IJCLA **ISSN 0976-0962**

International Journal of Computational Linguistics and Applications

Vol. 5 **No. 1** **Jan-Jun 2014**

Editor-in-Chief
Alexander Gelbukh

ISSN 0976-0962

International Journal of Computational Linguistics and Applications

Vol. 5 **No. 1** **Jan-Jun 2014**

International Journal of Computational Linguistics and Applications – IJCLA (started in 2010) is a peer-reviewed international journal published twice a year, in June and December. It publishes original research papers related to computational linguistics, natural language processing, human language technologies and their applications.

The views expressed herein are those of the authors. The journal reserves the right to edit the material.

Indexing: Cabell's Directory of Publishing Opportunities.

Editor-in-Chief: Alexander Gelbukh

Subscription: India: Rs. 2699
Rest of the world: US$ 249

Payments can be made by Cheques/Bank Drafts/International Money Orders drawn in the name of BAHRI PUBLICATIONS, NEW DELHI and sent to:

BAHRI PUBLICATIONS
1749A/5, 1st Floor, Gobindpuri Extension,
P. O. Box 4453, Kalkaji, New Delhi 110019
Telephones: 011-65810766, (0) 9811204673, (0) 9212794543
E-mail: bahrius@vsnl.com; bahripublications@yahoo.com
Website: http://www.bahripublications.com

Printed & Published by Deepinder Singh Bahri, for and on behalf of **BAHRI PUBLICATIONS**, New Delhi.

International Journal of Computational Linguistics and Applications

Vol. 5 **No. 1** **Jan-Jun 2014**

CONTENTS

SYNTAX, PARSING, AND TAGGING

Editorial

This issue of IJCLA presents papers on three topics: semantics and topic classification; sentiment analysis and truthfulness detection; and syntax, parsing, and tagging.

The first section consists of three papers devoted to semantics and topic classification. Extracting and using semantics, that is, the meaning of the text, is the main task of natural language understanding, and is gaining increasing importance in the majority of natural language processing tasks. Modern natural language processing systems are supposed to behave according to the meaning of the text irrespective from how and in what words this meaning is expressed.

Topic classification is a particular task of semantic interpretation. It consists in automatically recognizing the main topic of the given text—such as deciding whether a newspaper article or a blog post is on sports, politics, culture, science, etc.

D. Huynh *et al*. (Australia) propose a novel distributional measure of semantic similarity between words, using the local context. In addition to the use of words that appear in typical contexts of the two given words, they detect latent topics, which gives a more accurate measure of similarity when the contexts differ in words but are similar in topics. Their method outperforms other methods based purely on vector representation of texts, and is second best after a more sophisticated method that uses multi-prototypes.

L. Wolf *et al*. (Israel) describe a vector-based representation of words, known as word embedding, for two languages at the same time. This is useful in various ways. One is data sparseness: when data for one language is insufficient, information can be borrowed from another language for which there are more texts available. Another is disambiguation: data translated in a different language contains important information on the contextual meaning of ambiguous words. The authors describe the process of building a dataset analogous to the famous word2vec dataset provided by Google, but for a language for which much smaller amount of texts is available.

D. Inkpen and **A. H. Razavi** (Canada) develop a novel method for automatic detection of topics in texts. In this context, a topic is a group of words that have some relation with each other, often clearly felt or even interpretable for humans. Representing a document as a vector of topics instead of a vector of words leads to very significant dimensionality reduction and thus speedup of machine learning algorithms. The algorithm developed by the authors offers different levels of granularity of the topics, so that the users can balance speed of processing with accuracy of the representation.

The second section presents two papers devoted to sentiment analysis and truthfulness detection. Sentiment analysis is a relatively young but very actively developed and very popular area of natural language processing. A typical sentiment analysis tasks consists in detecting whether the text expresses positive or negative opinion about something or some emotion, again positive, such as joy or surprise, or negative, such as sadness, disgust, anger, or fear. Analysis of this type has very important practical and commercial applications: in order to make buying choices, consumers need to know what other people think or feel about a specific product or service; companies need to know what the users feel about their products or services; political parties need to know what voters think and feel about a candidate or political program.

Truthfulness detection is the task of automatically deciding whether a given text, such as a speech of a politician, is a lie or truth. The importance of such a task cannot be overemphasized and does not need to be explained here. Technically, the task is difficult because of limited availability of examples of real-world false texts, that is, real lies and not literary fiction texts. The methods that can be used in such a task are akin to those used to detect emotions; actually, what such a program can detect is an "emotion of lying", similar to what a physical lie detector measures.

C. Vania *et al*. (Indonesia) suggest a method for developing sentiment lexicons for languages for which not many texts are available, as well as the use of such lexicon for classification of texts in that language into positive or negative polarity. The method for compilation of the sentiment lexicon is based on the use of seed words translated from an existing polarity lexicon, in this case for English.

V. V. Datla *et al*. (USA and The Netherlands) present a technique to predict whether a short political statement is true or false. Since automatically checking the facts expressed in the statement is unfeasible, they guess the truthfulness of the text by the way it is expressed, using

only linguistic features. They achieve up to 59% accuracy, which is quite encouraging.

The last section consists of four paper devoted to syntax, parsing, and tagging. Syntactic structure of a sentence describes how the words in the sentence are grouped together, or—in a different view on syntactic phenomena—which words of the sentence add details to the meaning of other words. For example, in the sentence "John loves Mary", the words "loves Mary" are grouped together to describe a state of John, and the whole situation is described by grouping together "John" with the expression "loves Mary", which describes his state. Or, in another view on syntax, both words "John" and "Mary" add details to the word "love", thus describing a more specific type of the situation: specifically John's love and loving specifically Mary. This latter approach is called dependency parsing.

Accordingly, automatic detection of such relationships in a sentence, a process called parsing, helps understanding its meaning and plays important role in various tasks of automatic language processing.

In particular, tagging is a process of disambiguating the possible syntactic role of a word in context, that is, determining its part of speech and related properties, when the word is used in sentence: for example, the word "deep" may refer to an adjective in some contexts, to a noun in other, and to a verb yet in other contexts. Tagging is a simpler task than parsing and is much faster. With this, tagging is a step commonly used in the processing chain of many practical natural language processing applications. It is usually the first step of parsing, too, which greatly improves the speed of parsing.

O. Lacroix *et al*. (France) consider the dependency parsing formalism, which is very popular nowadays due to its adequacy for many applications. Accurate dependency parsers are trained on manually labelled text corpora. Manual labelling is a very expensive, tedious, and error-prone process. The authors describe a technique for automatically pre-labeling the corpus, so that the human annotators are offered the most probably labels, or a set of choices ordered by their probability, much like a word processor offers the user a choice of orthographic corrections for a word. Choosing a correct label of a pre-computed set, or most often just confirming the highest-ranked variant, is much faster than assigning labels from scratch.

B. Galitsky *et al*. (USA and Russia) present a method to build syntactic structures that extend to whole paragraphs instead of only one sentence. The trees of individual sentences are connected by co-referent

nodes into a larger graph. Such representation can be used for measuring similarity between texts, which in turn is at the core of a wide range of natural processing tasks such as information retrieval, text classification, and many others. With this extended structure, the authors obtain up to 8% improvement in accuracy of information retrieval of short texts, such as blog posts. The authors provide an open-source implementation of their algorithm.

O. Krůza and **V. Kuboň** (Czech Republic) describe a lightweight method for recognition of clauses and their relationship in complex sentences, relying only on morphological information. Such recognition may in the future improve the performance of syntactic parsers when dealing with complex sentences. In addition, fast and simple clause recognition is useful in those tasks that do not need complete, and thus costly, syntactic analysis—for example, in information extraction or information retrieval.

G. Orosz *et al*. (Hungary) give an extensive discussion of the lessons learned from tagging medical texts. They concentrate the discussion on Hungarian language, an under-resourced agglutinative language. The authors show how to extend and adapt existing resources to this task. They achieve about 50% reduction in the error rate. Their conclusions and advice would probably be useful for implementing tagging methods for medical domain in other under-resourced languages, especially agglutinative languages.

This issue of IJCLA will be useful for researchers, students, software engineers, and general public interested in natural language processing and its applications.

ALEXANDER GELBUKH
EDITOR IN CHIEF

RESEARCH PROFESSOR,
CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN,
INSTITUTO POLITÉCNICO NACIONAL, MEXICO
WEB: <WWW.GELBUKH.COM>

Semantics and Topic Classification

IJCLA VOL. 5, NO. 1, JAN-JUN 2014, PP. 11–25
RECEIVED 06/01/14 ACCEPTED 06/02/14 FINAL 19/06/14

Semantic Similarity Measure Using Relational and Latent Topic Features

DAT HUYNH, DAT TRAN, WANLI MA, AND DHARMENDRA SHARMA

University of Canberra, Australia

ABSTRACT

Computing the semantic similarity between words is one of the key challenges in many language-based applications. Previous work tends to use the contextual information of words to disclose the degree of their similarity. In this paper, we consider the relationships between words in local contexts as well as latent topic information of words to propose a new distributed representation of words for semantic similarity measure. The method models meanings of a word as high dimensional Vector Space Models (VSMs) which combine relational features in word local contexts and its latent topic features in the global sense. Our experimental results on popular semantic similarity datasets show significant improvement of correlation scores with human judgements in comparison with other methods using purely plain texts.

1 INTRODUCTION

In many language-based applications, it is crucial to be able to measure precisely the semantic similarity between words. VSMs have been used to represent word meanings in a vector that captures semantic and syntactic information of the word. Generated latent topics from a large plain text corpus have been used as vector features for semantic similarity measure [1, 2]. Syntactic/lexical patterns of word (word pairs) in local contexts have also been used as vector features for the similarity measure [3–5].

In this work, we utilize a large plain text corpus as the knowledge-domain to propose a new set of features for semantic similarity task. The

feature set is extracted by considering relational participants (features) surrounding a focus word and its latent topic features over a large plain text corpus. Therefore, a VSM representation of a word is modelled as a high dimensional vector of the combination of relational features and latent topic features. We have developed parameters of the combined features on the MTurk dataset [6] and tested on the popular semantic similarity datasets such as WS-353 [7] and RG-65 [8]. The experimental results confirm the significant improvement of our proposed method on semantic similarity measure in comparison to other corpus-based methods tested on the same datasets.

The paper is organized as follows: We first present the construction of distributed representation in Section 2. In Section 3, the task of word similarity measure is described. In Section 4, our experimental setups and results are discussed. Finally, the related work on semantic similarity measure is presented in Section 5.

2 Semantic Distributed Representation

Meanings of a word can be inferred from its surround contexts. Consider the following example describing the contexts of an unknown word "*tezgüino*" (the modified example from [9, 5]).

- A bottle of *tezgüino* is on the table.
- Mexican likes *tezgüino*.
- Strong *tezgüino* makes you drunk.
- We make *tezgüino* out of corn.

The contexts in which the word "*tezgüino*" is appeared suggest that the meanings of "*tezgüino*" may be a kind of alcoholic beverage that makes from "*corn*", gets people "*drunk*" and normally contains in "*bottle*". In other words, the meanings of a given word could be disclosed by considering its relational participants in local contexts such as "*bottle*", "*strong*", "*drunk*", and "*corn*". This intuitive idea is also the motivation for building the relation-based distributed representation.

2.1 *Meaning Representation Using Relational Word Features*

It has been confirmed that meanings of a word is determined by its surrounding contexts [3]. The surrounding contexts include relational associations between the word and others in contexts. While some relational associations hold the meanings over long distance, such as in the

pairs ("*tezgüino*", "*drunk*") and ("*tezgüino*", "*corn*"), others maintain the meanings when the word interacts with its adjacent neighbours, such as in the pairs ("*tezgüino*", "*strong*") and ("*tezgüino*", "*bottle*"). Given a word w_i, its semantic representation $v(w_i)$ is described as a sparse vector as follows:

$$v(w_i) = \langle w_i^1, w_i^2, \ldots, w_i^n \rangle \tag{1}$$

where w_i^k is the information value that reflects the degree of semantic association between the word w_i and its relational participant w_k. The parameter n is the size of word dictionary in the given text corpus. Futhermore, different corpus-based approaches come up with different information value measures. We used the point-wise mutual information (PMI) [10] to measure the degree of information value (association) between two different words in a relation. The information value w_i^k of the pair of words (w_i, w_k) is measured as follows:

$$w_i^k = \log \frac{p(w_i, w_k)}{p(w_i)p(w_k)} \tag{2}$$

$$p(w_i, w_k) = \frac{d(w_i, w_k)}{\sum\limits_{i,k=1\ldots n} d(w_i, w_k)} \tag{3}$$

$$p(w_i) = \frac{\sum\limits_{k=1\ldots n} d(w_i, w_k)}{\sum\limits_{i,k=1\ldots n} d(w_i, w_k)} \tag{4}$$

where $d(w_i, w_k)$ is the number of times that w_i and w_k co-occur in a relational association.

2.2 *Representation Using Latent Topic Features*

Word meanings have been successfully described using explicit topics such as Wikipedia concepts [11]. However, the method relies on the network structure of Wikipedia links, which hardily adapts to different domains as well as languages. In this work, we used the latent topics instead, which could be inferred using typical a generative topic model operated on a large plain text corpus. Several variants of topic model have been proposed such as Latent Semantic Analysis (LSA), and [1],

Latent Dirichlet Allocation (LDA) [2]. They are all based on the same fundamental idea that documents are mixtures of topics where a topic is a probability distribution over words, and the content of a topic is expressed by the probabilities of the words within that topic. On the task of semantic similarity measure, LDA has been confirmed for the better results than LSA [12]. In this work, we used LDA as the background topic model in building features for word representation. LDA performs the latent semantic analysis to find the latent structure of "topics" or "concepts" in a plain text corpus.

Given a focus word w_i and a latent topic t_j, the topic model produces the probability m_i^j that w_i belongs to the particular topic t_j. As the result, the topic representation of the word w_i is considered as a vector of latent topics, where each value of the vector is represented for the probability that w_i belongs to a particular topic t_j ($j = 1 \dots k$).

The topic representation of the word w_i is described as follows:

$$u(w_i) = \langle m_i^1, m_i^2, \dots, m_i^k \rangle \tag{5}$$

where k is the number of latent topics. The vector $u(w_i)$ is used to describe the meanings of the word w_i using latent topic information.

2.3 *Word Representation Using Combination of Relational Features and Latent Topic Features*

Given w_i as a focus word, meanings of the word w_i is represented as a n-dimensional vector $v(w_i)$ of relational words denoted $w_1 \dots w_n$ (see formula 1). Meanwhile, the focus word w_i is also represented as a k-dimensional vector $u(w_i)$ of latent topics denoted $t_1 \dots t_k$ (see formula 5). Therefore, the composition vector representation $c(w_i)$ of the word w_i is the linear concatenation of the relational feature vector $v(w_i)$ and the latent topic feature vector $u(w_i)$ as:

$$c(w_i) = \langle \alpha w_i^1, \dots, \alpha w_i^n, \beta m_i^1, \dots, \beta m_i^k \rangle \tag{6}$$

where n is the number of relational word features and k is the number of latent topics.

3 SEMANTIC WORD SIMILARITY

Our proposed content-based method of measuring semantic similarity was constructed using two different groups of features: relational words

in context and latent topics. These groups of features were tested separately and collectively. The following pre-processing steps were undertaken:

1. *Relation Extraction*: Relations surrounding words in contexts need to be extracted from a plain text repository. We designed a pattern-based extractor which single-passes through the plain texts and returns the extractions. Each extraction is a pair of a focus word and its relational participant, which have to match the following conditions:
   (a) The relational participant has to be a single noun, compound noun, or a name entity
   (b) If existed, the sequence in between the pair from the text has to match the following pattern:

   V+ | V+W*P | P

   where
   - V = (relative word | verb | particle | adverb)
   - W = (noun | adjactive | adverb | pronoun | determiner)
   - P = (preposition | particle | appositional modifier)

   These rules are expected to cover most of the local and long distance association between words in contexts.
2. *Word Representation*: Each focus word is represented by a set of relational participants. To reduce the number of relational associations, we retained those having considerable information value. Therefore, we applied a first filter on the relation frequency and a second filter on information value for each relation. There are three ways to construct the VSM of a word: relational feature VSM, latent topic feature VSM and combination feature VSM.
3. *Distance Measure*: To measure the semantic similarity between two words, we directly used the standard *Cosine* distance measure on the representation vectors. Given two words w_i and w_j, the semantic similarity between them is computed as:

$$sim(w_i, w_j) = \frac{v(w_i) \times v(w_j)}{\|v(w_i)\| \times \|v(w_j)\|} \tag{7}$$

4 Implementation Details

In this section we describe the implementation of our system.

Table 1. Experiment on MTruk for tuning parameters. The best Spearman's correlation score was obtained with $FF = 2$, $IVF = 1$, and $\frac{\alpha}{\beta} = \frac{1}{600}$ on both relational features and combination features. The related work's results on the same dataset was also presented. The knowledge-based methods are *italic*

Algorithm	$\rho \times 100$
Explicit Semantic Analysis [6]	59
Temporal Semantic Analysis [6]	**63**
Relational feature	**63**
Topic features	46
Combination Feature	**63**

4.1 *Benchmarks*

WordSimilarity-353 (WS-353) [7] dataset has been one of the largest publicly available collections for semantic similarity tests. This dataset consists of 353 word pairs annotated by 13 human experts. Their judgement scores were scaled from 0 (unrelated) to 10 (very closely related or identical). The judgements collected for each word pair were averaged to produce a single similarity score.

Several studies measured inter-judge correlations and found that human judgement correlations are consistently high $r = 0.88 - 0.95$ [13, 7]. Therefore, the outputs of computer-generated judgments on semantic similarity are expected to be as close as possible to the human judgement correlations.

Rubenstein and Goodenough dataset (RG-65) [8] consists of 65 word pairs ranging from synonym pairs to completely unrelated terms. The 65 noun pairs were annotated by 51 human subjects. All the noun pairs are non-technical words using scale from 0 (not-related) to 4 (perfect synonym).

MTurk[1] dataset contains 287 pairs of words [6]. Opposite to WS-353, a computer automatically draws the word pairs from words whose frequently occur together in large text domains. The relatedness of these pairs of words was then evaluated using human annotators, as done in the WS-353 dataset. We considered MTurk as a development dataset which was then used to find the range of optimal parameters. The selected parameters were tested on WS-353 and RG-65 datasets.

[1] http://www.technion.ac.il/∼kirar/Datasets.html

Table 2. The correlation results with different information value filter (IVF) tested on WS-353 dataset using Spearman's rank correlation (ρ). The best results were bolded. The results with underline were using parameters selected from the development dataset

IVF	Word features ($\rho \times 100$)	Combination features ($\rho \times 100$)	Topic features ($\rho \times 100$)
-3.0	60.58	62.97	66.78
-2.5	60.76	63.05	
-2.0	61.05	63.36	
-1.5	62.06	64.31	
-1.0	63.49	65.32	
-0.5	64.34	65.82	
0.0	63.73	65.07	
0.5	66.48	67.29	
1.0	**69.42**	70.19	
1.5	68.30	**70.79**	
2.0	64.60	70.12	
2.5	49.19	66.39	
3.0	26.93	55.94	

4.2 *Text Repository*

We used Wikipedia English XML dump of October 01, 2012. After parsing the XML dump[2], we obtained about 13GB of text from $5,836,084$ articles. As we expect to have a large amount of text data to increase the coverage of the method, we used first $1,000,000$ articles for our experiments.

To build the relational feature representation for each word, we applied the pattern-based extractor to extract pairs of the focus word and its relational participant. After the extraction, we obtained $53,653,882$ raw unique pairs which then were normalized by applying the stemming technique [14]. Finally, we obtained $47,143,381$ unique relations between words and their relational paritipants.

As there were a large number of rare words and pairs associated with each focus word, we applied two filters to leave out those we believed as noise. While the relation frequency filter (FF) is reponsible to remove rare relational pairs, the information value filter (IVF) is expected to leave out pairs with low information value. Any pair with their respective informa-

[2] We used Wikiprep as the main tool to convert Wikipedia format to XML plain text, http://sourceforge.net/projects/wikiprep/

Table 3. The correlation results with different information value filter (IVF) tested on 65 pairs of RG-65 dataset using Spearman's rank correlation (ρ). The best results were bolded. The results with underline were using parameters selected from the development dataset

IVF	Word features ($\rho \times 100$)	Combination features ($\rho \times 100$)	Topic features ($\rho \times 100$)
-3.0	73.64	72.47	63.93
-2.5	73.62	72.25	
-2.0	73.74	72.58	
-1.5	74.50	72.91	
-1.0	75.25	73.96	
-0.5	76.65	75.89	
0.0	77.12	76.53	
0.5	77.63	77.09	
1.0	79.72	79.16	
1.5	84.11	83.59	
2.0	**84.43**	**84.59**	
2.5	78.46	83.33	
3.0	59.64	79.88	

tion is equal or above the filter's threshold will be retained to contruct the representation of words.

To extract latent topic features, we used plain texts from the first $100,000$ Wiki documents to feed to LDA training model. The reasons for us to choose this smaller amount of documents as LDA training phrase was time consuming with large amount of documents. We expected to reduce the number of input documents and kept the word dictionary relatively large to cover most of the expected words. The plain text from these documents was removed stop-words and applied the stemming technique. Rare words were also removed by using document frequency threshold ($df = 5$). We obtained $190,132$ unique words from the given set of documents after pre-processing step. To build the LDA training model, we used GibbsLDA++[3] [15] with its standard configuration except $ntopic = 1,000$ as the number of expected latent topics.

Parameter Tuning: The MTruk dataset was used for parameter tuning. We evaluated our method using relational features, topic features, and combination features. After scanning the FF and IVF parameters as well as the $\frac{\alpha}{\beta}$ ratio on this dataset, we obtained the best Spearman's corre-

[3] http://gibbslda.sourceforge.net

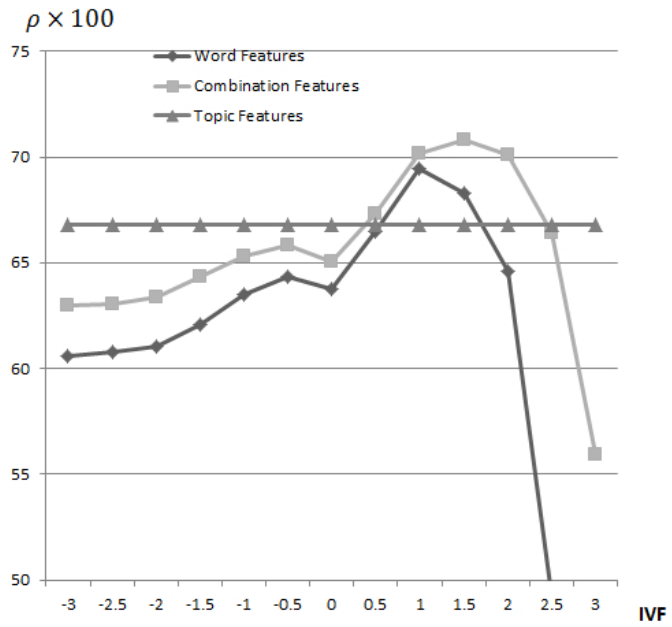

Fig. 1. The visualisation of experimental results from WS-353 dataset (see Table 2). The combination feature-based method outperformed the one using word features regardless IVF.

lation score $\rho = 63$ on both relational features and combination features with $FF = 2$, $IVF = 1$, and $\frac{\alpha}{\beta} = \frac{1}{600}$. Table 1 shows the results when the selected parameters were applied as well as the results of other related methods that have been tested on the same dataset. These tuning values were used when testing on WS-353 and RG-65 datasets.

4.3 *Evaluation*

In this section [4], we firstly discuss about the effectiveness of our method over different of standard datasets. Table 2 and 3 show the results of our experiments over three kinds of features. Overall, the method based on relational features outperformed those using topic features on WS-353

[4] The experimental results can be found at http://137.92.33.34/CICLING2014/

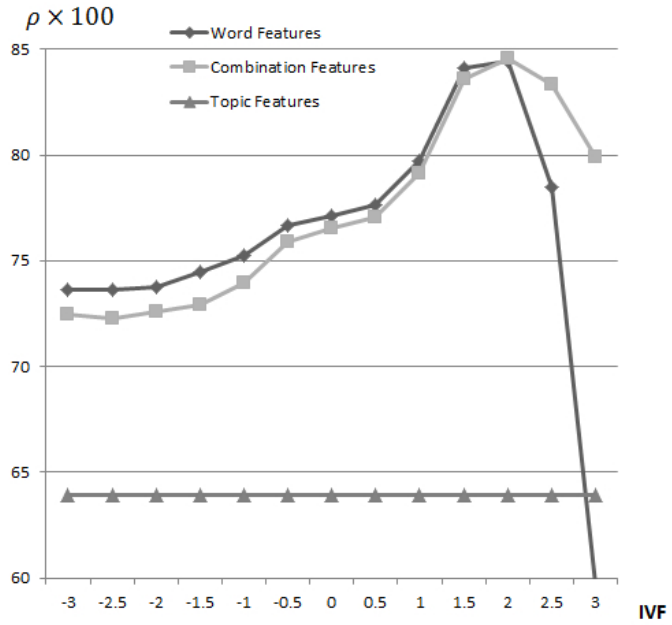

Fig. 2. The visualisation of experimental results on RG-65 dataset (see Table 3). The method using combination features is comparable the one based on word features.

dataset (69.42 vs. 66.78) and on RG-65 dataset (84.43 vs. 63.93). Particularly, when the relational features are combined with topic features in a single VSM, the performance of the combination method was improved in comparison with those using the type of features separately.

Moreover, Table 2 and 3 also confirms that the selected parameters from the development dataset potentially work really well on the WS-353 and RG-65 datasets. They produced significant improvement compared to those using the similar kinds of features (see Table 4).

It is notable to compare the performance of the proposed method to other related work on the same benchmarks. On the standard WS-353 dataset, our method outperforms to most of the semantic similarity methods using single VSM for word representation. Compare to other corpus-based methods in general, our approach also achieves the second high-

Table 4. The comparison results with different content-based methods on WS-353 and RG-65 datasets using Spearman's rank correlation (ρ). The knowledge-based methods are in italic. (†) denotes using parameters from the deveopment dataset. ($\star$) denotes the best results in our experiments

Algorithm	WS-353 ($\rho \times 100$)	RG-65 ($\rho \times 100$)
Syntactic Features [5]	34.80	78.8
Latent Topic Features (LSA) [7]	58.10	60.9
Latent Topic Features (LDA) [12]	53.39	–
Multi-Prototype [16]	**76.9**	–
Single-Prototype [16]	55.3	–
Multi-Prototype [17]	71.3	–
Learned Features [18]	49.86	–
Context Window Pattern ($\mathrm{WS} = 1$) [4]	69	89
Context Window Pattern ($\mathrm{WS} = 4$) [4]	66	**93**
Topic Features	66.78	63.93
Relational Features†	69.42	79.72
Combination Features†	70.19	79.16
Relational Features*	69.42	84.43
Combination Features*	70.79	84.59

est correlation score on this dataset after the multi-prototype VSM done by [16].

Additionally, the proposed method achieves the promising performance on RG-65 dataset on both word features and combination features. Interestingly, the topic feature-based method on Wikipedia outperforms to most of the other latent topic feature-based methods such as LSA and LDA on both WS-353 and RG-65 datasets.

Finally, in comparison to the work done by [5], one of the closest approaches to our work in term of feature engineering, the proposed method outperformed on both WS-353 and RG-65 datasets.

5 Related Work

Previous work in the field of semantic similarity is categorized as corpus-based and knowledge-based approaches. While the corpus-based methods utilize statistical techniques to measure the similarity between words using the pure text content of a given corpus, the knowledge-based approaches explore the embedded knowledge from a large repository such as Wordnet, networks of concepts from Wikipedia.

VSMs are mostly used to model the meaning of words. In frame of knowledge-base approaches, Gabrilovich et al. have proposed Explicit Semantic Analysis (ESA) [11], which represents word meanings as a vector of explicit Wikipedia concepts. The relatedness between words is measured by the distance between the respective vectors. Silent Semantic Analysis (SSA) was proposed by Hassan et al. [19]. SSA explores Wikipedia silent concepts which were then incorporated with the explicit Wikipedia concepts to model the word representation using VSMs.

One of the main differences between these methods and our approach is the way of estimating the degree of association between words. In ESA and SSA, word-word relations are defined indirectly using their relationship with Wikipedia concepts. However, the relation between words in our approaches is defined directly using the common relational participants within local contexts as well as their common latent topics.

In contrast to the knowledge-based methods, the content-based methods rely only on plain text. Latent Semantic Analysis (LSA) [1] was proposed to take into account word-document associations to present the semantic representation of words. LSA considers meanings of a word as a vector of latent topics and the similarity between words is measured by the distance of its represented vectors. Similarly, topic model Latent Dirichlet Allocation (LDA) [12] was used to to measure word semantic similarity. The fundamental idea that documents are mixtures of topics where a topic is a probability distribution over words. The similarity of words could be inferred by the associated of their common topics.

Agirre et al. used word patterns in context windows as the features. The method produced promising correlation results in RG-65 dataset and considerable results on WS-353 dataset with Windowsize (WS=1 and WS=4) [4]. Lin et al. [5] measured the similarity between words using the distributional lexical and syntactic patterns of words over a parsed corpus. The similarity between a pair of words was measured by the common between their distributions. The idea of feature engineering in this work is quite similar to our approach that using the local contexts to extract relations between words.

However, while these authors considered syntactic associations between a focus word and its adjacent words to produce the word's representation. We combined relational features and topic features to form a representation of words. Moreover, to reduce the influences of the noise in the semantic similarity measure, we applied different filters to retain information valuable relations. This has contributed to leverage the performance of our proposed method.

Recent work on feature learning has opened a new way of building word semantic representation automatically from the nature of language. Collobert et al. [18] proposed a deep learning framework for automatically building word meaning representations (word embeddings). Huang et al. [17] have successfully inherited the word embeddings to learn multiple word prototypes (multiple VSM represented for meanings of a word), which show the promising results on the task of semantic similarity. Similarly, Reisinger et al. [16] have proposed multi-prototype VSM for word meaning representation using text clustering. The method presents significant improvement performance on semantic similarity measure. However, they also confirmed that single word prototype is still having issues in gaining the performance of content-based semantic similarity measure.

6 CONCLUSION

We have presented an approach for semantic similarity measure using relational features and topic features. The method takes into account the relations between words in local contexts and latent topics information from global contexts. The experimental results have shown the positive contribution of relational features and topic features to the performance of corpus-based methods. Especially, their combination in modelling word representation yields the improvement results to most of the content-based methods on both tested datasets.

