Peggy Cellier
Thierry Charnois
Andreas Hotho
Stan Matwin
Marie-Francine Moens
Yannick Toussaint (Eds.)

Interactions between Data Mining and Natural Language Processing

1st International Workshop, DMNLP 2014
Nancy, France, September 2014
Proceedings

Volume Editors

Peggy Cellier
INSA Rennes, IRISA
Campus Beaulieu, 35042 Rennes cedex, France
E-mail: peggy.cellier@irisa.fr

Thierry Charnois
Université Paris 13 Sorbonne Paris Cité, LIPN CNRS
Av. J.B. Clément, 93430 Villetaneuse, France
E-mail: Thierry.Charnois@lipn.univ-paris13.fr

Andreas Hotho
University of Würzburg
Am Hubland, 97074 Würzburg, Germany
E-mail: hotho@informatik.uni-wuerzburg.de

Stan Matwin
Faculty of Computer Science, Dalhousie University
6050 University Ave., PO BOX 15000, Halifax, NS B3H 4R2, Canada
E-mail: stan@cs.dal.ca

Marie-Francine Moens
Department of Computer Science, KU Leuven
Celestijnenlaan 200A, B-3001 Heverlee, Belgium
E-mail: sien.moens@cs.kuleuven.be

Yannick Toussaint
INRIA Nancy Grand-Est, LORIA
615 Rue du Jardin Botanique, 54600 Villers-lès-Nancy, France
E-mail: Yannick.Toussaint@loria.fr

Preface

Recently, a new field has emerged taking benefit of both domains: Data Mining (DM) and Natural Language Processing (NLP). Indeed, statistical and machine learning methods hold a predominant position in NLP research[1], advanced methods such as recurrent neural networks, Bayesian networks and kernel based methods are extensively researched, and "may have been too successful (. . .) as there is no longer much room for anything else"[2]. They have proved their effectiveness for some tasks but one major drawback is that they do not provide human readable models. By contrast, symbolic machine learning methods are known to provide more human-readable model that could be an end in itself (e.g., for stylistics) or improve, by combination, further methods including numerical ones. Research in Data Mining has progressed significantly in the last decades, through the development of advanced algorithms and techniques to extract knowledge from data in different forms. In particular, for two decades Pattern Mining has been one of the most active field in Knowledge Discovery.

This volume contains the papers presented at the ECML/PKDD 2014 workshop: DMNLP'14, held on September 15, 2014 in Nancy. DMNLP'14 (Workshop on Interactions between Data Mining and Natural Language Processing) is the first workshop dedicated to Data Mining and Natural Language Processing cross-fertilization, *i.e* a workshop where NLP brings new challenges to DM, and where DM gives future prospects to NLP. It is well-known that texts provide a very challenging context to both NLP and DM with a huge volume of low-structured, complex, domain-dependent and task-dependent data. The objective of DMNLP is thus to provide a forum to discuss how Data Mining can be interesting for NLP tasks, providing symbolic knowledge, but also how NLP can enhance data mining approaches by providing richer and/or more complex information to mine and by integrating linguistic knowledge directly in the mining process.

Out of 23 submitted papers, 9 were accepted as regular papers amounting to an acceptance rate of 39%. In addition to regular contributions, two less mature works, which were still considered valuable for discussion, were accepted as posters and appear as extended abstract in this volume.

The high quality of the program of the workshop was ensured by the much-appreciate work of the authors and the Program Committee members. Finally, we wish to thank the local organization team of ECML/PKDD 2014, and more specifically Amedeo Napoli and Chedy Raïssy, and the ECML/PKDD 2014 workshop chairs Bettina Berendt and Patrick Gallinari.

September 2014

Peggy Cellier, Thierry Charnois
Andreas Hotho, Stan Matwin
Marie-Francine Moens, Yannick Toussaint

[1] D. Hall, D. Jurafsky, and C. M. Manning. Studying the History of Ideas Using Topic Models. In Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing, pp. 363–371, 2008

[2] K. Church. A Pendulum Swung Too Far. Linguistic Issues in Language Technology, Vol. 6, CSLI publications, 2011.

Organization

Program Chairs

Peggy Cellier	INSA Rennes, IRISA, France
Thierry Charnois	Université Paris 13, Sorbonne Paris cité, LIPN, France
Andreas Hotho	University of Kassel, Germany
Stan Matwin	Dalhousie University, Canada
Marie-Francine Moens	Katholieke Universiteit Leuven, Belgium
Yannick Toussaint	INRIA Nancy Grand-Est, LORIA, France

Program Commitee

Martin Atzmueller	University of Kassel, Germany
Delphine Battistelli	MoDyCo-Université Paris Ouest, France
Yves Bestgen	F.R.S-FNRS, Université catholique de Louvain, Belgium
Philipp Cimiano	University of Bielefeld, Germany
Bruno Cremilleux	Universit de Caen, France
Beatrice Daille	Laboratoire d'Informatique de Nantes Atlantique, France
Francois Jacquenet	Laboratoire Hubert Curien, France
Jiri Klema	Czech Technical University, Prague, Czech Republic
Yves Lepage	Waseda University, Japan
Amedeo Napoli	LORIA Nancy, France
Adeline Nazarenko	Université de Paris 13, LIPN, France
Claire Nédellec	Institut National de Recherche Agronomique, France
Maria Teresa Pazienza	University of Roma "Tor Vergata", Italy
Pascal Poncelet	LIRMM Montpellier, France
Stephen Poteet	Boeing, USA
Solen Quiniou	Laboratoire d'Informatique de Nantes Atlantique, France
Mathieu Roche	Cirad, TETIS, Montpellier, France
Arnaud Soulet	Université François Rabelais Tours, France
Steffen Staab	University of Koblenz-Landau, Germany
Koichi Takeuchi	Okayama University, Japan
Isabelle Tellier	Lattice, Paris, France
Johanna Völker	University of Mannheim, Germany
Xifeng Yan	University of California at Santa Barbara, USA
Pierre Zweigenbaum	LIMSI-CNRS, Paris, France

Additional Reviewers

Éric Kergosien	LIRMM, Montpellier, France

Table of Contents

Regular papers

Extended abstracts

Automatically Detecting and Rating Product Aspects from Textual Customer Reviews

Wouter Bancken, Daniele Alfarone and Jesse Davis

Department of Computer Science, KU Leuven
Celestijnenlaan 200A - box 2402, 3001 Leuven, Belgium
wouter.bancken@student.kuleuven.be
daniele.alfarone@cs.kuleuven.be
jesse.davis@cs.kuleuven.be

Abstract. This paper proposes a new approach to aspect-based sentiment analysis. The goal of our algorithm is to obtain a summary of the most positive and the most negative aspects of a specific product, given a collection of free-text customer reviews. Our approach starts by matching handcrafted dependency paths in individual sentences to find opinions expressed towards candidate aspects. Then, it clusters together different mentions of the same aspect by using a WordNet-based similarity measure. Finally, it computes a sentiment score for each aspect, which represents the overall emerging opinion of a group of customers towards a specific aspect of the product. Our approach does not require any seed word or domain-specific knowledge, as it only employs an off-the-shelf sentiment lexicon. We discuss encouraging preliminary results in detecting and rating aspects from on-line reviews of movies and MP3 players.

Keywords: aspect-based sentiment analysis, opinion mining, syntactic dependency paths, text mining

1 Introduction

Sentiment analysis is the task of detecting subjectivity in natural language. Approaches to this task mainly draw from the areas of natural language processing, data mining, and machine learning. In the last decade, the exponential growth of opinionated data on the Web fostered a strong interest in the insights that sentiment analysis could reveal. For example, companies can analyze user reviews on the Web to obtain a good picture of the general public opinion on their products at very little cost.

While the first efforts in sentiment analysis were directed towards determining the general polarity (positive or negative) of a certain sentence or document, the interest has recently shifted towards a more qualitative analysis, that aims to detect the different *aspects* of a topic towards which an opinion is expressed. For example, we may be interested in analyzing a movie review to capture the opinions of the reviewer towards aspects such as the plot, the cinematography,

In: P. Cellier, T. Charnois, A. Hotho, S. Matwin, M.-F. Moens, Y. Toussaint (Eds.): Proceedings of DMNLP, Workshop at ECML/PKDD, Nancy, France, 2014.

or the performance of a specific actor. The most challenging part in aspect-based sentiment analysis is that a system needs to detect the relevant aspects before these can be associated with a polarity.

In this paper we introduce Aspectator, a new algorithm for automatically detecting and rating product aspects from customer reviews. Aspectator can discover candidate aspects by simply matching few syntactic dependency paths, while other approaches [6, 14, 16, 21] require seed words in input and use syntactic dependencies or some bootstrapping technique to discover new words and the relations between them. Additionally, it does not require any domain-specific knowledge in input, but only few handcrafted syntactic dependency paths and an off-the-shelf sentiment lexicon. Consequently, the proposed system can detect and rate aspects of products in any domain, while many existing approaches [16, 21, 18] focus on domains for which machine-readable knowledge is available. Concretely, Aspectator combines a first high-recall step where candidate aspects are extracted from individual sentences through syntactic dependency paths, with a second and third high-precision steps, where aspect mentions are clustered and their sentiment scores are aggregated by leveraging an external sentiment lexicon.

In our opinion, the considered setting represents an ideal testbed for investigating interactions between natural language processing and data mining. Indeed, our focus is not on extracting the aspects discussed in a single sentence or document, which could be seen as a problem of deep text understanding, but on crunching hundreds of reviews of a specific product to capture the aspects that best summarize the opinions of a group of customers, which requires linguistic knowledge to extract information from single sentences, along with data mining expertise to make sense of large amounts of data.

2 Related Work

Historically, sentiment analysis has been concerned with assigning a binary classification to sentences or entire documents, that represents the polarity (i.e., the orientation) of the writer towards the discussed contents [13, 19]. Nevertheless, the overall polarity gives no indication about which *aspects* the opinions refer to. For this reason, in 2004 Hu and Liu [6] introduced the more interesting problem of aspect-based sentiment analysis, where polarity is not assigned to sentences or documents, but to single aspects discussed in them. In their approach, given a large number of reviews for a specific product, they first attempt to identify interesting product aspects by using association mining, and then attach a sentiment score to each aspect by exploiting a small seed set of opinion words, along with their synonyms and antonyms present in WordNet. Next, they use newly detected opinion words to extract additional infrequent product aspects. Instead of using association mining, our work will detect aspects through dependency paths, and will use an external sentiment lexicon to rate them. However, their work remains the most similar to ours, as in both cases the goal is to summarize a collection of reviews for a specific product by detecting the most interesting and discussed aspects, while most approaches focus on analyzing individual reviews.

Qiu et al. [14] continued to pursue the idea that opinion words can be used to detect product aspects and vice versa, focusing on single reviews. In their approach, a seed set of opinion words is combined with syntactic dependencies to identify product aspects and new opinion words. To detect the polarity of the newly identified opinion words, they consider the given polarities of the seed words and make the assumption that opinion words expressing a sentiment towards the same aspect in the same review share the same polarity. While Qiu et al. use syntactic dependencies solely to capture word sequences that contain aspects or opinion words already observed, our approach uses dependency paths to detect new product aspects, with the potential advantage of achieving higher coverage.

A different line of work on aspect-based sentiment analysis is based on topic models. Brody and Elhadad [3] have tried to use Latent Dirichlet Allocation (LDA) [2] to extract topics as product aspects. To determine the polarity towards each topic/aspect, they start from a set of seed opinion words and propagate their polarities to other adjectives by using a label propagation algorithm. Instead of treating aspect detection and sentiment classification as two separate problems, Lin and He [11] and Jo and Oh [8] directly integrate the sentiment classification in the LDA model, so that it natively captures the sentiment towards the topic/aspect. While these LDA-based approaches provide an elegant model of the problem, they produce topics that are often not directly interpretable as aspects, and thus require manual labelling to achieve a readable output.

The work discussed so far proposes domain-independent solutions for aspect-based sentiment analysis, where also our approach is positioned. However, several works make use of domain-specific knowledge to improve their results. For instance, Thet et al. [16] focus on aspect-based classification of movie reviews, and include as input for their algorithm movie-specific terms such as the name of the movie, the cast and the director. Additionally, they include some domain-specific opinion words as input for their algorithm. As expected, including domain-specific knowledge yields a more accurate sentiment classification. To make an example, the word "*unpredictable*" has a negative polarity in general English, but in the movie domain it is often used to praise the unpredictability of a storyline. Since all relevant aspects are given as input, they exclusively focus on detecting opinions towards the given aspects by (1) capturing new opinion words through syntactic dependencies, and (2) rating the product aspects based on an external sentiment lexicon and some given domain-specific opinion words.

Similarly, Zhu et al. [21] use product aspects and some aspect-related terms as input for their algorithm, but then attempt to discover new aspect-related terms by applying a bootstrapping algorithm based on co-occurrence between seed terms and new candidate terms. A sentiment score is again obtained by accessing an external sentiment lexicon. While our approach retains from these works the usage of an external lexicon, it requires neither labelled examples nor domain-specific knowledge, thus it has wider applicability.

