sentence count, sentence length, paragraph length, punctuation marks; (b) *lexical* indices related to sophisticated vocabulary work; (c) *syntactic* indices related to word and sentence-level analyses which include syntactic dependencies and part of speech tagging (POS); (d) *semantic* features focusing on the text's structure and centered on local and global cohesion; and (e) *discourse-centered* elements related to dialogism and discourse connectors.

This paper presents an extension of our previous analysis [2], with new improvements, new features, and a new dataset. The new enhancements brought to our system consist of the following features. First, the semantic models within the system were trained on a far larger and better designed corpus (The Corpus of Contemporary American English [COCA] – <https://corpus.byu.edu/coca/>) in order to better conceptualize the text. COCA is the largest available corpus of English and it contains over 560 million words from various categories, such as fiction, academic texts, newspaper, and popular magazines. COCA provides a better conceptualization of words, it covers broader categories and does not include repeated samples of texts which are common in other corpora such as TASA. Second, we do not remove essays from this analysis that contain only one paragraph. Third, we used a specific corpus of 15,496 essays for creating a comprehensive writing sample. Fourth, we improved our processing performance by optimizing the Natural Language Processing (NLP) pipeline from *ReaderBench* which is invoked only once now at document level, and not for each level individually. Fifth, we added a new granularity level for the provided feedback, namely word-level. In contrast to other systems (e.g., T.E.R.A [3]), *ReadMe* provides personalized feedback at four granularity levels and is oriented towards supporting students throughout the learning process.

## 2 Method

**Selected Corpora.** We derived our language features from a large corpus of independent essays collected from several sources. In total 15,496 essays were collected using a number of different prompts. Unlike other corpora commonly used in similar

analyses (i.e., TASA), the essays were all independent writing samples which means the writers could rely on their own background knowledge to write about the given prompt. Most of the essays came from the William and Flora Hewlett Foundation Automated Student Assessment Prize competition hosted in 2012 (<https://www.kaggle.com/c/ASAP-AES>); a smaller percentage of the essays were collected by the fourth author or were collected through the Writing-Pal intelligent tutoring system [4].

**Aggregating Complexity Indices into Principal Components.** More than 1,200 complexity indices were computed using the *ReaderBench* framework. A Principle Component Analysis (PCA) was applied to these indices in order to group the indices into a small number of components. Due to the large number of indices, multiple steps of index pruning were applied, as follows: *checking for normality* – indices with non-normal statistical distributions were eliminated; *elimination of localized indices* – indices with low linguistic coverage were removed; *elimination of outlier essays* – essays with at least 10% outliers indices were excluded (indices were considered outliers if their value deviates more than 2 standard deviations from the mean value); *elimination of strongly correlated indices* – indices that strongly correlate with others above and imposed threshold are eliminated using Pearson correlations. After all these steps, 32 complexity indices were introduced in the PCA. By grouping similar complexity indices into a smaller set of components, the PCA generated 6 components which explain 57.5% of the total variance including: *word complexity* (e.g., different age-of-acquisition lists, polysemy count), *local cohesion* (e.g., sentence - paragraph or sentence adjacency cohesion score), *global cohesion*, and normalized *word counts* related to sentiment polarity (both negative and positive).

**Personalized Feedback Generation through an Extensible Rule-Based Engine** We developed an extensible rule-based engine to provide personalized feedback to users, which supports two types of rules: rules based on the components generated by the PCA, and rules based on the values of individual complexity indices. Each rule has a maximum and minimum range set, and specific feedback messages if the document exceeds these boundaries. The range values from the components were computed based on the average plus/minus standard deviation values. The range values for the rules at word level are computed using Kuperman’s age-of-acquisition word list [5]. All rules and feedback messages were manually configured using a JSON file. Each granularity level has its own defined rules and feedback messages and each rule has more interchangeable feedback messages in order to avoid monotony.