REFERENCES

1. Dumais, S.T.: Latent semantic analysis. Annual review of information science and technology **38**(1) (2004) 188–230
2. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent Dirichlet allocation. The Journal of Machine Learning Research **3** (2003) 993–1022
3. Turney, P.D.: Mining the web for synonyms: PMI-IR versus LSA on TOEFL. In: Proceedings of the 12th European Conference on Machine Learning. (2001) 491–502
4. Agirre, E., Alfonseca, E., Hall, K., Kravalova, J., Paşca, M., Soroa, A.: A study on similarity and relatedness using distributional and WordNet-based approaches. In: Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Association for Computational Linguistics (2009) 19–27

5. Lin, D.: An information-theoretic definition of similarity. In: ICML. Volume 98. (1998) 296–304
6. Radinsky, K., Agichtein, E., Gabrilovich, E., Markovitch, S.: A word at a time: Computing word relatedness using temporal semantic analysis. In: Proceedings of the 20th International Conference on World Wide Web, ACM (2011) 337–346
7. Finkelstein, L., Gabrilovich, E., Matias, Y., Rivlin, E., Solan, Z., Wolfman, G., Ruppin, E.: Placing search in context: The concept revisited. In: Proceedings of the 10th International Conference on World Wide Web, ACM (2001) 406–414
8. Rubenstein, H., Goodenough, J.B.: Contextual correlates of synonymy. Communications of the ACM **8**(10) (1965) 627–633
9. Nida, E.A.: Componential analysis of meaning. Mouton The Hague (1975)
10. Dagan, I., Marcus, S., Markovitch, S.: Contextual word similarity and estimation from sparse data. In: Proceedings of Association for Computational Linguistics, Association for Computational Linguistics (1993) 164–171
11. Gabrilovich, E., Markovitch, S.: Computing semantic relatedness using Wikipedia-based explicit semantic analysis. In: IJCAI. Volume 7. (2007) 1606–1611
12. Dinu, G., Lapata, M.: Measuring distributional similarity in context. In: Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics (2010) 1162–1172
13. Budanitsky, A., Hirst, G.: Evaluating WordNet-based measures of lexical semantic relatedness. Computational Linguistics **32**(1) (2006) 13–47
14. Van Rijsbergen, C.J., Robertson, S.E., Porter, M.F.: New models in probabilistic information retrieval. Computer Laboratory, University of Cambridge (1980)
15. Phan, X.H., Nguyen, L.M., Horiguchi, S.: Learning to classify short and sparse text & web with hidden topics from large-scale data collections. In: Proceedings of the 17th International Conference on World Wide Web, ACM (2008) 91–100
16. Reisinger, J., Mooney, R.J.: Multi-prototype vector-space models of word meaning. In: Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Association for Computational Linguistics (2010) 109–117
17. Huang, E.H., Socher, R., Manning, C.D., Ng, A.Y.: Improving word representations via global context and multiple word prototypes. In: Proceedings of Association for Computational Linguistics, Association for Computational Linguistics (2012) 873–882
18. Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., Kuksa, P.: Natural language processing (almost) from scratch. The Journal of Machine Learning Research **12** (2011) 2493–2537
19. Hassan, S., Mihalcea, R.: Semantic relatedness using salient semantic analysis. In: AAAI. (2011)

DAT HUYNH
UNIVERSITY OF CANBERRA,
BRUCE, ACT 2617, AUSTRALIA
E-MAIL: <DAT.HUYNH@CANBERRA.EDU.AU>

DAT TRAN
UNIVERSITY OF CANBERRA,
BRUCE, ACT 2617, AUSTRALIA
E-MAIL: <DAT.TRAN@CANBERRA.EDU.AU>

WANLI MA
UNIVERSITY OF CANBERRA,
BRUCE, ACT 2617, AUSTRALIA
E-MAIL: <WANLI.MA@CANBERRA.EDU.AU>

DHARMENDRA SHARMA
UNIVERSITY OF CANBERRA,
BRUCE, ACT 2617, AUSTRALIA
E-MAIL: <DHARMENDRA.SHARMA@CANBERRA.EDU.AU>

IJCLA VOL. 5, NO. 1, JAN-JUN 2014, PP. 27–42
RECEIVED 07/01/14 ACCEPTED 06/02/14 FINAL 16/06/14

Joint word2vec Networks for Bilingual Semantic Representations

LIOR WOLF, YAIR HANANI, KFIR BAR, AND NACHUM DERSHOWITZ

Tel Aviv University, Israel

ABSTRACT

We extend the word2vec framework to capture meaning across languages. The input consists of a source text and a word-aligned parallel text in a second language. The joint word2vec tool then represents words in both languages within a common "semantic" vector space. The result can be used to enrich lexicons of under-resourced languages, to identify ambiguities, and to perform clustering and classification. Experiments were conducted on a parallel English-Arabic corpus, as well as on English and Hebrew Biblical texts.

1 INTRODUCTION

Semantic models of languages map words, or word senses, in the language under study to some space in which semantic relationships between them can be observed and measured. Within such models, one would expect synonymous words to appear "near" one another. If the model incorporates more than one language, then the translation of a word should also reside nearby.

Word2vec [1] is a recently developed technique for building a neural network that maps words to real-number vectors, with the desideratum that words with similar meanings will map to similar vectors. One problem with a straightforward learning scheme from words and their context is polysemy, since the resultant vector will be an average of the different senses of the word.

Some languages are more ambiguous than others. There are many homographs in English, fewer in Spanish (a more phonetic language than English), and many more in (unvocalized) Hebrew. Sometimes homographs are also homophones, being pronounced the same (e.g., "bank"); other times they are heteronyms that are distinguished in speech (e.g., "dove"). We show how to leverage word-aligned parallel texts to improve the quality of the inferred model.

In the next section, we provide some background on the use of bilingual models for single-language tasks and to aid in machine translation. Section 3 contains an explanation of our bilingual extension to the word2vec model builder. It is followed by a section on experimental results for word-aligned English-Arabic and English-Hebrew corpora and then a brief discussion of the implications.

2 BACKGROUND

Word sense disambiguation [2] is a difficult problem, partly because of the paucity of sense-tagged training data in most languages. Languages like Arabic and Hebrew pose a bigger challenge because of their unvowelized and undiacritized writing. In English and other European languages, senses are represented, in particular, by WordNet [3]. Although similar WordNet repositories exist for Arabic [4] and Hebrew [5], they are quite limited in scope. The only available sense-tagged corpus in Hebrew is the Bible. For Arabic, there is OntoNotes [6], a relatively large corpus of various genres, tagged for syntax and shallow semantic attributes, including word senses.

It is well known that there is little agreement between dictionaries as to the division of words into their senses. Some distinctions are subtle and often cross language boundaries, so may not be considered distinct senses by other lexicographers; others are substantive and are unlikely to share the same words in multiple languages. Accordingly, it has been argued (e.g., [7–9]) that sense distinctions can be derived from co-occurrence statistics across languages. To quote [2]: "Homograph distinctions do not require a lexicographer to locate them, since they are basically those that can be found easily in parallel texts in different languages".

In [10], the authors showed how one can train word2vec independently in two languages, then use some anchor points between the two languages (a limited number of corresponding words in the two languages) to find an affine mapping from one language model to the other.

The hope was that the resultant spaces can be used to enrich the dictionary by looking in the model of the second language for the word closest to the position of a word from the first language.

3 WORD2VEC

Word2vec belongs to the class of methods called "neural language models". Using a scheme that is much simpler than previous work in this domain, where neural networks with many hidden units and several nonlinear layers were normally constructed (e.g., [11]), word2vec [1] constructs a simple log-linear classification network [12]. Two such networks are proposed: the Skip-gram architecture and the Continuous Bag-of-words (CBOW) architecture. While the Skip-gram architecture is sometimes preferable, in our experience – probably due to the relatively small corpora we use, the CBOW architecture outperforms it. Therefore, we limit our exposition to the latter architecture.

3.1 *Continuous Bag-of-Words Architecture*

In the CBOW variant of word2vec, the network predicts each word based on its neighborhood – the five words preceding and the five words following that word. An input layer denotes the bag of words representation of the surrounding words, and contains one input element per glossary word. It is projected linearly to the hidden encoding layer. The hidden layer in then mapped to an output Huffman code representation of the given word. Once the network is trained, the projections from each input unit to the middle encoding layer are used to represent the associated glossary word. Interestingly, the resulting encoding not only captures meaningful word representations, where (unambiguous) words of similar meaning have nearby representations, but also captures, in a surprising manner, pairwise relations through simple arithmetic operations [1].

Next, we formalize the CBOW architecture and introduce the notation used throughout the paper. Word2vec models are learned from an input corpus. Every word that occurs at least three times in the corpus becomes part of the glossary. Let D be the size of the glossary. The goal of the word2vec procedure is to associate each of the D glossary words with a vector representation in $\mathbb{R}^L$. In all of our experiments, L is fixed at a value of 200.

Each training example i is an occurrence of a glossary word $w(i)$ in the corpus. During the learning phase, the CBOW network is trained to

map (through a hidden projection layer of size L) binary input vectors B_i^{in} of length D, to the matching output vectors B_i^{out} of length $\lg D$. The input vectors contain one element corresponding to each glossary entry, which denotes the existence of each glossary entry within the small neighborhood of size $2N$ surrounding example i in the text. In our experiments the window-size parameter N is fixed to be 5.

The binary output vector $B_i^{\text{out}} = H_{w(i)}$ is the Huffman code of the associated glossary entry. The Huffman codes H_j, $j = 1 \,..\, D$ of length $\lg D$ are constructed by considering the frequency of each glossary entry j. The use of Huffman codes follows a common practice in neural net language models [13]. We adhere to this practice, although its main advantage over encoding through balanced trees is to reduce evaluation time through an early stopping mechanism. This consideration is not a major concern in our system, where D is relatively small.

As mentioned above, the vector representation V_j of each glossary word, $j = 1 \,..\, D$, is its projection to the hidden layer; that is, it is the array of weights used to project the corresponding bit in the input vector B^{in} to the hidden layer. Intuitively, it captures the contribution of each glossary word to a layer that enables a relatively effective way of predicting the output vector. This is a semantic task, and the representation is therefore semantic. Also, by design, since linear projections are summed to capture local neighborhoods, the representation behaves well under addition.

3.2 *Bilingual Extension*

Very recently, a method was proposed for employing word2vec in the bilingual context [10]. The proposed method trains two word2vec models independently and then, using a seed list of pairs of words and their translation, the two spaces are aligned.

More specifically, assume that word w_k in the first language is translated as word w'_k in the second, $k = 1 \,..\, t$, where t is size of the seed set. Let the vectors $V_1, V_2, \ldots, V_D$ be the learned word2vec representations in the first language, and $V'_1, V'_2, \ldots, V'_{D'}$ the learned vectors of the second. A transformation T is then learned from the space of the second representation to the space of the first, such that the following measure is minimized:

$$\sum_{k=1}^{t} \left\| T V'_{w'_k} - V_{w_k} \right\|^2 .$$

This is a linear least squares problem. In [10] stochastic gradient descent was used to solve for the transformation T. In our experiments we employ conventional linear solvers.

4 JOINT WORD2VEC

The bilingual method of [10] is motivated by the observation that when visualizing the word vectors using PCA, the vector representations of similar words in different languages were related by a linear transformation. However, since there is no guarantee that the two spaces are indeed related by a linear transformation, it could be suboptimal to learn each space separately and then try to align the two. In addition, if the corpora used to train the two vector spaces are related, it might be beneficial to use one language to improve the representation of the other, for example, in cases of word sense ambiguities.

In our method, *joint word2vec*, two versions of the same text are used for training. One of the versions is set to be the baseline version, and the other is aligned to it using an HMM-based word-alignment model [14]. The result, for a specific text (which will be used in our next examples as well) is illustrated in Fig. 1. In this example, a Hebrew verse from Genesis is used as the baseline text, and the English King James translation is mapped to it. Some of the words in the English text are not mapped to any Hebrew words and vice versa.

The joint word2vec model is a variant of the word2vec model with some significant modifications in both structure and training procedure that allow for simultaneous learning of unified representations for both languages. The input layer of the joint model corresponds to a union of the two dictionaries. Let the glossary of the first (second) language, consisting of frequent-enough words, be of size D (D', respectively). The input layer of the joint architecture is of size $D + D'$, where the first D (last D') elements correspond to the entries of the first (second) glossary. The output layer is of size $\lg(D + D')$, and encodes the words of the two dictionaries based on their frequencies in the bilingual corpora. The hidden layer and the log-linear model structure remain as in the conventional word2vec model.

The training procedure is shown in Fig. 2. It consists of four stages for each input occurrence i of the first language. First, as in the conventional word2vec, the neighborhood of i (first language) is used to predict word i. Then, the same process is repeated for the second-language word that is aligned to i, using the second-language words that are aligned with the

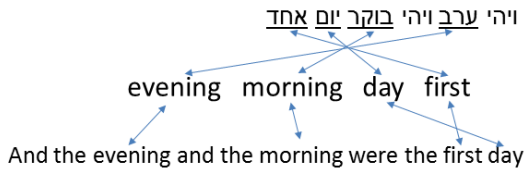

Fig. 1. A short alignment example (Genesis 1:5), which is also used in Fig. 2. The Hebrew verse (top row) is the baseline text to which the English verse at the bottom row is aligned. The words in the middle are those words that were found by the HMM model to match the Hebrew words they are connected to with an arrow.

first-language neighborhood of i. Naturally, this only occurs when there is a second-language word aligned with word i. Also, the aligned neighborhood could be smaller than the first-language neighborhood since not all first-language words are mapped to the second language.

These first two stages train two models using the same hidden layer. However, the two models are not tied together: each is using only the part of the input vectors B^{in} that corresponds to just one language. The next two stages tie the models together at the semantic level. In these stages, a word from one language is being predicted using the neighborhood in the other language, which helps make the model language-agnostic. First, the aligned word of the second language is predicted based on the neighborhood of word i in the first language. The input vector B^{in} is derived from data in the first language, and the output vector B^{out} is derived from second language data. Lastly, the process is reversed: the aligned second language neighborhood is used to predict and output vector B^{out} that is the Hoffman coding of word i in the first language.

Since training of the joint word2vec is four times slower than training of the conventional method, we modify the training procedure slightly. Whereas the original method employs stochastic gradient descent, which updates the network after observing each training samples, we introduced "mini-batches" and update the weights only after a small number of training examples. In between updates, the gradient is accumulated but not used to perform updates.

5 Evaluation

We compare the performance of the proposed joint word2vec to the performance of conventional word2vec and the performance of the bilingual

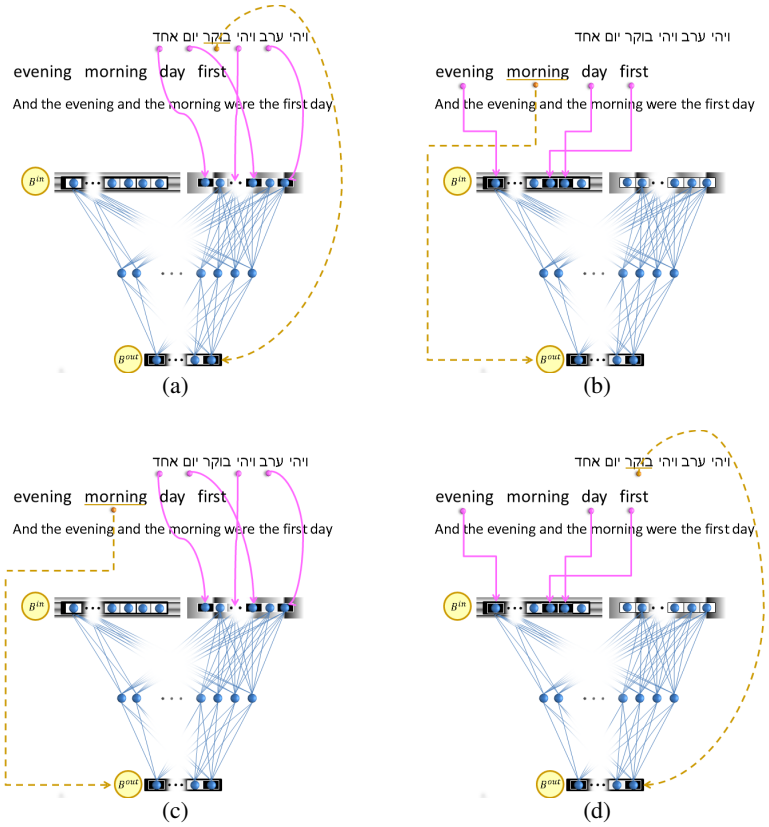

Fig. 2. An illustration of the joint word2vec network applied to a sample Hebrew/English verse during training. The input layer has two parts, corresponding to the two vocabularies. During training there are four steps (a–d): (a) Word i of the Hebrew text is predicted, that is, encoded at the output layer B_i^{out} in accordance to its Huffman encoding $H_{w(i)}$. The Hebrew neighborhood of word i is encoded through a bag-of-words scheme as the input layer B_i^{in}. The hidden layer is a linear projection layer and the entire architecture is a simple log-linear one. (b) Using the same architecture and the same hidden units, the process is repeated for predicting the English word that corresponds to the Hebrew word i from the English words that are mapped to the neighborhood of i. Note that this neighborhood typically differs from the neighborhood in the English text and is often smaller than the Hebrew neighborhood since some Hebrew words are not aligned to any English words. (c) The Hebrew neighborhood is used to predict the English word that matches word i. (d) The English neighborhood (see (b)) is used to predict the Hebrew word i.

model of [10] (Sect. 3.2). Given two texts, where the second is aligned to the first, the transformation of this bilingual method is estimated using all the words of the first text and their alignment, when they exist. Each pair is used once, regardless of the number of its occurrences.

Three benchmarks have been devised: (i) a *translation* benchmark that looks to see how close to each other a word and its translation are; (ii) a *homograph* benchmark; and (iii) a single-language *synonym* benchmark. The benchmarks are applied on two datasets, except for the third task, which is only applied to one dataset.

5.1 *Datasets*

In our experiments, as a second language, we used (transliterated) Arabic and Hebrew – both highly inflected languages. Among other interesting characteristics, Semitic languages in general are based on complicated derivational as well as inflectional morphologies. Furthermore, the lack of short vowels in writing increases the level of ambiguity of the written word. Words are derived from a root and a pattern (template), combined with prefixes and suffixes. The root consists of 3 or 4 consonants and the pattern is a sequence of consonants and variables for root letters. Using the same root with different patterns may yield words with different meanings. Words are then inflected for person, number and gender; proclitics and enclitics are added to indicate definiteness, conjunction, various prepositions, and possessive forms. On account of this morphological complexity, a single Arabic or Hebrew word often translates into several English words; for example, the English translation of the Arabic word *wbbytyn* is "and in his two houses".

Our first benchmark uses Arabic news stories that we aligned on the word level with their English sources. Overall, we have about 4M Arabic words. Due to Arabic's rich morphology, we preprocessed the Arabic text with MADA [15], a context-sensitive morphological analyzer that works on the word level, and then used TOKAN [15] to tokenize the text, additionally separating the definite article (*Al*) from nouns (and modifiers) and the future particle (*s*) from verbs. For example, the word *wbbyth*, "and in his house", gets tokenized like this: *w+ b+ byt +h*, each token respectively translated into: "and", "in", "house", and "his". Alignment proceeds on this token level.

The second dataset was constructed by aligning the Bible, in Hebrew, to its King James translation into English. The two resources are already aligned at the verse level, and contain 23,145 verses. The Hebrew text,

like in Arabic, was tokenized using a context-sensitive morphological analyzer and a tokenizer. We used the MILA [16] morphological analyzer, with probabilities obtained using EM-HMM training [17, 18]. The Hebrew tokenization scheme is similar to the one for Arabic. Both resources were aligned on the token level using GIZA++ [14], an implementation of the IBM word alignment models [19]. GIZA++ is a popular statistical machine translation toolkit that includes an HMM-based word-alignment model. It uses Baum-Welch training, and includes smoothing techniques for fertility (multiple-word translations of a single word).

For the synonym benchmark, we obtained the list of Biblical synonyms used in [20]. That list was compiled in the following manner: The King James translation of the Bible almost invariably translates synonyms identically. So, one can generally identify Hebrew synonyms by considering their translations. Word senses were derived from Strong's 1890 concordance [21], which lists separately every occurrence of each sense of each root that appears in the Bible. (Some anomalies due to polysemy in English were manually removed from the list by a Bible scholar.) The procedure yielded 529 synonym sets, ranging in size from 2 to 7 ("fear"), for a total of 1595 individual synonyms.

5.2 *Experiments*

The translation benchmark For each of the two datasets we identify pairs of glossary words that are uniquely mapped between the two texts, that is, we extract a list of words in the first text that are consistently aligned by GIZA++ to the same word in the second. Ideally, the vector representations of the words in each pair would be as close as possible; indeed the bilingual method of [10] directly minimizes the distance between the representations of words that are mapped between the two languages.

The distance computed in the first benchmark is measured for our method in the joint space, and in the method of [10] in each one of the spaces, after the computed linear transformation is applied. In word2vec experiments [1], cosine similarity (dot product of the normalized vectors) is often used. We comply with this by computing the distance between the normalized vectors, which also makes the comparison of distances between the various vector spaces valid.

The results of this benchmark are depicted in Table 1. Sometimes the joint word2vec outperforms the baseline algorithm and sometimes it is the other way around. This is remarkable given that the score that the

Table 1. The performance of various word2vec networks in the translation benchmark, in which distances of normalized vector representations of word pairs that are uniquely aligned between the two languages are compared. For the Bible (Heb) Dataset, the English text is aligned to the Hebrew one, and vice versa for Bible (Eng). Since the alignment is not symmetric, the pairs of words used for benchmarking differ between the two directions. The baseline algorithm [10] (Sect. 3.2) runs conventional word2vec on each corpus separately and then estimates a linear transformation that minimizes the L2 distance between the vector representations of aligned words. It is evaluated for both alignment directions. The joint word2vec algorithm is applied to both corpora at once. Shown are the mean distance between vector pairs normalized to have a norm of one and the standard error of these distances. A lower distance indicates greater proximity of the vector representations of translated words.

Data-set	Method	Normalized Distance $\pm$ Standard Error
News	2$\times$Vanilla W2V (Eng and Arab mapped to Eng space)	1.1534 ± 0.0000
	2$\times$Vanilla W2V (Arab and Eng mapped to Arab space)	1.1505 ± 0.0000
	Joint W2V (Eng+aligned Arab)	1.1644 ± 0.0000
Bible	2$\times$Vanilla W2V (Heb and Eng mapped to Heb space)	0.9739 ± 0.0002
(Heb)	2$\times$Vanilla W2V (Eng and Heb mapped to Eng space)	1.0066 ± 0.0002
	Joint W2V (Heb+aligned Eng)	0.9710 ± 0.0002
Bible	2$\times$Vanilla W2V (Eng and Heb mapped to Eng space)	0.8900 ± 0.0005
(Eng)	2$\times$Vanilla W2V (Heb and Eng mapped to Heb space)	0.8543 ± 0.0005
	Joint W2V (Eng+aligned Heb)	0.8790 ± 0.0006

baseline algorithm optimizes is directly related to the score employed in the benchmark and that the baseline algorithm is applied twice.

The homograph benchmark We now consider the words in one language that are mapped to multiple words in the second language. This can be (i) a result of synonyms in the second language, or, (ii) as is almost always the case when aligning the Hebrew Bible to the King James edition, a result of homographs in the first language. For the first case, we would like to have the vector representation of the word in the first space as close as possible to that of the vector representations in the second space, and the vector representations in the second space as close as possible. In the second case, since the model is additive, it makes sense to expect the vector representation of the word in the first language to be a linear combination of the vector representations of the vectors in the second language, that is, to be spanned by those vectors.

Since we have no means to determine which of the two cases is the source of the multiple glossary word alignments, and since the spanning criteria is also approximately satisfied in case two vectors are similar, this criteria is the basis of our success score. Let V_i be the vector representation of the word in the first language, and let $V_i^{'(k)}$, $k = 1..n$ be the $n > 1$ words that are aligned to it throughout its occurrences in the dataset. The score employed in this benchmark is the reconstruction error:

$$\left\| \frac{V_i}{\|V_i\|} - \sum_{k=1}^{n} \lambda_k V_i^{'(k)} \right\| ,$$

where the λ_k are the coefficients that minimize this error. The minimization is obtained by solving (least squares) a linear set of $L = 200$ equations in these n unknowns. Note again, that in order to adhere to previous work with word2vec, and to allow a fair comparison between different vector spaces, the normalized version of V_i is used.

Table 2 depicts the results of this benchmark. The joint word2vec method clearly outperforms the baseline method, and the difference in performance is significant at an extremely low p-value in both t-test and Wilcoxon signed rank test. We also report results for another variant of the reconstruction error in which the weights λ_k are constrained to be positive. This is motivated by the nature of the word2vec architectures, in which, during training, projections are added but never subtracted. Naturally, the added positiveness constraint increases the reconstruction error, but this increase seems to be moderate, and in our experiments the order of the scores obtained by the various algorithms does not change.

The synonym benchmark Lastly, we evaluated the quality of the learned word2vec representation on the Hebrew list of biblical synonyms. Eighty percent of the identified synonym pairs were included in our glossary, which contains only words that appear at least three times. For each of these we employ the distance between the normalized vector representations of the two words as the score.

This is a single language task and joint word2vec would outperform the baseline method only if learning jointly two languages successfully utilizes the information in the second language to improve the vector space of the first. The results in Table 3 suggest that this is indeed the case. A significant improvement is obtained in the joint word2vec method in comparison to the baseline method.

Table 2. The performance of various word2vec networks in the homograph benchmark, in which the glossary words in the first language that are aligned to multiple words in the second language are employed. The reconstruction error is the distance between the L2 normalized vector representation in the first language to the linear space spanned by the vector representations in the second language; lower is better. Also shown is the reconstruction error when the combination weights λ_k used to compute the linear combination are constrained to be positive. Due to the additional constraint, the reconstruction errors are aways bigger; however, qualitatively the results remain the same. See text and the caption of Table 1 for more details.

Data-set	Method	Reconstruction Error $\pm$ Standard Error	Reconstruction Error $\pm$ Standard Error (Positive Weights)
News	2$\times$Vanilla W2V (Eng + transformed Arab)	0.9387 ± 0.0000	0.9469 ± 0.0000
	2$\times$Vanilla W2V (Arab + transformed Eng)	0.9321 ± 0.0000	0.9388 ± 0.0000
	Joint W2V (Eng+aligned Arab)	0.8893 ± 0.0000	0.9000 ± 0.0000
Bible (Heb)	2$\times$Vanilla W2V (Heb and Eng mapped to Heb space)	0.8063 ± 0.0001	0.8477 ± 0.0001
	2$\times$Vanilla W2V (Eng and Heb mapped to Eng space)	0.8410 ± 0.0001	0.8797 ± 0.0000
	Joint W2V (Heb+aligned Eng)	0.7462 ± 0.0001	0.7613 ± 0.0001
Bible (Eng)	2$\times$Vanilla W2V (Eng and Heb mapped to Eng space)	0.9033 ± 0.0002	0.9228 ± 0.0002
	2$\times$Vanilla W2V (Heb and Eng mapped to Heb space)	0.8173 ± 0.0004	0.8696 ± 0.0004
	Joint W2V (Eng+aligned Heb)	0.6610 ± 0.0006	0.6896 ± 0.0006

6 DISCUSSION

In the experiments we conducted, both datasets are relatively small compared to the sizes typically employed to learn representations in word2vec models. Larger bilingual datasets are available; however, the processing required by the GIZA++ software for these datasets could not fit into our schedule. Fortunately, the joint word2vec software itself proved efficient enough and scales equally well as the open-source word2vec, upon which code base it is built. The up-side is that our experiments are the first to explore the utilization of word2vec on a scale possible for under-resourced languages, in which digitized texts and translations are scarce. It is in-

Table 3. The performance of various word2vec networks on the Bible-Synonyms benchmark. The distance between the normalized representations of each pair of Hebrew synonyms is computed. As can be seen, the joint word2vec outperforms the baseline method, although this is a single-language task. The results in the two last rows are added for completeness, as they are expected to be worse. It is interesting to note that even learning joint word2vec with Hebrew aligned to English, which processes the Hebrew text in a way that is likely to disrupt its structure, seems to outperform the baseline method.

Method	Normalized Distance $\pm$ Standard Error
Vanilla W2V (Heb)	1.1864 ± 0.0013
Joint W2V (Heb+aligned Eng)	1.0637 ± 0.0015
$2\times$Vanilla W2V (Heb mapped to Eng space)	1.2396 ± 0.0009
Joint W2V (Eng+aligned Heb)	1.1073 ± 0.0015

teresting that, at least at these scales, joint word2vec representations outperform vanilla word2vec learned vectors even at tasks requiring just one language.

While our exposition is focused on the bilingual case, it should be straightforward to generalize it to multiple languages. In fact, with little modifications, the joint word2vec system can employ heterogeneous inputs to learn a unified "Babel fish" semantic vector representation for a union of multiple dictionaries. During training, the weights can be updated from unilingual texts (conventional word2vec), bilingual texts (as done here), or aligned multi-lingual texts, each time updating the projection weights of the current word with accordance to the available data.

As an effective method to learn representations, word2vec is often seen as part of the "deep learning" trend, in which neural networks containing multiple layers and multiple non-linear transformations are used to learn state-of-the-art representations in domains such as computer vision and speech processing. While there is nothing deep in the word2vec architecture, by comparing it to deeper representations, the creators of the original architecture have been able to demonstrate that the simplest log-linear models are "as deep as is necessary", thereby complying with the road plan set forth by [22].

Still, the word2vec representation is wanting in some respects. For example, it would be desirable that homographs would be represented, not by one vector, but by a set of vectors. In the bilingual context, the automatic identification of homographs seems plausible. Recall that words

in one language that are aligned with multiple words in the second language are either homographs or are translated to a set of synonyms. As discussed in Sect. 5.2, the set of synonyms should form a tight cluster, which can be easily measured, for example, by computing the mean distance to the centroid of the set of vectors representing the aligned words.