3 Aspectator: a New Approach

Aspectator takes as input a collection of textual customer reviews for one specific product, and automatically extracts the most positive and the most negative aspects of the product, together with all the sentences that contribute to the sentiment polarity of each aspect. More precisely:

Given: a collection of textual reviews of one specific product

Extract:

- The n most *positive* product aspects, along with a list of all sentences containing positive and negative mentions of each aspect.
- The n most *negative* product aspects, along with a list of all sentences containing positive and negative mentions of each aspect.

Aspectator works in three steps, depicted in Fig. 1. First, it detects mentions of aspects and their associated opinion by matching handcrafted paths in dependency trees. Second, it clusters the different mentions of an aspect extracted in the first step by means of a WordNet-based similarity measure. Third, it attaches a sentiment score to each mention, and aggregates the scores from all mentions belonging to the same cluster in order to obtain a final sentiment score for each aspect.

Aspectator does not require labelled examples and it is domain-independent, thus it can run on any collection of reviews for a specific product. The only required external knowledge is in the form of ten handcrafted dependency paths and an English lexicon with a sentiment score for every word.

3.1 Detecting Product Aspects

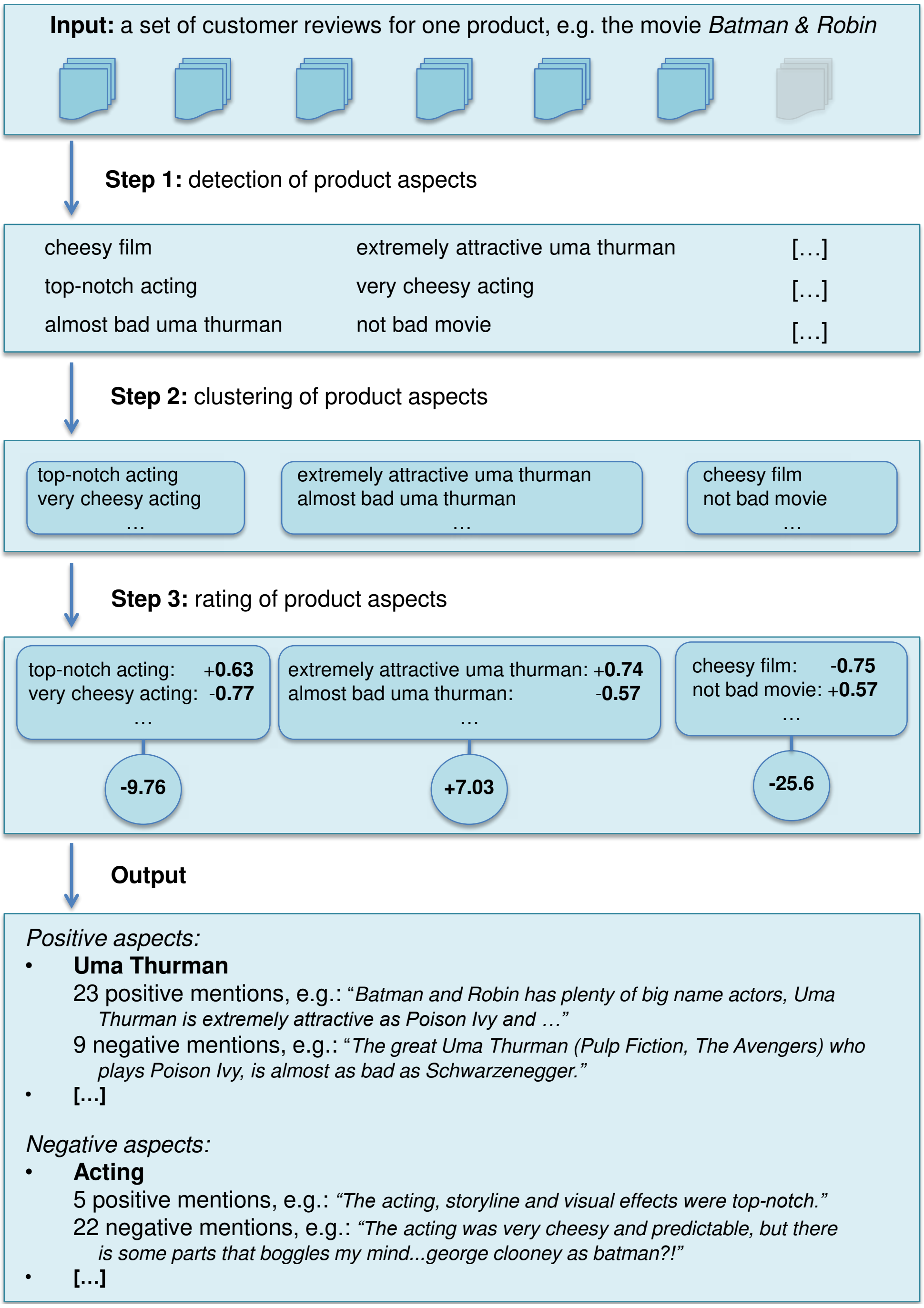
The objective of the first step is to extract from customer reviews mentions of a product aspect and the words that express the opinion of the writer towards that aspect. For instance, given the sentence:

"The action music used in the movie wasn't too good."

Aspectator extracts the following pair:

$$< \underbrace{\textbf{not}\ \text{too good}}_{\substack{\text{Sentiment}\\ \text{modifier}}} ; \underbrace{\text{action music}}_{\substack{\text{Aspect}\\ \text{mention}}} >$$

We call this an **opinion pair**, as the first part is the opinion of a reviewer towards the second part. The first part can optionally be negated, as in the above example, causing an inversion of the polarity expressed by the sentiment modifier.


Input: a set of customer reviews for one product, e.g. the movie *Batman & Robin*
Step 1: detection of product aspects
cheesy film
top-notch acting
almost bad uma thurman
extremely attractive uma thurman
very cheesy acting
not bad movie
[…]
[…]
[…]
Step 2: clustering of product aspects
top-notch acting
very cheesy acting
…
extremely attractive uma thurman
almost bad uma thurman
…
cheesy film
not bad movie
…
Step 3: rating of product aspects
top-notch acting: +**0.63**
very cheesy acting: -**0.77**
…
extremely attractive uma thurman: +**0.74**
almost bad uma thurman: -**0.57**
…
cheesy film: -**0.75**
not bad movie: +**0.57**
…
-9.76
+7.03
-25.6
Output

Positive aspects:

- **Uma Thurman**
  23 positive mentions, e.g.: *"Batman and Robin has plenty of big name actors, Uma Thurman is extremely attractive as Poison Ivy and …"*
  9 negative mentions, e.g.: *"The great Uma Thurman (Pulp Fiction, The Avengers) who plays Poison Ivy, is almost as bad as Schwarzenegger."*
- **[…]**

Negative aspects:

- **Acting**
  5 positive mentions, e.g.: *"The acting, storyline and visual effects were top-notch."*
  22 negative mentions, e.g.: *"The acting was very cheesy and predictable, but there is some parts that boggles my mind...george clooney as batman?!"*
- **[…]**

Fig. 1. ASPECTATOR's full pipeline, with example extractions from reviews for the movie *Batman & Robin*. Scores greater or lower than zero represent positive or negative sentiment polarity, respectively.

Aspectator extracts opinion pairs by using ten simple handcrafted dependency paths, in three steps:

1. For each sentence, Aspectator extracts a syntactic dependency tree by using the Stanford dependency parser [4, 10]. Fig. 2 shows the dependencies for the example sentence above.
2. Given a dependency tree, it attempts to extract a basic opinion pair composed by a single-word sentiment modifier and a single-word aspect mention by matching one of the five dependency paths shown in Table 1. For the example sentence, this step extracts the opinion pair $<$ *good* ; *music* $>$ through the dependency path $A \xleftarrow{nsubj} M \xrightarrow{cop} *$.
3. Given a matched opinion pair, it attempts to extend the match to neighbouring words by applying the additional dependency paths shown in Table 2. This allows to (1) capture multi-word expressions, such as "*action music*" and "*too good*" in the running example, and (2) capture negations, such as "*wasn't*" in the example. The final opinion pair for the running example becomes $<$ ***not*** *too good* ; *action music* $>$.

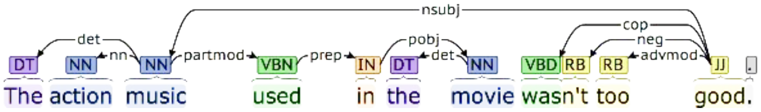


Fig. 2. Example of syntactic dependencies detected by the Stanford dependency parser.

Note that our approach leverages syntactic dependency paths for two purposes: (1) detecting aspect mentions and sentiment modifiers, and (2) discovering relations between them. This is a significant difference with other approaches that are based on syntactic dependencies. For example, Qiu et al. [14] only use syntactic dependencies to identify relations between word sequences that contain an aspect or an opinion word that has been detected before.

While our technique for extracting aspect mentions and sentiment modifiers yields high recall, its precision is low, since several irrelevant word sequences are captured. Nevertheless, the following steps allow our system to assign lower confidence to incorrect extractions, thus ultimately yielding accurate top-ranked extractions.

Table 1. Main dependency paths used by Aspectator to detect an aspect (A) and a sentiment modifier (M) that form an opinion pair $< M \,;\, A >$. Asterisks (*) are wildcards that can match any word.

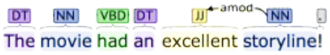
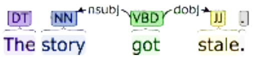
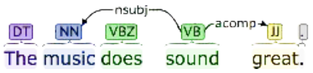
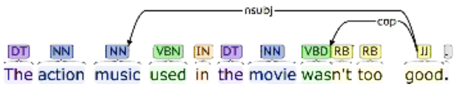
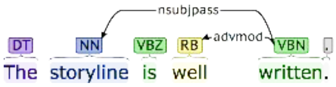
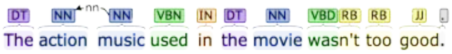
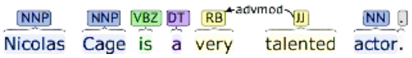
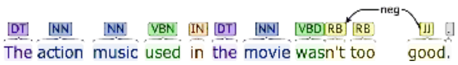
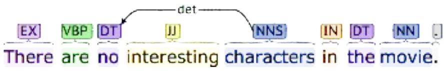
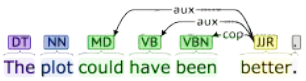
Type	**Dependency path**	**Example**
Adjectival modifier	$M \xleftarrow{amod} A$	 The movie had an excellent storyline!
Direct object	$A \xleftarrow{nsubj} * \xrightarrow{dobj} M$	 The story got stale.
Adjectival complement	$A \xleftarrow{nsubj} * \xrightarrow{acomp} M$	 The music does sound great.
Complement of a copular verb	$A \xleftarrow{nsubj} M \xrightarrow{cop} *$	 The action music used in the movie wasn't too good.
Adverbial modifier to a passive verb	$A \xleftarrow{nsubjpass} * \xrightarrow{advmod} M$	 The storyline is well written.

Table 2. Extensions to the dependency paths of Table 1 to deal with multi-word aspects (A) and multi-word sentiment modifiers (M), and to capture negations. Note that the fourth extension is the only one that imposes a lexical constraint, as it only triggers if the determiner is the word "no".

Type of extension	**Dependency path**	**Example**
Compound noun	$A' \xleftarrow{nn} A$	 The action music used in the movie wasn't too good.
Adverbial modifier	$M' \xleftarrow{advmod} M$	 Nicolas Cage is a very talented actor.
Simple negation	$* \xleftarrow{neg} M$	 The action music used in the movie wasn't too good.
Negation through "no" determiner	"no" $\xleftarrow{det} A$	 There are no interesting characters in the movie.
Negation through hypothetical phrase	$\begin{matrix} * \xleftarrow{aux} \\ * \xleftarrow{aux} \end{matrix} M \xrightarrow{cop} *$	 The plot could have been better.

3.2 Clustering Product Aspects

The goal of this step is to cluster the previously-extracted opinion pairs by searching for all semantically similar aspect mentions, independently from their sentiment modifier. For example, in the context of movie reviews, we would like to cluster together the opinion pairs < *very bad* ; *music* > and < *awesome* ; *soundtrack* >, as they both express opinions towards the same aspect of a movie.

To identify semantically similar aspect mentions, Aspectator uses a WordNet-based similarity metric called *Jcn* [7]. Zhai et al. [20] experimented with several WordNet-based similarity metrics in the context of clustering for aspect-based sentiment analysis, concluding that *Jcn* delivers the best results.