**An Interactive Visualization Interface.** Users can input an essay and receive personalized feedback at the four levels of granularity: document, paragraph, sentence and *word* level. Two color gradients are used to indicate the severity of identified problems: *red* – with darker shades meaning that the text fragment has more issues; *blue* – no rule was triggered; darker shades of blue mean that the segment is similar to the average values measured across the entire corpus, while lighter shades represent values closer to the margins of the [minimum, maximum] acceptable interval. Figure 1 shows feedback generated at *word* level at which each word is individually highlighted based on its severity value computed using Kuperman’s age-of-acquisition.



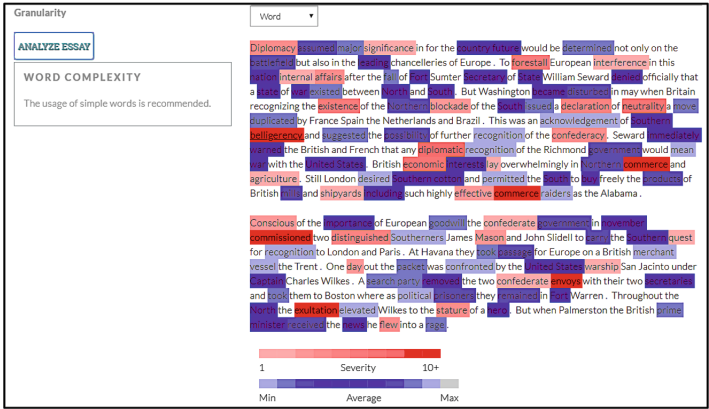


Fig. 1. ReadME – personalized feedback at word level. (Color figure online)

The words that are not highlighted are either stopwords (e.g., “the”, “for”, “in”) or words which are not present in the word list (e.g., named entities like “France”, “Spain”, or “William”). On mouseover, a feedback message for that corresponding specific word is generated in the left side (see Fig. 1 in which the “belligerency” word is underlined). Feedback given at the component level would include personalized messages for each dimension (e.g., “Words in your text tend to be too complex.” or “You should consider writing self-contained, cohesive paragraphs that do not contain too many new ideas.”).

### 3 Conclusions

Automated textual complexity indices found in ReaderBench can provide significant feedback to user’s in consideration of word complexity, syntax, local and global cohesion. This paper is an extension of our previous study [2], with new features and increased performance. Beyond the first release, the current version of the *ReadME* system provides feedback at four granularity levels and includes *word*-level feedback. In addition, the semantic models in this version were trained using the COCA corpus and a more comprehensive baseline of essay writing was established using the complexity indices applied to a larger and more specific corpus of essays.

For the time being, no experiments using the new version of *ReadME* have been performed. Thus, a first follow-up action is to perform experiments with students of various levels to examine the usability of the system and the effectiveness of the feedback provided. In addition, the system will be further enhanced by using specific training collections that reflect different student writing levels, introducing side-by-side feedback for multiple documents in order to compare them, including grammatical and mechanic errors in the rule-based engine, as well as support for multi-lingual analyses.

**Acknowledgments.** This research was partially supported by the ReadME project “Interactive and Innovative application for evaluating the readability of texts in Romanian Language and for improving users’ writing styles”, contract no. 114/15.09.2017, MySMIS 2014 code 119286, as well as the FP7 2008-212578 LTfLL project.