REFERENCES

1. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. CoRR **abs/1301.3781** (2013)
2. Ide, N., Wilks, Y.: Making sense about sense. In Agirre, E., Edmonds, P., eds.: Word Sense Disambiguation. Volume 33 of Text, Speech and Language Technology. Springer Netherlands (2006) 47–73
3. Miller, G.A.: Wordnet: A lexical database for English. Communications of the ACM **38** (1995) 39–41
4. Black, W., Elkateb, S., Vossen, P.: Introducing the Arabic WordNet project. In: Proceedings of the Third International WordNet Conference (GWC-06. (2006)
5. Ordan, N., Wintner, S.: Hebrew WordNet: A test case of aligning lexical databases across languages. International Journal of Translation **19**(1) (2007) 39–58
6. Hovy, E., Marcus, M., Palmer, M., Ramshaw, L., Weischedel, R.: Ontonotes: The 90% solution. In: Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers. NAACL-Short '06, Stroudsburg, PA, USA, Association for Computational Linguistics (2006) 57–60
7. Brown, P.F., Della Pietra, S.A., Della Pietra, V.J., Mercer, R.L.: Word-sense disambiguation using statistical methods. In: Proceedings of the 29th Annual Meeting on Association for Computational Linguistics. ACL '91, Stroudsburg, PA, USA, Association for Computational Linguistics (1991) 264–270
8. Gale, W., Church, K., Yarowsky, D.: Using bilingual materials to develop word sense disambiguation methods. In: Proceedings of the International Conference on Theoretical and Methodological Issues in Machine Translation. (1992) 101–112
9. Dagan, I., Itai, A.: Word sense disambiguation using a second language monolingual corpus. Computational Linguistics **20**(4) (1994) 563–596
10. Mikolov, T., Le, Q.V., Sutskever, I.: Exploiting similarities among languages for machine translation. CoRR **abs/1309.4168** (2013)
11. Bengio, Y., Ducharme, R., Vincent, P., Janvin, C.: A neural probabilistic language model. J. Mach. Learn. Res. **3** (March 2003) 1137–1155
12. Mnih, A., Hinton, G.: Three new graphical models for statistical language modelling. In: Proceedings of the 24th International Conference on Machine Learning. ICML '07, New York, NY, USA, ACM (2007) 641–648

13. Mikolov, T., Kombrink, S., Burget, L., Cernocky, J., Khudanpur, S.: Extensions of recurrent neural network language model. In: Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on. (2011) 5528–5531
14. Och, F.J., Ney, H.: A systematic comparison of various statistical alignment models. Computational Linguistics **29**(1) (2003) 19–21
15. Habash, N., Rambow, O., Roth, R.: MADA+TOKAN: A toolkit for arabic tokenization, diacritization, morphological disambiguation, POS tagging, stemming and lemmatization. In Choukri, K., Maegaard, B., eds.: Proceedings of the Second International Conference on Arabic Language Resources and Tools, Cairo, Egypt, The MEDAR Consortium (April 2009)
16. Itai, A., Wintner, S.: Language resources for Hebrew. Language Resources and Evaluation **42**(1) (March 2008) 75–98
17. Adler, M.: Hebrew Morphological Disambiguation: An Unsupervised Stochastic Word-based Approach. PhD thesis, Ben-Gurion University (2007)
18. Goldberg, Y., Adler, M., Elhadad, M.: EM can find pretty good HMM POS-taggers (when given a good start). In: Proceedings of ACL-08: HLT, Columbus, OH, Association for Computational Linguistics (June 2008) 746–754
19. Brown, P.F., Della-Pietra, S.A., Della-Pietra, V.J., Mercer, R.L.: The mathematics of statistical machine translation. Computational Linguistics **19**(2) (1993) 263–313
20. Akiva, N.: Style-Based Analysis of Natural Language Using Lexical Features. PhD thesis, Dept. Computer Science, Bar-Ilan Univ. (November 2013)
21. Strong, J.: The Exhaustive Concordance of the Bible. Jennings & Graham, Nashville (1890)
22. Bengio, Y., LeCun, Y.: Scaling learning algorithms towards AI. In Bottou, L., Chapelle, O., DeCoste, D., Weston, J., eds.: Large Scale Kernel Machines. MIT Press (2007)

LIOR WOLF
TEL AVIV UNIVERSITY,
ISRAEL
E-MAIL: <WOLF@CS.TAU.AC.IL>

YAIR HANANI
TEL AVIV UNIVERSITY,
ISRAEL
E-MAIL: <YAIR.HANANI@GMAIL.COM>

KFIR BAR
TEL AVIV UNIVERSITY,
ISRAEL
E-MAIL: <BARKFIR@YAHOO.COM>

NACHUM DERSHOWITZ
TEL AVIV UNIVERSITY,
ISRAEL
E-MAIL: <NACHUMD@POST.TAU.AC.IL>

IJCLA VOL. 5, NO. 1, JAN-JUN 2014, PP. 43–55
RECEIVED 06/01/14 ACCEPTED 06/02/14 FINAL 17/06/14

Topic Classification using Latent Dirichlet Allocation at Multiple Levels

DIANA INKPEN AND AMIR H. RAZAVI

University of Ottawa, Canada

ABSTRACT

We propose a novel low-dimensional text representation method for topic classification. Several Latent Dirichet Allocation (LDA) models are built on a large amount of unlabelled data, in order to extract potential topic clusters, at different levels of generalization. Each document is represented as a distribution over these topic clusters. We experiment with two datasets. We collected the first dataset from the FriendFeed social network and we manually annotated part of it with 10 general classes [1]. The second dataset is a standard text classification benchmark, Reuters 21578, the R8 subset (annotated with 8 classes). We show that classification based on our multi-level LDA representation leads to improved results for both datasets. Our representation catches topic distributions from generic ones to more specific ones and allows the machine learning algorithm choose the appropriate level of generalization for the task. Another advantage is the dimensionality reduction, which permitting the use of machine learning algorithms that cannot run on high-dimensional feature spaces. Even for the algorithms that can deal with high-dimensional features spaces, it is often useful to speed up the training and testing time by using the lower dimensionality.

1 INTRODUCTION

In order to improve the performance of text classification tasks, we always need informative and expressive methods to represent the texts [2, 3]. If we consider the words as the smallest informative unit of a text,

there is a variety of well-known quantitative information measures that can be used to represent a text. Such methods have been used in a variety of information extraction projects, and in many cases have even outperformed some syntax-based approaches. There are a variety of Vector Space Models (VSM) which have been well explained and compared, for example in [4]. However, these kinds of representations disregard valuable knowledge that could be inferred by considering the different types of relations between the words. These major relations are actually the essential components that, at a higher level, could express concepts or explain the main topic of a text. A representation method which could add some kind of relations and dependencies to the raw information items, and illustrate the characteristics of a text at different conceptual levels, could play an important role in knowledge extraction, concept analysis and sentiment analysis tasks.

In this paper, the main focus is on how we represent the topics of the texts. Thus, we select a LDA topic-based representation method, and we extend it to a multi-level representation that can automatically choose the appropriate level of generality. Then, we run machine learning algorithms on each representation (or combinations), in order to explore the most discriminative representation for the task of text classification, for the two datasets that we selected.

2 Related Work

In most text classification tasks, the texts are represented as a set of independent units such as unigrams / bag of words (BOW), bigrams and/or multi-grams which construct the feature space, and the text is normally represented only by the assigned values (binary, frequency or term TF-IDF[1]) [5]. In this case, since most lexical features occur only a few times in each context, if at all, the representation vectors tend to be very sparse. This method has two disadvantages. First, very similar contexts may be represented by different features in the vector space. Second, in short and medium-size texts, we will have too many zero features for machine learning algorithms, including supervised classification methods.

Blei, Ng and Jordan proposed the Latent Dirichlet Allocation (LDA) model and a Variational Expectation-Maximization algorithm for training their model. LDA is a generative probabilistic model of a corpus and the idea behind it is that the documents are represented as weighted relevancy

[1] term frequency / inverse document frequency

vectors over latent topics, where a topic is characterized by a distribution over words. These topic models are a kind of hierarchical Bayesian models of a corpus [6]. The model can unveil the main themes of a corpus which can potentially be used to organize, search, and explore the documents of the corpus. In LDA models, a *topic* is a distribution over the feature space of the corpus and each document can be represented by several topics with different weights. The number of topics (clusters) and the proportion of vocabulary that create each topic (the number of words in a cluster) are considered as two hidden variables of the model. The conditional distribution of words in topics, given these variables, for an observed set of documents, is regarded as the main challenge of the model.

Griffiths and Steyvers in 2004, applied a derivation of the Gibbs sampling algorithm for learning LDA models [7]. They showed that the extracted topics capture a meaningful structure of the data. The captured structure is consistent with the class labels assigned by the authors of the articles that composed the dataset. The paper presents further applications of this analysis, such as identifying *hot topics* by examining temporal dynamics and tagging some abstracts to help exploring the semantic content. Since then, the Gibbs sampling algorithm was shown as more efficient than other LDA training methods, e.g., variational EM and Expectation-Propagation [8]. This efficiency is attributed to a famous attribute of LDA namely, "the conjugacy between the Dirichlet distribution and the multinomial likelihood". This means that the conjugate prior is useful, since the posterior distribution is the same as the prior, and it makes inference feasible; therefore, when we are doing sampling, the posterior sampling become easier. Hence, the Gibbs sampling algorithms was applied for inference in a variety of models that extend LDA [9–13].

Recently, Mimno et al. presented a hybrid algorithm for Bayesian topic modeling in which the main effort is to combine the efficiency of sparse Gibbs sampling with the scalability of online stochastic inference [14]. They used their algorithm to analyze a corpus that included 1.2 million books (33 billion words) with thousands of topics. They showed that their approach reduces the bias of variational inference and can be generalized by many Bayesian hidden-variable models.

LDA topics models started to be used in various Natural Language Processing tasks. It was used, among other tasks, for native language identification [15], for learning word classes [16], and for opinion analysis [17]. Supervised versions were developed, named labelled LDA, and applied, for example, for authorship attribution [18]. Experiments that

used LDA topic models for a task of cross-language categorization of Wikipedia pages were presented in [19]. In this paper, we focus on the task of automatic text classification into a set of generic topics/subjects using multiple LDA models in the same time, in order to achieve different levels of generalization. Smet at al. [19] also used multiple LDA models (with 10 to 200 topics, with an increment of 20). We developend our method without being aware of their work, initally. The task and datasets that we used are different.

3 DATASETS

In order to properly evaluate our new multi-level LDA text representation, we conducted experiment with two datasets of different genres (social media text and newspaper text).

The first dataset that we prepared for our experiments consists of threads from the FriendFeed social network. We collected main postings (12,450,658) and their corresponding comments (3,749,890) in order to obtain all the discussion threads (a thread consists of a message and its follow up comments). We filtered out the threads with less than three comments. We were left with about 24,000 threads. From these, we used 4,000 randomly-selected threads as background source of data, in order to build the LDA model. We randomly selected 500 threads out of the 4000 and manually annotated them with 10 general classes[2], to use as training and test data for the classification task. The 10 classes are: *consumers, education, entertainment, lifestyle, politics, relationships, religion, science, social_life* and *technology*. We will make the dataset available (the whole corpus that we collected and the manually-annotated part).

We observed that the 10 class labels (general topics) are distributed unevenly over the dataset of 500 threads, in which we had 21 threads for the class *consumers*, 10 threads for *education*, 92 threads for *entertainment*, 28 threads for *incidents*, 90 threads for *lifestyle*, 27 threads for *politics*, 58 threads for *relationships*, 31 threads for *science*, 49 threads for *social_activities*, and 94 threads for *technology*.

The second dataset that we chose for our experiments is the well-known R8 subset of the Reuters-21578 collection (excerpted from the UCI machine learning repository), a typical text classification benchmark. The data includes the 8 most frequent classes of Reuteres-21578;

[2] We used only one annotator, but we had a second annotator check a small subset, in order to validate the quality of annotation. In future work, we plan to have a second annotator label all the 500 threads.

hence the topics that will be considered as class labels in our experiments are *acq, crude, earn, grain, interest, money, ship* and *trade*.

In order to follow Sebastiani's convention [3], we also call the dataset R8. Note that there is also a R10 dataset, and the only difference between R10 and R8 is that the classes *corn* and *wheat*, which are closely related to the class *grain*, were removed. Table 1 shows the distribution of documents per class and the split into training and test data for the R8 subset.

Table 1. Class distribution of training and testing data for R8.

Class	No. of Training Docs	No. of Test Docs	Total
Acq	1596	696	2292
Earn	2840	1083	3923
Grain	41	10	51
Interest	190	81	271
Money-fx	206	87	293
Ship	108	36	144
Trade	251	75	326
Crude	253	121	374
Total	5485	2189	7674

4 METHOD

We trained LDA models for each of the two datasets: the 4000 threads from FriendFeed and the R8 text data. LDA models have two parameters whose values need to be chosen experimentally: the number of topic clusters and the number of words in each cluster. We experimented with various parameter values of the LDA models. The number of cluster is particularly difficult to choose, since it reflects the level of generality of the extracted topics / concepts.

For the first dataset, the number of words in each cluster was set to maximum 15 (because for higher values, the weights of the words in the clusters became very small). For the number of topics, we chose several values: 10, 20, 40, 80, 160, and 320. Therefore we build 6 LDA models. We started with 10 topics because we have 10 classes, then we doubled the number of LDA topics at every model. Instead of choosing one of the models, we used all of them in order to represent each text at multiple levels of generalization at the same time. In this way, we let the classifiers choose the best features for the task.

In LDA models, polysemous words can be member of more than one topical cluster, while synonymous words are normally gathered in the same topics. An example of LDA topic cluster for the first model is: "Google", "email", "search", "work", "site", "services", "image", "click", "page", "create", "contact", "connect", "buzz", "Gmail", "mail". This could be labeled as *Internet*.

As mentioned, our 500 threads were manually annotated with the 10 generic classes. These classes, enumerated in section 3, are a manually generalized version of the top 50 LDA clusters into the 10 generic categories that proved to be sufficient during the manual annotation of the data. For the above example, the annotator placed it under the *technology* and *social_life* categories. The classification task is therefore multi-class, since a thread can be in more than one class. We trained binary classifiers for each class, and averaged the results over all classes.

For the second dataset, R8, we experimented with several parameter values for the number of clusters in the LDA models: 8, 16, 32, 64, 128, and 256 (thus we built 6 models). We chose 20 words in each cluster (because for higher values the weights were becoming too small). The reason we started with 8 clusters is that there are 8 classes in the annotated data. Then we doubled the number of topics several times. Similarly to the representation used for the first dataset, we combined all the models in the feature representation (the multi-level LDA-based representation), leaving up to the classifier to choose an appropriate level of generalization.

For the classification task on both datasets, we chose several classifiers from Weka [20], including Naive Bayes (NB) because it is fast and works well with text, SVM since it is known to obtain high performance on many tasks, and decision trees because we can manually inspect the learned tree.

We applied these classifiers on simple bag-of-words (BOW) representation, on LDA-based representations of different granularities, and on an integrated representation concatenating the BOW features and the LDA features. The values of the LDA-based features for each document are the weights of the clusters associated to the document by the LDA model (probability distributions).

5 Experiments and Results

The results on the first dataset are presented in Table 2. After stopword removal and stemming, the bag-of-words (BOW) representation contained

6573 words as features (TF-IDF values). The lower-dimensional representation based on LDA contained 630 features (10 + 20 + 40 + 80 + 160 + 320), whose values are the weights corresponding to the topic clusters. For the combined representation (BOW integrated with the LDA topics) the number of features was 7203 (6573+630).

The baseline of any classification experiment over this dataset may be considered as 18.8%, for a trivial classifier that puts everything in the most frequent class, *technology*.

On this dataset, due to its relatively small size, we conducted the classification evaluations using stratified 10-fold cross-validations (this means that the classifier is trained on nine parts of the data and tested on the remaining part, then this is repeated 10 times for different splits, and the results are averaged over the 10 folds). We performed several experiments on a range of classifiers and parameter values for each representation, to check the stability of a classifier's performance. We changed the *seed*, a randomization parameter of the 10-fold cross-validation, in order to avoid the accidental over-fitting. The values reported in Table 2 are the accuracies of the classification over all classes.

Table 2. Results on the FriendFeed dataset for different classifiers and representations, by cross-validation.

Representation / Classifier	Accuracy
Baseline	18.8%
BOW / SVM	72.22%
LDA Topics / SVM	75.13%
LDA+BOW / SVM	**80.40%**
BOW / NB	75.93%
LDA Topics / NB	74.63%
LDA+BOW / NB	77.39%
BOW / DT	69.33%
LDA Topics / DT	73.11%
LDA + BOW / DT	75.69%

The SVM classifier was the best for the task. The multi-level LDA-based representation achieved an accuracy of 75.13% compared to the BOW representation at 72.20%. Note that for the BOW representation, the best classifier was Naive Bayes, with an accuracy of 75.93%, but this is due to the use of a variant called complement Naive Bayes that compensates for data imbalance. For the combine LDA and BOW repre-

sentation, SVM achieved the best accuracy of 80.40%. When using the low-dimensional LDA representation only, the accuracy goes down a bit, but still at the same level as BOW and it has the advantage that the classifiers are faster and other classifiers could be used (that do not usually run on high-dimensional data).

Table 3 presents detailed results for each class, for the best run (SVM classifier with LDA + BOW representation). We present the rate of true positives, the rate of false positives, the precision, recall and F-measure for each class. Since the accuracy results over all the classes is good, we wanted to see if the results are good for all classes, or if they vary by class. We can see that there are a few classes that seem to be more challenging for the classifier: *entertainment* and *lifestyle*. This could be due to these two classes being a bit ambiguous, with overlapping vocabulary and topics among the two of them or with the other classes. Perhaps *lifestyle* might be considered a bit too vague as a class label. The *technology* class is also on the low side; perhaps the vocabulary of this class overlaps with other classes too, since we are using a lot of technology for entertainment and other purposes.

Table 3. Results on the FriendFeed dataset for each class, for the SVM classifier (LDA+BOW representation), by cross-validation.

TP Rate	FP Rate	Precision	Recall	F-Measure	Class
1.000	0.001	0.989	1.000	0.994	consumers
1.000	0.000	1.000	1.000	1.000	education
0.459	0.040	0.558	0.484	0.504	entertainment
0.993	0.027	0.797	0.989	0.891	incidents
0.419	0.045	0.479	0.469	0.439	lifestyle
1.000	0.001	0.989	1.000	0.994	politics
0.787	0.050	0.629	0.797	0.710	relationships
1.000	0.001	0.989	1.000	0.984	science
0.921	0.019	0.829	0.941	0.878	social_activities
0.553	0.026	0.678	0.593	0.606	technology

The results on the second dataset, R8, are shown in Table 4, for classifiers trained on the training parts of the data and tested on the test part. After stopword removal and stemming, the BOW representation (TF-IDF values) contained 17387 words as the feature space. We experimented with each LDA representation separately, without good results; therefore we chose the combined 6-level representation wit 504 features (8 + 16 +

32 + 64 + 128 + 256), corresponding to the LDA models with 8, 16, 32, 64, 128, and 256 clusters. For the integrated representation BOW with LDA topics we had 17891 features (the 504 LDA topics plus the 17387 word features).

The average classification accuracy is very high, compared to a baseline of 49.47% (of a simplistic 8-way classifier that always chooses the most frequent class, *earn* in this dataset). The SVM classifier achieved the best results. These values are in line with state-of-the art results reports in the literature. We can compare our results with other reported classification results of the same dataset. According to the best of our knowledge, the accuracy of our integrated representation method on the Reuters R8 dataset, 97%, is higher than any simple and combinatory representation method from related work, which reports accuracies of 88%–95% [21–23], while 96% was reached with SVM on a complex representation method based on kernel functions and Latent Semantic Indexing [24].

For our SVM classifier, the LDA-based representation achieved better accuracy (95.89%) than the BOW representation (93.33%). This is due to the multi-level representation. When we experimented with each level separately, the accuracies dropped considerably. The best results over all the experiments were for SVM with the combined BOW and LDA-based representation (97.03%), though the representation based only on LDA is not far behind and it has the advantage of lower dimensionality.

Table 4. Results on the R8 dataset, on the test data (2189 documents).

Representation / Classifier	Accuracy
Baseline	49.47%
BOW / SVM	93.33%
LDA Topics / SVM	95.89%
LDA+BOW / SVM	**97.03%**
BOW / NB	95.20%
LDA Topics / NB	94.61%
LDA+BOW / NB	95.52%
BOW / DT	91.54%
LDA Topics / DT	91.78%
LDA + BOW / DT	92.10%

For more complete experiments, as a second scenario on the R8 dataset, we also trained and tested the same set of classifiers using 10-fold cross-validation on the whole dataset, to check the stability of the results

when training and testing sets are rotationally changed by stratified 10-fold cross-validation. The results are presented in Table 5 and they show a similar trend as the results from Table 4. The SVM classifier with LDA + BOW representation achieved the best accuracy, while the representation based only on the multi-level LDA is not far behind and it is better than the BOW representation.

In Table 6 we show detailed results for each class, for the best classifier, SVM with the combined feature representation, for the cross-validation setting. We can see that performance is very good for all classes, with one exception for the class *grain*. This is probably due to the lower number of instances of this class in the training and test data compared to the other classes.

Table 5. Results on the R8 dataset, by cross-validation on the whole data.

Representation / Classifier	Accuracy
Baseline	51.00%
BOW / SVM	94.67%
LDA Topics / SVM	95.89%
LDA+BOW / SVM	**97.29%**
BOW / NB	94.91%
LDA Topics / NB	92.57%
LDA+BOW / NB	94.59%
BOW / DT	90.40%
LDA Topics / DT	91.73%
LDA + BOW / DT	91.88%

6 Conclusions and Future Work

As our experimental results show, we can achieve good classification results by using a low-dimensional representation based on the multi-level LDA. This representation has the advantage that allows the use of classifiers or clustering algorithms that cannot run on high-dimensional feature spaces. By using a multi-level representation (different generalization levels) we achieved better results than the BOW representation on both datasets, with the SVM classifier.

The combined BOW and LDA features representation achieved the best classification performance, and it can be used when there memory

Table 6. Results on the R8 dataset for each class, for the SVM classifier (LDA+BOW representation), by cross-validation.

TP Rate	FP Rate	Precision	Recall	F-Measure	Class
0.982	0.020	0.954	0.982	0.968	acq
0.988	0.011	0.990	0.988	0.989	earn
0.794	0.000	0.986	0.794	0.880	grain
0.914	0.002	0.921	0.914	0.917	interest
0.884	0.002	0.931	0.884	0.907	money-fx
0.878	0.001	0.950	0.878	0.913	ship
0.945	0.003	0.931	0.945	0.938	trade
0.912	0.001	0.977	0.912	0.943	crude

is not a concern, for classifiers that are able to cope with the large vector spaces. Even in this case, the training and test times can be reduced by using only the LDA based representation.

Our results show that the first dataset is more difficult to classify than the second dataset. The reason is that it consists of social media texts, which are very noisy. In future work, we plan to experiment with more training data for the FriendFeed dataset (automatically annotated via the mapping of LDA clusters into the 10 classes), and to design new representation and classification methods that are more appropriate for this kind of data.

ACKNOWLEDGMENTS The authors wish to thank Ontario Centres of Excellence and to The Natural Sciences and Engineering Research Council of Canada for the financial support. We thank Lana Bogouslavski for annotating the FriendFeed data and Dmitry Brusilovsky for his insights in the experiments related to the classification of social media texts.

REFERENCES

1. Razavi, A.H., Inkpen, D., Brusilovsky, D., Bogouslavski, L.: General topic annotation in social networks: A Latent Dirichlet Allocation approach. In: Canadian Conference on Artificial Intelligence. (2013) 293–300
2. Pan, X., Assal, H.: Providing context for free text interpretation. In: IEEE International Conference on Natural Language Processing and Knowledge Engineering. (2003) 704–709
3. Sebastiani, F.: Classification of text, automatic. In Brown, K., ed.: The Encyclopedia of Language and Linguistics, Volume 14, 2nd Edition. Elsevier Science Publishers, Amsterdam, NL (2006) 457–462

4. Turney, P.D., Pantel, P.: From frequency to meaning: Vector space models of semantics. Journal of Artificial Intelligence Research **37**(1) (2010) 141–188
5. Spark Jones, K.: A statistical interpretation of term specificity and its application in retrieval. Journal of Documentation **1** (1972) 11–21
6. Blei, D.M., Griffiths, T.L., Jordan, M.I., Tenenbaum, J.B.: Hierarchical topic models and the nested Chinese restaurant process. In: Advances in Neural Information Processing Systems, MIT Press (2004) 2003
7. Griffiths, T.L., Steyvers, M.: Finding scientific topics. Volume 101, Suppl. 1 of Proceedings of the National Academy of Sciences of the United States of America. National Academy of Sciences (2004)
8. Minka, T., Lafferty, J.: Expectation-propagation for the generative aspect model. In: Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence. UAI'02, San Francisco, CA, USA, Morgan Kaufmann Publishers Inc. (2002) 352–359
9. Wang, X., McCallum, A.: Topics over time: A non-Markov continuous-time model of topical trends. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD '06, New York, NY, USA, ACM (2006) 424–433
10. McCallum, A., Corrada-Emmanuel, A., Wang, X.: Topic and role discovery in social networks. In: Proceedings of the 19th International Joint Conference on Artificial Intelligence. IJCAI'05, San Francisco, CA, USA, Morgan Kaufmann Publishers Inc. (2005) 786–791
11. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent Dirichlet allocation. Journal of Machine Learning Research **3** (2003) 993–1022
12. Blei, D.M., Lafferty, J.D.: Topic models. Text mining: Classification, clustering, and applications **10** (2009) 71
13. Li, W., McCallum, A.: Pachinko allocation: Dag-structured mixture models of topic correlations. In: Proceedings of the 23rd International Conference on Machine Learning. ICML'06, New York, NY, USA, ACM (2006) 577–584
14. Mimno, D.M., Hoffman, M.D., Blei, D.M.: Sparse stochastic inference for Latent Dirichlet Allocation. In: Proceedings of the 29th International Conference on Machine Learning (ICML 2012). (2012)
15. Jojo, S.m., Wong, M., Dras, M.J.: Topic modeling for native language identification. In: Proceedings of the Australasian Language Technology Association Workshop, Canberra, Australia (2011) 115–124
16. Chrupala, G.: Efficient induction of probabilistic word classes with LDA. In: Proceedings of the Fifth International Joint Conference on Natural Language Processing (IJCNLP 2011). (2011) 363–372
17. Zhao, W.X., Jiang, J., Yan, H., Li, X.: Jointly modeling aspects and opinions with a MaxEnt-LDA hybrid. In: Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing. EMNLP'10, Stroudsburg, PA, USA, Association for Computational Linguistics (2010) 56–65
18. Ramage, D., Hall, D., Nallapati, R., Manning, C.D.: Labeled LDA: A supervised topic model for credit attribution in multi-labeled corpora. In: Proceedings of the 2009 Conference on Empirical Methods in Natural Language

Processing: Volume 1. EMNLP '09, Stroudsburg, PA, USA, Association for Computational Linguistics (2009) 248–256
19. Smet, W.D., Tang, J., Moens, M.F.: Knowledge transfer across multilingual corpora via latent topics. In: Proceedings of the 15th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining - Volume Part I. PAKDD'11, Berlin, Heidelberg, Springer-Verlag (2011) 549–560
20. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2005)
21. Chen, Y.L., Yu, T.L.: News classification based on experts' work knowledge. In: Proceedings of the 2nd International Conference on Networking and Information Technology (IPCSIT 2011), vol.17, Singapore, IACSIT Press (2011)
22. Aggarwal, C.C., Zhao, P.: Towards graphical models for text processing. Knowledge Information Systems **36**(1) (2013) 1–21
23. Cardoso-Cachopo, A., Oliveira, A.L.: Combining LSI with other classifiers to improve accuracy of single-label text categorization. In: Proceedings of the First European Workshop on Latent Semantic Analysis in Technology Enhanced Learning-EWLSATEL. (2007)
24. Yuan, M., Ouyang, Y., Xiong, Z.: A text categorization method using extended vector space model by frequent term sets. Journal of Information Science and Engineering **29**(1) (2013) 99–114

DIANA INKPEN
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE,
UNIVERSITY OF OTTAWA,
OTTAWA, ON, K1N 6N5, CANADA
E-MAIL: <DIANA.INKPEN@UOTTAWA.CA>

AMIR H. RAZAVI
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE,
UNIVERSITY OF OTTAWA,
OTTAWA, ON, K1N 6N5, CANADA
E-MAIL: <ARAZA082@UOTTAWA.CA>

Sentiment Analysis and Truthfulness Detection

IJCLA VOL. 5, NO. 1, JAN-JUN 2014, PP. 59–72
RECEIVED 07/01/14 ACCEPTED 06/02/14 FINAL 18/06/14

Sentiment Lexicon Generation for an Under-Resourced Language

CLARA VANIA, MOH. IBRAHIM, AND MIRNA ADRIANI

Universitas Indonesia, Indonesia

ABSTRACT

Sentiment analysis and opinion mining are actively explored nowadays. One of the most important resources for the sentiment analysis task is sentiment lexicon. This paper presents our study in building domain-specific sentiment lexicon for Indonesian language. Our main contributions are (1) methods to expand sentiment lexicon using sentiment patterns and (2) a technique to classify the polarity of a word using the sentiment score. Our method is able to generate sentiment lexicon automatically by using a small seed of sentiment words, user reviews, and part-of-speech (POS) tagger. We develop the lexicon for Indonesian language using a set of seed words translated from English sentiment lexicon and expand them using sentiment patterns found in the user reviews. Our results show that the proposed method can generate additional lexicon with sentiment accuracy of 77.7%.

KEYWORDS: Sentiment lexicon, natural language processing, under-resourced language, lexicon generation.

1 INTRODUCTION

Sentiment analysis or opinion mining is one of the most active research areas today. The rapid growth of social media such as Twitter, Facebook, forum discussions, etc., has made a huge amount of opinionated data available on the web. People share their opinion about things they like or dislike on the web. A person who wants to buy a particular product

searches for its review on the web. Organizations conduct survey or research to analyze public opinions. As a result, opinion mining has been used to track public opinions toward entities, i.e products, events, individuals, organizations, topics, etc.

One of the most important resources for sentiment analysis task is sentiment lexicon. Sentiment lexicon consists of words with its polarity, whether it is positive or negative. For example, "good" is considered as positive word and "bad" as negative word. While there are many English sentiment lexicons available on the web, sentiment lexicons in other languages can be considered very limited or even unavailable. This made research in sentiment analysis quite difficult for non-English documents. Therefore, developing sentiment lexicon in other languages is very important.

According to Liu [10], sentiment lexicon generation can be divided into three approaches, namely manual approach, dictionary-based approach, and corpus-based approach. The first approach is built manually by human and thus requires considerable resources. The second approach is dictionary-based approach, where a set of seed words is created manually and then expanded by using a dictionary (thesaurus, WordNet, etc). The corpus-based approach also uses manually labeled seed words and then expanded using available corpus data.