Jcn is based on the principle that two terms are similar if their least common subsumer (LCS) in the WordNet taxonomy has high information content (IC). For instance, the terms *(car, bicycle)*, having LCS "*vehicle*", are more similar than *(car, fork)*, having LCS "*artifact*", because "*vehicle*" is a more informative term than "*artifact*". Formally, the *Jcn* similarity between two terms t_1 and t_2 is defined as:

$$Jcn(t_1, t_2) = \frac{1}{IC(t_1) + IC(t_2) - 2 \cdot IC(LCS(t_1, t_2))} \tag{1}$$

where $LCS(t_1, t_2)$ is the least common subsumer of t_1 and t_2 in WordNet, and the information content of a term is equivalent to:

$$IC(t) = -log\ P(t) \tag{2}$$

where $P(t)$ is the probability of observing, in a large English corpus, the term t or any term subsumed by t in the WordNet hierarchy. The higher the probability of observing a term t or any of its subsumed terms, the lower the information content of t.

Concretely, in order to cluster opinion pairs, Aspectator first computes the *Jcn* similarity for every possible pair of aspect mentions, by using an implementation available in the WS4J library [15]. Next, it normalizes all mentions by stemming them, in order to increase data density. When two terms map to the same root, for instance "*act*" and "*acting*", a comparison with another term is made by picking the stem that maximizes the *Jcn* similarity. Finally, Aspectator uses the pairwise similarity values as input for the K-Medoids clustering algorithm [9], which will return clusters of opinion pairs, with each cluster representing a collection of opinions towards a single aspect. K-Medoids is preferred over K-Means because it can compute the centroid of a cluster without the need of defining a mean.

3.3 Rating Product Aspects

In the final stage of our approach, each cluster receives a sentiment score, which represents the overall emerging opinion of a group of customers towards a specific aspect of a product. Concretely, Aspectator undertakes three sub-steps for each cluster:

1. For each opinion pair in the cluster, it assigns an individual sentiment score to each word that composes the sentiment modifier. For instance, given the opinion pair < *just plain stupid* ; *action music* >, it attaches three individual scores to "*just*", "*plain*" and "*stupid*".
2. It combines the scores for the individual words into a single score for the entire sentiment modifier, e.g., "*just plain stupid*".
3. It extracts a final sentiment score for the entire cluster by aggregating the scores of all sentiment modifiers.

Step 1. In order to obtain a sentiment score for individual words, ASPECTATOR uses the external sentiment lexicon SentiWordNet [1]. SentiWordNet extends WordNet by attaching three scores to each *synset*:[1] a positive sentiment score, a negative sentiment score and a neutrality score. These three scores always sum to 1. For example, the word "*mediocre*", in the sense of "lacking exceptional quality or ability" has the scores 0.25, 0.125 and 0.625 as its positive, neutral and negative score, respectively.

For simplicity, our approach does not use three different sentiment scores, but combines them in one score in the range [-1,1] by subtracting the negative score from the positive score. The neutrality score is thus ignored, as "almost neutral" opinions will have a score close to zero, and consequently will have no significant impact in the following aggregation steps. Instead of performing word sense disambiguation, ASPECTATOR simply aggregates the sentiment scores of all the synsets in which a word w appears, as follows:

$$score(w) = \frac{\sum_{i=1}^{n} score(synset_i)/i}{\sum_{i=1}^{n} 1/i} \tag{3}$$

where $i \in \mathbb{N}$ is the rank of a synset in WordNet based on the synset's frequency in general English, and $synset_i$ is the i^{th} synset of w in the ranking. Intuitively, dividing a synset's score by i allows our approach to give higher weight to synsets that are more likely to represent the right sense of the word w in a certain context, given their overall higher popularity in English.

Step 2. The word-level scores obtained in the previous step are then combined into a single score for the entire sentiment modifier by adopting an approach based on the work of Thet et al. [16]. Specifically, ASPECTATOR takes as initial score the sentiment score of the rightmost (i.e., most specific) word in the sentiment modifier. Then, it iteratively uses the score of each preceding word to either intensify or attenuate the current score depending on the polarity of the considered words, remaining in the range [-1,1]. Concretely, the score for a sentiment modifier composed by words $w_n \ w_{n-1} \dots w_1 \ w_0$ is computed as:

[1] A synset is a group of synonymous words, corresponding to a node in the WordNet hierarchy.

$$\begin{aligned} score(w_i \ldots w_0) = \ & score(w_{i-1} \ldots w_0) - (score(w_{i-1} \ldots w_0) \cdot |score(w_i)|) \\ & \text{if } score(w_{i-1} \ldots w_0) > 0 \text{ and } score(w_i) < 0 \end{aligned} \tag{4a}$$

$$\begin{aligned} score(w_i \ldots w_0) = \ & \sigma \cdot \Big(|score(w_{i-1} \ldots w_0)| + (1 - |score(w_{i-1} \ldots w_0)|) \cdot |score(w_i)| \Big) \\ & \text{with } \sigma = sign(score(w_{i-1} \ldots w_0)) \\ & \text{otherwise} \end{aligned} \tag{4b}$$

In case the sentiment modifier is negated, the resulting sentiment score is multiplied by -1 to obtain the opposite polarity.

Equation (4b) models the general case, where the next word w_i in the iterative procedure intensifies the current score functionally to $|score(w_i)|$. This follows Thet et al.'s observation that (1) words with the same polarity tend to intensify each other (e.g., "*super nice*", "*terribly stupid*"), and (2) a negative current score becomes more negative when the next word has positive score (e.g., "*super bad*"). Equation (4a) is introduced to handle the particular case in which the current score is positive and the next word to be processed is negative (e.g., "*hardly interesting*"). In this case, applying (4b) would make the final score more positive, while a negative modifier should make the score less positive.

As a full example, we show how our iterative procedure computes a sentiment score for the opinion pair < *just plain stupid* ; *action music* >:

Example opinion pair: < *just plain stupid* ; *action music* >

$$\begin{array}{llll} & w_2 & w_1 & w_0 \\ \text{Individual scores:} & 0.07 & 0.12 & -0.51 \end{array}$$

$$\text{score}(\textit{plain stupid}) = (-1) \cdot \Big(0.51 + (1 - 0.51) \cdot 0.12\Big) = -0.57$$

$$\text{score}(\textit{just plain stupid}) = (-1) \cdot \Big(0.57 + (1 - 0.57) \cdot 0.07\Big) = -0.60$$

Thus, the resulting sentiment score for the aspect mention *"action music"* in this example is -0.60.

Step 3. Lastly, Aspectator computes a final sentiment score for each aspect, by summing the scores computed in the previous step for all sentiment modifiers belonging to the aspect's cluster. A simple algebraic summation supports the intuition that few strongly positive/negative opinions should result in a sentiment score comparable to the one of many weakly positive/negative opinions. We refer back to Fig. 1 for a complete example.

In order to produce the final output, Aspectator ranks the aspects by their sentiment score, and returns only the n most positive and the n most negative aspects, where n is specified by the user. This ranking places at the top the most interesting aspects, i.e., the ones that (1) are frequently mentioned in the reviews, and (2) are subjected to strong positive or negative opinions of the reviewers.

This has also the advantage that many incorrect opinion pairs extracted in the first step of the pipeline (Sect. 3.1) will be excluded from the final output, as they typically have very few mentions and are not associated with strong opinions.

4 Experiments

In this section, we present a preliminary evaluation of Aspectator. The objective of our experiments is to address the following questions:

1. Can our approach detect interesting and relevant product aspects?
2. Can our approach provide meaningful evidence that supports the sentiment score assigned to each aspect?

Additionally, we discuss the main sources of error of our approach.

4.1 Methodology

Aspectator's output was manually evaluated on a portion of two public datasets from different domains by two annotators, out of which only one was a co-author.

The first dataset is a collection of movie reviews taken from Amazon,[2] published by McAuley and Leskovec [12]. Since manual evaluation is required, we sampled ten movies to create a validation set and a test set, in the following way. First, we considered only the 50 movies with the highest number of reviews, as we want to the test the ability of our algorithm to summarize a large amount of data for a single movie. Since most of these movies have a majority of positive reviews, in order to obtain a more balanced dataset we first took the five movies with the highest number of negative reviews, and then randomly sampled five other movies from the remaining set. This resulted in a collection of 700 to 850 reviews for each movie.

The second dataset consists of reviews of MP3 players taken from Amazon,[3] published by Wang, Lu and Zhai [17]. From this dataset we selected the five products with the highest number of reviews in the dataset, obtaining a collection of 500 to 770 reviews for each MP3 player.

From these samples, we used eight movies and three MP3 players as our validation set, and the remaining two movies and two MP3 players as our test set. We used the validation set to determine the optimal k for the K-Medoids clustering applied in Sect. 3.2, which should ideally be equal to the total number of unique aspects appearing in a set of reviews. We found that the optimal k is 0.9 times the number of aspect mentions to be clustered. For instance, if 1700 aspect mentions have been identified for a certain product, we set $k = 1530$.

We used the test set consisting of two movies and two MP3 players to manually evaluate our algorithm. For each product, two annotators were given a form with the ten most positive and the ten most negative product aspects, along

[2] `http://snap.stanford.edu/data/web-Movies.html`

[3] `http://sifaka.cs.uiuc.edu/~wang296/Data/LARA/Amazon/mp3/`

with the six sentences containing the three most positive and three most negative mentions of each aspect. The annotators were asked to mark each product aspect and each sentence mentioning the aspect as either correct or incorrect. For simplicity, in the form given to the annotators each aspect was only represented by the aspect mention appearing most frequently in the reviews. An aspect is considered correct if it is an interesting and relevant aspect for the considered product, such as "*battery life*" and "*display*" for an MP3 player, or "*storyline*" and the name of an actor for a movie. A sentence listed by our algorithm for a certain aspect is considered correct only if (1) the sentence mentions the considered aspect, and (2) the sentence expresses an opinion towards the considered aspect that matches the polarity extracted by our algorithm for that specific opinion pair.

4.2 Results

The accuracy of the top-n aspects is shown in Fig. 3. On average, the 10 most positive and the 10 most negative aspects were considered to be correct in 72.5% of the cases. The sentences mentioning an aspect were only considered correct in 59.8% of the cases. However, this last result can be studied more closely. Table 3 shows the accuracy of these sentences in function of the polarity of both aspects and sentences. Clearly, the detected sentences are generally more accurate when the aspect and the corresponding sentence have the same polarity. This is due to the fact that for an aspect there are typically many more sentences with a matching polarity than sentences with the opposite polarity, so when the top-3 sentences are drawn from a larger number of sentences, these tend to have higher accuracy.

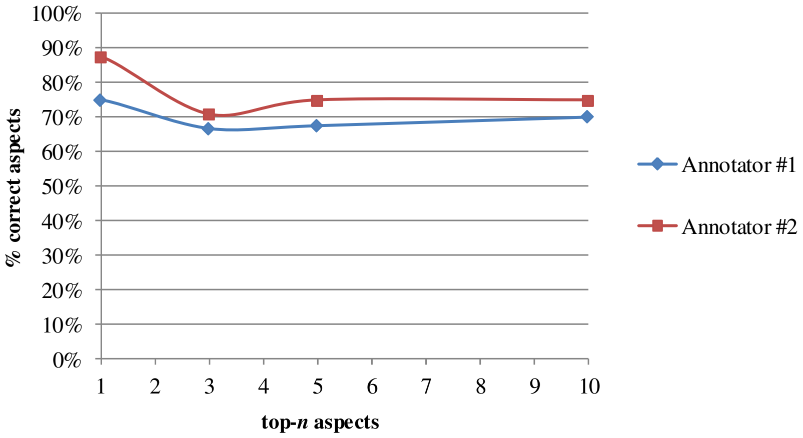


Fig. 3. Percentage of top-1, top-3, top-5, top-10 aspects marked as correct by two annotators.

Table 3. Percentages of correct sentences split accordingly to the polarity of the sentence and the product aspect.

	Pos. sentence pos. aspect	Neg. sentence pos. aspect	Neg. sentence neg. aspect	Pos. sentence neg. aspect
Annotator #1	86.9%	47.8%	66.7%	39.1%
Annotator #2	85.0%	53.0%	58.3%	41.3%
Average	85.9%	50.4%	62.5%	40.2%

Table 4. Positive and negative aspects for the movie *Batman & Robin*. Each aspect is represented as a cluster of its different mentions. Aspects shown in italics were marked as incorrect by the two annotators, while † indicates a disagreement between annotators.

Batman & Robin		
	Positive aspects	**Negative aspects**
1	*book*	movie, film, ...
2	Uma Thurman	dialogue, dialog
3	*job, occupation, ...*	† line
4	actor, thespian	acting, act, ...
5	Alicium Silverstone	† review, reviewer
6	performance, perform, ...	plot
7	Bruce Wayne	*guy*
8	effect, consequence, ...	script
9	costume	*thing*
10	*way*	character, depict, ...