## References

1. Dascalu, M., Dessus, P., Bianco, M., Trausan-Matu, S., Nardy, A.: Mining texts, learner productions and strategies with *ReaderBench*. In: Peña-Ayala, A. (ed.) Educational Data Mining. SCI, vol. 524, pp. 345–377. Springer, Cham (2014). [https://doi.org/10.1007/978-3-319-02738-8\\_13](https://doi.org/10.1007/978-3-319-02738-8_13)
2. Botarleanu, R.-M., Dascalu, M., Sirbu, M.-D., Crossley, S.A., Trausan-Matu, S.: *ReadME* – generating personalized feedback for essay writing using the *ReaderBench* Framework. In: Knoche, H., Popescu, E., Cartelli, A. (eds.) SLERD 2018 2018. SIST, vol. 95, pp. 133–145. Springer, Cham (2019). [https://doi.org/10.1007/978-3-319-92022-1\\_12](https://doi.org/10.1007/978-3-319-92022-1_12)
3. McNamara, D.S., Graesser, A., Cai, Z., Dai, J.: Coh-Metrix Common Core TERA version 1.0, vol. 2018 (2013). <http://coh-metrix.commoncoretera.com>
4. Roscoe, R.D., McNamara, D.S.: Writing Pal: Feasibility of an intelligent writing strategy tutor in the high school classroom. *J. Edu. Psychol.* **105**(4), 1010 (2013)
5. Kuperman, V., Stadthagen-Gonzalez, H., Brysbaert, M.: Age-of-acquisition ratings for 30,000 English words. *Behav. Res. Methods* **44**(4), 978–990 (2012)



# Feature Selection Based on Logistic Regression for 2-Class Classification of Multidimensional Molecular Data

Sebastian Student<sup>1(✉)</sup>, Alicja Płuciennik<sup>1,2</sup>, Michał Jakubczak<sup>1</sup>,  
and Krzysztof Fajarewicz<sup>1</sup>

<sup>1</sup> Institute of Automatic Control, Silesian University of Technology,  
Gliwice, Poland

`sebastian.student@polsl.pl`

<sup>2</sup> WASKO S.A., Gliwice, Poland

**Abstract.** This paper describes a classification system which uses feature selection method based on logistic regression algorithm. As a feature elimination criterion the variance inflation factor of the statistical logistic regression model is used. The experimental results show that this method can be successfully applied for feature selection in classification problem of multidimensional microarray data.

**Keywords:** Feature selection · Logistic regression · Classification  
Cancer diagnosis · Gene expression signatures

## 1 Introduction

Using high-throughput molecular techniques, researchers can study the activity of thousands of genes simultaneously. Using different molecular techniques we can assess gene expression measured by DNA microarrays or RNA-Seq technique, DNA methylation levels measured by DNA methylation microarrays or protein and phosphoprotein levels measured by reverse phase protein arrays. Cancer classification based on high-throughput molecular techniques is one of the most challenging problems in nowadays molecular data research. Over the past decades, a wide range of classification algorithms has been proposed in the literature to tackle various classification problems. Classification algorithms can be divided into two categories: binary and multi-class classifiers. This work is focused on binary problems, but the methodology can also be adjusted to multi-class classification problems.

Here we propose a feature selection method based on logistic regression method. Logistic regression is a popular classification method and has an explicit statistical interpretation where probabilities of class membership, for example different cancer phenotypes, can be assessed. However, in most gene expression studies, the number of genes typically far exceeds the number of samples. This situation is called high-dimensional and low sample size problem. In this case

the logistic regression method cannot be used to estimate the regression parameters directly. For that reason, we have used additional preselection step based on standard feature selection methods.

## 2 Methodology

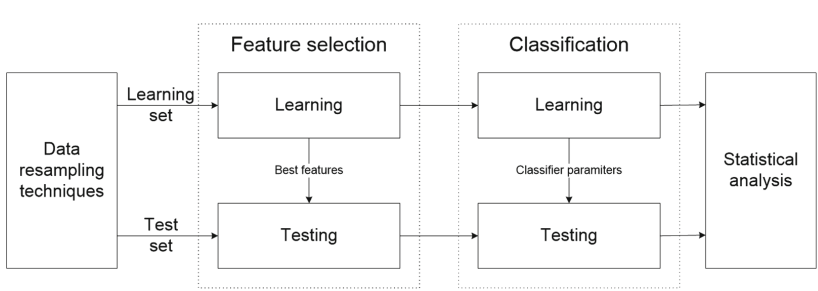
### 2.1 Feature Selection—RegVIF Method

The proposed method of feature selection is based on iterative variable elimination from a logistic regression model. The procedure is as follows. p-value for Wald z-score is calculated for each variable in the model. This score is commonly used for importance assessment of variables in given model, where the null hypothesis is that a variable is not influencing the model. The lower p-value, the more significantly a particular variable influences the model. As a next step, the variance inflation factor (VIF) is calculated for each variable. VIF allows assessing multicollinearity of variables. High values of VIF may indicate multicollinearity of these variables, but there is no clear way to choose a threshold [6, 8]. Both scores: VIF and Wald test p-value are multiplied and the feature with the highest product is selected for elimination. Then the new logistic regression model with new coefficients is re-calculated.