Many research works on sentiment lexicon generation have been done. Most of the research work is applied in English, while for other languages the research is still growing. Turney and Littman [18] use queries to find candidate English sentiment lexicons from Web search engine. Kanayama and Natsukawa [7] propose an unsupervised method to detect polar clause in domain-specific documents. Qiu et al. [16] use double propagation to expand the sentiment lexicon and extract opinion target in a document. Pérez-Rosas et al. [14] apply dictionary-based approach to build Spanish sentiment lexicon. Kaji and Kitsuregawa [5] uses massive HTML corpus to build Japanese sentiment lexicon. In their work, they use structural clues to find polar sentence from Japanese HTML documents. Banea et al. [1] propose a method for constructing sentiment lexicons for low-resourced language.

In this paper, we apply corpus-based approach to build Indonesian sentiment lexicon for a specific target domain. While most of sentiment lexicon generation techniques rely on the availability of WordNet, in our case it is not feasible because of the limitation of Indonesian language resources. Our proposed methods depend on the availability of English sentiment lexicon, machine translation, part-of-speech (POS) tagger and online user reviews. Our main contributions in this paper are:

1. Methods to expand the sentiment lexicon using automatic translation services and simple pattern-based approaches. We use available English sentiment lexicon and translate them into Indonesian language. To expand the lexicon, we use user reviews from user-generated content (UGC) and social media data, as they are available and can be collected easily.
2. Techniques to filter sentiment words and scoring function to determine the polarity of each word.

In this work, we show that although the language resources are limited, we can use other resources, which can be collected easily to build the lexicon. UGC and social media are quite popular nowadays and available in almost every language. Those data also contains many public opinions and very suitable for sentiment analysis research.

2 INDONESIAN SENTIMENT LEXICON GENERATION

2.1 *Seed Lexicon*

Many research about sentiment lexicon generation use seed words to build the lexicon. Some use manually built seed lexicon [9] and some others use seed words taken from dictionary (e.g., [2, 4, 6, 8]). In this study, we use an available English sentiment lexicon, which has been widely used in many sentiment analysis research works. The lexicon that we used in this experiment is OpinionFinder[1] [21] and SentiWordNet.[2] In the OpinionFinder, each word is assigned with its polarity; positive, negative, or objective. It also gives label strong or weak subjectivity to each word. SentiWordNet is another English sentiment lexicon developed by [4]. This lexicon is built in accordance with WordNet. Each synset is assigned with its subjectivity score. SentiWordNet defines three score for each synsets; positive, negative, and objective score.

In this study, we aim to build sentiment lexicon with positive and negative subjectivity. We begin by selecting initial seed words to building the lexicon. We select terms from OpinionFinder with *strong positive* / ***negative polarity***. For SentiWordNet, we select adjective synsets with *highest subjectivity score* (in this experiment we take terms with score

[1] http://mpqa.cs.pitt.edu/opinionfinder
[2] http://sentiwordnet.isti.cnr.it

above 0.7). These selection criteria help us to choose terms with strong polarity and use it as seed words.

As we want to build lexicon for Indonesian language, we translate those seed words into Indonesian. Several problems which occur are terms which do not have its corresponding terms in Indonesian and some English terms which have the same translation in Indonesian. The same problem also happened in the previous research by Wiebe et al. [21], Wan [19] in using translation to build sentiment lexicon. In this study, we simply eliminate terms that do not have its corresponding translation in Indonesian language.

In order to get expansion terms with high precision, we have to ensure that the selected seed words are opinion words. Therefore, we conduct two stages of manual evaluation which consist of translation and subjectivity evaluation. For translation evaluation, we eliminate words that have no translation in Indonesian. Duplicate translations and mistranslated word also removed from the seeds. To evaluate the subjectivity, we conduct manual evaluation for each word to check whether the translated word contains the same polarity with the English word.

Table 1 shows the statistics of our seed lexicon. After evaluation, there are 291 positive words and 517 negative words which we used as seed words.

Table 1. Statistics of Seed Words

Source Lexicon	#English words	#Translated Words	#Seed Words
Positive Words	2071	1161	**291**
Negative Words	4637	2392	**517**

2.2 *Sentiment Lexicon Expansion*

SENTI-PATTERN (SP). People tend to have similar patterns to express their opinion. For domain-specific sentiment analysis, these patterns are useful to analyze opinions about a particular entity. For example, in book reviews, we can find opinions such as "*This book is great*" or "*This book is awfull*". Although those opinions have opposite subjectivity, the sentences use the same pattern that clearly states opinions about the book.

In the first approach, we want to find sentiment patterns that are usually found in the user reviews. In the previous study, Pantel and Pennacchiotti [12] use generic patterns to extract semantic relations from raw text. In this study, our hypothesis is that sentiment patterns that are frequently used by many reviewers can be used to extract new sentiment

words. Fig. 1 shows the extraction process of SP. In the first step, documents (user reviews) will be divided into sentences. After that, we develop a list of n-grams (n = 3) along with its frequency that are found in the corpus. We filter the n-grams by only taking n-grams which contains seed words. Any seed word found in the n-grams is then replaced by the same tag, i.e. [SENT] to indicate sentiment word position in the n-gram. Top-N n-grams with highest frequency (we use N=50) are then used as senti-patterns. Fig. 2 shows example of sentiment patterns found in the corpus.

We expand the seed words by searching the senti-patterns in the corpus to find candidate sentiment words. At this step, we do not classify the word polarity as the patterns can contains opinion words with various polarities. The polarity classification will be done at the filtering step.

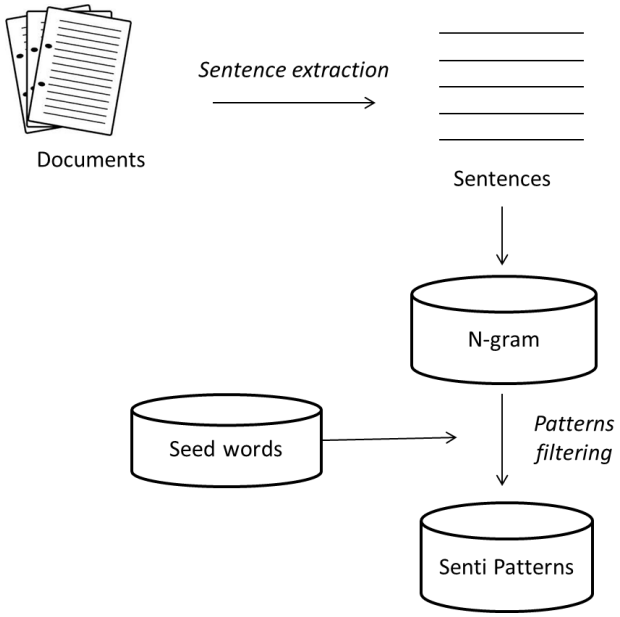

Fig. 1. Senti-Pattern (SP) extraction process

tempat yang [SENT]
('[SENT] place')

Fig. 2. Example of Senti-Pattern

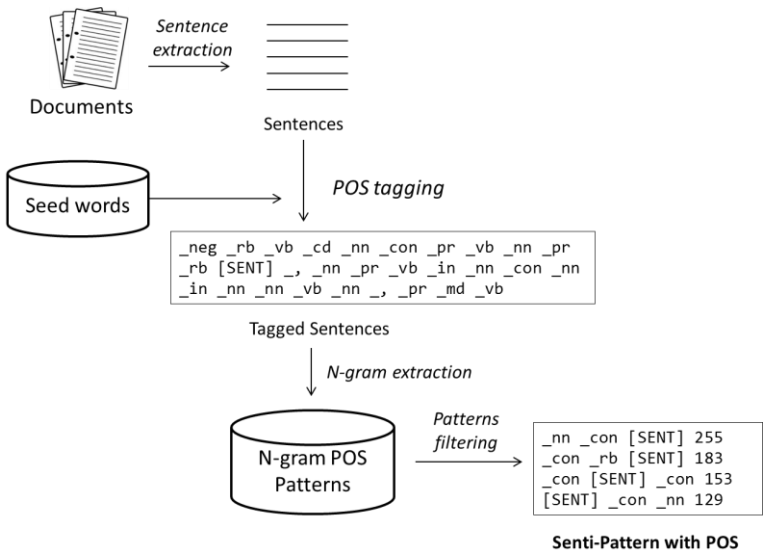

Fig. 3. Senti-Pattern with POS (SP-POS) extraction process

SENTI-PATTERN WITH PART-OF-SPEECH (SP-POS) The SP approach find new candidate words by using patterns that usually occurred in the documents. However, the approach depends on exact matching to find new words. In the second approach (SP-POS) we try to develop more general patterns by using the *Part-Of-Speech* (POS) information of the n-grams.

At the first step, we apply sentence extraction on the documents. Next, we conduct POS tagging in order to mark every word in the sentence with its part-of-speech, based on its definition and context. We use Indonesian POS Tagger developed by Pisceldo et al. [15], which uses probabilistic approach. In the tagging process, seed words found in the sentences are tagged with a special tag, '[SENT]'. After that, we extract n-grams (n = 3) from the tagged sentences. We ranks the n-grams based on their frequency and take top-N (N = 50) n-grams that contain [SENT] tag as SP-POS. Fig. 3 depicts the overall process to extract SP-POS.

After we develop the SP-POS patterns, we create parallel corpus, which consists of original sentences and its corresponding tagged sentences (without using seed words). We match the SP-POS patterns to the tagged sentences, and find n-gram that suitable with the SP-POS patterns, except the [SENT] tag, which can match any word. Finally, we look for the original words that fit the [SENT] tag in the parallel corpus and add them to the candidate lexicon.

Table 2. Negation and Transitional Words

Negation words	*Tidak* (not), *enggak* (not), *nggak* (not), *engga* (not), *ga* (no), *gak* (no), *gag* (no), *bukan* (not), *tiada* (no), *non* (not), *tak* (not), *kagak* (no), *kaga* (non)
Transitional words	*Tetapi* (but), *melainkan* (but), *padahal* (whereas), *sedangkan* (while), *tapi* (but), *namun* (however), *sebaliknya* (otherwise)

EXPANSION USING SENTENCE POLARITY (SPO) The next approach expands the seed lexicon using sentence polarity. (Terra and Clarke, 2003) propose technique to find words that have high similarity based on their co-occurrence in the corpus. Using the same idea, we try to find new sentiment words by its occurrences in polar sentences. A sentence is a polar sentence if it contains seed word(s). We assume that the occurrence information will implicitly define the relationship between seed words and candidate word.

EXTRACTING SENTENCE POLARITY To expand the seed lexicon, first we filter sentences that contain seed words. By default, the sentence polarity follows the seed word polarity. We also include some cases that may change the polarity of a sentence by searching transitional and negation words.

TRANSITIONAL WORDS We detect transitional words that appeared in the sentences. A subjective sentence may contain more than one polarity, as people can state what they like and dislike in one sentence. For example, "While the food is expensive, the taste is very delicious". In that sentence, we can find two kind of sentiment with different polarity. The reviewer likes the food but does not like the price of that food. Here, we list words that may change the polarity of a sentence. For this kind of sentence, we split the sentence into two sub-sentences with different polarity.

NEGATION WORDS We also detect negation words, such as "no" and "not" in the sentences. Negation words are used to detect polarity shifting.

SELECTING CANDIDATE WORDS After extract the polarity of sentence, we calculate polarity score of each word in the sentence. We adopt the

Pointwise Mutual Information (PMI) (Church and Hanks, 1989) to estimate polarity value. For each word *w* in the corpus, we calculate its two polarity score, positive polarity (*pos_polarity*) and negative polarity (*neg_polarity*). Sentiment polarity of a word *w* will have higher value when it frequently occurred in sentiment sentences. Sentiment polarity value is estimated as follows:

$$pos\_polarity(w) = \log_2 \frac{p(w,pos)}{f(w) \cdot \frac{f(pos)}{N}}, \quad (1)$$

$$neg\_polarity(w) = \log_2 \frac{p(w,neg)}{f(w) \cdot \frac{f(neg)}{N}}, \quad (2)$$

where *p*(*w, pos*) is the occurrence likelihood of word *w* in the positive sentences, *f*(*w*) is the frequency of sentences which contain word w, *f*(*pos*) is frequency of positive sentences, and N is total number of sentences. The same definition applied for negative polarity. We compute *p*(*w, pos*) *and p*(*w,neg*) as follows:

$$p(w,pos) = \frac{f_{(w,pos)}}{N}, \quad (3)$$

$$p(w,neg) = \frac{f_{(w,neg)}}{N}, \quad (4)$$

where $f_{(w,pos)}$ is the frequency of positive sentences that contain word *w*.

2.3 *Filtering Expansion Terms*

OPINION FILTERING We apply opinion filtering to remove non-sentiment words from the candidate sentiment words. Several study on sentiment analysis show that adjective words is effective to increase accuracy [3, 13]. In this phase, we simply remove non-adjective words based on our random observation that sentiment words are usually adjectives.

SENTIMENT DETECTION As lexicon expansion only collect candidate sentiment words without determining its polarity, in this step we detect sentiment of each candidate word. We detect the polarity of a word by calculate its two sentiment score, *sent_pos*(*w*) and *sent_ neg*(*w*):

$$sent_{pos(w)} = \frac{\frac{P(x|pos)}{P(pos)}}{\frac{P(x|pos)}{P(pos)} + \frac{P(x|neg)}{P(neg)}}, \quad (5)$$

$$sent_{neg(w)} = \frac{\frac{P(x|neg)}{P(neg)}}{\frac{P(x|pos)}{P(pos)} + \frac{P(x|neg)}{P(neg)}}, \quad (6)$$

where $P(x|pos)$ is the number of positive seed words in positive documents and $P(x|neg)$ is the number of positive seed words in negative documents. $P(pos)$ and $P(neg)$ is the number of positive and negative documents. A word is considered positive if its positive score is higher than negative score and vice versa.

3 Experimental Results

3.1 *Dataset*

In this study, we use three kind of dataset collected from social media data. We focus on domain specific sentiment lexicon, so we collect data from tourism domain. Dataset used in this experiment are collected from TripAdvisor[3], Twitter, and OpenRice[4]. TripAdvisor and OpenRice are user generated content (UGC) which contains reviews about Indonesian tourism and restaurants. We collect reviews from both sites and assign polarity value (negative or positive) based on the review ratings. As they use rating scale 1–5, we assign review with ratings (1–2) as negatives and (4–5) as positives. For Twitter data, we collect tweets using query about tourism sites in Indonesia. As building human annotated Twitter corpus requires considerable resources, we collect Twitter corpus using query that contains emoticons, i.e :-), :), :(, :-(, etc. We assume that a tweet is a subjective if it is contain emoticons and classify the tweets using positive and negative emoticons. Statistics of our dataset are shown in Table 3.

[3] http://www.tripadvisor.co.id
[4] http://id.openrice.com

Table 3. Dataset Statistics

Source	# positive reviews	# negative reviews
TripAdvisor	1139	229
OpenRice	3553	297
Twitter	8435	3381

3.2 *Lexicon Evaluation*

Lexicon evaluation was done manually by two annotators with Kappa value 0.729, which is considered substantial agreement. Both annotators judge the subjectivity and polarity for each candidate word. In subjective evaluation, annotators are asked to judge whether a candidate word is a sentiment word or not. Furthermore, for polarity evaluation, annotators are asked to judge whether a candidate word is positive or negative.

EXPANSION RESULTS From the lexicon expansion phase, all the three approaches can generate a number of candidate lexicons. SP and SP-POS generate a fair number of words as they use exact matching with patterns. SPo generates a large number of candidate words, because it uses word occurrences in sentences. The result of seed expansion process is shown in Table 4. This table shows the percentage (%inc) of lexicon increment relative to the initial lexicon (seed words).

Table 4. Seed Expansion Results

Dataset	%inc		
	SP	SP-POS	SPo
TripAdvisor	86%	132%	2624%
Twitter	203%	168%	5682%
Openrice	185%	172%	4740%

FILTERING RESULTS Tables 5 and 6 report the filtering result. We use two evaluation metrics; %inc to shows the number of new candidate words relative to the initial seed words and %acc to shows the accuracy of candidate words.

The opinion filtering results are shown in Table 5. As seen from the tables, after opinion filtering, SP generates candidate words with highest accuracy (89%) but with lowest expansion (23.98%). This is because SP generates specific sentiment patterns that not always occurred in the document (exact matching). On the other hand, SP-POS achieves 71.63%

accuracy but can expand the lexicon with the highest percentage at 105.33%. SP-POS can generate more candidate lexicon because it uses generalized patterns, which use part-of-speech information. SPo yields lowest accuracy (41.91%) with lexicon increment at 92.12%. SPo fails to generate candidate words with good accuracy because it rely on word occurrence in sentences, so that any words that frequently appear can become candidate sentiment words. The algorithm finds the correlation between words with assumption that a review sentence will contains more than one sentiment word. However, based on our observation, a review sentence does not always contain more than one sentiment words.

Table 5. Opinion Filtering Results

Dataset	SP		SP-POS		SPo	
	%inc	%acc	%inc	%acc	%inc	%acc
TripAdvisor	16.95%	97.5%	73.73%	75.60%	76.27%	41.51%
Twitter	16.75%	75.0%	101.83%	69.67%	86.13%	45.41%
Openrice	38.24%	94.5%	140.44%	69.63%	113.97%	38.80%
All	23.98%	**89.00%**	**105.33%**	71.63%	92.12%	41.91%

From the dataset perspective, SP and SP-POS generates best result with TripAdvisor dataset because it contains reviews with good sentence structure. SPo produces best result with Twitter because the algorithm does not count on the sentence structure and Twitter has the highest number of documents to construct correlation between seed words and candidate words.

The sentiment detection results are shown in Table 6. From the overall results, we can see that polarity detection accuracy for positive words is always better than negative words. This is because the dataset that we used in this study contains more positive documents then negative documents. Based on the results, we can see that our approach to detect polarity of a word produce consistent accuracy for all kind of dataset.

Table 6. Sentiment Detection Results

Dataset	TripAdvisor		Twitter		OpenRice	
	positive	negative	positive	negative	positive	negative
SP	91.20%	66.70%	90.50%	61.10%	91.60%	57.90%
SP-POS	77.70%	84.90%	76.20%	60.60%	84%	61%
SPo	56.50%	43.00%	52.90%	52.40%	50.30%	47.30%
All	**75.13%**	**64.87%**	**73.20%**	**58.03%**	**75.30%**	**55.40%**

4 CONCLUSION

In this paper, we propose our approaches in building domain specific sentiment lexicon for Indonesian language. Our main contributions are: (1) methods to expand sentiment lexicon using sentiment patterns; and (2) techniques to classify the polarity of a word using sentiment score.

The process start by translating English sentiment words to build seed lexicon. The seed lexicon is then expanded using senti-patterns (SP and SP-POS) and similarity with polar sentence (SPo) to produce candidate sentiment words. Finally, we apply two stages of filtering process, opinion filtering and sentiment detection to generate final list of expanded sentiment lexicon.

We test our proposed methods to build Indonesian sentiment lexicon for tourism domain with three kind of dataset which is different in the level of sentence structure. Yet, using the same techniques, it is also possible to implement this technique in other under-resourced languages, which can provide seed lexicon, POS tagger, and user reviews.

REFERENCES

1. Banea, C., Mihalcea, R., Wiebe, J.: A Bootstrapping Method for Building Subjectivity Lexicons for Languages with Scarce Resources. In: LREC (2008)
2. Brooke, J., Tofiloski, M., Taboada, M.: Cross-linguistic sentiment analysis: From English to Spanish. In: Proceedings of the 7th International Conference on Recent Advances in Natural Language Processing, Borovets, Bulgaria (2009) 50–54
3. Chesley, P., Vincent, B., Xu, L., Srihari, R.K.: Using verbs and adjectives to automatically classify blog sentiment. Training 580 (2006) 233
4. Esuli, A., Sebastiani, F.: SentiWordNet: A publicly available lexical resource for opinion mining. In: Proceedings of LREC (2006) 417–422
5. Kaji, N., Kitsuregawa, M.: Building Lexicon for Sentiment Analysis from Massive Collection of HTML Documents. In: EMNLP-CoNLL (2007) 1075–1083
6. Kamps, J., Marx, M.J., Mokken, R.J., De Rijke, M.: Using wordnet to measure semantic orientations of adjectives (2004)
7. Kanayama, H., Nasukawa, T.: Fully automatic lexicon expansion for domain-oriented sentiment analysis. In Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics (2006) 355–363

8. Kim, S.-M., Hovy, E.: Determining the sentiment of opinions. In: Proceedings of the 20th International Conference on Computational Linguistics (2004) 1367
9. Klebanov, B.B., Burstein, J., Madnani, N., Faulkner, A., Tetreault, J.: Building subjectivity lexicon(s) from scratch for essay data. In: Computational Linguistics and Intelligent Text Processing, CICLing 2012, LNCS, Springer (2012) 591–602
10. Liu, B.: Sentiment analysis and opinion mining. Synth. Lect. Hum. Lang. Technol **5** (2012) 1–167
11. Mihalcea, R., Banea, C., Wiebe, J.: Learning multilingual subjective language via cross-lingual projections. In: Annual Meeting of the Association for Computational Linguistics (2007) 976
12. Pantel, P., Pennacchiotti, M.: Espresso: Leveraging generic patterns for automatically harvesting semantic relations. In: Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics (2006) 113–120.
13. Peng, W., Park, D.H.: Generate adjective sentiment dictionary for social media sentiment analysis using constrained nonnegative matrix factorization. Urbana **51** (2004) 61801.
14. Pérez-Rosas, V., Banea, C., Mihalcea, R.: Learning Sentiment Lexicons in Spanish. In: LREC (2012) 3077–3081
15. Pisceldo, F., Manurung, R., Adriani, M.: Probabilistic Part-of-Speech Tagging for Bahasa Indonesia. In: The Third International MALINDO Workshop, co-located Event ACL-IJCNLP (2009)
16. Qiu, G., Liu, B., Bu, J., Chen, C.: Opinion word expansion and target extraction through double propagation. Computational Linguistics **37** (2011) 9–27
17. Terra, E., Clarke, C.L.: Frequency estimates for statistical word similarity measures. In: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology, volume 1 (2003) 165–172
18. Turney, P., Littman, M.L.: Unsupervised learning of semantic orientation from a hundred-billion-word corpus (2002)
19. Wan, X.: Using bilingual knowledge and ensemble techniques for unsupervised Chinese sentiment analysis. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing (2008) 553–561
20. Wan, X.: Co-training for cross-lingual sentiment classification. 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 1 (2009) 235–243
21. Wiebe, J., Mihalcea, R.: Word sense and subjectivity. In: Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics (2006) 1065–1072.

CLARA VANIA
FACULTY OF COMPUTER SCIENCE,
UNIVERSITAS INDONESIA,
INDONESIA
E-MAIL: <C.VANIA@CS.UI.AC.ID>

MOH. IBRAHIM
FACULTY OF COMPUTER SCIENCE,
UNIVERSITAS INDONESIA,
INDONESIA
E-MAIL: <MOCH.IBRAHIM@UI.AC.ID>

MIRNA ADRIANI
FACULTY OF COMPUTER SCIENCE,
UNIVERSITAS INDONESIA,
INDONESIA
E-MAIL: <MIRNA@CS.UI.AC.ID>

IJCLA VOL. 5, NO. 1, JAN-JUN 2014, PP. 73–86
RECEIVED 06/01/14 ACCEPTED 06/02/14 FINAL 05/04/14

Linguistic Features Predict the Truthfulness of Short Political Statements

VIVEK V. DATLA,[1] KING-IP LIN,[1] AND MAX M. LOUWERSE[2]

[1] *University of Memphis, USA*
[2] *Tilburg University, The Netherlands*

ABSTRACT

Checking the truth value of political statements is difficult. Fact checking computationally has therefore not been very successful. An alternative to checking the truth value of a statement is to not consider the facts that are stated, but the way the statement is expressed. Using linguistic features from seven computational linguistic algorithms, we investigated whether truth-false statements and the definitiveness with which the statement is expressed can be predicted using linguistic features. In a training set we found that both distinctiveness and truthfulness of the statement predicted linguistic variables. These variables corresponded to those mentioned in deception literature. Next, we used a new set of political statements and determined whether the same linguistic variables would be able to predict the definitiveness and truthfulness of the statement. Given the fact that the political statements are short, one-sentence statements, allowing for a large variability in linguistic variables, discriminant analyses showed that the function obtained from the training set allowed for an accurate classification of $57 - 59\%$ *of the data. These findings are encouraging, for instance for first analysis on the truth value and verifiability of political statements.*

1 INTRODUCTION

In a speech, Rick Santorum, runner for the Republican presidential nomination, 2011, said: *"[T]hey have voluntary euthanasia in the Netherlands, but half the people who are euthanized every year, and it's 10 percent of all deaths for the Netherlands, half of those people are euthanized*

involuntarily at hospitals because they are older and sick." Political statements like these might sound convincing and definite. Yet, it is unclear whether these statements are actually true. Finding out whether they are takes a considerable amount of investigative work.

One can investigate the number of deaths and euthanizations in the Netherlands and conclude that the statement is false. Such a task can perhaps be performed computationally, whereby a computational algorithm interprets a statement, tracks down the facts, and compares the truth value of these facts. However, a successful algorithm is not yet on the computational linguistic horizon [1]. An alternative might lie not so much in identifying the truth value of the facts being stated, but in investigating the style a statement is expressed in.

Speakers generally follow guidelines for a smooth conversation, summarized by [2] in four maxims of communication. The maxim of quantity postulates that the speaker should not say more, or less, than what is needed, the maxim of relation postulates the speaker should be relevant to the purposes of the conversation, and the maxim of manner postulates the speaker should be clear and orderly. Importantly for the current paper, the maxim of quality states "do not say what you believe to be false" and "do not say that for which you lack adequate evidence".

Some political statements, like the one by Rick Santorum quoted earlier, happen to be false and lack any evidence. We can investigate whether the style of the statement gives away cues that indicate the speaker is not quite sure that he says what s/he believes to be true and only says that for which s/he has adequate evidence. Such an investigation on determining the definitiveness and the truthfulness of political statements is the topic of this paper.

A politician may use a formal style of language in order to create the impression that s/he presents precise, objective information, while s/he really wants to hide the exact details of his/her policy [3]. If the speakers purposefully violate Grice's maxim of quality, they leave non-linguistic and linguistic footprints in their attempts to hide the truth [4].

There is of course a distinction between not quite telling the truth and actual deceiving. A speaker might not tell the truth because s/he does not have the facts readily available but needs to say something, or because the facts cannot be stated because of political, strategic, or social reasons (maxim of relevance). However, regardless of the motivating behind hiding the truth, the non-linguistic and linguistic footprints in the speaker's attempts to hide the truth might actually be the same in deception and non-truth telling: in both cases the speaker has an increased cognitive

load because of not wanting to tell the truth, even though the motivations behind not telling the truth might be different.

There are various studies that report on verbal and non-verbal footprints left behind in deception. For instance, in the context of police interviews people telling a lie used fewer illustrations, had an increase in pauses, and an increase in the latency period, most likely due to the increased cognitive load, i.e., the focus on both not telling the truth and the actual telling the truth [5]. A review of 116 deception studies by [4] showed that lies had more verbal and vocal uncertainty, less verbal and vocal immediacy, were more ambivalent, less plausible and had less logical structure, with less contextual embedding.

DePaulo et al. [4] found that deceptive communication had fewer first-person singular pronouns, fewer third-person pronouns, more negative emotion words (e.g., hate, anger, enemy), fewer exclusive words (e.g., but, except), and more motion verbs (e.g., walk, move, go). [6] investigated statements from speakers who were asked to be deceptive in asynchronous computer-mediated communication (CMC). Participants were asked to write stories on five different topics, with one group of participants asked to not tell the truth. The untrue stories consisted of fewer words, fewer first person pronouns, more questions, and more words pertaining to senses (e.g., see, listen). This finding is consistent with [7] findings.

DePaulo et al. [4] argued that the motivation to not tell the truth plays an important role in the linguistic features of the statements. The settings of typical laboratory experiments lack a participant's motivation. That is, when untrue statements in society are investigated the stakes are higher. In the case of a politician not telling the truth could mean the difference between being considered credible or not, between voted into office or not. It can therefore be expected that the verbal footprints are more easily to detect than when speakers are less motivated to tell or hide the truth.

Much of the literature investigating linguistic cues in statements where the speaker says what he/she believes to be false uses passages or paragraphs. Indeed, if verbal footprints of not telling the truth are left, they will be more obvious when more data from a speaker is available. However, there often is only limited data available. Twitter messages, Facebook comments, or other brief comments do not allow for lengthy text. In addition while the overall message of a conversation may not be false, individual statements within it may be inaccurate or fabricated. Therefore, even though it might be easier to detect deception in large text samples [6,

7], the current study investigated whether linguistic cues from short political statements also predict definitiveness and truthfulness.

We used statements from politifact.com [8], because they consist of recent relatively short—on average approximately 18 words—statements from a variety of politicians that are checked on their accuracy.

Politifact is a project of the Tampa Bay Times, a Florida-based media organization, which won the prestigious Pulitzer Prize for its fact checking during the 2008 presidential election campaigns. Statements include those by members of congress, state legislators, governors, mayors, the president, cabinet secretaries, lobbyists, people who testify before Congress etc. Politifact uses the following categories to represent the truthfulness of a statement:

- True: The statement is accurate
- Mostly true: The statement is accurate but needs clarification
- Half true: The statement is partially accurate
- Mostly false: The statement contains an element of truth but ignores some information
- False: the statement is not accurate
- Pants on fire: The statement is absurdly false

These six categories allow for two pieces of information. First, statements can be categorized in true and false statements. However, we can do this on the basis of a strict criterion (true versus false and pants on fire) or a more lenient criterion (true and mostly true versus false, mostly false and pants on fire). We also did a different analysis by making a distinction in truthfulness (regardless of whether the distinction is made based on a strict or lenient criterion) is the definitiveness of the truth or false value. Half true statements are half true and half false, and therefore are not definitive; it can be expected that stylistic cues give away to what extent the speaker expresses a statement in a more (true/false) or less (half true/false) way; see Table 1.

The Politifact statements and the 2 definitiveness $\times$ 2 truthfulness (strict and lenient) categories allow us to train the linguistic characteristics of each political statement on its respective category. The resulting categorization function from this training can then be tested on a test set of new political statements.

2 Linguistic Features

To train the model, a wide range of computational linguistic dimensions was selected, including syntactic and semantic algorithms. These algo-

Table 1. Overview of the politifact categories

		True	False
Definitive	**Strict**	true	false, pants on fire
	Lenient	true, mostly true	false, mostly false, pants on fire
Indefinitive		half-true	

rithms can, generally, be classified into general structural (e.g., word count), syntactic (e.g., connectives) and semantic (e.g., word choice) dimensions of language. Five different algorithms were used, categorized in Figure 1.

For general linguistic features, we used the frequency of 67 linguistic features used in [9]. These features in primarily operate at the word level (e.g., parts-of-speech) and can be categorized as tense and aspect markers, place and time adverbials, pronouns and pro-verbs, questions, nominal forms, passives, stative forms, subordination features, prepositional phrases, adjectives and adverbs, lexical specificity, lexical classes, modals, specialized verb classes, reduced forms and dispreferred structures, and co-ordinations and negations.