When evaluating the product aspects, the annotators agreed in 88% of the cases, with an inter-annotator agreement of $\kappa = 0.69$ according to Cohen's kappa score. When evaluating the sentences containing mentions, the annotators agreed in 89% of the cases, with $\kappa = 0.785$. Table 4 shows an example of the aspects manually evaluated for the movie "*Batman & Robin*". A sentence marked as correct positive mention for the aspect "*performance*" in the same movie is:

> *"Though not classic villains, Arnold Schwarzenegger as Mr. Freeze and Uma Thurman as Poison Ivy give engaging performances."*

while for the same aspect a negative mention is:

> *"The performances were horrendous, roll call: George Clooney, Chris O'Donnel, Alicia Silverstone and Arnold."*

4.3 Error Analysis

We conclude the experimental section by reporting the main sources of errors for our approach.

Ambiguity of word polarity. The polarity of an opinion word can vary according to its context. In some cases, SentiWordNet does not cover all possible senses of a word. For instance, SentiWordNet only contains positive sentiment scores for the word "*laughable*", while in a sentence such as *"The acting is laughable."* the intended sentiment is clearly negative.

In some other cases, even though SentiWordNet covers also the correct sense of a word, ASPECTATOR picks the wrong polarity. This is due to the fact that, for simplicity, our algorithm does not perform word sense disambiguation, but instead computes a sentiment score for a term as a weighted sum of the scores of all possible senses of the term. For example, SentiWordNet contains several positive sentiment scores for different senses of the word "*joke*", and only one negative score. By using a weighted sum, the overall sentiment score is positive, while in a sentence such as *"The dialogue was just a joke."* the word is used with a negative connotation.

Inclusion of objective statements. The presence of opinion words in a review does not necessarily imply that the reviewer is expressing an opinion. For instance, when describing the antagonist of a movie, reviewers often use words with a negative connotation without the intention of expressing any judgement. This is the case in sentences like *"Javier Bardem is an extremely creepy bad guy."*, where ASPECTATOR incorrectly concludes that a strongly negative opinion is expressed towards Javier Bardem.

Limitations of dependency paths. The ten handcrafted dependency paths sometimes fail to capture the full sentiment of a sentence. To make a concrete example, consider the sentence *"Uma Thurman was really sexy as Poison Ivy... and that's about it."*. If the first part of the sentence was considered in isolation, a human reader would interpret it as a positive opinion about Uma Thurman, and ASPECTATOR does the same. Nevertheless, the second part of the sentence reveals a negative attitude of the reviewer, which our simple dependency paths cannot capture.

Incorrect dependency parsing. A last major source of error is introduced by the Stanford dependency parser. Some of the errors are caused by the inability of the Stanford parser to deal with the imprecise, colloquial language typically adopted in on-line product reviews. To make an example, from the sentence *"Man, this film is bad."* the parser indicates that the opinion word "*bad*" refers to "*man*", and not "*film*".

Additionally, the Stanford parser is not always able to detect compound nouns, as terms like "*hard drive*" are considered to be adjective-noun pairs. This makes ASPECTATOR interpret the adjective as an opinion expressed towards the noun, while the compound noun simply represents an aspect mention with no associated opinion.

5 Conclusions

We presented Aspectator, a novel algorithm for aspect-based sentiment analysis that takes in input a collection of customer reviews for a specific product, and automatically extracts the most positive and the most negative aspects, together with evidence that supports the extractions. Aspectator first harvests candidate aspects and the associated opinions by matching ten simple handcrafted dependency paths, then clusters together mentions of the same aspect, and finally computes a sentiment score that expresses the overall orientation towards each aspect. Our approach is domain-independent and does not require any labelled example, thus it can be adopted to analyze customer reviews for products in unseen domains.

In a preliminary evaluation, we show that on average the 72.5% of the extracted aspects are relevant, and that sentences that adhere to the overall polarity of each aspect are correct in 74.2% of the cases. This percentage drops to 45.3% when the sentence polarity does not match the overall aspect polarity. Furthermore, we found that most errors in our pipeline are caused by the ambiguity and the complexity of the colloquial language adopted in the reviews.

For future work, we are interested in verifying whether, starting from few example opinion pairs, we can learn the dependency paths that we now handcraft, and discover additional ones that generalize well across multiple domains. Additionally, an extended (and comparative) evaluation is required.

Acknowledgements

Daniele Alfarone and Jesse Davis acknowledge the generous support of the Research Fund K.U. Leuven (CREA/11/015 and OT/11/051), EU FP7 Marie Curie Career Integration Grant (#294068), and FWO-Vlaanderen (G.0356.12).

References

1. Baccianella S., Esuli A., Sebastiani F., Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *Proceedings of LREC*, volume 10, pages 2200–2204, 2010.
2. Blei D. M., Ng A. Y., Jordan M. I., Latent dirichlet allocation. *Journal of Machine Learning research*, 3:993–1022, 2003.
3. Brody S., Elhadad N., An unsupervised aspect-sentiment model for online reviews. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 804–812. Association for Computational Linguistics, 2010.
4. De Marneffe M.-C., MacCartney B., Manning C. D., Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*, volume 6, pages 449–454, 2006.

5. Ganu G., Elhadad N., Marian A., Beyond the stars: Improving rating predictions using review text content. In *Proceedings of the 12th International Workshop on the Web and Databases*, 2009.
6. Hu M., Liu B., Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 168–177. ACM, 2004.
7. Jiang J. J., Conrath D. W., Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings of ROCLING X*, 1997.
8. Jo Y., Oh A. H., Aspect and sentiment unification model for online review analysis. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 815–824. ACM, 2011.
9. Kaufman L., Rousseeuw P. J., Finding groups in data: an introduction to cluster analysis. John Wiley & Sons, 2009.
10. Klein D., Manning C. D., Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pages 423–430, 2003.
11. Lin C., He Y., Joint sentiment/topic model for sentiment analysis. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, pages 375–384, 2009.
12. McAuley J. J., Leskovec J., From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews. In *Proceedings of the 22nd International Conference on World Wide Web*, pages 897–908, 2013.
13. Paltoglou G., Thelwall M., A study of information retrieval weighting schemes for sentiment analysis. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1386–1395, 2010.
14. Qiu G., Liu B., Bu J., Chen C., Expanding domain sentiment lexicon through double propagation. In *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence*, volume 9, pages 1199–1204, 2009.
15. Shima H., WS4J WordNet Similarity for Java. `https://code.google.com/p/ws4j/`, 2014.
16. Thet T. T., Na J.-C., Khoo C. S., Aspect-based sentiment analysis of movie reviews on discussion boards. *Journal of Information Science*, 36(6):823–848, 2010.
17. Wang H., Lu Y., Zhai C., Latent aspect rating analysis without aspect keyword supervision. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 618–626. ACM, 2011.
18. Wang H., Lu Y., Zhai C., Latent aspect rating analysis on review text data: a rating regression approach. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2010.
19. Yessenalina A., Yue Y., Cardie C., Multi-level structured models for document-level sentiment classification. In *Proceedings of the 2010 conference on Empirical Methods in Natural Language Processing*, pages 1046–1056, 2010.
20. Zhai Z., Liu B., Xu H., Jia P., Clustering product features for opinion mining. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 347–354. ACM, 2011.
21. Zhu J., Wang H., Zhu M., Tsou B. K., Ma M., Aspect-based opinion polling from customer reviews. *IEEE Transactions on Affective Computing*, 2(1):37–49, 2011.
22. Zhu X., Ghahramani Z., Learning from labeled and unlabeled data with label propagation. Technical report, Technical Report CMU-CALD-02-107, Carnegie Mellon University, 2002.

Mining Meaning from Text by Harvesting Frequent and Diverse Semantic Itemsets

Luigi Di Caro and Guido Boella

Department of Computer Science
University of Turin, Italy
{dicaro,boella}@di.unito.it

Abstract. In this paper, we present a novel and completely-unsupervised approach to unravel meanings (or senses) from linguistic constructions found in large corpora by introducing the concept of *semantic vector*. A semantic vector is a space-transformed vector where features represent fine-grained semantic information units, instead of values of co-occurrences within a collection of texts. More in detail, instead of seeing words as vectors of frequency values, we propose to first explode words into a multitude of tiny semantic information retrieved from existing resources like WordNet and ConceptNet, and then clustering them into frequent and diverse patterns. This way, on the one hand, we are able to model linguistic data with a larger but much more dense and informative semantic feature space. On the other hand, being the model based on basic and conceptual information, we are also able to generate new data by querying the above-mentioned semantic resources with the features contained in the extracted patterns. We experimented the idea on a dataset of 640 millions of triples *subject-verb-object* to automatically inducing senses for specific input verbs, demonstrating the validity and the potential of the presented approach in modeling and understanding natural language.

Keywords: Natural Language Understanding, Distributional Semantics, Diverse Itemset Mining

1 Introduction

Most Computational Linguistics applications may need semantic information to improve their effectiveness. Semantic resources are often constructed with automatic approaches, since manual building of ontologies is not feasible on large scale [24, 42, 19, 37, 31, 5, 7].

Distributional Semantics (DS) is nowadays one of the frontiers in this field [4, 22, 8, 2, 14]. DS derives from the *Distributional Hypothesis* introduced by Z. Harris [23], where Vector Space Models (VSMs) represent its main expression [39]. The current availability of huge corpora like ukWac [17] makes these approaches particularly efficient. Data Mining (DM) techniques leveraging on VSMs have been successfully applied on text since many decades on Topic Extraction-related

In: P. Cellier, T. Charnois, A. Hotho, S. Matwin, M.-F. Moens, Y. Toussaint (Eds.): Proceedings of DMNLP, Workshop at ECML/PKDD, Nancy, France, 2014.

tasks [10, 1, 13]. Specifically, terms become interconnected by similarity scores forming concept-like entities, i.e., words clusters sharing similar contexts [9]. DS refines traditional DM on text, since it considers language as a grammar-based type of data. However, DS still sees linguistically-refined tokens as the basic bricks in VSMs, suffering of an intrinsic limitation: a wide range of words and grammar constructions is actually rarely used. Even in very large corpora, there is little chance of finding statistically-significant patterns that can be used to carve out meanings out of them. This is known as the long tail problem [45]. Moreover, DM starts from linguistic items to develop semantic information, without reusing it for further analysis.

This work is based on an interdisciplinary approach that relies on Conceptual Spaces [20], a theory developed by P. Gardenfors in the Cognitive Science field, through which concepts are represented by vectors whose features are cognitive axes that humans naturally use to give meaning to their perceptions. In this sense, rather than considering VSMs of linguistic symbols we will consider VSMs of extensive semantic information associated with them, derived from different sources. Our methodology leverages on the wealth of resources available on the web concerning semantics, like Linked Open Data (e.g., DBPedia[1], Freebase[2], etc.), linguistic resources (e.g., WordNet [29], ConceptNet [44], BabelNet [32], etc.), Semantic Web technologies (e.g., FRED [15], TPALO [33], etc.), and automatic parsing of large corpora like Wikipedia to map linguistic contexts into semantic features.

An initial proof-of-concept of the proposal is given by recent research in which the transformation of terms into top-level hypernyms carried to improvement in several computational tasks, as in [28, 18]. While this is in line with this paper, this transformation only involves terminological abstractions by means of IS-A substitutions. In fact, this contribution represents a large generalization that takes into account a wider spectrum of conceptual relationships. The outcomes of this work are threefold:

- a new methodology that introduces the concept of *semantic vectors*
- a novel technique for mining frequent and *diverse* itemsets based on *set cover problem* [41], implemented with an heuristic approach.
- a model that generalizes over existing linguistic constructions with low resource requirements that is also able to generate new linguistic data

The paper first presents the motivations and the goals of this work. Then, the approach is explained in terms of methodology and algorithms. An evaluation phase is then presented, showing the data and the pursued objectives. A final part of conclusions and future work ends the paper.

[1] http://dbpedia.org/About
[2] http://www.freebase.com/

2 Background and Related Work

In Computational Linguistics, recent advances and successful experiences of statistical distributional approaches can be found in different tasks. The IBM Watson Question Answering system[3] is maybe the most recent and well-known direct result. This also explains the fortunate and growing trend of available semantic resources often constructed with automatic approaches, since manual building of ontologies is not feasible on large scale. Currently, many systems actually try to automatically extract semantic knowledge from texts by means of three possible generic approaches: distributional analysis, pattern-based approaches, and Machine Learning techniques.

Nowadays, semantic information extraction is currently approached by distributional analysis of linguistic items over specific contexts [34] or by starting from seeds and patterns to build ontologies from scratch [31]. In some cases, linguistic items are substituted by super-senses (i.e., top-level hypernyms) [28]. However, such generalization should be applied taking into account a wider notion of semantic context introduced by related cognitive theories [21], that has not been addressed by current computational approaches.

Distributional Semantics (DS) is nowadays one of the frontiers[4] within the Computational Linguistics field [3]. DS derives from the *distributional hypothesis* introduced by Z. Harris in 1954 [23]. Vector Space Models (VSMs) [39], proposed by Gerald Salton in the seventies, are the main expression of this idea. Data Mining (DM) techniques fully leveraging on VSMs and Latent Semantic Analysis (LSA) [16] have been successfully applied on text since many decades on topic extraction-related tasks, often producing concept-like entities, i.e., words clusters sharing similar contexts [9].