For the presented RegVIF algorithm a different stopping criterion might be applied and we have checked a few of them. The first one is to stop procedure when AIC (Akaike Information Criterion) is increasing, which indicates a loss in the relative quality of the model for the given data. The second one is when maximum VIF reaches a value below an arbitrary threshold. The last one is when a desired number of variables is achieved. The last variant was used in the present study for comparison of the method to other methods. The logistic regression model is suitable for both variable types: categorical and numeric. In the present study, the algorithm was applied on numerical values of normalized gene expression dataset. The superabundance of features in logistic regression model requires genes preselection for the saturated model. For that reason two methods, T-test and fold change based feature selection method, were used.

### 2.2 Classifier Design

Design and used algorithms of the classification system for large-scale data described in the previous section constitute challenging problems. We need to choose the right classifier design, including data preprocessing, choice of a proper classification and feature selection methods. In this paper, we develop a selection method based on logistic regression and compare the results with the most used methods and a different number of selected features. Our classifier is based on the classification/validation scheme proposed in [1], implemented in [2] and is presented in Fig. 1. The same scheme has been also used recently for comparing different data fusion strategies [3]. Additionally, we used the following filter methods for feature selection: the lowest p-values in Student's t-test and the



**Fig. 1.** Classification/validation scheme used for comparing different models.

highest values of logarithmic fold change (FC). The classifiers for each desired number of features were validated using 500 iterations of bootstrap.

As a classification method we have used Support Vector Machines (SVM) algorithm [4,5] with linear kernel. This method is reported as one of the best classifier for multidimensional genomic data analyzed in this paper.

### 3 Data Set Description

In this article, we have used publicly available microarray gene expression dataset of human glioma samples. The data set consists of 50 gliomas with 22 anaplastic oligodendrogliomas and 28 glioblastomas. This dataset was normalized with RMA algorithm and annotated with genecards derived annotations for custom probeset definitions *gahgu95av2* [10]. After normalization, the dataset contains 8359 gene probes. The dataset can be freely downloaded from NCBI Gene Expression Omnibus GEO [9].

### 4 Results

RegVIF feature selection method have been compared to other methods by estimation the accuracy of test sets obtained in 500 iterations of bootstrap technique. In the molecular data classification problem not only the accuracy rate is important, but also how small is the number of selected genes. Hence, we have calculated accuracy for different cardinality of selected feature (gene) set. Based on this one can see how many genes is needed to obtain good prediction accuracy.

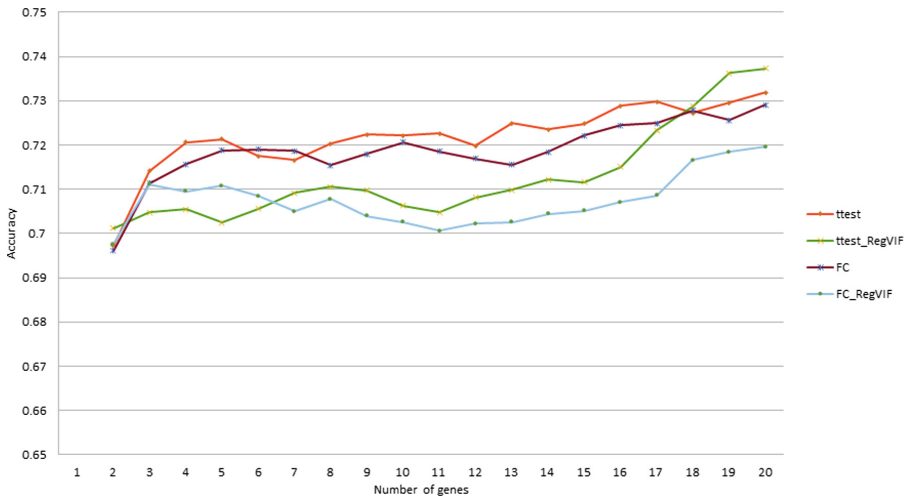
For two types of malignant brain cancers, glioblastoma (WHO, grade IV) and anaplastic oligodendroglioma (WHO, grade III), the calculated accuracy of classification for a range of up to 20 features was between 0.695 and 0.74—see Fig. 2. The best accuracy rate was obtained for our RegVIF feature selection method and t-test preselection. In this case, the best result was observed for 20 selected features. The RegVIF selection also has a higher sensitivity and