For WordNet [10] 150,000 words in 44 base types were selected, including 25 primitive groups for nouns (e.g. time, location, person etc.), 15 for verbs (e.g. communication, cognition, etc.), 3 groups of adjectives and 1 group of adverbs.

The linguistic category model (LCM) gives insight into the interpersonal language use. The model consists of a classification of interpersonal (transitive) verbs that are used to describe actions or psychological states and adjectives that are employed to characterize persons. In order to capture the various emotions expressed by the statement we have used the emotion words given by [11], classified into two classes broadly basic emotions (anger, fear, disgust, happiness etc.) and complex emotions (guilt, pity, tenderness etc.). The basic emotions indicate no cognitive load hence they are also called as raw emotions, whereas the complex emotions indicate cognitive load.

Interclausal relationships were captured using [12] parameterization, including positive additive, (also, moreover), negative additive (however, but), positive temporal (after, before), negative temporal (until), and causal (because, so) connectives. In order to get the frequencies of the words we have used CELEX database [13]. The CELEX database consists of 17.9

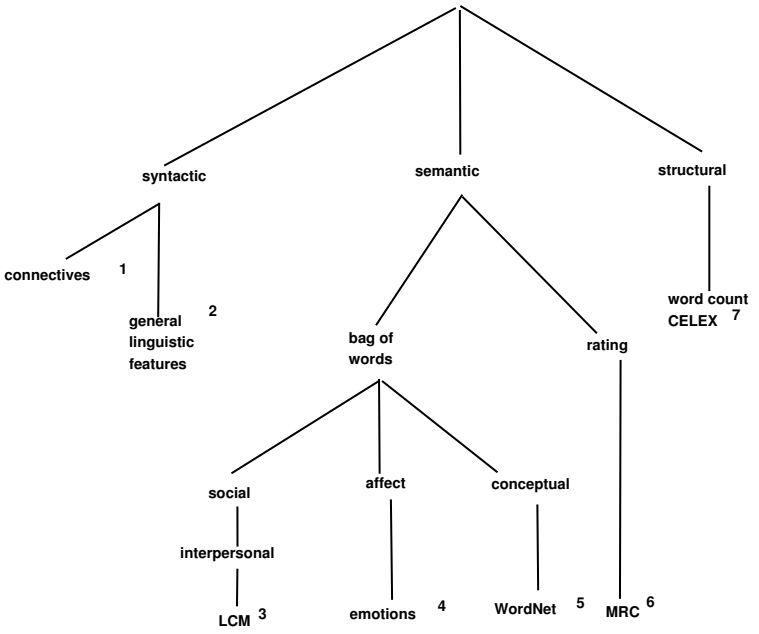

Fig. 1. Overview of computational linguistic algorithms used. [1]Louwerse (2002), [2]Biber (1988), [3]Semin & Fiedler (1991), [4]Johnson-Laird & Oatley (1989), [5]Miller et al. (1990), [6]Coltheart (1981), [7] Baayen, Piepenbrock, & Gulikers (1995)

million words taken from both spoken (news wire and telephonic conversations) and written (newspapers and books) corpora.

In addition, we used the MRC Psycholinguistic Database [14], to get linguistic measures such as familiarity, concreteness, imaginability and meaningfulness. For each political statement collected from Politifact.com [8] we processed the features for the 7 computational linguistic algorithm, normalized for the number of words per statements, and the scores were treated as a vector.

3 TRAINING

A total of 1576 political statement were downloaded from Politifact.com sentences) as training data. These political statements came from April, 2012. The break down of the various categories for the training data are

as follows: 21% true, 19% mostly true, 22% half true, 15% mostly false, and 23% false.

4 Results and Discussion

A mixed effects regression model was run on each of the linguistic features with the category as independent variable and individual speaker as a random factor, to avoid any speaker bias [13]. The model was fitted using the restricted maximum likelihood estimation (REML) for the dependent. F-test denominator degrees of freedom were estimated using the Kenward-Roger's degrees of freedom adjustment to reduce the chances of Type I error [15].

We predicted that those patterns found in the deception studies discussed earlier would be found in the computational linguistic scores.

4.1 *Truthfulness*

As the results of the mixed effect regression model in Table 3 shows, truthfulness explained the variance of 20 linguistic variables, with similar patterns and variables for the strict and lenient categories.

The results show that various verb categories (cognitive, communicative, modals, predicated modals) explain the difference in truthfulness, a finding in line with the idea that these verbs increase verbal immediacy and cognitive load. The results in Table 3 are also in line with [4, 6, 7, 16, 17] who have all shown that negative emotions are related to deception, in our analysis emotions came to significance while classifying between true and false in the lenient case.

To put the findings reported in Table 3 in perspective, we linked each of the findings to a corresponding finding in the deception literature using the studies reported in Table 2.

4.2 *Definitiveness*

As the results of the mixed effect regression model in Table 4 shows, definitiveness explained the variance of 20 linguistic variables. Importantly, the direction of the significant linguistic features is similar across the strict and lenient categories. In both strict and lenient cases, variables such as concreteness, word count, variety in the tokens in the statement, positive connectives, has shown up to be significant. The results indicate that if the statement is more concrete or has high imagery score then

Table 2. Relevant Deception Studies. The number listed in the first column corresponds to the number used in the first column of the results.

ref	Literature
1	Newman et al., (2003) [7]
2	Tausczik & Pennebaker, (2010) [16]
3	Hancock, et al, (2007) [6]
4	DePaulo et al., (2003) [4]
5	Toma & Hancock (2010) [17]
6	Louwerse, et al., (2010) [18]

Table 3. Variables that explain truthfulness of a political statements. First columns gives references to deception literature. Superscript in the second column gives reference to the computational linguistic model. Last columns give the t-values to show the direction of the effect (**: $p \leq 0.01$, *: $0.01 \leq p \leq 0.05$).

		Condition	
Ref	Language Features	Strict	Lenient
1, 3, 4 ,5	Positive connectives[1]	-2.13*	-3.04**
1, 2, 3, 4, 5	Caused Emotions[4]	1.07	2.01*
1 ,2	Social verbs[5]	1.59	2.90**
1, 2, 3, 4, 5	Emotion verbs[5]	1.17	2.93**
1, 2, 3, 4, 5	Cognitive verbs[5]	1.69	3.56**
1, 2	Communication verbs[5]	2.40*	4.18**
1	Possession verbs[5]	1.99*	3.03**
2	Prepositions[2]	-2.6**	-3.49**
	Second person pronouns[2]	2.59*	3.29**
1	Modal verbs[2]	1.74	3.8**
	Numbers[2]	-2.02*	-2.99**
	CELEX frequency[7]	-2.57*	-3.2**
1, 3, 4 ,5	Temporal positive connectives[1]	-1.14	-3.24**
1, 3, 4 ,5	Additive positive connectives[1]	-2.17*	-2.9**
1, 3, 4 ,5	Temporal connectives[1]	-1.16	-2.81**
	Private verbs[2]	2.34*	2.04*
1	Predicated Modality[2]	1.50	3.05**
	Emphatics[1]	-2.17*	-1.71
3,5	Brown Frequency[6]	2.34*	2.1*

it is more likely to be an indefinite(half true) statement, than a definite (true/false) statement. This corresponds with the literature. [16] indicate pronoun use, emotionally toned words, and prepositions and conjunctions that signal cognitive load are linked to behavioral and emotional

Table 4. Language features that help in detecting definitiveness of the truth value in the statement. First columns gives references to deception literature. Superscript in the second column gives reference to the computational linguistic model. Last columns give the t-values to show the direction of the effect (**: $p \leq 0.01$, *: $p \leq 0.05$).

		Condition	
Ref	Language Features	Strict	Lenient
3, 5	Word Count[7]	-3.96**	-3.79**
4	Token types[7]	3.23*	3.6*
	Concreteness with type[6]	-2.08*	-2.3*
	Concreteness with token[6]	-2.71*	-2.7*
1, 3, 4, 5	Positive connectives[1]	-2.64*	-2.4*
1, 3, 4 ,5	Additive positive connectives[1]	-2.30*	-2*
1, 3, 4, 5	Additives[1]	-2.54*	-1.85
	Consumption Verbs[5]	-2.20*	-2.04*
1	Communication Verbs[5]	-2.08*	-2.2*
1, 2, 3, 4, 5,6	State Action Verbs[3]	-2.49*	-1.57
	Public Verbs[2]	2.48*	2*
1	Prepositions[2]	-3.66**	-2.9**
1	Auxiliary Verbs[2]	2.54**	1.64

outcomes. Similarly, [7] indicate that self-references, negative emotion words and cognitive complexity play an important role when people try to deceive. In our analysis we find that connectives help in classifying between definite and indefinite sentences, with higher frequencies of connectives yielding more complex sentences and consequently higher cognitive load.

5 TESTING

A total of 1597 political statements from January 2013 were downloaded as a test set. The breakdown of the various categories for the test data are as follows: 14% true, 30.8% false, 16.7% mostly-true, 15.2% mostly-false and 22.4% half-true statements.

As the sizes of the categories of statements are not equal, this makes the discriminant analysis classify all the instances of the classes to the post popular class in data. In order to make sure that we have not made a special selection of statements that make the two classes, we conducted 1000 Monte Carlo simulations on both truthfulness and definiteness cases, and also in their strict and lenient sub cases, to pick two equal classes, for

discriminant analysis. This also helps to improve the robustness of our classification.

In order to make sure that we do not overfit the model with all the variables that came to significance in the mixed effect regression model, we selected a random set from the 1000 sets created by the Monte Carlo simulations from the strict case of both, True Vs. False, and Definite (true/false) Vs. Indefinite(half true) classification. Then selected the smallest set of variables in each case that classify them into their respective classes. In case of True Vs. False, we got the best result for the classification by taking the variables, social verbs, modal verbs, numbers, private verbs and brown frequency.

In the case of classification between definite and Indefinite statements, we got the best classification on the random set, by using the variables word count, prepositions, token types, concreteness with token types, positive connectives and additives.

We have used the same variables for the classification in the strict and the lenient case. The results of the classification are in shown in the Table 5 and Table 6. In case of classifying between the True and False statements, communication verbs (announce, argue, express etc.), social verbs (observe, upgrade, permit etc.) and modal verbs (can, could, may etc.) which indicate cognitive load and verbal immediacy were significant. These categories of words are also referred in the deception literature [7]; [16], indicating that even in constricted context these categories help in classifying true and false statements.

In case of classifying between the definite and indefinite statements in both strict and lenient case, we are able to classify significantly between the cases with accuracy of about 58% on an average over 1000 runs. The classification is more significant in the strict case compared to the lenient case. Table 5 shows the results averaged over 1000 runs for strict and lenient case, which we obtained for classifying the statements into true and false. The results indicate that we need more context in classifying true and false statements, as in the lenient case we are able to classify between the true and false cases more significantly.

Table 6 indicates the accuracy for classifying between definite and indefinite statements in strict and lenient cases. Even though the accuracy on average over 1000 runs is only 59% given that chance is 50%, it is a significant result as we are analyzing the statements with very few words. The classification in the strict case and the lenient cases the classification are significant. The significance of classification is smaller in the lenient

Table 5. Classification between true and false statements

Strict	True	False	Overall
True	42.6%	57.4%	56.8%
False	28.9%	71.1%	
$\chi^2(1, N = 5) = 16.89, p < 0.024$			
Lenient	True	False	Overall
True	44.7%	55.3%	55.9%
False	32.8%	67.2%	
$\chi^2(1, N = 5) = 23.67, p < 0.0041$			

Table 6. Classification between Definite and Indefinite Statements

Strict	True	False	Overall
True	59.2%	41.8%	57.9%
False	43.5%	56.5%	
$\chi^2(1, N = 6) = 27.54, p < 0.002$			
Lenient	True	False	Overall
True	56.6%	43.4%	55.9%
False	44%	56%	
$\chi^2(1, N = 6) = 17.4, p < 0.04$			

case, this is due to the fact the mostly-true, half-true, and mostly false are contiguous on the deception scale.

6 Conclusion and Future Work

This study investigated whether linguistic features can be predicting the truth value and the definitiveness of the truth value in short political statements. Using a training set of one-sentence political statements, we investigated whether linguistic features obtained from seven computational linguistic algorithms across syntactic, semantic and structural dimensions showed a relationship with truthfulness and definitiveness. We thereby used a strict criterion and a more lenient criterion. Results showed that a similar set of linguistic variables explained these categories. In a testing set of a new set of political statements we then tested whether the same variables explained the truthfulness and definitiveness categories. The results showed they did, supporting the conclusion that linguistic features can help determining to what extent political statements are true and to what extent this decision can be made with certainty.

Interestingly, the linguistic variables that have been identified as predictors of truthfulness and definitiveness match the variables that have been identified as linguistic cues to deception. A large body of literature has investigated whether deceivers leave linguistic footprints in their deception. However, rather than with the purpose of deceiving, we assume that the speakers of the short political statements of the current study had valid reasons to not quite tell the truth. The findings reported here might not overwhelm. A 55-60% discrimination score is not high. Yet, the fact that such a score is significant, that the variables behind the score are consistent across training and testing, and that this score is obtained with a small language unit (about one sentence), makes the findings reported in the current study remarkable nonetheless.

The computational linguistic means of predicting truthfulness and definitiveness should certainly not stand on their own in evaluating short political statements. However, they can fulfill a supporting role. Computational linguistic algorithms such as the ones discussed can identify whether statements can be easily checked and whether there is an initial likelihood that supporting evidence can or cannot be found. In the day and age of Twitter and Facebook with many short statements, having a tool that filters whether a statement is the truth and nothing but the truth or not, might be very welcome.

REFERENCES

1. Mihalcea, R., Strapparava, C.: The lie detector: Explorations in the automatic recognition of deceptive language. In: Proceedings of the ACL-IJCNLP 2009 Conference Short Papers. ACLShort '09, Stroudsburg, PA, USA, Association for Computational Linguistics (2009) 309–312
2. Grice, H.P.: Logic and conversation. Syntax and Semantics **3** (1975) 41–58
3. Heylighen, F., Dewaele, J.M.: Variation in the contextuality of language: An empirical measure. Foundations of Science **7**(3) (2002) 293–340
4. DePaulo, B.M., Lindsay, J.J., Malone, B.E., Muhlenbruck, L., Charlton, K., Cooper, H.: Cues to deception. Psychological bulletin **129**(1) (2003) 74
5. Vrij, A., Edward, K., Roberts, K., Bull, R.: Detecting deceit via analysis of verbal and nonverbal behavior. Journal of Nonverbal Behavior **24**(4) (2000) 239–263
6. Hancock, J.T., Curry, L.E., Goorha, S., Woodworth, M.: On lying and being lied to: A linguistic analysis of deception in computer-mediated communication. Discourse Processes **45**(1) (2007) 1–23
7. Newman, M.L., Pennebaker, J.W., Berry, D.S., Richards, J.M.: Lying words: Predicting deception from linguistic styles. Personality and Social Psychology Bulletin **29**(5) (2003) 665–675

8. TampaBay Times: Politifact.com. Published online (April 2012)
9. Biber, D.: Variation across speech and writing. Cambridge University Press (1991)
10. Miller, G.A., Beckwith, R., Fellbaum, C., Gross, D., Miller, K.: Five papers on WordNet. In Fellbaum, C., ed.: WordNet: An Electronic Lexical Database. MIT Press (May 1998)
11. Johnson-laird, P.N., Oatley, K.: The language of emotions: An analysis of a semantic field. Cognition and Emotion **3**(2) (1989) 81–123
12. Louwerse, M.: An analytic and cognitive parametrization of coherence relations. Cognitive Linguistics **12**(3) (2001) 291–316
13. Baayen, H.R., Piepenbrock, R., Gulikers, L.: The CELEX lexical database. Release 2 (CD-ROM). Linguistic Data Consortium, University of Pennsylvania, Philadelphia, Pennsylvania (1995)
14. Coltheart, M.: The MRC psycholinguistic database. The Quarterly Journal of Experimental Psychology **33**(4) (1981) 497–505
15. Littell, R., Stroup, W., Freund, R.: SAS for Linear Models,: Fourth Edition. Wiley series in probability and statistics. SAS Institute (2002)
16. Tausczik, Y.R., Pennebaker, J.W.: The psychological meaning of words: LIWC and computerized text analysis methods. Journal of Language and Social Psychology **29**(1) (2010) 24–54
17. Toma, C.L., Hancock, J.T.: Reading between the lines: Linguistic cues to deception in online dating profiles. In: Proceedings of the 2010 ACM Conference on Computer Supported Cooperative Work. CSCW '10, New York, NY, USA, ACM (2010) 5–8
18. Louwerse, M.M., Lin, K., Drescher, A., Semin, G.: Linguistic cues predict fraudulent events in a corporate social network. In Ohlsson, S., Catrambone, R., eds.: Proceedings of the 32nd Annual Conference of the Cognitive Science Society, Austin, TX, The organization, Cognitive Science Society (2010) 961–966

VIVEK V. DATLA
DEPTARTMENT OF COMPUTER SCIENCE,
UNIVERSITY OF MEMPHIS,
TN 38152, USA
E-MAIL: <VVDATLA@MEMPHIS.EDU>

KING-IP LIN
INSTITUTE FOR INTELLIGENT SYSTEMS,
UNIVERSITY OF MEMPHIS,
365 INNOVATION DRIVE, MEMPHIS, TN 38152, USA
E-MAIL: <DAVIDLIN@MEMPHIS.EDU>

MAX M. LOUWERSE
TILBURG CENTRE FOR COGNITION AND COMMUNICATION (TICC),
TILBURG UNIVERSITY,
PO BOX 90153, 5000 LE, TILBURG, THE NETHERLANDS
E-MAIL: <MLOUWERSE@MEMPHIS.EDU>

Syntax, Parsing, and Tagging

IJCLA VOL. 5, NO. 1, JAN-JUN 2014, PP. 89–103
RECEIVED 06/01/14 ACCEPTED 06/02/14 FINAL 16/06/14

Label Pre-annotation for Building Non-projective Dependency Treebanks for French

OPHÉLIE LACROIX, DENIS BÉCHET, AND FLORIAN BOUDIN

Université de Nantes, France

ABSTRACT

The current interest in accurate dependency parsing make it necessary to build dependency treebanks for French containing both projective and non-projective dependencies. In order to alleviate the work of the annotator, we propose to automatically pre-annotate the sentences with the labels of the dependencies ending on the words. The selection of the dependency labels reduces the ambiguity of the parsing. We show that a maximum entropy Markov model method reaches the label accuracy score of a standard dependency parser (MaltParser). Moreover, this method allows to find more than one label per word, i.e. the more probable ones, in order to improve the recall score. It improves the quality of the parsing step of the annotation process. Therefore, the inclusion of the method in the process of annotation makes the work quicker and more natural to annotators.

1 INTRODUCTION

Dependency-based methods for syntactic parsing have become increasingly popular in natural language processing in recent years [1]. Most proposed approaches for dependency parsing are data-driven and require large sets of manually annotated sentences, called treebanks. Obviously, annotating such data is very costly and time consuming. One usual way to alleviate the burden of manual annotation is to automatically pre-annotate the data, so that annotators only have to validate pre-annotated sentences.

Available treebanks for French are constituency treebanks that were converted into dependency ones, e.g. [2]. While the conversion method

that was used is able to generate non-projective dependency structures [3], constituency trees are always projective. It is then not surprising that converted dependency trees do not reflect non-projective relations. As a consequence, data-driven dependency parsers trained on these converted treebanks fail to produce non-projective dependency structures. Developing a French treebank that contains non-projective trees is therefore necessary for improving parsing accuracy.

A relevant work in this direction is that of Dikovsky [4]. Dikovsky proposed a framework for jointly constructing a treebank and a grammar for French (CDGFr). The result of this work is a treebank consisting of 3030 sentences annotated with dependency structures (projective and non-projective) along with an annotation environment called CDG Lab [5].

In CDG Lab, annotating a sentence is a three-step process. The first step is the manual pre-annotation of the sentence. It consists of selecting either a grammatical class or a dependency label for each word through a selection form. The computational time of the second step, the dependency analysis, is exponentially proportional to the number of selected labels per word. So, the selection of one label per word restrains the search space of the grammar-based analysis and then make the analysis practical. The last step is a manual validation.

Filling the selection form is a tedious task for the annotators. In this paper, we propose to automatize the sentence pre-annotation step in order to alleviate the work of the annotator through the building of large dependency treebanks. We replace the selection form by a method using a maximum entropy Markov model to provide dependency labels and selecting one or more dependency labels for each word depending on their probability score. The method reaches the label accuracy scores of a standard data-driven parser, MaltParser [6], in addition to providing more than one label per word. Moreover, this number can be controlled to impact positively the grammar-based dependency parsing. Then, the parsing step becomes a trade-off between the preservation of a high recall score and acceptable parsing time in order to reduce the error correction rate and therefore the whole time of the annotation process. Finally, the use of the automatic label pre-annotation tool facilitates and speeds up the creation of new large French dependency treebanks containing both projective and non-projective trees.

The rest of this paper is organized as follows. In Section 2, we first review the related work on methods for building dependency treebanks. In Section 3, we present the background of dependency parsing and de-

scribe the process of annotating a sentence in CDGLab. Then, we detail our automatic pre-annotation method in Section 4 and examine the results in Section 5. Finally, we discuss the benefit of the pre-annotation process in the building of dependency treebanks in Section 6 and conclude.

2 Related Work

Dependency treebanks are now available for many languages [7]. Depending on the available tools and resources, sentences may be fully or partially annotated with Part-Of-Speech tags and dependency relations.

On the one hand, conversion methods can be applied to convert constituency treebanks to dependency ones. The converted treebanks require no or very few corrections after conversion due to the quality and quantity of the syntactic and grammatical information given by the original constituency treebanks. Such kind of method has been applied to French and also to English through the building of the Penn Treebank [8].
On the other hand, the development of large treebanks requires several automatic and manual steps. The automated steps occur on various levels of analysis (segmentation, POS-tagging, parsing) and require the validation of the annotators. The benefit over the conversion methods is to be independent from other formalisms like the constituent one. For example, the annotation process of the Prague Dependency Treebank [9] includes its own level of analysis (e.g. morphological, analytical, tectogrammatical). Furthermore, many tasks on treebanks building exploit the performance of a data-driven dependency parser, such as the MaltParser [6] : this is the case for various work (e.g. for Indonesian [10], Latin [11], Turkish [12]) to pre-annotate their data.

An example of a dependency treebank built from scratch is the speech dependency treebank for French. Here, Cerisara et al. [13] perform a manual segmentation step before the tagging and parsing steps. Nevertheless, in our work, we do not want to use a converted treebank to train a model because it does not include non-projective trees. And, in order to provide trees consistent with the CDGFr, we do not use a data-driven parser.

The automatic pre-annotation process often includes POS-tagging. In the case of non-projective dependency parsing, Alfared et al. [14] showed that the upstream disambiguation of POS-tagging is not sufficient to disambiguate in grammar-based parsing. Our annotation process uses a pre-annotation step to select one or more dependency labels for each word

as well as the POS. The spirit of this particular task is in the way of supertagging [15]. However, here we want to predict a single information, the dependency label rather than a complex structure like a type (Categorial Grammar) or an elementary tree (Tree-Adjoining Grammar). The complexity of this task is halfway between POS-tagging and supertagging.

3 ANNOTATION FRAMEWORK

3.1 *Background*

The dependency representation allows representing both projective and non-projective relations that exist in natural languages. A dependency tree containing at least one non-projective dependency is called non-projective. For a dependency $h \xrightarrow{l} d$ the label l represents the function binding the head h with the dependent d. Such a dependency is non-projective if at least one word located between the head and the dependent of the dependency does not depend on the head. Figure 1 presents an example of a non-projective dependency tree where the non-projective dependency connects a verb with a distant clitic.

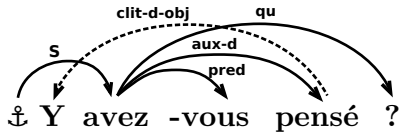

Fig. 1. Dependency tree for the sentence "Y avez-vous pensé ?" ("Did you think about it ?") The clitic "y" ("it") depends on the verb "pense" ("think"). It refers to the indirect object of the verb (dative case).

The categorial dependency grammar of French [4], used to build the trees, has 116 different dependency labels. All dependencies with the same label describes specific information about the syntax of French. Most of the dependency labels can be gathered into larger syntactic groups describing more general information. For example, objects are separated into 7 dependency labels differentiating the grammatical cases (dative, accusative, etc.). Most of the dependency labels (89) are exclusively associated with projective dependencies. But some of them can be associated both with projective and non-projective dependencies. Among the 23

dependency labels that can be combined with non-projective dependencies, the most frequent ones are clitics, negatives, objects, reflexives and copredicates. Four dependency labels are exclusively associated to non-projective dependencies, they are particular cases of aggregation, copula, comparison and negation.

The categorial dependency grammar of French, the non-projective dependency treebank and the parsing and treebank development environment that we use in this study is not yet publicly available, we have recovered them directly from the authors. In our work we use a treebank made up of sentences of various grammatical styles. A large part of these sentences (64%) were initialy used to develop the grammar of French. The whole corpus gathers several corpora composed of sentences from newspaper, 19^{th} and 20^{th} century literary works and plain language. We will call the joining of these treebanks the CDG Treebank. It is composed of 3 030 sentences (42691 words). Each sentence is paired with a dependency tree. The number of non-projective trees reaches 41.2% of the CDG Treebank. But, among all the dependencies, the non-projective ones represent 3.8% of all dependencies. The rate of non-projective dependencies varies from 1.3% to 4.9% according to the corpus.[1]

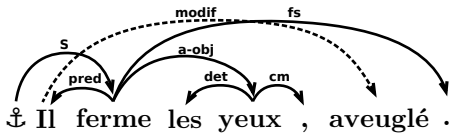

Fig. 2. Dependency tree for the sentence "Il ferme les yeux, aveuglé." ("He closes his eyes, blinded."). The modifier "aveuglé" ("blinded") depends on the subject "il" ("he").

3.2 *Annotation Process*

The annotation process we propose includes four steps :

– sentence segmentation;

[1] The sentences that were used to develop the grammar were choosed to cover all the syntactic phenomenon of French including the non-projective ones. Consequently, the rate of non-projective dependencies is more significant in the development corpus.

– automatic label prediction;
– dependency analysis;
– validation of the dependency tree.

The sentence segmentation is performed through a segmentation module which selects the longer lists of tokens recognizable by the lexical version of the CDGFr[2]. Here, we focus on the second step for which the role is to select the proper dependency label of each word[3]. Here, we call "the label of a word" the label of the dependency coming from the head to the word that is the dependent of the dependency. These labels correspond to the grammar's categories. For example, the labels of the words for the sentence used in Figure 2 are shown in Figure 3.

Il	**ferme**	**les yeux**	**,**	**aveuglé**	**.**
pred	S	det a-obj	cm	modif	fs

Fig. 3. The corresponding labels of the words for the sentence "Il ferme les yeux, aveuglé."

In addition, each word is associated with a set of possible grammatical classes. The grammatical classes are extended tags (compared to classical part-of-speech tags) used by the CDGFr to categorize the words. The possible labels depend on the set of the possible classes of a word. Among these given possibilities, our goal is to select one or more labels for each word that are consistent with the categorial dependency grammar. The automatic procedure of this particular step is explained in detail in Section 4.

This pre-annotation step reduces the ambiguity of the next step, the grammar-based dependency analysis. Here, a CKY-based algorithm is applied to find all the possible dependency trees for the sentence. With the label pre-annotation, the analyser only considers the rules in adequacy with the selected categories (labels). This way, the number of generated dependency tree candidates greatly decreases. For example, the analysis

[2] A recognizable list of tokens is a list constituting a lexical unit and not included in a black list which excludes some frequent errors of concatenation.

[3] We call words the possible combination of tokens that form a lexical unit. For instance, "Président Bill Clinton" has three tokens but corresponds to one word.

of the sentence presented in Figure 2 generates 1518 (projective or non-projective) dependency trees without restrictions on labels. Selecting the proper labels reduces the number of possible dependency trees to 2.

Finally, the fourth step of the annotation process is validation. The task of the annotator involves annotating positively or negatively the dependencies of the resulting dependency tree and selecting the proper label and segmentation of the words for which a wrong label or segmentation was selected. Afterwards, a new analysis (iteration step) is performed taking account the annotations to approach the correct dependency tree (consistent with the grammar). This step can be performed as often as necessary and can include again the different steps of the pre-annotation process.

4 SENTENCE PRE-ANNOTATION

Automatic label pre-annotation is the core of our annotation process and requires information about words and their grammatical context. Accordingly, we start by tagging the Part-Of-Speech tags.

4.1 *POS-Tagging*

The categorial dependency grammar makes use of 18 grammatical classes to categorize the words (e.g. noun, verb) and 10 for punctuation marks (e.g full stop, semicolon). The disjunction of some classes (e.g punctuation, particular verb types) is not necessary from a tagging point of view and can be ambiguous. Moreover, this tagset is not in adequacy with the tagset standardly used in French POS-taggers. Thus, in order to use a standard POS tagger and a tagset standardly used by the (French) community, we decided to convert our tagset into the TREEBANK+ tagset. This tagset consists in 28 tags extended from the classical tags used by the French Treebank [2], known to be efficient for parsing [16]. Furthermore, this tagset is used by MElt, a well studied French POS-tagger that achieves more than 97% accuracy on French [17].

Most of the grammatical classes correspond to TREEBANK+ tags, but some classes (e.g expletives, collocations, partitives) have no equivalent tags. These ones would make a direct conversion ambiguous. Therefore, we decided to conduct a mixed conversion. First, we tag automatically the whole corpus with the MElt tagger. Second, we correct the tags using basic rules for correction referring to the (non-ambiguous) original

grammatical classes annotated in the CDG Treebank. The rate of correction on the tagset conversion reaches 6%. The most frequent errors are due to the ambiguity existing with adjectives acting as common nouns or past participle verbs acting as adjectives. Furthermore, some errors appear because of the differences between the sentences of the training corpus, a variant of the French Treebank [2], used by the MElt tagger and the sentences of the CDG Treebank [4]. The newly converted data are used in the label tagging experiments.

4.2 *Label Pre-Annotation*

Here, the goal is to find the labels but not the dependencies associated with the words. This automatic step should alleviate the work of the annotators. We need to use a rapid method to conduct the tagging. The parsing methods, trying to find both the label and the dependency, achieve equivalent scores (label accuracy) to these obtained by a method dedicated to tagging. However, we want to produce, for each word, a restricted list of the best labels with their probability scores. Therefore, among the probabilistic graphical models we choose the maximum entropy Markov model (MEMM) [18] to achieve this task because of its speed and the fact that the words are tagged independently[5].