Current research in DS focuses on the exploration of the different impact of parameters such as *context type* (i.e., text regions vs. linguistic items), *window* (context extension), *frequency weighting strategy* (e.g., number of occurrences, Pointwise Mutual Information, etc.), *dimensionality reduction* (e.g., Latent Semantic Analysis, Random Indexing, etc.), and *similarity measure* (e.g., Cosine similarity, Jaccard's coefficient, etc.). Then, it produces co-occurrences matrices (or tensors) that model the semantics of the tokens by means of weights distributions.

DS refines traditional DM on text, since it considers language as a grammar-based type of data instead of simple unstructured paragraphs. However, DS still sees linguistically-refined tokens (words, lemmas, part-of-speech, etc.) as the basic bricks in VSMs, suffering of an intrinsic limitation: a wide range of words and grammar constructions is actually rarely used.

On the contrary, this work concerns a radical departure from this direction, releasing the assumption made by all approaches to rely on linguistic items (either terms or more context-aware tokens). The current methodology still starts

[3] http://www.ibm.com/smarterplanet/us/en/ibmwatson/

[4] See also the ERC project COMPOSES leaded by Marco Baroni. http://clic.cimec.unitn.it/composes/

from syntax and strings of text to extract semantics, while it would be more reasonable to have an automated approach which also leverages on the existing semantic resources it produces as further input. The idea at the basis of the proposed approach is to conceptually restructure the problem of DS under the light of research in Cognitive Science. The above-mentioned theory of Conceptual Spaces introduced by Peter Gardenfors is about a concept representation motivated by the notions of conceptual similarity and prototype theory [38]. A *conceptual space* is a multi-dimensional feature space where points denote objects, and regions define concepts. Its bases are composed by quality dimensions, which denote basic features in which concepts and objects can be compared, such as weight, color, taste and so on. Symbolic representations are particularly weak at modeling concept learning, which is paramount for understanding many cognitive phenomena. Concept learning is closely tied to the notion of similarity, which is also poorly served by the symbolic approach.

Taking inspiration from this vision of language, as basic bricks of DS we substitute linguistic items with a representation of their meaning in terms of sets of quality dimensions. In detail, each word can be represented by a set of semantic relationships and properties that define a specific concept by means of physical and behavioral facts. For instance, a cat has legs, claws, paws, eyes, etc. (properties); then, it usually chases mouses and it sees in the dark (behaviour); it is an animal and a feline (taxonomical information), and it can have many other relations like the places where it can be habitually found.

The Conceptual Spaces (CS) framework developed in the Cognitive Sciences field by [20] is based on a vectorial representation of concepts whose features are cognitive axes through humans naturally give meaning to their perceptions. CS is directly connectable with VSMs since it is a particular type of VSMs where features represent the conceptual level. Our approach is about injecting semantics into tokens towards a concept-level feature set. One of the most important brick in almost all Computational Linguistics tasks is the computation of similarity scores between texts at different levels: terms, sentences and discourses. As recently discussed in the literature [36], semantic similarity needs to be cross-level.

In the DS current view, the triple *subject-verb-object* extracted from the sentence "*the cat climbs a tree*" is equally seen as the triple extracted from "*the monkey climbs a tree*", since the two subjects share the same linguistic context. In this work, instead, the two situations will be differentiated and therefore more deeply understood: in the first case, it will be able to correlate the fact of having claws with the ability of climbing a tree; in the second case, this will happen for the presence of prehensile feet. This is due to the introduction of semantics within the process of distributional analysis itself. In fact, they share physical body parts with a similar kind of functionality. Since only specific semantic information are useful at a time, this new approach can also filter out non-relevant information (for instance, the fact that both are mammals with fur and teeth does not suggest the ability to climb a tree). Nowadays, the extraction of these features can be done due to the huge availability of semantic resources.

Once linguistic items are replaced by semantic representations, it becomes possible to reuse the methodology itself having as input the larger basis of semantic information created by the system, thus creating a positive feedback cycle and enlarging the possibilities of the system. We call this concept as *semantic loop*, and, to the best of our knowledge, it is the first attempt to go beyond single-processing systems that connect syntax with semantics towards recursive processing of extracted semantics. For example, the action of "*seeing*" can show a correlation with the fact of having eyes. Nowadays, the link between actions and properties of subject and objects are not used while they actually provide significant information for deeper language understanding.

This paper presents an approach that also relies on the concept of *diversity*. Diversity has been taken into account mostly in Information Retrieval (IR) scenarios, where systems become aware of the need of obtaining search results that cover different aspects of the data [12, 11]. However, this concept can be also useful in different contexts like clustering [30] and recommendation [40]. In spite of this, within the Pattern Mining (PM) and Association Rules (AR) areas, to the best of our knowledge, diversity has not been faced yet. Since our system architecture needs to manage the output of these techniques with the additional goal of producing frequent patterns that are able to cover different aspects of the input, we also revisited them in this sense.

This shift in the basic bricks opens new research questions and challenges concerning Data Mining methodologies: the problem of correlating atomic linguistic items becomes to correlate sets of features, where only some of them are actually significant. Thus, the new challenges become to understand:

- which features need to be filtered out
- which features can be combined to approximate concepts (according to Conceptual Spaces)

The advantages of the proposed research direction are the following:

- the integration of semantic information within the internal steps of the current methodology can create a virtuous loop through which semantic resources can be automatically extended.
- linguistic items are fragmented into minimal conceptual information that enables statistical approaches to overcome the problem of low-frequency words occurrences. In fact, even in very large corpora, there is little chance of finding statistically-significant patterns that can be used to carve out meanings out of text. This is known as the *long tail* problem. Statistical approaches are usually not able to propagate existing information belonging to frequent data to such *long tail*. One of the aim of this proposal is to define a linguistic framework in which both rare and frequent words are fragmented into more basic facts on which reason on, avoiding low-frequency issues.
- the use of multilingual resources will have an impact on the creation of more powerful semantic resources, that will be more independent by word-to-word translations. Within the DS field, a minimal step in this direction has already been done by means of transformations of words into general concepts

or super-senses. However, this only involves terminological abstractions by means of IS-A relationship substitutions. In fact, our proposal represents a generalization of this since it considers a wider spectrum of conceptual relationships. For example, a person can assume the role of living entity, doctor or student in the context of breathing, making surgical interventions, and studying mathematics respectively. The point is that only specific properties are activated by the context at a time, so we avoid to assign fixed top-level hypernyms for all the cases. In addition to this, the simple generalization of a linguistic item does not extend the current analysis of correlations between atomic tokens.

The outcomes of such novel approach can be many:

- a new methodology that introduces the concept of *semantic loop*, i.e., iterative use of extracted semantics as input for further extension of semantic resources
- new semantic resources, created by the use of the proposed methodology
- revisitations of Data Mining techniques for dealing with a new and more complex type of data with respect to standard VSMs applied on text
- the proposed contribution can also have impact on how semantic knowledge can be re-used or inherited from data in different languages. For instance, in case there is no translation for two words in two different languages, it will be possible to leverage their semantic information to link them automatically. Only translation at concept level it will be needed (i.e., translation of the new feature space). Thus, the semantic loop can work also for alignment of different languages.

3 Approach

Our proposal concerns an automatic methods to build a large-scale semantic framework based on a concept-level distributional analysis of the semantics contained in plain texts. Our methodology avoids manual constructions of ontologies which is known to be unfeasible. On the contrary, the method goes towards a direct and extensive exploitation of the wealth of available resources regarding semantics. In particular, it leverages different types of knowledge that can be used to transform words (intended as lemmas or generic linguistic items, from now on) into sets of extended and fine-grained semantic information. The resulting explosion of such heterogeneous knowledge, coming from different sources and methods, create a new challenge: how to align, filter, and merge it in order to feed Vector Space models with semantics, as opposite to lexical entities.

3.1 Semantic Basis

In this paper, we started focusing on ConceptNet [43], a semantic crowdsourced knowledge. In detail, the Open Mind Common Sense project developed by MIT collected unstructured common-sense knowledge by asking people to contribute over the Web. ConceptNet, a semantic graph created from a parsing of such knowledge, is its final outcome. In contrast with linguistic resources like WordNet [29], ConceptNet contains semantic information more related to common-sense facts. For this reason, it has a wider spectrum of semantic relationships but a much more sparse coverage due to the non-methodological approach that was used to build it. For instance, among the more unusual types of relationships (24 in total), it contains information like "*ObstructedBy*" (i.e., referring to what would prevent it from happening), "*CausesDesire*" (i.e., what does it make you want to do), and "*MotivatedByGoal*" (i.e., why would you do it). In addition, it also has classic relationships like "*is_a*" and "*part_of*" as in most linguistic resources. An at-a-glance view of these semantic relations is shown in Table 1.

Table 1. The relations in ConceptNet, with example sentences in English.

Relation	**Example sentence**
IsA	NP is a kind of NP.
LocatedNear	You are likely to find NP near NP.
UsedFor	NP is used for VP.
DefinedAs	NP is defined as NP.
HasA	NP has NP.
SymbolOf	NP represents NP.
CapableOf	NP can VP.
ReceivesAction	NP can be VP.
Desires	NP wants to VP.
HasPrerequisite	NPjVP requires NPjVP.
CreatedBy	You make NP by VP.
MotivatedByGoal	You would VP because you want VP.
PartOf	NP is part of NP.
CausesDesire	NP would make you want to VP.
Causes	The effect of VP is NPjVP.
MadeOf	NP is made of NP.
HasFirstSubevent	The first thing you do when you VP is NPjVP.
HasSubevent	One of the things you do when you VP is NPjVP.
AtLocation	Somewhere NP can be is NP.
HasLastSubevent	The last thing you do when you VP is NPjVP.
HasProperty	NP is AP.

In spite of this, the approach can work with other resources. For example, another type of knowledge that can have an high impact on our semantic integration comes from Linked Open Data (LOD). One of the most used LOD resources in Computational Linguistics is DBPedia, a dataset containing data directly extracted from Wikipedia. It contains more than 3 million concepts described by 1 billion triples, including descriptions in several languages. Other knowledge bases are UMBEL (i.e., a 20k subjects ontology derived from OpenCyc), GeoNames (i.e., descriptions of geographical features), and several others. Then, WordNet [29] is a large lexical database of English nouns, verbs, adjectives and adverbs that can further extend the semantic basis. All the words are therein grouped into sets of synonyms (also called synsets), each expressing a distinct concept. WordNet contains also a set of relationships that link the synsets. To make some examples, synsets can be used to extrapolate "*same_as*" properties from synonyms, then hypernyms can be mapped into "*is_a*" taxonomical information, while meronyms can be seen as "*part_of*" features.

3.2 Data for Distributional Analysis

In order to experiment the validity of the approach, we had the need of computing a distributional model starting from a large collection of texts. However, instead of parsing corpora from scratch, we used a dataset of *subject-verb-object* (SVO) triples generated as part of the NELL project[5]. This dataset contains a set of 604 million triples extracted from the entire dependency-parsed corpus *ClueWeb09* (about 230 billion tokens)[6]. The dataset also provides the frequency of each triple in the parsed corpus. We integrated a Named Entity Recognition module to transform proper names into generic semantic classes, like people and organizations[7].

3.3 Algorithm

In this section, we explain the details of the approach. In particular, the algorithm is composed by three different phases: (1) the data pre-processing step with the generation of two transactional databases (transactions of items, as in the fields of Frequent Itemset Mining and Association Rules [6]) that we also call *semantic vectors*; (2) the extraction of frequent, closed, and *diverse* itemsets (we will briefly introduce the meaning of all these names in the next paragraphs); and finally (3) the creation of *semantic verb models*, that generalize and automatically induce *senses* from entire linguistic constructions at sentence-level.

Transactional Databases Generation The first step of the algorithm regards the generation of the *semantic vectors*, i.e., vectors whose features represent conceptual and semantic facts rather than document- or context-occurrences. Since the aim of the system is to capture senses from data, we start from the root of the meaning, that is the verb. So, for a specific input verb v, we parse all the SVO triples in the datasets that have a frequency higher than a set threshold[8], and we only take those who are morphological variations of v. Then, for each one of these triples, we query ConceptNet with the subject-term and

[5] http://rtw.ml.cmu.edu/resources/svo/

[6] http://lemurproject.org/clueweb09/

[7] We used the Stanford NLP toolkit available at http://www-nlp.stanford.edu/.

[8] In our experiments we considered SVO triples that occur at least 100 times in the whole ClueWeb09 corpus, in order to remove noisy data.

the object-term, retrieving all their semantic information that will later build the new semantic space. Table 2 shows an example of the information collected in this phase.

Table 2. An example of subject- and object-terms semantic transformation for one triple of the verb "*to learn*" (*student-learns-math*). This represents one row of the two transactional databases.

Subject-term	Subject semantic features	Object-term	Object semantic features
student	*CapableOf*-study, *AtLocation*-at_school, *IsA*-person, *Desires*-learn, *PartOf*-class, *CapableOf*-read_book, ...	**math**	*IsA*-subject, *HasProperty*-useful_in_business, *UsedFor*-model_physical_world, *ReceivesAction*-teach_in_class, ...