specificity rates in comparison to plain t-test selection. They are presented in Table 1.

The presented study dataset shows the problem of binary classification for two of gliomas which are challenging for histological classification [7]. In comparison to the previous work [11], where the accuracy for 20 features was 0.86, our results have lower values. However, this is due to different methodology of the overall classification system evaluation. We used the structure presented in Fig. 1, which exclude the possibility of information leak and eliminate the optimistic bias from the classification quality assessment.

**Table 1.** Best classification accuracy rate for all tested methods

Selection method	Accuracy	Sensitivity	Specificity	nFeatures
ttest	0.732	0.7764	0.6856	20
FC	0.729	0.7615	0.7054	19
ttest RegVIF	0.736	0.7771	0.6989	20
FC RegVIF	0.7196	0.7554	0.6904	20



**Fig. 2.** Bootstrap based classification accuracy by successive gene set reduction selected for different feature selection methods of the SVM classifier

## 5 Summary

In this paper, we have presented results of new RegVIF feature selection method which used the logistic regression model with stopping criterion based on

assumed number of selected features. The main advantage of this method is that we can use it for numeric, categorical and dummy variables in contrast to the most commonly used filter methods like t-test or fold change based feature selection. On the other hand, the logistic regression based feature selection is sensitive to the number of probes and the number of features ratio. In the case of the small number of samples and large number of variables the Wald test p-value increases to 1. In such situation The preselection of features is necessary to use in the logistic regression model. The logistic regression based feature selection can be adopted for multiclass problems using one versus one (ovo) or one versus rest (ovr) approaches.

**Acknowledgments.** This research was supported by Polish National Centre for Research and Development under grant No. NCBR Strateged2/267398/4/NCBR/2015.

## References

1. Student, S., Fujarewicz, K.: Stable feature selection and classification algorithms for multiclass microarray data. *Biol. Direct* **7**(33), 1–20 (2012)
2. Fajarewicz, K., et al.: Large-scale data classification system based on Galaxy Server and protected from information leak. In: Nguyen, N.T., Tojo, S., Nguyen, L.M., Trawiński, B. (eds.) *ACIIDS 2017*. LNCS (LNAI), vol. 10192, pp. 765–773. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-54430-4\\_73](https://doi.org/10.1007/978-3-319-54430-4_73)
3. Pojda, K., Jakubczak, M., Student, S., Świerniak, A., Fajarewicz, K.: Comparing different data fusion strategies for cancer classification. In: Rocha, Á., Guarda, T. (eds.) *ICITS 2018*. AISC, vol. 721, pp. 417–426. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-73450-7\\_40](https://doi.org/10.1007/978-3-319-73450-7_40)
4. Boser, B.E., Guyon, I.M., Vapnik, V.: A training algorithm for optimal margin classifiers. In: *Fifth Annual Workshop on Computational Learning Theory*, Pittsburgh (1992)
5. Brown, M.P.S., et al.: Knowledge based analysis of microarray gene expression data by using support vector machines. *Proc. Natl. Acad. Sci.* **97**(1), 262–267 (2000)
6. Vu, D.H., Muttaqi, K.M., Agalgaonkar, A.P.: A variance inflation factor and backward elimination based robust regression model for forecasting monthly electricity demand using climatic variables. *Appl. Energy* **140**, 385–394 (2015)
7. Van den Bent, M.J.: Interobserver variation of the histopathological diagnosis in clinical trials on glioma: a clinicians perspective. *Acta Neuropathol.* **120**(3), 297–304 (2010)
8. O'Brien, R.: A caution regarding rules of thumb for Variance Inflation Factors. *Qual. Quant.* **41**, 673–690 (2007)
9. Clough, E., Barrett, T.: The gene expression omnibus database. *Methods Mol. Biol.* **1418**, 93–110 (2016)
10. Ferrari, F., et al.: Novel definition files for human GeneChips based on GeneAnnot. *BMC Bioinform.* **8**, 446 (2007)
11. Nutt, C.L., et al.: Gene expression-based classification of malignant gliomas correlates better with survival than histological classification. *Cancer Res.* **63**(7), 1602–1607 (2003)