To predict the labels, we try different combinations of features and test results. The features result in a combination of information from the lexical and grammatical context (a window size of 7 around the words and of 11 around the POS-tags). Then, we retrieve the 20 best labels for each word from the tagging. The list of labels is pruned from the labels which are not in the list of possible labels.

4.3 *Label Sorting*

Our model allows to keep control over the number of labels assigned to each word. In order to reduce the ambiguity, we want to eliminate the bad

[4] One of the problems is that the training corpus contains very few imperative sentences and the CDG Treebank contains significantly more. Then, MElt is not able to find most of the imperative verbs. A lot of imperative verbs are tagged as indicative verbs. Others are tagged as nouns because this conjugated form are often located at the start of sentences with a first capital letter. and often tags the personal pronoun “tu” (“you”) as a verb because “tu” is also a conjugated form of the verb “taire” (“keep quiet”).

[5] We use the software Wapiti [19] which is able to deal with a large tagset.

labels (i.e. the less probable ones) from the list of possible labels while preserving a high recall score. Each label (associated with a word) gets a probability score from the pre-annotation step. So, for a word, the idea is to eliminate the labels for which the probability score p_{max} is lower than $\alpha.p_{max}$ where p_{max} is the probability of the best label (the more probable one) and $\alpha \in [0, 1]$.

5 Experiments and Results

5.1 *Experimental Settings*

To evaluate the label pre-annotation process, we conduct a 10-fold cross-evaluation on the CDG treebank. Each experiment is performed on sentences POS-tagged with Melt.

To estimate the results, we calculate the precision of the label pre-annotation at rank one. It means, we calculate the percentage of words for which the first assigned label (i.e. the more probable) is the correct label. This precision corresponds to the label accuracy (LA) calculated on the output of a dependency parsing. Furthermore, we want to find a trade-off between increasing the recall on label accuracy and preserving a small number of labels per word. So, we evaluated the interest of the label sorting by varying the α parameter and connecting the recall with the number of labels assigned to each word.

5.2 *Results of the Label Pre-Annotation*

Table 1 presents the results of the label pre-annotation. The scores do not reach the scores of projective dependency parsing of French that achieve more than 88% label accuracy. Actually, the scores are not comparable because of the constitution of the treebanks exploited in standard work. These commonly exploited French dependency treebanks, in addition to being projective, contain more sentences and use a smaller label set. Thus, to establish a baseline, we trained a transition-based parser, the MaltParser [6], on the sentences of the CDG Treebank. To exploit the potential of MaltParser we tested its available algorithms for non-projective dependency parsing. The best scores result from the use of the *covnon-proj* algorithm and optimized features. Our label pre-annotation scores are slightly better than the label accuracy obtained from the data-driven dependency analysis.

Table 1. Evaluation of the label pre-annotation comparing our method with the performances of MaltParser

	Label accuracy			**Sentence accuracy**		
	All	Proj.	Non-proj.	All	Proj.	Non-proj.
Our method	84.7	84.9	78.6	24.4	26.4	22.1
MaltParser	83.0	83.6	69.2	24.3	26.3	21.7

An interesting question is whether the non-projectivity affects the results. Table 1 also shows the accuracy of the label pre-annotation on the words for which a projective dependency ends on, in the original dependency tree, and for which a non-projective dependency ends on. The accuracy on words associated with a non-projective dependency achieves a lower score than for the words associated with a projective dependency. But, we note that our method achieves a better score than the MaltParser, on the words originally attached with a non-projective dependency. However, due to the small number of non-projective dependencies in the treebank (4%) the global score is weakly affected. The lower scores for words associated with non-projective dependencies can be explained by the fact that they are often attached by distant dependencies. This is not the case of current negation or clitization but some labels such as the aggregation or co-predication commonly attach distant words. Moreover, the context of distant dependents greatly differs from a sentence to another and thus cannot be learnt by the model.

Table 1 shows as well the accuracy on the sentences for both projective and non-projective ones. The non-projective dependencies do not represent a large part of the dependencies but are spread on many sentences (40%). So the effect of the non-projectivity on the accuracy on sentences is important.

A closer look shows that the best scores of accuracy among the different labels are achieved by the most frequent labels. They cover the most general syntactic function of French as subject, accusative object, determiners, modifiers, genitive prepositions. Likewise, the less frequent labels, describing very particular syntactic roles, are often subcategories of more general functions as the copulas, the auxiliaries, the object, etc. There are 34 labels appearing less than 20 times in the corpus which represents almost one third of the labels. These rare labels are not found at high ranks. This problem shows the importance of the label sorting. It allows to reach the second or more probable label for each word accord-

ing to a given threshold. Figure 4 presents the results of the label sorting method as described in section 4.3. It highlights the progression of the recall according to the average number of more probable labels retrieved per word. For this experiment, the α parameter varies from 1 to 5.10^{-5}.

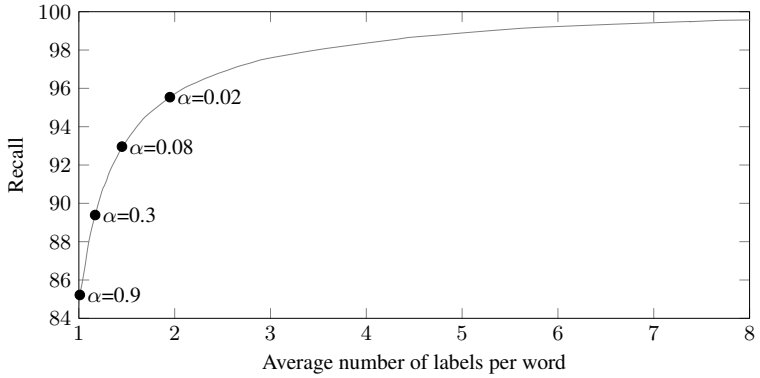

Fig. 4. Evaluation of the recall depending on the average number of labels found applying a sorting based on the probability scores

5.3 *Benefits of the Label Pre-Annotation on the Parsing Step*

We evaluate the effect of the label pre-annotation on the parsing step of the annotation process (i.e. parsing with categorial dependency grammar). We present, in Table 2, the best parsing score we could obtain and the parsing time induced. The evaluation is performed on the CDG Treebank. The attachment scores are computed on the best dependency tree of each parses.[6]

The first experiments are performed using the values of α indicated in Figure 4. For each experiment, the α parameter is fixed for the whole corpus. The last experiment is performed varying the α parameter according to the length of each parsed sentence. The longer the sentence, the higher (restrictive) α is. The first threshold starts with $\alpha = 0.006$, allowing high recall scores for short sentences (< 10 words). Then, intermediate floors are defined until the last one ($\alpha = 0.9$) which preserves

[6] The best dependency tree is the tree having the most correct dependencies

a small number of labels per word in order to parse the longest sentences (> 50 words) in reasonable time. Overall, defining progressive floors allows to find a trade-off between the parsing time and the attachment scores on the whole corpus. The goal is to conduct both a pre-annotation and a dependency parsing which are both accurate and not to long, in order to speed up the annotation process and alleviate the work of the annotator.

Table 2. Evaluation of the dependency parsing using the pre-annotation tool to assign one or more labels to each word. We present the best labelled attachment score (LAS) and the best unlabelled attachment score (UAS) that could be reached with this method.

Label sorting	Labels/words	Scores		Time (sec./sentence)
		LAS	UAS	
Fixed α	1.01	77.62	83.59	0.3
	1.17	81.10	86.47	0.8
	1.45	87.40	91.34	2.3
	1.95	91.94	94.62	7.2
Progressive α	2.04	90.16	92.89	3.0

We notice that the attachment scores increase slowly while the parsing time increases exponentially using a fixed α. The parsing time is decent for short sentences but explodes for long sentences when the number of pre-annotated labels per word is too large. The use of a progressive α is an interesting alternative which increases the attachment scores for short sentences (i.e. better chances to get the correct dependency tree increasing the label recall) and decreases the parsing time for the long ones (allowing to build at least one tree in a reasonable time).

6 Discussion about the Annotation Process

In order to estimate the impact of the pre-annotation step in the development of a dependency treebank we propose to annotate a small set of sentences from the different sub-corpora of the French treebank Sequoia [20]. We evaluate qualitatively the annotation process for two methods. The first one is the method using our automatic pre-annotation process,

and the second one is the manual annotation process that uses word's label selection form. For a fair comparison of the methods, the annotation is performed on equivalent sentences (i.e. equivalent lengths).

The annotation of the sentences shows that our methodology is more suitable for the annotators. An advantage of the automatic pre-annotation is the possibility to skip the fastidious step of pre-selecting the labels. The annotators only have to validate the dependency trees. The benefit of the pre-annotation process is concrete on sentences of average and small length (< 35) but minor on very long sentences. But overall, the average time saved with the first method is around half of the second.

Moreover, the assessment of the annotation highlights that some sentences of the Sequoia treebank are non-projective. The dependency annotation reveals the distant relations and the non-projective constructions that the constituent can not reveal. Around 28% of the annotated sentences have at least one non-projective dependency.

7 Conclusion and Future Work

We show that the scores of a label tagging method using a maximum entropy Markov model are equivalent to the label accuracy scores obtained with a standard data-driven dependency parser. These scores do not reach the scores reported in works on projective dependency parsing because finding the non-projective dependencies is a difficult task. However, the method reaches interesting recall scores which allow to retrieve the right labels while keeping control over the ambiguity reduction. Consequently, this automatic pre-annotation tool included in the whole annotation process relieves the work of the annotators. Part of the time is saved and the annotation process is more accessible. Avoiding the pre-annotation step is greatly appreciated even if the validation step requires some corrections.

What is more, the evaluation of dependency parsing using the pre-annotation tool shows that we could obtain good scores on non-projective dependency parsing. We plan to improve the sorting of the dependency trees in order to propose a complete parser which is able to deal with non-projective constructions and reach appropriate scores.

Acknowledgment The authors thank Danièle Beauquier and Alexander Dikovsky, who provided treebanks that they have built.

REFERENCES

1. Kübler, S., McDonald, R., Nivre, J.: Dependency parsing. Synthesis Lectures on Human Language Technologies **1**(1) (2009) 1–127
2. Abeillé, A., Clément, L., Kinyon, A.: Building a treebank for French. In: Proceedings of the Language Resources and Evaluation Conference. LREC 2000, Athens, Greece (May 2000)
3. Candito, M., Crabbé, B., Denis, P.: Statistical French dependency parsing: Treebank conversion and first results. In: Proceedings of the Language Resources and Evaluation Conference. LREC 2010, Valletta, Malta (May 2010)
4. Dikovsky, A.: Categorial dependency grammars: From theory to large scale grammars. In: Proceedings of the International Conference on Dependency Linguistics. DEPLING 2011 (September 2011)
5. Alfared, R., Béchet, D., Dikovsky, A.: CDG Lab: A toolbox for dependency grammars and dependency treebanks development. In: Proceedings of the International Conference on Dependency Linguistics. DEPLING 2011, Barcelona, Spain (September 2011) 272–281
6. Nivre, J., Hall, J., Nilsson, J., Chanev, A., Eryigit, G., Kübler, S., Marinov, S., Marsi, E.: MaltParser: A language-independent system for data-driven dependency parsing. Natural Language Engineering **13** (6 2007) 95–135
7. Buchholz, S., Marsi, E.: CoNLL-X shared task on multilingual dependency parsing. In: Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X), New York City, Association for Computational Linguistics (June 2006) 149–164
8. Marcus, M.P., Marcinkiewicz, M.A., Santorini, B.: Building a large annotated corpus of English: The Penn treebank. Computational Linguistics **19**(2) (June 1993) 313–330
9. Hajič, J., Hajičová, E., Pajas, P., Panevová, J., Sgall, P., Hladká, B.V.: Prague Dependency Treebank 1.0 (final production label) (2001)
10. Green, N., Larasati, S.D., Žabokrtský, Z.: Indonesian dependency treebank: Annotation and parsing. In: Proceedings of the 26th Pacific Asia Conference on Language, Information, and Computation. PACLIC 2012, Bali, Indonesia (November 2012) 137–145
11. McGillivray, B., Passarotti, M., Ruffolo, P.: The Index Thomisticus treebank project: Annotation, parsing and valency lexicon. TAL **50**(2) (2009) 103–127
12. Atalay, N.B., Oflazer, K., Say, B., Inst, I.: The annotation process in the Turkish treebank. In: Proceedings of the 4th International Workshop on Linguistically Interpreted Corpora. LINC-03, Budapest, Hungary (April 2003)
13. Cerisara, C., Gardent, C., Anderson, C.: Building and exploiting a dependency treebank for French radio broadcast. In: Proceedings of the Ninth International Workshop on Treebanks and Linguistic Theories. TLT9, Tartu, Estonia (November 2010)
14. Alfared, R., Béchet, D.: On the adequacy of three POS taggers and a dependency parser. In: Proceedings of the 13th International Conference on

Computational Linguistics and Intelligent Text Processing (CICLing 2012), Part I. Number 7181 in Lecture Notes in Computer Science, New Delhi, India (March 2012) 104–116
15. Bangalore, S., Joshi, A., eds.: Complexity of Lexical Descriptions and its Relevance to Natural Language Processing: A Supertagging Approach. MIT Press (2010)
16. Crabbé, B., Candito, M.: Expériences d'analyse syntaxique statistique du français. In: Proceedings of the Joint Conference JEP-TALN-RECITAL 2008, Avignon, France (2008)
17. Denis, P., Sagot, B.: Coupling an annotated corpus and a morphosyntactic lexicon for state-of-the-art POS tagging with less human effort. In: Proceedings of the Pacific Asia Conference on Language, Information and Computation. PACLIC 2009, Hong Kong, China (2009)
18. Ratnaparkhi, A.: A maximum entropy model for part-of-speech tagging. In: Proceedings of the 1st Conference on Empirical Methods in Natural Language Processing. EMNLP 1996, Pennsylvania, USA (May 1996)
19. Lavergne, T., Cappé, O., Yvon, F.: Practical very large scale CRFs. In: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics. ACL 2010 (July 2010) 504–513
20. Candito, M., Seddah, D.: Le corpus Sequoia : annotation syntaxique et exploitation pour l'adaptation d'analyseur par pont lexical (The Sequoia corpus: Syntactic annotation and use for a parser lexical domain adaptation method) [in French]. In: Proceedings of the Joint Conference JEP-TALN-RECITAL 2012, volume 2: TALN, Grenoble, France (June 2012) 321–334

OPHÉLIE LACROIX
LINA,
UNIVERSITÉ DE NANTES,
44322 NANTES CEDEX 3, FRANCE
E-MAIL: <OPHELIE.LACROIX@UNIV-NANTES.FR>

DENIS BÉCHET
LINA,
UNIVERSITÉ DE NANTES,
44322 NANTES CEDEX 3, FRANCE
E-MAIL: <DENIS.BECHET@UNIV-NANTES.FR>

FLORIAN BOUDIN
LINA,
UNIVERSITÉ DE NANTES,
44322 NANTES CEDEX 3, FRANCE
E-MAIL: <FLORIAN.BOUDIN@UNIV-NANTES.FR>

IJCLA VOL. 5, NO. 1, JAN-JUN 2014, PP. 105–116
RECEIVED 02/01/14 ACCEPTED 06/02/14 FINAL 11/06/14

Extending Tree Kernels towards Paragraphs

BORIS GALITSKY,[1,2] DMITRY ILVOVSKY,[2] AND SERGEY O. KUZNETSOV[2]

[1] *Knowledge Trail Inc., USA*
[2] *Higher School of Economics, Russia*

ABSTRACT

We extend parse tree kernels from the level of individual sentences towards the level of paragraph to build a framework for learning short texts such as search results and social profile postings. We build a set of extended trees for a paragraph of text from the individual parse trees for sentences. It is performed based on coreferences and Rhetoric Structure relations between the phrases in different sentences. Tree kernel learning is applied to extended trees to take advantage of additional discourse-related information. We evaluate our approach, tracking relevance improvement for multi-sentence search, comparing performances of individual sentence kernels with the ones for extended parse trees. The search problem is formulated as classification of search results into the classes of relevant and irrelevant, learning from the Bing search results, used as a baseline and as a training dataset.

1 INTRODUCTION

Convolution tree kernel [6] defines a feature space consisting of all subtree types of parse trees and counts the number of common subtrees as the syntactic similarity between two parse trees. They have found a number of applications in several natural language tasks, e.g. syntactic parsing re-ranking, relation extraction [50], named entity recognition [8] and Semantic Role Labeling [53], pronoun resolution [49], question classification [52] and machine translation.

The kernel's ability to generate large feature sets is useful to model quickly new and not well-understood linguistic phenomena in learning machines. However, it is often possible to design manually features for linear kernels that produce high accuracy and low computation time, whereas the complexity of tree kernels may prevent their application in real scenarios.

Many learning algorithms, such as SVM can work directly with kernels by replacing the dot product with a particular kernel function. This useful property of kernel methods, that implicitly calculates the dot product in a high-dimensional space over the original representations of objects such as sentences, has made kernel methods an effective solution to modeling structured objects in NLP. A number of NL tasks require computing of semantic features over paragraphs of text containing multiple sentences. Doing it in a sentence pairwise manner is not always accurate, since it is strongly dependent on how information (phrases) is distributed through sentences.

An approach to build a kernel based on more than a single parse tree has been proposed, however for a different purpose than treating multi-sentence portions of text. To compensate for parsing errors [51], a convolution kernel over *packed* parse forest is used to mine syntactic features from it directly. A packed forest compactly encodes exponential number of n-best parse trees, and thus containing much more rich structured features than a single parse tree. This advantage enables the forest kernel not only to be more robust against parsing errors, but also to be able to learn more reliable feature values and help to solve the data sparseness issue that exists in the traditional tree kernel.

On the contrary, in this study we form a tree from a tree forest of sequence of sentences in a paragraph of text. In learning settings where texts include multiple sentences, structures that include paragraph-level information need to be employed. We demonstrate that in certain domains and certain cases discourse structure is essential for proper classification of texts.

2 FROM REGULAR TO EXTENDED TREES

For every arc that connects two parse trees, we derive the extension of these trees, extending branches according to the arc (Fig. 1).

In this approach, for a given parse tree, we will obtain a set of its extension, so the elements of kernel will be computed for many extensions

instead of just a single tree. The problem here is that we need to find common sub-trees for a much higher number of trees than the number of sentences in text, however by subsumption (sub-tree relation) the number of common sub-trees will be substantially reduced.

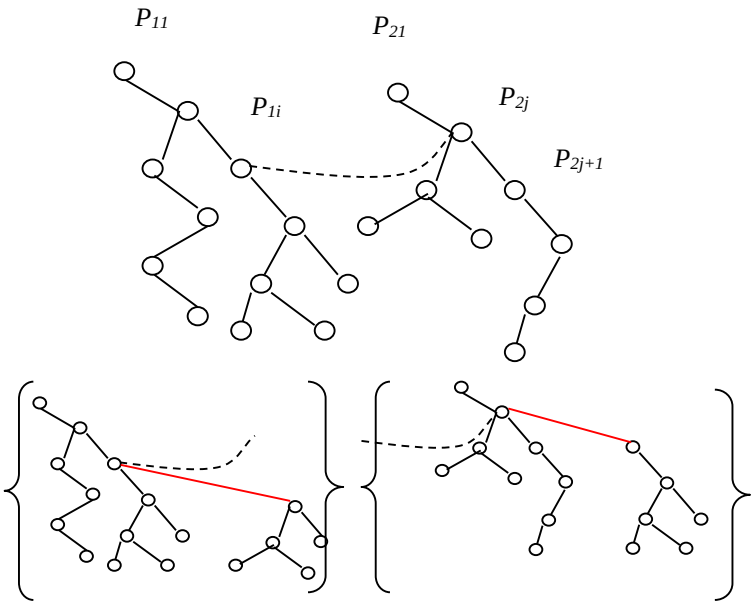

Fig. 1. An arc that connects two parse trees for two sentences in a text (top) and the derived set of extended trees (bottom).

If we have two parse trees P_1 and P_2 for two sentences in a paragraph, and a relation R_{12}: $P_{1i} \rightarrow P_{2j}$ between the nodes P_{1i} and P_{2j}, we form the pair of extended trees $P_1 * P_2$:

…, P_{1i-2}, P_{1i-1}, P_{1i}, P_{2j}, P_{2j+1}, P_{2j+2}, …,
…, P_{2j-2}, P_{2j-1}, P_{2j}, P_{1i}, P_{1i+1}, P_{2i+2}, …,

which would form the feature set for tree kernel learning in addition to the original trees P_1 and P_2.

The algorithm for building an extended tree for a set of parse trees T is presented below:

Input:

1. Set of parse trees T
2. Set of relations R, which includes relations R_{ijk} between the nodes of T_i and T_j: $T_i \in T$, $T_j \in T$, $R_{ijk} \in R$. We use index k to range over multiple relations between the nodes of parse tree for a pair of sentences

Output**:** the exhaustive set of extended trees E

Set $E = \varnothing$;
For each tree $i = 1 : |T|$
  For each relation R_{ijk}, $k = 1 : |R|$
    Obtain T_j;
    Form the pair of extended trees $T_i * T_j$;
    Verify that each of the extended trees do not have a super-tree in E;
    If verified, add to E;
Return E.

Note that the resultant trees are not the proper parse trees for a sentence, but they still form an adequate feature space for tree kernel learning.

To obtain the inter-sentence links, we employed the following sources:

1. Co-reference tools from Stanford NLP [41, 26].
2. Rhetoric relation extractor based on the rule-based approach to finding relations between elementary discourse units [15, 16]. We combined manual rules with automatically learned rules derived from the available discourse corpus by means of syntactic generalization.

3 Implementation of Paragraph Learning

The evaluation framework described here is implemented as an OpenNLP contribution. It relies on the following systems:

– OpenNLP/Stanford NLP parser;
– Stanford NLP Coreference;
– Bing search;
– Wrapper of kernel learner [36].

One of the use cases of this OpenNLP.similarity component is a Java wrapper for tree kernel algorithms implemented in C++. It allows seamless integration of tree kernel algorithms into other open source systems available in Java for search, information retrieval, and machine learning. Moreover, tree kernel algorithms can be embedded into Hadoop framework in the domains where offline performance is essential. Code and libraries described here are also available at http://code.google.com/p/relevance-based-on-parse-trees and http://svn.apache.org/repos/asf/opennlp/sandbox/opennlp-similarity.

4 Complexity Estimation

To estimate the complexity of building extended trees, let us consider an average case with five sentences in each paragraph and 15 words in each sentence. We have on average 10 inter-sentence arcs, which give us up to 20 extended trees formed from two sentences, and 60 extended trees formed from three sentences. Hence, we have to apply tree learning to up to 100 trees (of a bigger size) instead of just 5 original trees. We observe that kernel learning of extended trees has to handle at least 20 times bigger input set.

However, most of the smaller subtrees are repetitive and will be reduced in the course of dimensionality reduction. In addition, in an industrial search application where phrases are stored in an inverse index, the generalization operation can be completed in constant time, irrespectively of the size of index [29]. In case of map-reduce implementation of generalization operation, for example, using Cascading framework, the time complexity becomes constant with the size of candidate search results to be re-ranked [9].

5 Evaluation of Multi-sentence Classification in Search Domain

To confirm that using a set of extended parse trees for paragraphs leverages additional semantic information compared to a set of parse trees for all sentences in a paragraph, we perform an evaluation of relevance in search domain. We apply the same type of tree kernel learning for a paragraph, obtaining parse trees by following two ways:

1. As a baseline, we take all trees for sentences in paragraphs;
2. As an expected improvement, we take all extended trees in a paragraph.

We then compare the classification results as obtained by tree kernel algorithm, applied to the two above sources. We select a search domain that allows us an unlimited set of paragraph-level text initiating search. However, tree kernels are used to solve search relevance problem as classifying candidate answers to be relevant or not. We use Bing search engine API for all web mining and search baseline tasks.

Since a benchmarking database for answering complex multi-sentence questions is not available, we form our own dataset for product-related opinions. The question-answering problem is formulated as finding information on the web, relevant to a user posting / opinion expression in a blog, forum, or social network. We generate a set of queries as short paragraphs of text and run Bing web search engine API to find a candidate set of answers and form a training set.

The classification problem is formulated as classifying a set of search results into the classes of relevant and irrelevant. The respective training dataset is formed from the set of highly ranked answers (as a positive, relevant set) and the set of answers with lower rank (as a negative, irrelevant set). Some randomly selected other candidates are classified, given this training dataset. For each candidate search result, we use its snippet as obtained by Bing and the respective portion of text extracted from the webpage. This experiment is based on the suggestion that top (bottom) Bing results are somehow relevant (irrelevant) to the initial query despite that they can be ordered in a wrong way.

For the purpose of this evaluation, it is not essential to provide the best possible set of answers. Instead, we are concerned with the comparison of relevance improvement by using extended parse tree, as long as the evaluation settings of question answering are identical.

The training/evaluation datasets is formed from search results in the following way. We obtain a first hundred search results (or less if hundred is not available). We select 1–20 (or first 20%) of search results as a positive set, and 81–100 as a negative set. Search results 21–80 form the basis of evaluation dataset, from which we randomly select 10 texts to be classified into the classes of positive or negative. Hence, we have the ratio 4:1 between the training and evaluation datasets.

To motivate our evaluation setting, we rely on the following observations. In case of searching for complex multi-sentence queries, relevance indeed drops abruptly with proceeding from the first 10–20 search results, as search evaluation results demonstrated [15, 16]. The order of

search results in first 20% and last 20% does not affect our evaluation. Although the last 20% of search results is not really a "gold standard," it is nevertheless a set that can be reasonably separated from the positive set. If such separation is too easy or too difficult, it would we hard to evaluate adequately the difference between regular parse trees and extended trees for text classification. Search-based approach to collect texts for evaluation of classification allows reaching maximum degree of experiment automation.

It turned out that the use of tail search results as negative set helps to leverage the high level semantic and discourse information. Negative examples, as well as positive ones, include most keywords from the queries. However, the main difference between the positive and negative search results is that the former include much more coreferences and rhetoric structures similar to the query, than the latter set. Use of extended trees was beneficial in the cases where phrases from queries are distributed through multiple sentences in search results.

We conducted two independent experiments for each search session, classifying search result snippets and original texts extracted from webpages. For the snippets, we split them into sentence fragments and built extended trees for these fragments of sentences. For original texts, we extracted all sentences for snippet fragments and built extended trees for these sentences.

Training and classification occurs in the automated mode, and the classification assessment is conducted by the members of research group guided by the authors. The assessors only consulted the query and answer snippets.

We use the standard parameters of tree sequence kernels from http://disi.unitn.it/moschitti/Tree-Kernel.htm [36]. The latest version of tree kernel learner was obtained from the author.

Table 1. Evaluation results for products domain

Products		Basic kernels [36]	Extended kernels
Text from the page	Precision	0.568	0.587
	Recall	0.752	0.846
	F-measure	0.649	0.675
Snippets	Precision	0.563	0.632
	Recall	0.784	0.831
	F-measure	0.617	0.670

Table 2. Evaluation results for popular answers domain

Answers		Basic kernels [36]	Extended kernels
Text from the page	Precision	0.517	0.544
	Recall	0.736	0.833
	F-measure	0.601	0.628
Snippets	Precision	0.595	0.679
	Recall	0.733	0.790
	F-measure	0.625	0.707

Evaluation results show visible improvement of classification accuracy achieved by extended trees. Stronger increase of recall in comparison to precision can be explained by the following. It is due to the acquired capability of extended trees to match phrases from the search results distributed through multiple sentences, with questions.

6 CONCLUSIONS

In this study we compared two sets of linguistic features:

– The baseline, parse trees for individual sentences,
– Parse trees and discourse information,

and demonstrated that the enriched set of features indeed improves the classification accuracy, having the learning framework fixed. This improvement varies from 2 to 8% in different domains with different structure of texts. To tackle such enriched set of linguistic features, an adjustment of tree kernel algorithm itself was not necessary.

Traditionally, machine learning of linguistic structures is limited to keyword forms and frequencies. At the same time, most theories of discourse are not computational, they model a particular set of relations between consecutive states. In this work, we attempted to achieve the best of both worlds: learn complete parse tree information augmented with an adjustment of discourse theory allowing computational treatment.

The experimental environment, multi-sentence queries and the evaluation framework is available at http://code.google.com/p/relevance-based-on-parse-trees.