Then, we associate each semantic information to a unique *id* and construct two transactional databases: one for the semantic information of the subjects, and one for the objects. An example of result of the first phase is shown in Table 3.

Table 3. An example of the two transactional databases created for the verb "to learn" and the ID-label association table.

Transactional DB of the subjects	Transactional DB of the objects
1 34 67 90	2 4 6 23 67 87 122 198
3 4 12 36 59 88 90 91	42 54 67 87 122 124
34 67 45	2 6 54 67 87
...	...

ID	Associated Semantic information
1	isa-young_person
2	atlocation-classroom
3	atlocation-at_school
4	capableof-learn
...	...

Diverse Itemsets Mining Once the transactional databases are built for a specific verb "v", we use techniques belonging to the field of Frequent Itemset Mining to extract *frequent patterns*, i.e, semantic features that frequently co-occur in our transactional databases.

The description of the problem is the following: let $I = i_1, i_2, ..., i_n$ be a set of *items* (i.e., our semantic features) and D be a multiset of transactions, where each transaction t is a set of items such that $t \subseteq I$. For any $X \subseteq I$, we say that a transaction t contains X if $X \subseteq t$. The set X is called *itemset*. The set of all $X \subseteq I$ (the powerset of I) naturally forms an itemset *lattice*. The count of an itemset X is the number of transactions in D that contain X. The *support* of an itemset X is the proportion of transactions in D that contain X.

An itemset X is called *frequent* if its support is greater than or equal to some given percentage threshold s, where s is called *minimum support*.

When the database contains a significant number of large frequent itemsets, mining all of them can be very expensive, since the space of itemsets to generate can be huge. However, if any subset of a frequent itemset is frequent, it can be sufficient to discover only all the maximal frequent itemsets (MFIs). A frequent itemset X is called maximal if there does not exist a frequent itemset Y such that $X \subseteq Y$. Mining frequent itemsets can thus be reduced to mining a "border" in the itemset lattice. All itemsets above the border are infrequent and those that are below the border are all frequent. Another type of frequent itemset, called *closed frequent itemset* (CFI), was proposed in [35]. A frequent itemset X is *closed* if none of its proper supersets have the same support.

In our experimentation, we used the library called SPMF for finding closed frequent itemsets[9], applying the CHARM algorithm [46]. This is done for both the transactional databases (subject and object databases associated to the verb 'v'). Since our aim is to capture all the linguistic *senses*, i.e., the different meanings connectable to the use of a specific verb, we also need to obtain itemsets that cover all the items that are found in frequent itemsets. In other words, we want to extract *diverse itemsets*, i.e., a minimal set of frequent and closed itemsets that cover all the frequent items. The concept of *diversity* has been mostly used in Information Retrieval tasks, and to the best of our knowledge there is no attempt in capturing "kind of" diverse itemsets in the current literature.

In order to produce these novel types of frequent itemsets, we viewed the problem as a *set cover problem* [41], implementing an heuristic-based approach to face it. Given a set of elements $U = i_1, i_2, ..., i_m$ (called the universe) and a set S of n sets whose union equals the universe, the set cover problem is to identify the smallest subset of S whose union equals the universe. The only parameter of the algorithm is the percentage of diversity *div* that the candidate itemsets must have with respect to the ones already selected. The main cycle of the algorithm is then over the closed itemsets, starting from the ones with the highest cardinality (i.e., the ones that cover most of the items). For each candidate itemset, if its current percentage of diversity overtakes *div*, it is added to the result set. In our experiments, we set its initial value to 0.5 (candidate itemsets must have a half of their items that are not already present in the selected itemsets). In case the insertion phase ends without having covered all the items that are contained in the input closed itemsets, the value decreases of a certain factor *alpha* (set to 0.1, in our experiments). This way, the algorithm assures its termination.

Verb Model Construction In the final phase, once obtained the frequent and diverse itemsets for both the two transactional databases, we connect all the

[9] http://www.philippe-fournier-viger.com/spmf/index.php

subject-itemsets with all the object-itemsets, weighting the connection according to the their co-occurrences in the same triples of the original dataset.

The *semantic verb model* constructed for a specific verb "v" is thus a set of weighted connections between *frequent and diverse* semantic features belonging to the subjects of "v" and *frequent and diverse* semantic features of the objects of "v". On the one hand, this is a way to summarize the semantics suggested by the verb. On the other hand, it is also a result that can be used to generate new data by querying existing semantic resources with such semantic subject- and object-itemsets. Still, this can be done without looking for words similar to frequent subjects and objects, but by finding new subjects and objects that, even if not similar in general, have certain semantic information that fill a specific context.

The resulting models are automatically calculated, and they are very concise, since in all the large and sparse semantic space only few features are relevant to certain meanings (headed by the verb). This is also in line with what stated in [26] where the authors claimed that semantics is actually structured by low-dimensionality spaces that are covered up in high-dimensional standard vector spaces.

4 Experiments and Results

In this section we present the result of the approach on different cases. In particular, we extracted all the triples in the dataset containing different verbs like *to play*, *to eat*, *to sing*, and so forth. Then, for each of these verbs we executed the algorithm and extracted the models, i.e., sets of weighted pairs of *diverse* subject- and object-itemsets. Table 4 shows some examples of the automatically extracted semantic information.

In the experiments, we wanted to evaluate the quality of the constructed models and their ability to generalize over the input data also taking into account their size in comparison with classic word-based vector spaces.

On the one hand, the approach is able to model the meanings expressed by complete verbal phrases with minimal resource requirements, as shown in Figure 1. In fact, starting from hundreds of verbal instances, the method produces itemsets with a feature space much smaller then common word spaces in which words and chunks are represented by vector spaces of the order of thousands of features. For instance, in the presented example, with a minimum support of 0.05 (i.e., 5%), the resulting model is constituted by 4 diverse itemsets for the objects and 24 for the subjects, with an average itemset cardinality of 18.5 and 12.6 respectively, covering more than 50% of the semantic features of all the input triples.

On the other hand, we calculated the *coverage* of the extracted models, that is the percentage of triples *subject-verb-object* in the input data in which at least one item is included in the extracted diverse itemsets. These results are shown in Figure 2. Notice that the coverage of the diverse itemsets is always equals to

Table 4. Examples of the main semantic information that are automatically induced for subjects and objects, tested on various verbs.

Verb	Subject semantic features	Object semantic features
to pay	*isa*-person	*relatedto*-money
to read	*isa*-person *notcapableof*-fly *desires*-clothe *capableof*-think *capableof*-love *capableof*-talk_to_each_other *desires*-privacy *partof*-society *capableof*-voice_opinion *hasa*-name	*usedfor*-read *atlocation*-library *atlocation*-newspaper
to visit	*isa*-person	*atlocation*-city *aTlocation*-museum *partof*-web_site *usedfor*-entertainment
to eat	*isa*-mammal *capableOf*-fear_death *capableOf*-cook_dinner *capableOf*-run *capableOf*-eat *capableof*-pay_bill *atLocation*-earth ...	*atlocation*-oven *usedfor*-eat *atlocation*-store *hasproperty*-delicious *atlocation*-tree *isa*-food *atlocation*-restaurant ...
to play	*notcapableof*-fly *isa*-mammal *capableof*-think *atlocation*-earth *desires*-laugh *capableof*-hear_noise *capableof*-experience_joy *partof*-society ...	*atlocation*-movie_theater *hasproperty*-fun *atlocation*-theatre *isa*-story *usedfor*-entertainment *hasproperty*-entertain *capableof*-tell_story *usedfor*-learn ...
to sing	*isa*-person *capableof*-think *capableof*-love *atLocation*-earth	*partof*-album *usedfor*-pleasure_yourself *atlocation*-record *usedfor*-have_fun *hasproperty*-melodic *usedfor*-express_feel_and_emotion *isa*-composition_of_music *createdby*-composer *atlocation*-on_cd *usedfor*-entertainment ...

the coverage of the closed itemsets, even if the formers are less than (or equal to) the latters.

To the best of our knowledge, this is the first attempt to model entire linguistic constructions *subject-verb-object* in terms of *senses* at sentence-level automatically carved out from the data by deeply analyzing co-occurrent fine-grained semantic information instead of lexical and syntactic chunks. We think that further efforts on this direction can importantly change the vision and the horizon of current Natural Language Understanding goals as well as the management of large collections of textual data with concise, generative, and semantics-based models.

5 Conclusions

This contribution represents a first effort to pass from standard word-vectors to *semantic vectors*. This causes the raise of new challenges, like the alignment

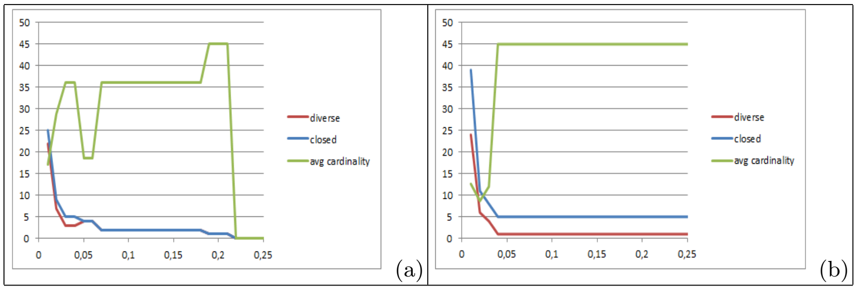


Fig. 1. Size of closed (blue line) and diverse (red line) itemsets w.r.t. minimum support, and average number of itemset cardinality (green line). The plot on the left (a) is for the subjects of the example verb "*to sing*", while the plot on the right (b) is for its direct objects.

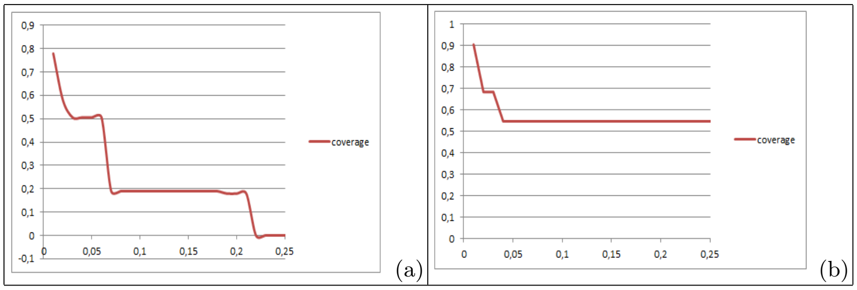


Fig. 2. Coverage w.r.t. minimum support. The plot on the left (a) is for the subjects of the example verb "*to sing*", while the plot on the right (b) is for its direct objects.

and the filtering of heterogeneous semantic information. Still, such shift in the basic bricks also concerns Data Mining techniques, since the problem of correlating linguistic items becomes to correlate sets of semantic features, where only some of them are actually significant. In this paper, we presented an approach that connect Natural Language Processing techniques (Lexico-syntactic analysis, syntactic parsing[10] and Named Entity Recognition) with Pattern Mining approaches like Frequent Itemset Mining and the cover set problem.

To produce *semantic vectors*, we started by using ConceptNet, one of the largest semantic resource currently available. In spite of this, in future work we will also come back to lexico-syntactic parsing of large corpora like Wikipedia for the extraction of further semantic information directly from text.

The impact of this new research direction can be extremely high. The main question this proposal wants to engender is the following: what if computational systems can directly reason on semantics instead of syntax? Future NLP

[10] We refer here to the used *subject-verb-object* input structures.

technologies could move away from language through more complex meaning understanding, also dealing with unseen and low-frequency words.

By reducing commonly-huge vector spaces based on linguistic items into synthetic conceptual matrices, we also attack the Big Data problem for textual databases. For example, if we think at the term "*color*", a linguistic-based vectorial representation would contain hundreds of terms that usually co-occur with it, such as "*pastel*", "*dark*", "*light*", "*red*", "*brilliant*", and so forth. In Wikipedia, for instance, we found more than 500 adjectival lemmas that co-occur with this term. On the other hand, the concept of "color" can be potentially represented by few dimensions. For instance, the HSV scheme uses only three dimensions: brightness, hue, and saturation.

We evaluated the approach by its ability to reduce the space and generalize over the input data. In future work, we will also measure the approach on tasks like Ontology Learning and Question Answering. This paper also introduces a the concept of *semantic loop*, i.e., the recursive use of extracted semantics as input for further extensions. The use of this methodology can create new and extended semantic resources.

Finally, we will leverage techniques for data compression like Multi Dimensional Scaling (MDS) [27], Principal Component Analysis (PCA) [25] and tensors decompositions to actually transform combinations of properties into reduced-spaces capturing the more significant part of the data (due to their ability to approximate information while preserving the maximum level of their expressivity). Cognitive psychology has deeply used such techniques in a wide variety of applications where the explanation of cognitive processes can be derived directly from them.