## Author Index

- Agre, Gennady 3  
Ahmad, Tariq 16  
Ahmed, Hanady 16  
Alshahrani, Amal 83  
Altmami, Nouf Ibrahim 255  
Amelio, Alessia 152  
Angelova, Galia 60, 115  
Atanasov, Atanas 126  
Avdjieva, Irena 275
- Balodis, Kaspars 25  
Bohus, Martin 141  
Botarleanu, Robert-Mihai 281  
Boumarafi, Yazid 197  
Boycheva, Svetla 36  
Brodic, Darko 152
- Chelcioiu, Ionut Daniel 207  
Chiru, Costin-Gabriel 163  
Corlatescu, Dragos 207  
Crossley, Scott A. 281
- Dascalu, Mihai 207, 281  
de Matos, David Martins 93  
Deksne, Daiga 25  
Draganov, Ivo R. 152
- Fujarewicz, Krzysztof 286
- Genest, David 218  
Georgiev, Milen 240  
Gimbel, Stephan 141  
Goerg, Nora 141
- Hardalov, Momchil 48, 126  
Humm, Bernhard G. 141
- Jakubczak, Michał 286  
Janković, Radmila 152
- Kanishcheva, Olga 60  
Keskinova, Simona 3
- Khaluf, Yara 260  
Kopev, Daniel 126  
Koprinkova-Hristova, Petia 73  
Koychev, Ivan 48, 126  
Krachunov, Milko 173  
Kulawik, Maciej 265
- Loiseau, Stéphane 218
- Martins, Bruno 104  
Menai, Mohamed El Bachir 255  
Mihaylov, Iliyan 186  
Mitov, Kristiyan 126
- Nakov, Preslav 48, 115, 126  
Nikolov, Ventsislav 270  
Nikolova, Ivelina 60  
Nisheva, Maria 173, 186  
Nowak, Robert M. 265
- Osenova, Petya 73
- Paraschiv, Ionut Cristian 207  
Petrov, Daniel 3  
Peychev, Deyan 275  
Pluciennik, Alicja 286  
Popov, Alexander 73  
Posea, Vlad-Valentin 163
- Ramsay, Allan 16, 83  
Ray, Thomas 240  
Ribeiro, Eugénio 93  
Ribeiro, Ricardo 93, 104  
Robert, Adrian 218  
Rodrigues, Frederico 104
- Salhi, Yakoub 197, 228  
Schüller, Martin 141  
Shimohara, Katsunori 240  
Simoens, Pieter 260  
Simov, Kiril 73  
Sirbu, Maria-Dorinela 281



Stefchov, Evgeni [115](#)

Steffens, Marc [141](#)

Student, Sebastian [286](#)

Tanev, Ivan [240](#)

Trausan-Matu, Stefan [207](#), [281](#)

Van Havermaet, Stef [260](#)

Vassilev, Dimitar [173](#), [186](#)

Vonderlin, Ruben [141](#)

Zlatkova, Dimitrina [126](#)