REFERENCES

1. Abney, S.: Parsing by Chunks, Principle-Based Parsing, Kluwer Academic Publishers (1991) 257–278
2. Bhasker, Bharat, K. Srikumar: Recommender systems in e-Commerce. CUP (2010)
3. Bron, Coen; Kerbosch, Joep: Algorithm 457: finding all cliques of an undirected graph, Commun. ACM (ACM) **16**(9) (1973) 575–577
4. Byun, Hyeran, Seong-Whan Lee: Applications of Support Vector Machines for pattern recognition: A survey. In: Proceedings of the First International Workshop on Pattern Recognition with Support Vector Machines, SVM'02, Springer, London, UK, (2002) 213–236
5. Cascading. en.wikipedia.org/wiki/Cascading; www.cascading.org/2013
6. Collins, M., and Duffy, N.: Convolution kernels for natural language. In: Proceedings of NIPS (2002) 625–632
7. Conte, D., P. Foggia, C. Sansone, and M. Vento.: Thirty years of graph matching in pattern recognition. International Journal of Pattern Recognition and Artificial Intelligence **18**(3) (2004) 265–298
8. Cumby, C., Roth, D.: Kernel methods for relational learning. In: Proceedings of ICML 2003 (2003)
9. Dean, J.: Challenges in Building Large-Scale Information Retrieval Systems. research.google.com/people/jeff/WSDM09-keynote.pdf (2009)
10. Domingos, P. and Poon, H.: Unsupervised semantic parsing. In: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, Singapore, ACL (2009)
11. Ehrlich H.-C., Rarey M.: Maximum common subgraph isomorphism algorithms and their applications in molecular science: review. Wiley Interdisciplinary Reviews: Computational Molecular Science **1**(1) (2011) 68–79
12. Fukunaga, K.: Introduction to statistical pattern recognition (2nd ed.), Academic Press Professional, Inc., San Diego, CA (1990)
13. Furukawa, K.: From deduction to induction: Logical perspective. The Logic Programming Paradigm. Springer (1998)
14. Galitsky, B.: Natural language question answering system: Technique of semantic headers. In: Advanced Knowledge International, Australia (2003)
15. Galitsky, B., Daniel Usikov, Sergei O. Kuznetsov: Parse thicket representations for answering multi-sentence questions. In: 20th International Conference on Conceptual Structures, ICCS 2013 (2013)
16. Galitsky, B., Ilvovsky, D. Kuznetsov, S., Strok, F.: Improving text retrieval efficiency with pattern structures on parse thickets. In: Workshop on Formal Concept Analysis Meets Information Retrieval at ECIR 2013, Moscow, Russia (2013)
17. Galitsky, B., Josep Lluis de la Rosa, Gábor Dobrocsi: Inferring the semantic properties of sentences by mining syntactic parse trees. Data and Knowledge Engineering **81–82** (2012) 21–45

18. Galitsky, B., Kuznetsov S, Learning communicative actions of conflicting human agents. J. Exp. Theor. Artificial Intelligence **20**(4) (2008) 277–317
19. Galitsky, B., Machine learning of syntactic parse trees for search and classification of text. Engineering Application of Artificial Intelligence **26**(3) (2012) 1072–1091
20. Haussler, D. 1999. Convolution kernels on discrete structures.
21. Jarvelin, Kalervo, Jaana Kekalainen: Cumulated gain-based evaluation of IR techniques. ACM Transactions on Information Systems **20**(4) (2002) 422–446
22. Jurafsky, D., Martin, J. Speech and language processing. An introduction to natural language processing, computational linguistics, and speech recognition (2008)
23. Kann, V.: On the approximability of the maximum common subgraph problem. In: (STACS '92), Springer, London, UK (1992) 377–388
24. Kim, Jung-Jae, Piotr Pezik and Dietrich Rebholz-Schuhmann. MedEvi: Retrieving textual evidence of relations between biomedical concepts from Medline. Bioinformatics **24**(11) (2008) 1410–1412.
25. Kohavi, Ron: A study of cross-validation and bootstrap for accuracy estimation and model selection. In: International Joint Conference on Artificial Intelligence IJCAI 1995 (1995)
26. Lee, Heeyoung, Angel Chang, Yves Peirsman, Nathanael Chambers, Mihai Surdeanu and Dan Jurafsky: Deterministic coreference resolution based on entity-centric, precision-ranked rules. Computational Linguistics **39**(4), 2013.
27. Lee, Heeyoung, Marta Recasens, Angel Chang, Mihai Surdeanu, and Dan Jurafsky: Joint Entity and Event Coreference Resolution across Documents. In: Proceedings of EMNLP-CoNLL (2012)
28. Levy, Roger and Galen Andrew: Tregex and Tsurgeon: tools for querying and manipulating tree data structures. In: Proceedings of the 5th International Conference on Language Resources and Evaluation, LREC 2006 (2006)
29. Lin, J., Chris Dyer: Data-intensive text processing with MapReduce. Morgan & Claypool Publishers (2010)
30. Mann, William C., Christian M. I. M. Matthiessen and Sandra A. Thompson: Rhetorical structure theory and text analysis. In: Discourse Description: Diverse linguistic analyses of a fund-raising text. John Benjamins (1992) 39–78.
31. Manning, Chris and Hinrich Schütze: Foundations of Statistical Natural Language Processing, MIT Press. Cambridge, MA (1999)
32. Marcu, D.: From discourse structures to text summaries. In: Proceedings of ACL Workshop on Intelligent Scalable Text Summarization, Madrid, Spain (1997) 82–88
33. Mill, J.S.: A system of logic, ratiocinative and inductive. London (1843)
34. Mitchell, T.: Machine Learning. McGraw Hill (1997)
35. Montaner, M.; Lopez, B.; de la Rosa, J. L.: A taxonomy of recommender agents on the Internet. Artificial Intelligence Review **19**(4) (2003) 285–330

36. Moschitti, A.: Efficient convolution kernels for dependency and constituent syntactic trees. In: Proceedings of the 17th European Conference on Machine Learning, Berlin, Germany (2006)
37. Moschitti, Alessandro, Daniele Pighin, and Roberto Basili: Tree kernels for semantic role labeling. Computational Linguistics **34**(2) (2008) 193–224
38. Polovina S., John Heaton: An Introduction to conceptual graphs. AI Expert (1992) 36–43
39. Punyakanok, V., Roth, D., Yih, W.: Mapping dependencies trees: an application to question answering. In: Proceedings of AI & Math, Florida, USA (2004)
40. Punyakanok, V., Roth, D., Yih, W.: The Necessity of Syntactic Parsing for Semantic Role Labeling. In: Proceedings of IJCAI-05 (2005)
41. Recasens, Marta, Marie-Catherine de Marneffe, and Christopher Potts: The life and death of discourse entities: Identifying singleton mentions. In: Proceedings of NAACL 2013 (2013)
42. Robinson J.A.: A machine-oriented logic based on the resolution principle. Journal of the Association for Computing Machinery **12** (1965) 23–41
43. Sun, J., Min Zhang, Chew Lim Tan: Tree Sequence Kernel for Natural Language. AAAI-25 (2011)
44. Sun, J.; Zhang, M.; and Tan, C.: Exploring syntactic structural features for sub-tree alignment using bilingual tree kernels. In: Proceedings of ACL (2010) 306–315
45. Trias i Mansilla, A., J.L. de la Rosa i Esteva : Asknext: An agent protocol for social search. Information Sciences **190** (2012) 144–161
46. Vismara, Philippe, Benoît, Valery: Finding Maximum Common Connected Subgraphs Using Clique Detection or Constraint Satisfaction Algorithms. Modelling, Computation and Optimization in Information Systems and Management Sciences, Springer (2008)
47. Wu, Jiangning, Zhaoguo Xuan, Donghua Pan: Enhancing text representation for classification tasks with semantic graph structures. International Journal of Innovative Computing, Information and Control **7**(5(B)) (2011) 2689–2698
48. Yan, X., Han, J.: gSpan: Graph-based substructure pattern mining. In: Proceedings of the IEEE International Conference on Data Mining, ICDM'02, IEEE Computer Society (2002) 721–724
49. Yang X. F., Su J., Chew C. L.: Kernel-based pronoun resolution with structured syntactic knowledge. In: COLING-ACL'2006. (2006)
50. Zelenko, D., Aone, C., Richardella, A.: Kernel methods for relation extraction. Journal of Machine Learning Research **3** (2003) 1083–1106
51. Zhang M., Zhang H., Li H., Convolution Kernel over Packed Parse Forest. In: Proceedings of ACL-2010 (2010)
52. Zhang, Dell, Wee Sun Lee.: Question classification using Support Vector Machines. In: Proceedings of the 26th ACM SIGIR (2003) 26–32
53. Zhang, M., Che, W., Zhou, G., Aw, A., Tan, C., Liu, T., Li, S.: Semantic role labeling using a grammar-driven convolution tree kernel. IEEE Transactions on Audio, Speech, and Language Processing **16**(7) (2008) 1315–1329

BORIS GALITSKY
KNOWLEDGE TRAIL INC.
SAN JOSE, CA, USA
AND
NATIONAL RESEARCH UNIVERSITY HIGHER SCHOOL OF ECONOMICS,
KOCHNOVSKI PR. 3, MOSCOW, 125319, RUSSIA
E-MAIL: <BGALITSKY@HOTMAIL.COM>

DMITRY ILVOVSKY
NATIONAL RESEARCH UNIVERSITY HIGHER SCHOOL OF ECONOMICS,
KOCHNOVSKI PR. 3, MOSCOW, 125319, RUSSIA
E-MAIL: <DILVOVSKY@HSE.RU>

SERGEY O. KUZNETSOV
NATIONAL RESEARCH UNIVERSITY HIGHER SCHOOL OF ECONOMICS,
KOCHNOVSKI PR. 3, MOSCOW, 125319, RUSSIA
E-MAIL: < SKUZNETSOV@HSE.RU>

IJCLA VOL. 5, NO. 1, JAN-JUN 2014, PP. 117–128
RECEIVED 07/01/14 ACCEPTED 06/02/14 FINAL 17/06/14

Automatic Recognition of Clauses

OLDŘICH KRŮZA AND VLADISLAV KUBOŇ

Charles University in Prague, Czech Republic

ABSTRACT

This paper describes an attempt to solve the problem of recognizing clauses and their mutual relationship in complex Czech sentences on the basis of limited information, namely the information obtained by morphological analysis only. The method described in this paper may be used in the future for splitting the parsing process into two phases, namely (1) Recognizing clauses and their mutual relationships; and (2) Parsing the individual clauses. This approach should be able to improve the result of parsing long complex sentences.

1 INTRODUCTION

Despite the progress achieved in recent years in the field of natural language parsing it still makes sense to seek alternative approaches to the problem. Parsing is a procedure *performed by a human or a computer* whose input are words of a sentence, typically with morphological annotation, and whose output is a syntactical structure on these words, in our case represented by a dependency tree. Clearly, a sentence consists of words, just like a human body consists of cells. Nobody sane would describe a human body as a mere cluster of cells though. A body apparently consists of organs, and those consist of cells. Even if a sentence can hardly have 10^{14} words[1], we believe the relation between words, clauses and a sentence is similar to that of cells, organs and a body.

Clauses are in many aspects autonomous and many grammatical relationships are either limited within a clause, or are among clauses as atomic units. For example, Petkevič [1] states that

[1] This is roughly how many cells a human body has.

- word order,
- valency and valency scopes, adjunct scopes, scopes of modifications (local, temporal, concessive, etc.),
- agreement and its scope,
- analytical predicate and its scope,
- constituent coordination and its scope,
- internal coreference/anaphor

are all intra-clausal relationships.

In our approach we suggest to step aside from the traditional path of performing full-fledged parsing immediatelly after morphological tagging of individual word forms and to attempt to detect the organs of complex sentences, individual clauses, first, and to determine their mutual relationships prior to the actual parsing phase. The next phse could then, subsequently, concentrate on parsing individual clauses one by one. Not only does this more refined sequence of steps make a lot more linguistic sense, but it also decreases the number of words a parser has to consider at a time and thus simplifies the task. We hope that this approach could also speed up the whole procedure and raise the overall efficiency.

2 Building a Model for Clause Recognition

Previous work on clause detection has focused on English and was mostly limited to finding clause boundaries, not their dependency relations like we do. One occasion of focus to clause identification was the CoNLL-2001 shared task [2, 3]. Meanwhile, Prague Depencency Treebank in its version 2.5 contains data that mark up clauses based on the research on sentence *segments* [4]. The clauses have been annotated by an automatic procedure that uses the surface syntax layer to identify the clause that each token belongs to [5]. This approach differs from the one presented in this paper in the type of information used. It exploits the information from already analyzed sentences and their surface syntactic trees and thus it cannot be used for splitting the parsing process into more phases. On the other hand, our approach aims at using the morphological information only for automatic clause identification.

The task of automated clause recognition is not trivial from the programming point of view. We'd like to use a statistical approach using machine learning. However, the task is not easily mapped to standard regression or classification that machine learning has standard procedures for. The output of clause recognition should include:

1. the set of clauses,
2. the dependency, and coordination relations among clauses,
3. distribution of words to individual

clauses.

2.1 *Incremental Approach*

Our first attempt at solving the task was an incremental solution, where sub-tasks would be defined, each solvable as a classification or regression task. The final result would then be obtained by chaining the subtasks.

The first step was the identification of clause borders. We trained a simple discriminative model based on features that drew on previous research of sentence segments described in [6]. Segments happen to be mostly bordered by conjunctions or punctuation in Czech and we expolited this. We have used the following features for our clause boundary model:

- word's lemma,
- whether the right neighbor is a *separator*,
- whether the left neighbor is a separator,
- whether the left neighbor is a conjunction.

A *separator* is a punctuation mark or one of the coordinate conjuntions that do not enforce a comma (*a*, *nebo*, *ani*, *i*).

The set of lemmata was limited to a small set of those that represent conjunctions, punctuation, and single-letter words.

These features were chosen so that they use all the morphological information that the surface-syntax-tree-to-clauses converter described in Krůza, Kuboň 2009 uses. This converter made it possible to use the data from the analytical layer of the Prague Dependency Treebank as training data.

The predicted feature was whether a word is at the boundary of a clause. Specifically, whether it is the starting word of a clause, whether it is the final word of a clause, whether this word is just before the start of a contained sub-clause and whether this word is just after the end of a contained sub-clause.

Despite its simplicity, this model achieved about 75% precision. Even though it likely had much room for improvement, we abandoned this approach completely. A non-negligible error rate in the first step alone would spoil the chain as the errors in the individual steps would accumulate.

2.2 *Using MST Parser for Clause Recognition*

The experience with the incremental approach clearly indicated that the idea of dividing the recognition task into more steps should be abandoned in favor of a more "holistic" approach that will try to solve the task in a single step. Because the clause recognition is to a certain extent a similar task to full-fledged parsing (the main difference being the size of units we are working with), using a standard parser with adapted data for this task seemed to be a natural solution. We have decided to use the MST parser [7] for this task due to its quality which had been already demonstrated for several languages of various types, Czech and English being among them.

The standard form of data MST parser is the following: the words and punctuation marks (tokens) represent the nodes of a syntactic tree. Each word has four types of values:

1. an original word form
2. a morphological tag
3. a parent node
4. a label of a dependency edge going towards the parent node.

The adaptation of the data for our experiment had been relatively straightforward. Since the whole parse of a sentence always has a tree structure (at least in the FGD formalism, which we adhere to [8]), and since each clause is formed by a subtree of the parse, the clauses themselves must evidently also form a dependency tree. Now, if we see the relation of a word belonging to a clause as a special case of dependency, then the whole output structure (i.e. the three points listed above) can be seen as one dependency tree.

We need the nodes of the tree to represent the clauses on one hand and the tokens of the clauses on the other hand. We'll use the tokens of the sentence to represent both things. So the set of tokens will be the set of nodes of the tree. Each token will represent itself, and some tokens will also represent the clause they belong to. The choice of which token shall represent a clause is clear as each clause has, according to our definition, a *head* among its tokens.

In the tree, we will distinguish between simple tokens and heads of clauses with dependency types: (1) *token dependency* and (2) *clause dependency*. A word that has a clause dependency to its parent represents a clause. Other words can have token dependencies to such a word, thus representing that the dependant is included in the clause represented by

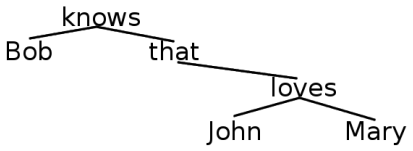

Fig. 1. Standard dependency tree, simplified for illustrating our point.

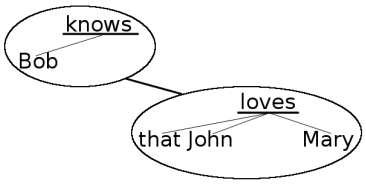

Fig. 2. Dependency tree adapted to capture clausal dependencies. Thin lines denote token dependencies, thick lines denote clause dependencies. Underlined words denote clause heads.

the parent word. Figures 1 and 2 illustrate the difference between a standard syntax tree and an adapted tree that captures clausal relationships on simple made-up English examples.

Krůza, Kuboň 2009 elaborated on details of representing clause structure. Two major factors make the situation more complicated than outlined above: coordination and clauses sharing words. A brief recapitulation follows.

A dependency relationship of one clause to another is represented by clause dependency of the head either to the head of the parent clause or to a word included in the parent clause. The former variant is simpler but the latter can capture which token of the parent clause is the parent of the head of the child clause in the parse. This information does not strictly belong to the clause relationships, where clauses are seen as depending on each other, not on tokens, but choosing to preserve this piece of information in the clause tree is an option that can save a lot of trouble in later stages. Figures 3 and 4 illustrate the difference between the two alternatives.

Beside dependency relations between clauses, two clauses can also be coordinated. This frequent phenomenon, which is in fact a real nightmare for the dependency paradigm, is represented by a special depen-

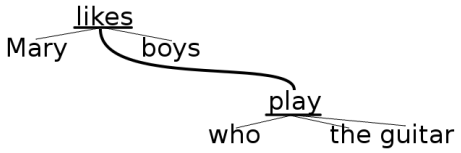

Fig. 3. Illustration of the variant where the dependency relationship of two clauses is delegated on the two heads. The alternative is simpler and seems cleaner in the sense of keeping the clauses atomic.

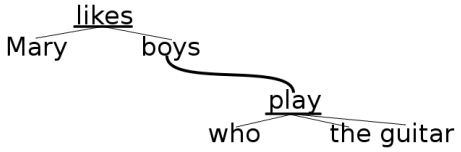

Fig. 4. Illustration of the variant where the dependency relationship of two clauses is delegated on the head of the dependent clause and its actual antecedant from the parent clause. The alternative introduces heterogeny but can be very useful in the stage of reconstructing the actual sentence parse from inter-clausal and intra-clausal trees.

dency type of both coordinated clauses on the coordination. Thus, we introduce a new type of tree nodes. Now, beside clauses and tokens, clause coordinations also occur. They are denoted by having the *coordination dependency*. The coordinated clauses are denoted by having the *member dependency* type.

Table 1 shows an example of the data format for MST parser.

Often, a noun phrase or another constituent forms a modifier shared by both coordinated clauses. Such subtrees are represented by a node of the tree with a special *part dependency* type. A tree with all dependency types is shown in Figure 5. Notice that this short example with all types of dependencies is quite extreme. Typically, a clause-structure tree is much flatter, which is one of the differences compared to more complex trees resulting from full-fledged parsing.

Having such trees, we trained the MST parser to recognize them. The best results were achieved with a second-order model and non-projective algorithm.

Table 1. A sample sentence with formating for the MST parser: "Šetřete peníze, netelefonujte, faxujte! Je tento reklamní slogan pravdivý? [Save money, don't call, fax! Is this advertising slogan true?]"

```
Šetřit         peníze ,           telefonovat_:T
i-P-           NIP4   :---        i-P-
Clause-MEMBER  Word   Coord.-CHILD Clause-MEMBER
3              1      0           5

,              faxovat_:T    !
:---           i-P-          :---
Coord.-MEMBER  Clause-MEMBER Word
3              5             0

být           tento reklamní slogan pravdivý ?
B-S-          DYS1  AIS1     NIS1   AIS1     :---
Clause-Child  Word  Word     Word   Word     Word
0             1      1       1      1        0
```

2.3 *Baseline*

Since our work on this type of automated clause recognition in Czech was pioneer, we had no direct comparison with other approaches. Our task is simpler than full-fledged parsing and it is our ultimate goal to improve automated parsing. So deriving the clauses from the result of an existing parser offers a fitting baseline for us.

We used the development test data from CoNLL 2009, which contains MST-predicted heads and labels. The clauses have been derived using the algorithm presented in [6].

3 EVALUATION

To use the standard precision and recall scores to evaluate clause recognition, we need to define what makes for a correctly-recognized clause. We use two criteria:

1. whether the head is correctly recognized, and
2. whether the component, i.e. set of words of the clause is correctly assigned.

In addition, we also use a conjunction of both criteria. We call these the scores *a head-based score*, *a component-based score* and *a head-and-component-based score*.

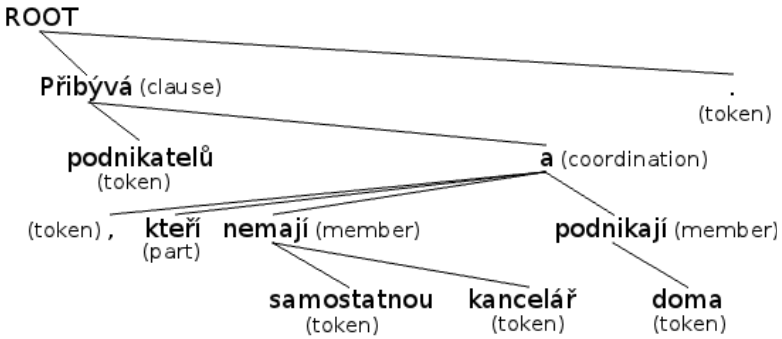

Fig. 5. Clause-structure tree as fed to MST parser. The-number-grows of-businessmen, who don't-have separate office and work at-home. (The number of businessmen who don't have a separate office and work at home, grows.)

None of these, however, gives a continuous measure of the word distribution. We want a way to measure that a clause that has 12 out of 15 words correctly assigned is better than one that has only 9 of 15. For this purpose, we introduce a so-called *fuzzy score*. It is defined by the following algorithm:

1. Initialize an empty precision score array.
2. Initialize an empty recall score array.
3. For each predicted clause P find a clause G in gold standard such that no other clause of gold standard contains more tokens belonging to P than G does.
   Push the evaluation of the following formula to the precision score array: ((#words that P and G share – #words present in P but absent in G) / #words in P)
   End for.
4. For each clause in gold standard G find a predicted clause P such that no other predicted clause contains more tokens belonging to G than P does.
   Push the evaluation of the following formula to the recall score array: ((#words that G and P share – #words present in G but absent in P) / #words in G)
   End for.
5. Return mean values of the precision and recall score arrays.

The idea behind the fuzzy score is taken from precision and recall measures. Precision constituent is simulated by looking at each found

Table 2. Evaluation of the MST-based model on dtest data

Method	Precision	Recall
fuzzy	0.95	0.95
head	0.94	0.94
component	0.78	0.78
head+comp	0.78	0.78

Table 3. Baseline results

Method	Precision	Recall
fuzzy	0.87	0.90
head	0.89	0.86
component	0.59	0.57
head+comp	0.58	0.57

clause, finding the gold clause sharing the most tokens, and calculating the overlap ratio. Analogically, recall is simulated by looking at each gold clause and comparing it to an adequate predicted clause. Since we're comparing two coverages of a given set (a sentence), the score cannot reach zero. The worst case for the recall constituent would be identifying each token as a clause, whereas the worst case for the precision constituent would be identifying the whole sentence as one big clause.

The fuzzy score does its job in measuring component overlap but ignores everything else, so we use it in addition to the aforementioned measures. Table 2 shows the evaluation of the MST-based model. Table 3 shows the evaluation of the baseline.

3.1 *Improving the MST-based Model*

Seeing how much worse the model does at assigning words to clauses in comparison to identifying the clauses themselves and their heads, we were looking into ways of improving the assignment of words into correct clauses. When using the MST in projective mode, the most errors had been done where there was a discontinuity in a clause, e.g. when a clause spanned words 1 to 3 and 6 to 9. Switching to the non-projective algorithm raised the score a touch but there was no observable tendency in the remaining errors any more. We have therefore made an attempt to employ machine learning again and we have trained another model specifically for distributing the words into the pre-identified clauses. This

new model was applied on top of the MST-generated clause structure, so we could use the original MST prediction as a feature.

This new *component model* had couples of words as observation samples. Both words of such couples always belonged to the same sentence; one of them was a head of a clause and the other one was not (in the MST prediction). The predicted feature was whether the first word represents the clause that the other word belongs to. We have defined a set of 148 morphological, lexical and MST-prediction-based features, and planned to use a statistical feature-selection method to get the optimal feature space.

However, we have quickly encountered technical limitations: the training data set was simply too large to fit into the memory. All attempts to train a model failed, either not finishing even after ten days (e.g. k-nearest-neighbor) or crashed on depleting memory (all linear models including SVM). Finally we have tried the C5.0 decision tree. It finished in mere 16 minutes. The induced decision tree was only employing 38 features, which was a neat (though possibly suboptimal) feature selection.

Applying the component model raised the component-based score by 0.1%. The fuzzy score has been raised by 0.0004%. C5.0 was reporting its error rate at 3.9%, which is not bad and certainly not easy to beat by tweaking the feature set.

Because of the negligible contribution, we decided not to use the component model. The obvious explanation for the small contribution seems to be the strictness of the component score, where a clause is only considered correctly identified, when its set of components is correctly assigned. One token off or extra and the clause is not counted as a success. The decent fuzzy score confirms that the token distribution has actually reached a level where improvements are hard to get.

4 CONCLUSION

The paper presents an experiment with a method for automatical clause detection using a specially-trained MST parser. A custom measure rate has been defined to evaluate the recognition. The method outperforms deriving clauses from full-fledged automated parsing with MST. What remains yet to be seen is whether parsing the detected clauses would yield better results than parsing the sentences in a classical way.

ACKNOWLEDGMENTS This work was supported by the Grant of Czech Science Foundation (GAČR) No. P202/10/1333 and by the Charles Uni-

versity Grant Agency (GAUK), Grant No. 920913. In this work we used language resources developed, stored, and distributed by the LINDAT / CLARIN project of the Ministry of Education, Youth and Sports of the Czech Republic (project LM2010013). Participation in the conference was supported by Foundation of Vilem Mathesius.

REFERENCES

1. Petkevič, V.: Clause identification based on corpora of contemporary Czech. In: Gramatika a korpus / Grammar & Corpora 2007. (2007)
2. Tjong, E.F., Sang, K., Déjean, H.: Introduction to the CoNLL-2001 shared task: Clause identification. In Daelemans, W., Zajac, R., eds.: Proceedings of CoNLL-2001, Toulouse, France (2001)
3. Stevenson, S., Carreras, X.: Proceedings of CoNLL-2009. Association for Computational Linguistics, Boulder, Colorado (2009)
4. Lopatková, M., Homola, P., Klyueva, N.: Annotation of sentence structure: Capturing the relationship between clauses in Czech sentences. Language Resources and Evaluation (2012) 25–36
5. Bejček, E., Panevová, J., Popelka, J., Straňák, P., Ševčíková, M., Štěpánek, J., Žabokrtský, Z.: Prague Dependency Treebank 2.5 – a revisited version of PDT 2.0. In: Proceedings of Coling 2012. (2012) 231–246
6. Krůza, O., Kuboň, V.: Automatic extraction of clause relationships from a treebank. In: Computational Linguistics and Intelligent Text Processing (CICLing 2009). Number 5449 in LNCS. Springer (2009) 195–206
7. McDonald, R., Pereira, F., Ribarov, K., Hajič, J.: Non-projective dependency parsing using spanning tree algorithms. In: HLT/EMNLP'05: Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing, Morristown, NJ, USA, Association for Computational Linguistics (2005) 523–530
8. Sgall, P.: Generativní popis jazyka a česká deklinace. Academia, Prague, Czech Republic (1967)

OLDŘICH KRŮZA
INSTITUTE OF FORMAL AND APPLIED LINGUISTICS,
FACULTY OF MATHEMATICS AND PHYSICS,
CHARLES UNIVERSITY IN PRAGUE,
MALOSTRANSKÉ NÁMĚSTÍ 25, PRAGUE, CZECH REPUBLIC
E-MAIL: <KRUZA@UFAL.MFF.CUNI.CZ>

VLADISLAV KUBOŇ
INSTITUTE OF FORMAL AND APPLIED LINGUISTICS,
FACULTY OF MATHEMATICS AND PHYSICS,
CHARLES UNIVERSITY IN PRAGUE,
MALOSTRANSKÉ NÁMĚSTÍ 25, PRAGUE, CZECH REPUBLIC
E-MAIL: <VK@UFAL.MFF.CUNI.CZ>

IJCLA VOL. 5, NO. 1, JAN-JUN 2014, PP. 129–145
RECEIVED 30/12/13 ACCEPTED 06/02/14 FINAL 17/06/14

Lessons Learned from Tagging Clinical Hungarian

GYÖRGY OROSZ, ATTILA NOVÁK, AND GÁBOR PRÓSZÉKY

Pázmány Péter Catholic University and MTA-PPKE
Hungarian Language Technology Research Group, Hungary

ABSTRACT

As more and more textual resources from the medical domain are getting accessible, automatic analysis of clinical notes becomes possible. Since part-of-speech tagging is a fundamental part of any text processing chain, tagging tasks must be performed with high accuracy. While there are numerous studies on tagging medical English, we are not aware of any previous research examining the same field for Hungarian. This paper presents methods and resources which can be used for annotating medical Hungarian and investigates their application to tagging clinical records. Our research relies on a baseline setting, whose performance was improved incrementally by eliminating its most common errors. The extension of the lexicon used raised the overall accuracy significantly, while other domain adaptation methods were only partially successful. The presented enhancements corrected almost half of the errors. However, further analysis of errors suggest that abbreviations should be handled at a higher level of processing.

KEYWORDS: *Medical text processing, PoS tagging, morphological disambiguation, domain adaptation, clinical notes.*

1 INTRODUCTION

Hospitals produce a huge amount of clinical notes that have solely been used for archiving purposes and have generally been inaccessible to researchers. However, nowadays medical resources are becoming available,

enabling computer scientist to support medical researchers. As natural language processing (NLP) algorithms are getting more and more accurate, their usage can not just cut costs but can also boost medical research. PoS tagging is a fundamental task of computational linguistics: labeling words with their part-of-speech is essential for further processing algorithms. While tagging of general texts is well-known and considered to be solved, most of the commonly used methods usually fail on medical texts.

English has been the main target of many NLP applications up to the present time, thus less-resourced languages, which are usually morphologically complex, often fell beyond the scope. Similarly, there are just a few studies attempting to annotate non-English medical texts. Thus, the processing of Hungarian clinical records has very little literature. Moreover, there is not any research on tagging such texts. Therefore, this study aims to investigate how existing techniques can be used for the morphological tagging of Hungarian clinical records presenting possible pitfalls of a medical text processing chain.

This paper is structured as follows. The background of our research is described in the next section. Then a corpus is presented which has been created for development and evaluation purposes. In Section 4, we detail the baseline morphological disambiguation setting used, which is commonly employed for Hungarian. Afterwards, we present the most frequent errors made by the baseline tagger and we describe and evaluate the enhancements that were carried out on the text processing chain. Finally, Section 6 provides the final conclusions.

2 Parsing of Biomedical Texts

2.1 *Biomedical Tagging*

Processing of biomedical texts has an extensive literature, since there are numerous resources accessible. In contrast, much less manually annotated corpora of clinical texts are available. Most of the work in this field has been done for English, and only a few attempts have been published for morphologically rich languages (e. g. [1, 2]).

A general approach for biomedical PoS tagging is to employ supervised learning algorithms, which require manually annotated data. In the case of tagging biomedical texts, domain-specific corpora are used either alone [3–5] or in conjunction with a (sub)corpus of general English [6–8] as training data. While using texts only from the target domain yields

acceptable performance [3–5], several experiments have shown that accuracy further increases with incorporating annotated sentences from the general domain as well [9, 6]. A general observation is that the more data is used from the reference domain, the higher accuracy can be achieved (e. g. [10]). On the contrary, Hahn and Wermter argue for training learners only on general corpora [11] (for German). Further on, there are studies on selecting training data (e. g. [12]) that increase the accuracy. What is more, there are taggers (such as [13]) which learn from several domains in a parallel fashion, thus the model selection decision is delayed until the decoding process.

Using target-specific lexicons is another way of adapting taggers, as they can improve tagging performance [6, 14]. Some of these studies extend existing PoS dictionaries [15], while others build new ones specific to the target domain [5]. All of the experiments using such resources yield significantly reduced error rates.