References

1. Alvanaki, F., Sebastian, M., Ramamritham, K., Weikum, G.: Enblogue: emergent topic detection in web 2.0 streams. In: Proceedings of the 2011 ACM SIGMOD International Conference on Management of data. pp. 1271–1274. ACM (2011)
2. Baroni, M.: Composition in distributional semantics. Language and Linguistics Compass 7(10), 511–522 (2013)
3. Baroni, M., Bernardi, R., Zamparelli, R.: Frege in space: A program for compositional distributional semantics. Submitted, draft at http://clic. cimec. unitn. it/composes (2013)
4. Baroni, M., Lenci, A.: Distributional memory: A general framework for corpus-based semantics. Computational Linguistics 36(4), 673–721 (2010)
5. Biemann, C.: Ontology learning from text: A survey of methods. In: LDV forum. vol. 20, pp. 75–93 (2005)
6. Borgelt, C.: Frequent item set mining. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 2(6), 437–456 (2012)
7. Buitelaar, P., Cimiano, P., Magnini, B.: Ontology learning from text: An overview. Ontology learning from text: Methods, evaluation and applications 123, 3–12 (2005)
8. Cabrera, J.M., Escalante, H.J., Montes-y Gómez, M.: Distributional term representations for short-text categorization. In: Computational Linguistics and Intelligent Text Processing, pp. 335–346. Springer (2013)

9. Candan, K., Di Caro, L., Sapino, M.: Creating tag hierarchies for effective navigation in social media. In: Proceedings of the 2008 ACM workshop on Search in social media. pp. 75–82. ACM (2008)
10. Cataldi, M., Di Caro, L., Schifanella, C.: Emerging topic detection on twitter based on temporal and social terms evaluation. In: Proceedings of the Tenth International Workshop on Multimedia Data Mining. p. 4. ACM (2010)
11. Cataldi, M., Di Caro, L., Schifanella, C.: Immex: Immersive text documents exploration system. In: Content-Based Multimedia Indexing (CBMI), 2011 9th International Workshop on. pp. 1–6. IEEE (2011)
12. Cataldi, M., Schifanella, C., Candan, K.S., Sapino, M.L., Di Caro, L.: Cosena: a context-based search and navigation system. In: Proceedings of the International Conference on Management of Emergent Digital EcoSystems. p. 33. ACM (2009)
13. Chen, Y., Amiri, H., Li, Z., Chua, T.S.: Emerging topic detection for organizations from microblogs. In: Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval. pp. 43–52. ACM (2013)
14. Croce, D., Storch, V., Annesi, P., Basili, R.: Distributional compositional semantics and text similarity. In: Semantic Computing (ICSC), 2012 IEEE Sixth International Conference on. pp. 242–249. IEEE (2012)
15. Draicchio, F., Gangemi, A., Presutti, V., Nuzzolese, A.G.: Fred: From natural language text to rdf and owl in one click. In: The Semantic Web: ESWC 2013 Satellite Events, pp. 263–267. Springer (2013)
16. Dumais, S.T.: Latent semantic analysis. Annual review of information science and technology 38(1), 188–230 (2004)
17. Ferraresi, A., Zanchetta, E., Baroni, M., Bernardini, S.: Introducing and evaluating ukwac, a very large web-derived corpus of english. In: Proceedings of the 4th Web as Corpus Workshop (WAC-4) Can we beat Google. pp. 47–54 (2008)
18. Flati, T., Navigli, R.: Spred: Large-scale harvesting of semantic predicates. In: Proceedings of 51st Annual Meeting of the Association for Computational Linguistics, Sofia, Bulgaria (2013)
19. Fortuna, B., Mladenič, D., Grobelnik, M.: Semi-automatic construction of topic ontologies. Semantics, Web and Mining pp. 121–131 (2006)
20. Gärdenfors, P.: Conceptual spaces: The geometry of thought. MIT press (2004)
21. Gibson, J.: The concept of affordances. Perceiving, acting, and knowing pp. 67–82 (1977)
22. Grefenstette, E., Sadrzadeh, M.: Experimental support for a categorical compositional distributional model of meaning. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing. pp. 1394–1404. Association for Computational Linguistics (2011)
23. Harris, Z.: Distributional structure. Word 10(23), 146–162 (1954)
24. Hearst, M.: Automatic acquisition of hyponyms from large text corpora. In: Proceedings of the 14th conference on Computational linguistics-Volume 2. pp. 539–545. Association for Computational Linguistics (1992)
25. Jolliffe, I.: Principal component analysis. Wiley Online Library (2005)
26. Karlgren, J., Holst, A., Sahlgren, M.: Filaments of meaning in word space. In: Advances in Information Retrieval, pp. 531–538. Springer (2008)
27. Kruskal, J.B., Wish, M.: Multidimensional scaling, vol. 11. Sage (1978)
28. Lenci, A.: Carving verb classes from corpora. Word Classes. A cura di Raffaele Simone e Francesca Masini. Amsterdam-Philadelphia: John Benjamins p. 7 (2010)
29. Miller, G.A.: Wordnet: a lexical database for english. Communications of the ACM 38(11), 39–41 (1995)

30. Morik, K., Kaspari, A., Wurst, M., Skirzynski, M.: Multi-objective frequent termset clustering. Knowledge and information systems 30(3), 715–738 (2012)
31. Navigli, R., Velardi, P., Faralli, S.: A graph-based algorithm for inducing lexical taxonomies from scratch. In: Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume Three. pp. 1872–1877. AAAI Press (2011)
32. Navigli, R., Ponzetto, S.P.: Babelnet: Building a very large multilingual semantic network. In: Proceedings of the 48th annual meeting of the association for computational linguistics. pp. 216–225. Association for Computational Linguistics (2010)
33. Nuzzolese, A.G., Gangemi, A., Presutti, V., Draicchio, F., Musetti, A., Ciancarini, P.: Tìpalo: A tool for automatic typing of dbpedia entities. In: The Semantic Web: ESWC 2013 Satellite Events, pp. 253–257. Springer (2013)
34. Padó, S., Lapata, M.: Dependency-based construction of semantic space models. Computational Linguistics 33(2), 161–199 (2007)
35. Pasquier, N., Bastide, Y., Taouil, R., Lakhal, L.: Discovering frequent closed itemsets for association rules. In: Database TheoryICDT99, pp. 398–416. Springer (1999)
36. Pilehvar, M.T., Jurgens, D., Navigli, R.: Align, disambiguate and walk: A unified approach for measuring semantic similarity. In: Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL 2013) (2013)
37. Ponzetto, S., Strube, M.: Deriving a large scale taxonomy from wikipedia. In: Proceedings of the national conference on artificial intelligence. vol. 22, p. 1440. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999 (2007)
38. Rosch, E.: Principles of categorization. Concepts: core readings pp. 189–206 (1999)
39. Salton, G., Wong, A., Yang, C.S.: A vector space model for automatic indexing. Commun. ACM 18(11), 613–620 (Nov 1975), `http://doi.acm.org/10.1145/361219.361220`
40. Sigurbjörnsson, B., Van Zwol, R.: Flickr tag recommendation based on collective knowledge. In: Proceedings of the 17th international conference on World Wide Web. pp. 327–336. ACM (2008)
41. Slavík, P.: A tight analysis of the greedy algorithm for set cover. In: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing. pp. 435–441. ACM (1996)
42. Snow, R., Jurafsky, D., Ng, A.: Learning syntactic patterns for automatic hypernym discovery. Advances in Neural Information Processing Systems 17 (2004)
43. Speer, R., Havasi, C.: Representing general relational knowledge in conceptnet 5. In: LREC. pp. 3679–3686 (2012)
44. Speer, R., Havasi, C.: Conceptnet 5: A large semantic network for relational knowledge. In: The Peoples Web Meets NLP, pp. 161–176. Springer (2013)
45. Wu, F., Hoffmann, R., Weld, D.S.: Information extraction from wikipedia: Moving down the long tail. In: Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 731–739. ACM (2008)
46. Zaki, M.J., Hsiao, C.J.: Charm: An efficient algorithm for closed itemset mining. In: SDM. vol. 2, pp. 457–473. SIAM (2002)

Text Mining for Open Domain Semi-Supervised Semantic Role Labeling

Quynh Ngoc Thi Do[1], Steven Bethard[2], and Marie-Francine Moens[1]

[1] Katholieke Universiteit Leuven, Belgium
[2] University of Alabama at Birmingham, United States

Abstract. The identification and classification of some circumstance semantic roles like Location, Time, Manner and Direction, a task of Semantic Role Labeling (SRL), plays a very important role in building text understanding applications. However, the performance of the current SRL systems on those roles is often very poor, especially when the systems are applied on domains other than the ones they are trained on. We present a method to build open domain SRL system, in which the training data is expanded by replacing its predicates by words in the testing domain. A language model, which is considered as a text mining technique, and some linguistic resources are used to select from the vocabulary of the testing domain the best words for the replacement. We apply our method on the case study of transferring a semantic role labeler trained on the news domain to the children story domain. It gives us valuable improvements over the four circumstance semantic roles Location, Time, Manner and Direction.

1 Introduction

Playing an essential role in text understanding, *Semantic Role Labeling* is the task of natural language processing that specifies "Who did What to Whom, and How, When and Where?" in text [12].

For example, the processing of the sentence "Mary gave Peter a book at school yesterday" should result in the identification of a "giving" event with "Mary" as the *Agent* of the event, "Peter" as the *Recipient* and "a book" as the *Item being given*. The *Location* of the "giving" event, or where it took place, is "at school" and the *Time* of the event is "yesterday".

In this paper, we call an event ("giving" event) in a sentence the *semantic frame*, the verb or noun that evokes the frame ("gave") the *predicate*, the words ("Mary", "Peter", "a book", "at school", "yesterday") that play a role in the event the *arguments* and their roles ("Agent", "Recipient", "Thing being given", "Location", "Time") the *semantic roles*.

The task of semantic role labeling is to detect the event, to identify its arguments and assign the correct semantic roles to them. Thanks to the availability

In: P. Cellier, T. Charnois, A. Hotho, S. Matwin, M.-F. Moens, Y. Toussaint (Eds.): Proceedings of DMNLP, Workshop at ECML/PKDD, Nancy, France, 2014.

of semantic annotated resources (e.g. PropBank[3], FrameNet[4]), supervised machine learning approaches have been very successful in constructing automatic semantic role labellers. Assuming the predicates are already given, those systems can reach an F1[5] score of 85% when the training and testing data are in the same domain. But, when testing on other domains, the scores often drop significantly[6].

Text mining is the task of automatic discovery of new, previously unknown information from unstructured document collections. Meanwhile, a language model tries to capture the properties of a language, and predicts the next word in a word sequence. It is trained on a collection of unlabeled texts, and therefore is considered as a text mining technique. Recently, we some attempts to use such language models in a semi-supervised setting for semantic recognition [5], [7], in which, other words or a statistical class of words provided by the language model, that could be exchanged at a certain position in a sentence or phrase, enriches the feature vectors used in training, or they are used to create training examples artificially. However, there is no principled way to use such language information.

In this paper, we develop a methodology to generate additional training data for SRL by replacing selected verbal predicate words in training examples using a language model. For each selected predicate in the training examples, from the vocabulary of the domain that the SRL is applied on, a list of replacement words which we believe can occur at the same position as the selected word, are generated. We introduce and explore a variety of features for identifying how words should be replaced, including predicate vs. argument status, POS, WordNet related words, and a replacement score based on a language model. As for experiment, we present a case study of improving the performance of a SRL system trained on the news domain when applying to the children story domain. The case study is based on our ongoing European project "Machine Understanding for interactive StorytElling" (MUSE)[7]. One of the fundamental goals of MUSE is to detect actors, actions, plots in children stories, and render them as 3D worlds. SRL with its function of identifying the events in texts plays an essential role in solving our problem. Among the set of semantic roles, some circumstance semantic roles like Location, Time etc. are very important to understand the full meaning of an event, while the performance of the current SRL systems on them is often very poor, especially when testing on a domain other than the one they are trained on. Thus, in our case study, we target to improve SRL on the four PropBank circumstance roles: AM-LOC (Location), AM-TMP (Time), AM-MNR (Manner) and AM-DIR (Direction).

In the next sections, we present related work (Section 2), linguistic resources

[3] http://verbs.colorado.edu/ mpalmer/projects/ace.html

[4] https://framenet.icsi.berkeley.edu

[5] Harmonic mean of recall and precision.

[6] http://ufal.mff.cuni.cz/conll2009-st/

[7] http://www.muse-project.eu/

(Section 3), underlying assumptions, objectives and task definition (Section 4), methodology (Section 5), case study (Section 6), and conclusion (Section 7).