Concerning tagging algorithms, researchers tend to prefer already existing applications, such as the OpenNLP toolkit[1], which is the basis of the cTakes system [4]; while Brill's method [16] and TnT [17] are widely used (e.g. [11, 4, 10]) as well. There are other HMM-based solutions which have been shown to perform well [9, 6, 15, 11, 3, 2, 14] on such texts. Besides, a number of experiments have revealed [7, 14, 5] that domain-specific OOV words are primarily responsible for a reduced tagging performance. Thus successful methods employ either guessing algorithms [9, 15, 2, 14, 5] or broad-coverage lexicons (as detailed above). Beyond supervised algorithms, other approaches were also shown to be effective: Miller et al. [8] use semi-supervised methods; Dwinedi and Sukhadeve build a tagger system based only on rules [18]; while Ruch et al. propose a hybrid system [14]. Further on, domain adaptation methods (such as EasyAdapt [19] or ClinAdapt [7]) also perform well. However, they need an appropriate amount of manually annotated data from the target domain, which limits their applicability.

2.2 *Tagging General Hungarian*

For Hungarian, tagging experiments generally rely on the Szeged Corpus [20] (SZC), since this is the only contemporary linguistic resource that is manually annotated with morphological tags and is freely available. It contains about 1.2 million words from six different genres, but

[1] `http://opennlp.apache.org/`

does not involve texts from the biomedical domain. The original annotation of the corpus uses the MSD scheme proposed by the MULTEXT-East project [21]. Besides this, other morphosyntactic coding systems are commonly used as well. One of them is the system employed by the HuMor morphological analyzer [22], whose labels are composed of morpheme tags. Another annotation scheme is named KR, which is the default of morphdb.hu [23], a freely available Hungarian morphological resource. Although the Szeged Corpus contains only MSD codes, there are also automatically converted variants of it that use the latter schemes.

For agglutinating languages such as Hungarian, labeling a word only with its part-of-speech tag is not satisfactory (as described in [24]), since further parsing methods require full morphosyntactic labels and lemmata as well. Consequently, tagger tools must perform full morphological disambiguation which also involves lemmatization. For Hungarian, such tools are the following:

PurePos [24], an open-source full morphological disambiguation system which is able to incorporate the knowledge of a morphological analyzer (MA), thus providing state-of-the-art accuracy above 98%. The system is based on statistical trigram tagging algorithms, but it is extended to employ language-specific rule-based components effectively.

`magyarlanc` [25], a freely available[2] language processing chain for morphological and dependency parsing of Hungarian that contains several language-specific components. Its morphological disambiguation module is based on the Stanford tagger [26] and incorporates a MA based on morphdb.hu.

2.3 *Processing Clinical Hungarian*

There are only a few studies on processing Hungarian medical records. Siklósi et al. [27, 28] presented a system that is able to correct spelling errors in clinical notes. A resolution method for clinical abbreviations was also presented by them [29], in which they used pattern matching methods on domain-specific lexicons. Recently, Orosz et al. introduced [30] a partly unsupervised algorithm for segmenting tokens and sentences in clinical texts: their approach is a combination of collocation extraction algorithms and rule-based methods.

[2] It is available only without the source code.

As far as we know, no study exists either investigated possible approaches or established a proper method for tagging clinical Hungarian. Therefore we aim to examine special properties of clinical notes first, then to develop a disambiguation methodology. The experiments described below use methods that rely on an error analysis of the baseline system (in Section 4), while also incorporate ideas from previous studies (cf. Section 2.1).

3 THE CLINICAL CORPUS

First of all, special properties of clinical texts need to be considered. Such records are created in a special environment, thus they differ from general Hungarian in several respects. These attributes are the following (cf. [30, 29, 27]): *a*) notes contain a lot of erroneously spelled words, *b*) sentences generally lack punctuation marks and sentence initial capitalization, *c*) measurements are frequent and have plenty of different (erroneous) forms, *d*) a lot of (non-standard) abbreviations occur in such texts, *e*) and numerous medical terms are used that originate from Latin.

Since there was no corpus of clinical records available that was manually annotated with morphological analyses, a new one was created for testing purposes. This corpus contains about 600 sentences, which were extracted from the notes of 24 different clinics. First, the textual parts of the records were identified (as described in [27]), then the paragraphs to be processed were selected randomly. Then manual sentence boundary segmentation, tokenization and normalization was performed, which were aided by methods detailed in [30]. Manual spelling correction was carried out by using suggestions provided by the system of Siklósi et al. [28]. Finally, morphological disambiguation was performed: the initial annotation was provided by PurePos, then its output was checked manually.

Several properties of the corpus created differ from general ones. Beside characteristics described above, the corpus contains numerous *x* tokens which denote multiplication and are labeled as numerals. Latin words and abbreviations are analyzed regarding their meaning: e.g. *o.* denotes *szem* ‘eye’, thus it is tagged with N.NOM. Further on, names of medicines are labeled as singular nouns. Finally, as missing sentence final punctuation marks were not recovered in the test corpus, these are not tagged either.

The corpus was split into a development and a test set (see Table 1). The first part was employed for development purposes, while the methods

Table 1. Size of the clinical corpus created

	Sentences	Tokens
Development set	240	2230
Test set	333	3155

Table 2. Distribution of errors caused by the baseline algorithm – dev. set

Class	Frequency
Abbreviations and acronyms	49.17%
Out-of-vocabulary words	27.27%
Domain-specific PoS of word forms	14.88%
Other	0.06%

detailed below were evaluated against the second part. Evaluation was carried out by calculating per-word accuracy omitting punctuation marks.

4 The Baseline Setting and the Analysis of its Errors

Below we introduce the baseline tagging chain. First we describe its components, then the performance of the tagger is evaluated by detailing the most common error types. Concerning the parts of the chain we follow the work of Orosz et al. [24]. Thus *(morphosyntactic tag, lemma)* pairs represent the analyses of HuMor, which are then disambiguated by PurePos. However, the output of the MA is extended with the new analyses of x in order to fit the corpus to be tagged.

This baseline text processing chain produced 86.61% token accuracy on the development set, which is remarkably lower than tagging results for general Hungarian using the same components (96–98% [31]). Measuring the ratio of the correctly tagged sentences revealed that less than the third (28.33%) of the sentences were tagged correctly. This amount indicates that the models used by the baseline algorithm are weak for such a task. Therefore, errors made by the baseline algorithm are investigated first to reveal how the performance could be improved.

Table 2 shows that the top error class is the mistagged abbreviations and acronyms. A reason for the high number of such errors is that most of these tokens are unknown to the tagger. Moreover, abbreviations usually refer to medical terms that originate from Latin.

Another frequent error type is caused by the out-of-vocabulary (OOV) words. This observation is in accordance with the PoS tagging results for

Table 3. Evaluation of the enhancements – test set

ID	Method	PoS tagging	Lemmatization	Morph. disambig.
0	Baseline system	90.57%	93.54%	88.09%
1	0 + Lexicon extension	93.89%	96.24%	92.41%
2	1 + Handling abbreviations	**94.81%**	**97.60%**	**93.73%**
3	2 + Training data selection	94.25%	97.36%	93.29%

medical English (as described above). Similarly, in the case of Hungarian, most of the OOV tokens are specific to the clinical domain and often originate from Latin. However, several inflected forms of such terms also exist in clinical notes due to agglutination. Therefore, listing only medical terms and their analyses could not be a proper solution. This problem requires complex algorithms.

Furthermore, the domain-specific usage of general words leads the tagger astray as well. Frequently, participles are labeled as verbs such as *javasolt* 'suggested' or *felírt* 'written'. In addition, numerous mistakes are due to the lexical ambiguity that is present in Hungarian (such as *szembe* which can refer to 'into an eye' or 'toward/against').

Our investigation shows that most of the errors of the baseline system can be classified into the three categories above. We can use the categorization above to enhance the performance of the system by eliminating the typical sources of errors.

5 Incremental Improvements

Based on the observations above, systematic changes were carried out to improve the tagging accuracy of the chain. First, the processes of lexicon extension and algorithmic modifications are described, then an investigation is presented aiming to find the optimal training data. Each enhancement is evaluated against the test corpus. Table 3 contains the part-of-speech tagging, lemmatization and the whole morphological tagging performance of each system.

5.1 *Extending the Lexicon of the Morphological Analyzer*

Supervised tagging algorithms commonly use augmented lexicons in order to reduce the number of out-of-vocabulary words (see Section 2.1). In the case of Hungarian, this must be performed at the level of the MA.

Here we describe the process which was carried out to extend the lexicon of the HuMor analyzer.

The primary source for the extension process was a spelling dictionary of medical terms [32] that contained about 90000 entries. Beside this, a freely available list of medicines [33] of about 38000 items was used as well. Since neither of these resources contained any morphological information concerning these words, such analyses were created. For this, we followed an iterative process which included both human work and automatic algorithms. The steps of our workflow were the following: (1) a set of word forms was prepared and analyzed automatically (detailed below); (2) the analyses were checked and corrected manually; (3) the training sets of the supervised learning methods were extended with the results of step (2). Before each iteration, compounds of known items were selected to be processed first. This enhancement reduced the time spent on manual correction and granted the consistency of the database created. In the end, approximately 41000 new entries were added to the lexicon of the HuMor analyzer.

Since Latinate words can either be written as pronounced in Hungarian[3] or can appear with the original Latin spelling, having both variants is necessary. Most of the entries in the dictionary had both the Hungarian and Latin spelling variants, but this was not always the case. Language identification of the words was carried out to distinguish Hungarian terms from the ones that have Greek, Latin, English or French spelling. For this, an adapted version of TextCat [34] was involved in the iterative process to decide whether a word is Hungarian or not. If it was necessary, missing Hungarian spelling variants were produced using letter-to-sound rules of Latinate words as they are generally pronounced in Hungarian and were added semi-automatically to the lexicon.

As for the calculation of the morphological analyses, the guesser algorithm of PurePos was employed. Separate modules were employed for each language, thus language-specific training sets were maintained for them as well. In Hungarian, the inflection paradigm depends on vowel harmony and the ending of the word as it is pronounced, thus the pronunciation of foreign words had to be calculated first. This could be carried out using the same simple hand-written rules implementing Latin grapheme-to-phoneme correspondences that were used to generate missing Hungarian spelling variants.

[3] An example is the Latin word *dysplasia* [displa:zia], which can be spelled as *diszplázia* in Hungarian.

The lexicon extension process above reduced the OOV word ratio from 34.57% to 26.19% (development set), and resulted in an accuracy of 92.41% (test set). Since the medical dictionary [32] contained abbreviated words as well, this process could also decrease the number of mistagged abbreviations.

5.2 *Dealing with Acronyms and Abbreviations*

Despite the changes in Section 5.1, numerous errors made by the enhanced tagger were still connected to abbreviations. Thus we first examined erroneous tags of abbreviated terms, then developed methods aiming to improve the performance of the disambiguation chain.

A detailed error analysis revealed that some of the erroneous tags of abbreviated terms were due to the over-generating nature of HuMor, which could be reduced by a filtering method. For words with full stops an analysis was considered to be false if its lemma was not an abbreviation. This modification increased the overall accuracy significantly, reducing the number of errors by 9.20% on the development set (cf. "Filtering" in Table 5).

Another typical error type was the erroneous tagging of unknown acronyms. Since PurePos did not employ features that could deal with such cases, these tokens were left to the guesser. However, acronyms should have been tagged as singular nouns. Thus a pattern matching component relying on surface features could fix their tagging (see "Acronyms" in Table 5).

The rest of the errors were mainly connected to those abbreviations that were both unknown to the analyzer and had not been seen previously. For this, the distribution of the labels of abbreviations in the development data is compared to that of the Szeged Corpus (see Table 4 below). While there are several common properties between the two columns (such as the ratio of adverbs), discrepancies occur even more often. One of them is the ratio of adjectives, which is significantly higher in the medical domain than in general Hungarian. Comparing the values, it must be noted that 10.85% of the tokens are abbreviated in the development set, while the same ratio is only 0.37% in the Szeged Corpus.

Since the noun tag was the most frequent amongst abbreviations, a plausible method was to assign N.NOM to all of these tokens (cf. "UnkN" in Table 5) and to keep the original word forms as lemmata. This baseline method resulted in a surprisingly high error rate reduction of 31.54%.

Table 4. Morphosyntactic tag frequencies of abbreviations – dev. set

Tag	Clinical texts	Szeged Corpus
N.NOM	67.37%	78.18%
A.NOM	19.07%	3.96%
CONJ	1.27%	0.50%
ADV	10.17%	11.86%
Other	2.12%	5.50%

Table 5. Comparison of the approaches aiming to handle acronyms and abbreviations – dev. set

ID	Method	Morph. disambig.
0	Medical lexicon	90.11%
1	0 + Filtering	91.02%
2	1 + Acronyms	91.41%
3	2 + UnkN	**94.12%**
4	2 + UnkUni	92.82%
5	2 + UnkMLE	94.01%

Another approach was to model the analyses of abbreviations with data observed in Table 4. The first experiment ("UnkUni") employed a uniform distribution of labels for abbreviations present in the development set as an emission probability distribution. Thus all the tags (A.NOM, A.PRO, ADV, CONJ, N.NOM, V.3SG, V.PST_PTCL) were used with equal probability as a sort of guessing algorithm.

Beside this, a better method was to use a maximum likelihood estimation for calculating a priori probabilities ("UnkMLE"). In this case, relative frequency estimates were calculated for all the tags above. While the latter approaches could increase the overall performance, none of them managed to reach the accuracy of the "UnkN" method (cf. Table 5).

5.3 *Choosing the Proper Training Data*

Since many studies showed (cf. Section 2.1) that the training data used significantly affects the result of the annotation chain, we investigated the usage of sub-corpora available in the Szeged Corpus. Several properties of the corpus were examined (cf. Table 6) in order to find the training dataset that fits best for tagging clinical Hungarian. Measurements regarding the development set were calculated manually where it was necessary.

Table 6. Properties of training corpora

Corpus	Avg. sent. length	Abbrev. ratio	Unknown ratio	Perplexity	
				Words	Tags
Szeged Corpus	16.82	0.37%	**1.78%**	2318.02	22.56
Fiction	12.30	0.10%	2.44%	995.57	32.57
Compositions	13.22	0.14%	2.29%	1335.90	30.78
Computer	20.75	0.14%	2.34%	854.11	22.89
Newspaper	21.05	0.20%	2.10%	1284.89	**22.08**
Law	23.64	1.43%	2.74%	**824.42**	29.79
Short business news	23.28	0.91%	2.50%	859.33	27.88
Development set	9.29	10.85%	–	–	–

First of all, an important attribute of a corpus is the length of its sentences. Texts having shorter sentences tend to have simpler grammatical structure, while longer sentences are grammatically more complex. Further on, clinical texts have a vast amount of abbreviations, thus the ratio of abbreviations is also relevant during the comparison.

Furthermore, the accuracy of a tagging system is strongly related to the ratio of unknown words, thus these proportions were calculated for the development set using the vocabulary of each training corpus (see Table 6). This ratio could function as a similarity metric, but entropy-based measures work better [35] in such scenarios. We use perplexity, which is calculated here as follows: trigram models of word and tag sequences are trained on each corpus using Kneser-Ney smoothing, then all of them are evaluated against the development set[4].

Measurements show that there is no such part of the Szeged Corpus which has as much abbreviated terms as clinical texts have. Likewise, sentences written by clinicians are significantly shorter than the ones in any of the genres present in the Szeged Corpus. Neither the calculations above, nor the ratio of unknown words suggest that we should use subcorpora for training. However, the perplexity scores contradict this: sentences from the law domain have the most phrases in common with clinical notes, while news texts have the most similar grammatical structures.

Therefore, all sub-corpora were involved in the evaluation, which was carried out by employing all of the enhancements described in previous sections. Results showed that training on news texts resulted in the highest accuracy. However, it was not able to outperform the usage of the whole corpus.

[4] The SRILM toolkit [36] was employed for the calculations.

Table 7. Evaluation of the tagger using the subcorpora as training data – test set

Corpus	Morph. disambiguation accuracy
Szeged Corpus	**93.73%**
Fiction	92.01%
Compositions	91.97%
Computer	92.73%
Newspaper	**93.29%**
Law	92.17%
Short business news	92.69%

6 CONCLUSIONS

In this study, resources and methodologies were introduced that enabled us to investigate morphological tagging of clinical Hungarian. First, a test corpus was created and was compared in detail with a general Hungarian corpus. This corpus also allowed for the evaluation of numerous tagging approaches. These experiments were based on the PurePos tagger tool and the HuMor morphological analyzer. Errors made by the baseline morphological disambiguation chain were investigated, then several enhancements were carried out aiming at correcting the most common mistakes of the baseline algorithm. Amongst others, we extended the lexicon of the morphological analyzer and introduced several methods to handle the errors caused by abbreviations.

The baseline setup labeled every eighth token erroneously. Although this tagging chain is commonly used for parsing general Hungarian, it resulted in mistagged medical sentences in two thirds of the cases. In contrast, our enhancements raised the ceiling of the tagging accuracy to 93.73% by eliminating almost half (47.36%) of the mistakes. Deeper investigation revealed that this error rate reduction was mainly due to the usage of the extended lexicon, which significantly decreased the number of the out-of-vocabulary tokens. While this research did not manage to find decent training data for tagging clinical Hungarian, it showed that neither part of the Szeged Corpus was able to outperform the whole as a training corpus. Finally, results of tagging abbreviations suggest that abbreviated terms should not be tagged directly. They should be resolved first or should be labeled with a uniform tag.

The main limitation of this research is the corpus used. It contains a few hundred sentences, which is only enough to reveal the main pitfalls of the tagging method. Furthermore, most of the domain adaptation methods

rely on target-specific corpora that have several thousands of sentences. Taking these into consideration, further investigation should involve more manually annotated data from the medical domain.

In sum, commonly used methodologies alone fail to tag Hungarian clinical texts with a satisfactory accuracy. One of the main problems is that such algorithms are not able to deal with the tagging of abbreviations. However, our results suggests that the usage of an extended lexicon considerably increases the accuracy of an HMM tagger.

ACKNOWLEDGMENT We would like to thank Nóra Wenszky for comments on preliminary versions of this paper. The work was partially supported by TÁMOP – 4.2.1.B – 11/2/KMR-2011-0002 and TÁMOP – 4.2.2/B – 10/1–2010–0014.

REFERENCES

1. Oleynik, M., Nohama, P., Cancian, P.S., Schulz, S.: Performance analysis of a POS tagger applied to discharge summaries in Portuguese. Studies in health technology and informatics **160**(2) (2009) 959–963
2. Røst, T.B., Huseth, O., Nytrø, Ø., Grimsmo, A.: Lessons from developing an annotated corpus of patient histories. Journal of Computing Science and Engineering **2**(2) (2008) 162–179
3. Pakhomov, S.V.S., Coden, A., Chute, C.G.: Developing a corpus of clinical notes manually annotated for part-of-speech. International Journal of Medical Informatics **75**(6) (2006) 418–429
4. Savova, G.K., Masanz, J.J., Ogren, P.V., Zheng, J., Sohn, S., Schuler, K.K., Chute, C.G.: Mayo clinical text analysis and knowledge extraction system (cTAKES): Architecture, component evaluation and applications. Journal of the American Medical Informatics Association **17**(5) (2010) 507–513
5. Smith, L.H., Rindflesch, T.C., Wilbur, W.J.: The importance of the lexicon in tagging biological text. Natural Language Engineering **12**(4) (2006) 335–351
6. Coden, A., Pakhomov, S.V., Ando, R.K., Duffy, P.H., Chute, C.G.: Domain-specific language models and lexicons for tagging. Journal of Biomedical Informatics **38**(6) (2005) 422–430
7. Ferraro, J.P., Daumé, H.I., DuVall, S.L., Chapman, W.W., Harkema, H., Haug, P.J.: Improving performance of natural language processing part-of-speech tagging on clinical narratives through domain adaptation. Journal of the American Medical Informatics Association (2013) 931–939
8. Miller, J., Torii, M., Vijay-Shanker, K.: Building domain-specific taggers without annotated (domain) data. In: Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning. (2007) 1103–1111

9. Barrett, N., Weber-Jahnke, J.: A token centric part-of-speech tagger for biomedical text. In Peleg, M., Lavrač, N., Combi, C., eds.: Artificial Intelligence in Medicine. Volume 6747 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2011) 317–326
10. Pestian, J., Itert, L., Duch, W.: Development of a pediatric text-corpus for part-of-speech tagging. In Klopotek, M.A., Wierzchon, S.T., Trojanowski, K., eds.: Intelligent Information Processing and Web Mining. Advances in Soft Computing, Springer (2004) 219–226
11. Hahn, U., Wermter, J.: Tagging medical documents with high accuracy. In Zhang, C., Guesgen, H.W., Yeap, W.K., eds.: PRICAI 2004: Trends in Artificial Intelligence. Volume 3157 of Lecture Notes in Computer Science., Springer (2004) 852–861
12. Liu, K., Chapman, W., Hwa, R., Crowley, R.S.: Heuristic sample selection to minimize reference standard training set for a part-of-speech tagger. Journal of the American Medical Informatics Association **14**(5) (2007) 641–650
13. Choi, J.D., Palmer, M.: Fast and robust part-of-speech tagging using dynamic model selection. In: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2, Association for Computational Linguistics, The Association for Computer Linguistics (2012) 363–367
14. Ruch, P., Baud, R., Bouillon, P., Robert, G.: Minimal commitment and full lexical disambiguation: Balancing rules and hidden Markov models. In: Proceedings of the 2nd Workshop on Learning Language in Logic and the 4th Conference on Computational Natural Language Learning, Association for Computational Linguistics (2000) 111–114
15. Divita, G., Browne, A.C., Loane, R.: dTagger: A POS tagger. In: AMIA Annual Symposium Proceedings, American Medical Informatics Association (2006) 200–203
16. Brill, E.: A simple rule-based part of speech tagger. Proceedings of the Third Conference on Applied Natural Language Processing **28**(4) (1992) 152–155
17. Brants, T.: Tnt – a statistical part-of-speech tagger. In: Proceedings of the Sixth Conference on Applied Natural Language Processing, Universität des Saarlandes, Computational Linguistics, Association for Computational Linguistics (2000) 224–231
18. Dwivedi, S.K., Sukhadeve, P.P.: Rule-based part-of-speech tagger for homoeopathy clinical realm. IJCSI International Journal of Computer Science **8**(4) (July 2011) 350–354
19. Daumé III, H.: Frustratingly easy domain adaptation. In: Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, Prague, Czech Republic, Association for Computational Linguistics (June 2007) 256–263
20. Csendes, D., Csirik, J., Gyimóthy, T.: The Szeged corpus: A POS tagged and syntactically annotated Hungarian natural language corpus. In Sojka, P., Kopecek, I., Pala, K., eds.: Text, Speech, and Dialog. Volume 3206 of Lecture Notes in Computer Science. Springer (2004) 19–23

21. Erjavec, T.: Multext-east: Morphosyntactic resources for central and eastern european languages. Language Resources and Evaluation **46**(1) (March 2012) 131–142
22. Prószéky, G.: Industrial applications of unification morphology. In: Proceedings of the Fourth Conference on Applied Natural Language Processing. ANLC '94, Morristown, NJ, USA, Association for Computational Linguistics (October 1994) 213–214
23. Trón, V., Halácsy, P., Rebrus, P., Rung, A., Vajda, P., Simon, E.: Morphdb.hu: Hungarian lexical database and morphological grammar. In: Proceedings of the Fifth Conference on International Language Resources and Evaluation, Genoa (2006) 1670–1673
24. Orosz, G., Novák, A.: PurePos 2.0: A hybrid tool for morphological disambiguation. In: Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2013), Hissar, Bulgaria (2013) 539–545
25. Zsibrita, J., Vincze, V., Farkas, R.: magyarlanc: A toolkit for morphological and dependency parsing of Hungarian. In: Proceedings of Recent Advances in Natural Language Provessing 2013, Hissar, Bulgaria, Association for Computational Linguistics (2013) 763–771
26. Toutanova, K., Klein, D., Manning, C., Singer, Y.: Feature-rich part-of-speech tagging with a cyclic dependency network. In Hearst, M., Ostendorf, M., eds.: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology, Edmonton, Canada, Association for Computational Linguistics (2003) 173–180
27. Siklósi, B., Orosz, G., Novák, A., Prószéky, G.: Automatic structuring and correction suggestion system for Hungarian clinical records. In De Pauw, G., de Schryver, G.M., Forcada, M.L., M. Tyers, F., Waiganjo Wagacha, P., eds.: 8th SaLTMiL Workshop on Creation and use of basic lexical resources for less-resourced languages, Istanbul (2012) 29–34
28. Siklósi, B., Novák, A., Prószéky, G.: Context-aware correction of spelling errors in Hungarian medical documents. In Dediu, A.H., Martín-Vide, C., Mitkov, R., Truthe, B., eds.: Statistical Language and Speech Processing. Volume 7978 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2013) 248–259
29. Siklósi, B., Novák, A. In: Detection and Expansion of Abbreviations in Hungarian Clinical Notes. Volume 8265 of Lecture Notes in Artificial Intelligence. Springer-Verlag, Heidelberg (2013) 318–328
30. Orosz, G., Novák, A., Prószéky, G.: Hybrid text segmentation for Hungarian clinical records. In: Lecture Notes in Artificial Intelligence. Advances in Artificial Intelligence and Its Applications. Springer, Berlin Heidelberg (2013) 306–317
31. Orosz, G., Novák, A.: Purepos – an open source morphological disambiguator. In Sharp, B., Zock, M., eds.: Proceedings of the 9th International

Workshop on Natural Language Processing and Cognitive Science, Wroclaw (2012) 53–63
32. Fábián, P., Magasi, P.: Orvosi helyesíírási szótár. Akadémiai Kiadó, Budapest (1992)
33. Főigazgatóság, O.G.I.: Forgalomba hozatali engedéllyel rendelkező allopátiás és homeopátiás készítmények. `http://www.ogyi.hu/generalt_listak/tk_lista.csv` Online; accessed 20-December-2012.
34. Cavnar, W.B., Trenkle, J.M.: N-gram-based text categorization. In: Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval, Las Vegas, US (1994) 161–175
35. Kilgarriff, A., Rose, T.: Measures for corpus similarity and homogeneity. In Ide, N., Voutilainen, A., eds.: Proceedings of the Third Conference on Empirical Methods for Natural Language Processing, Association for Computational Linguistics (1998) 46–52
36. Stolcke, A.: Srilm – an extensible language modeling toolkit. In Hansen, J.H.L., Pellom, B.L., eds.: Proceedings International Conference on Spoken Language Processing, ISCA (November 2002) 257–286

GYÖRGY OROSZ
FACULTY OF INFORMATION TECHNOLOGY AND BIONICS,
PÁZMÁNY PÉTER CATHOLIC UNIVERSITY,
50/A PRÁTER STREET, 1083 BUDAPEST, HUNGARY
AND
MTA-PPKE HUNGARIAN LANGUAGE TECHNOLOGY RESEARCH GROUP,
50/A PRÁTER STREET, 1083 BUDAPEST, HUNGARY
E-MAIL: <OROSZGY@ITK.PPKE.HU>

ATTILA NOVÁK
FACULTY OF INFORMATION TECHNOLOGY AND BIONICS,
PÁZMÁNY PÉTER CATHOLIC UNIVERSITY,
50/A PRÁTER STREET, 1083 BUDAPEST, HUNGARY
AND
MTA-PPKE HUNGARIAN LANGUAGE TECHNOLOGY RESEARCH GROUP,
50/A PRÁTER STREET, 1083 BUDAPEST, HUNGARY
E-MAIL: <NOVAK.ATTILA@ITK.PPKE.HU>

GÁBOR PRÓSZÉKY
FACULTY OF INFORMATION TECHNOLOGY AND BIONICS,
PÁZMÁNY PÉTER CATHOLIC UNIVERSITY,
50/A PRÁTER STREET, 1083 BUDAPEST, HUNGARY
AND
MTA-PPKE HUNGARIAN LANGUAGE TECHNOLOGY RESEARCH GROUP,
50/A PRÁTER STREET, 1083 BUDAPEST, HUNGARY
E-MAIL: <PROSZEKYITK.PPKE.HU>

Author Index

Elena Lloret
Bernardo Magnini
Cerstin Mahlow
Suresh Manandhar
Diana Mccarthy
Alexander Mehler
Rada Mihalcea
Evangelos Milios
Dunja Mladenic
Marie-Francine Moens
Masaki Murata
Preslav Nakov
Costanza Navarretta
Roberto Navigli
Vincent Ng
Joakim Nivre
Attila Novák
Kjetil Nørvåg
Kemal Oflazer
Constantin Orasan
Ekaterina Ovchinnikova
Ivandre Paraboni
Saint-Dizier Patrick
Maria Teresa Pazienza
Ted Pedersen
Viktor Pekar
Anselmo Peñas
Octavian Popescu
Marta R. Costa-Jussà
German Rigau
Horacio Rodriguez
Paolo Rosso
Vasile Rus
Kepa Sarasola
Roser Sauri
Hassan Sawaf
Satoshi Sekine
Serge Sharoff
Grigori Sidorov
Kiril Simov
Vivek Kumar Singh
Vaclav Snasel
Thamar Solorio
Efstathios Stamatatos
Carlo Strapparava
Tomek Strzalkowski
Maosong Sun
Stan Szpakowicz
Mike Thelwall
Jörg Tiedemann
Christoph Tillmann
George Tsatsaronis
Dan Tufis
Olga Uryupina
Karin Verspoor
Manuel Vilares Ferro
Aline Villavicencio
Piotr W. Fuglewicz
Savas Yildirim

ADDITIONAL REFEREES FOR THIS VOLUME

Mahmoud Abunasser
Naveed Afzal
Iñaki Alegria
Hanna Bechara
Houda Bouamor
Janez Brank
Chen Chen
Víctor Darriba
Owen Davison
Ismaïl El Maarouf
Mahmoud El-Haj
Milagros Fernández-Gavilanes

Daniel Fernández-González
Corina Forascu
Kata Gabor
Mercedes García Martínez
Diman Ghazi
Rohit Gupta
Francisco Javier Guzman
Kazi Saidul Hasan
Radu Ion
Zahurul Islam
Milos Jakubicek
Antonio Jimeno
Olga Kolesnikova
Mohammed Korayem
Tobias Kuhn
Majid Laali
Yulia Ledeneva
Andy Lücking
Tokunbo Makanju
Raheleh Makki
Akshay Minocha
Abidalrahman Moh'D
Zuzana Neverilova
An Ngoc Vo
Mohamed Outahajala
Michael Piotrowski
Soujanya Poria
Francisco Rangel
Amir Hossein Razavi
Francisco Ribadas-Pena
Alvaro Rodrigo
Armin Sajadi
Paulo Schreiner
Djamé Seddah
Karan Singla
Vit Suchomel
Aniruddha Tammewar
Yasushi Tsubota
Francisco José Valverde Albacete
Kassius Vargas Prestes
Tim Vor der Brück
Tadej Štajner