2 Related Work

Semi-supervised approaches to semantic role labeling recently have received the attention of the computational linguistics community. Information from language models have been used as extra features to improve the performance of SRL. [17] use deep learning techniques based on semi-supervised embeddings to improve a SRL system. [3] pursue this track further and use a deep neural network architecture to obtain good word representations in the form of word-embeddings. The word embedding defines the related words which are the result of the neural network training and are usually referred to as language models. [16] use word embeddings obtained by recurrent neural networks to recover the syntactic structure of a sentence, but the method is not applied to semantic role labeling. Along these lines, a number of language models with hidden layers have been developed based on generative probabilistic approaches and applied to semantic role labeling. [5] define a latent words language model as a graphical model where at each word position in a text the distribution of exchangeable words are generated. The authors use a hidden Markov language model with dependencies defined on two previous and two following words in the discourse, and in a subsequent paper, [4] explain approximate methods to train such a model among which is Gibbs sampling. In this model, each hidden variable or latent word generates a distribution over the entire vocabulary of the training data set. The model improves the performance of SRL on the CoNLL 2008 dataset especially when few training data are given to the learner. [7] propose a hidden Markov model that learns the distribution of a hidden variable that can take K different values, and the hidden variable is dependent on the previous hidden variable in the sentence. In contrast to [5], each hidden variable can generate a span or sequence of words instead of a single word. The span contains the sequence of words for the word under consideration and the predicate. Each latent variable represents a distribution of categories of words. The model is trained with a Baum-Welch algorithm. In both [5] and [7], the respectively most probable hidden word or category of words is used as an extra feature to describe the feature vector used in the recognition. In [7], several "hidden" features are used each being the result of a different initialization of the Baum-Welch algorithm. Although appealing, these latent words language models have disadvantages. The model of [5] yields a distribution over all vocabulary words raising the need to make a selection of possibly the most probable ones when using them in the feature representation. The model of [7] relies on a fixed number of categories (or latent topics) that form the hidden variables, but it is not clear how to choose such a number especially when word spans of different sizes are used as observed variables. In this paper, we aim at using a more flexible approach where such free parameters are replaced by the use of linguistic knowledge.

Besides the semi-supervised approaches that extend the feature set of SRL, there are other attempts to generate new training examples automatically by using unlabeled data. [6] automatically generate training examples by considering the lexical and syntactic similarity between labeled and unlabeled sentences as a graph alignment problem. use the language model of [5] to generate new training examples by replacing the head word of temporal expression training examples in the task of temporal expression recognition.

None of the above works consider both structural similarity and language models as a source of evidence for generating training examples, nor do they evaluate different approaches to similarity depending on the roles sought. In contrast to most of the above works, we evaluate the proposed methods when porting the learned model to texts from a domain that is different from the one the semantic role labeler was trained on.

3 Linguistic resources

The ***Penn Proposition Bank*** (PropBank) [13] provides a corpus annotated with semantic roles. In this resource, a semantic frame which is evoked by a verb is represented as a role set: Each role set is linked to a specific sense of the verb. Therefore, each verb has several role sets corresponding to its possible senses. The list of role sets and their semantic roles for each verb is defined in a frame file. For example, in the sentence "Mary gave Peter a book at school yesterday", the role set *give*.01 with the meaning of "transfer" evoked by the verb "give", has three main arguments $A0$, $A1$ and $A2$ that are "Agent", "Theme", and " Recipient", respectively: "[Mary *A0*] gave (*give.01*) [Peter *A2*] [a book *A1*] [at school *AM-LOC*] [yesterday *AM-TMP*]".

In ***VerbNet*** [15], English verbs are grouped into different classes, adapting the previous verbal classification of [8]. Each verbal class takes different thematic roles and certain syntactic constraints that describe their superficial behavior. VerbNet's hierarchical verb classes establish a set of possible thematic roles [8]. However, the semantic roles in VerbNet are more thematic than the ones in PropBank. For example, in VerbNet, *Agent* label is used instead of $A0$ label as in PropBank. *Patient* and *Theme* can be referred to the label $A1$ of PropBank.

Table 1. Main PropBank semantic roles

Role	**Description**
A0	Agent - extern argument
A1	Patient/Theme - intern argument
A2	Indirect object / beneficiary / instrument / attribute / end state
AM-LOC	Location (where?)
AM-TMP	Temporal marker (when?)
AM-MNR	Manner
AM-DIR	Direction

In Table 2, there is an example of the information that VerbNet contains for the class *give* − 13.1.1. Members of this class share the same syntactic patterns (NP V NP PP) with corresponding thematic roles (Agent V Recipient Theme Asset). Thus, two verbs "give" and "sell" in the two sentences "Mary gave Peter a book for 20 EUR" and "Mary sold Peter a book for 20 EUR" which have the same syntactic pattern, evoke two semantic frames with the same semantic role patterns as follows:
"[Mary *Agent*] gave (*give.01*) [Peter *Recipient*] [a book *Theme*] [for 20 EUR *Asset*]"
"[Mary *Agent*] sold (*sell.01*) [Peter *Recipient*] [a book *Theme*] [for 20 EUR *Asset*]"
SemLink[8] is a project whose aim is to link together different lexical resources via a set of mappings. These mappings will make it possible to combine the different information provided by these different lexical resources for tasks such as inferencing. The mapping between VerbNet and PropBank is available in SemLink. Each frame in PropBank is linked to a suitable VerbNet class and each role label in the PropBank frame is mapped to a VerbNet role label. Table 3 shows a mapping from the PropBank role set "give.01" to the VerbNet class "13.1.1".

Table 2. VerbNet class Give-13.1.1

Class Give-13.1.1
Roles: Agent, Theme, Recipient, Asset
Members: give, hawk, hock, lease, pawn, rent, sell
Frame: NP V NP PP.asset (*Agent* V *Recipient* *Theme* {at, for, on} *Asset*)...

WordNet [11] is a large lexical database of English. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations. Noun and verb synsets are arranged into hierarchies. The main relation among words in WordNet is synonymy, as between the words "shut" and "close" or "car" and "automobile". Each of WordNet's 117000 synsets is linked to other synsets by means of a small number of "conceptual" relations.

Table 3. Mapping from PropBank role set "give.01" to VerbNet class "13.1.1"

PropBank role set="give.01"	**VerbNet class="13.1.1"**
PropBank role label	**VerbNet role label**
A0	Agent
A1	Theme
A2	Recipient

[8] http://verbs.colorado.edu/semlink/

The most frequently encoded relation among synsets is the super-subordinate relation (also called hyperonymy, hyponymy or IS-A relation). It links more general synsets like *furniture*, *piece_of_furniture* to increasingly specific ones like *bed* and *bunkbed*.

4 Underlying assumptions, objectives and task definition

Semi-supervised learning is very difficult to accomplish in natural language processing tasks. In general, it is successful when the labeled training data contain seed examples that are representative for the whole data set and when the cluster hypothesis holds, that is, when a suitable similarity metric can correctly cluster the unlabeled examples with the labeled seed examples [2]. With regard to semantic role labeling, in order for the cluster hypothesis to hold, it requires that "similar" or exchangeable syntactic structures and lexical words found in the labeled and unlabeled examples cluster the linguistic phrases that form a specific semantic role.

We assume that a language model (e.g., [10], [4]) with valuable generic information on both frequent and infrequent legitimate linguistic expressions, gives us exchangeable words in context. The exchangeable words are considered as a cluster of words playing the same role on forming a specific semantic role.

In this respect, the goals of this paper are to:

- Set up a methodology for choosing unlabeled examples, guessing their labels, then using them as new training data to improve the performance of a semantic role labeler.
- Evaluate the methodology in our case study: when the SRL model is trained on news domain and applied on children story domain.

The notation of the symbols used in this paper is given in Table 4. The task of the semi-supervised semantic role labeler discussed in this paper is to learn

Table 4. Denotation of the symbols used in this paper.

Symbol	**Meaning**
$\mathbf{S}_l$	Set of manually annotated sentences
$\mathbf{S}_t$	Testing set
$\mathbf{S}_{ul}$	Set of unlabeled sentences used to train the language model
$\mathbf{S}_u$	Set of unlabeled sentences generated automatically
$\mathbf{S}_{nl}$	Set of automatically annotated semantic frames of $\mathbf{S}_u$
$\mathbf{S}_{temp}$	Set of tuples of (sentence, word to be replaced, list of replacement words)
$\mathbf{S}_{sl}$	Set of semantic frames selected for the replacement
$\mathbf{V}$	Vocabulary of $\mathbf{S}_t$
N	Maximum number of replacement words for replacement candidate
z	Context window used to calculate replacement score

from a set of manually annotated sentences, a set of unannotated sentences, a language model and some linguistic resources, a model that assigns semantic roles to the set of semantic frames of sentences in a test set. A sentence may contain more than one frame. Each semantic frame consists of one predicate that evokes the frame and several arguments that play a role in the frame. Predicates and arguments may be composed of more than one word. In this paper, we work with *verbal* predicates, and use *head word labeling*, which means if an argument consists of more than one word then the semantic role is assigned to only the head word. For instance, if "in the park" is the argument playing the role AM-LOC, then only the head word of the phrase "in the park", "in", is labeled with the label AM-LOC. Given a sentence s composed of n words $w_1, w_2, ..., w_n$, for each semantic frame f in s, each word w_i ($i \in \{1, 2, ...n\}$) has received a label $r_i \in \mathbf{R} \cup \{NULL\}$ during the manual annotation for training, or will receive a label $r_i \in \mathbf{R} \cup \{NULL\}$ during testing or evaluation, where $\mathbf{R}$ is a set of predefined semantic roles and *NULL* means empty label. If $r_i \neq NULL$, then w_i is the head of an argument of f with r_i as the semantic role. In the approach that we describe in this paper, $\mathbf{R}$ is the set of PropBank semantic roles (see Table 1).

Instead of training a SRL system on the given manually annotated sentences $\mathbf{S}_l$, we generate a set of new training examples by using a language model trained on a set of unannotated sentences, then use them together with $\mathbf{S}_l$ as the training data for the SRL.

5 Methodology

The steps of our methodology to generate new training examples and train a SRL system are shown in Figure 1.

Given a manually annotated sentence set $\mathbf{S}_l$ which can be used as training data, a set of unannotated sentences $\mathbf{S}_{ul}$, a vocabulary $\mathbf{V}$ including words in the domain of the testing data $\mathbf{S}_t$, a language model L, and some linguistic resources, we create new training examples and train the SRL system as follows: First, L is trained on $\mathbf{S}_{ul}$. Then, L, $\mathbf{S}_l$, $\mathbf{V}$ and the linguistic resources are used to generate new training examples $\mathbf{S}_{nl}$. The algorithm to generate new training examples is presented below. Finally, the SRL system is trained on $\mathbf{S}_l \cup \mathbf{S}_{nl}$.

5.1 General model for generating new training instances

In what follows, we present our general model to generate new training instances for the semantic role labeling task. The detail of our algorithm is given in Figure 1 (in dashed rectangle) and Algorithm 1. It consists of five main steps:

Step 1. Selecting data for replacement. We select a set of semantic frames from $\mathbf{S}_l$ used for the replacement. The selection can be performed in different ways depending on specific case studies. For each of the selected semantic frame, we choose its predicate as the word to be replaced. For example, given a sentence "Mary gave a book to Peter at school", the semantic frame "giving" has "gave" as predicate. "gave" is the word selected for the replacement.

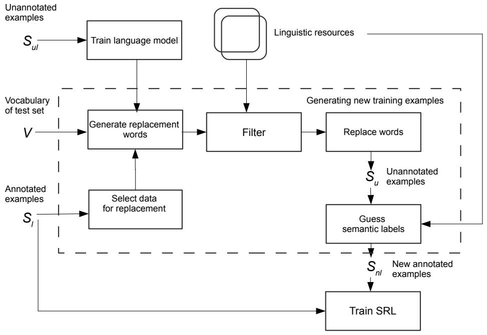


Fig. 1. Overview of the methodology to generate new training instances and train a SRL system.

Step 2. Generating replacement words for each word selected to be replaced. A statistical language model assigns a probability to a sequence of m words by means of a probability distribution. For each word selected to be replaced, we use the language model L trained on $\mathbf{S}_{ul}$, and the vocabulary $\mathbf{V}$ of the testing domain, to generate a list of replacement words. Given a sentence composed of $w_1, w_2, ..., w_n$, with w_i is the word to be replaced, for each $nw_j \in \mathbf{V}$, the score of replacing w_i by nw_j is calculated by the probability of the sequence of words $w_{i-z}, w_{i-z+1}, ..., w_{i-1}, nw_j, w_{i+1}, w_{i+2}, ..., w_{i+z}$ obtained by putting nw_j in the context of w_i where z is size of the context window taken into account:
$ReplacementScore(w_i, nw_j) = P(w_{i-z}, w_{i-z+1}, ..., w_{i-1}, nw_j, w_{i+1}, w_{i+2}, ..., w_{i+z})$
This probability score is calculated by the language model. It is used to rank the replacement words in our algorithm. Since the size of $\mathbf{V}$ may be very large and the words at the end of the list may have a very low score which often represents noise, only N words that have highest scores are chosen. After this step, we receive a ranked list of the top N replacement words for each candidate of the replacement.

Step 3. Applying filters to reduce noise in the list of replacement words. There may be a great deal of noise in the replacement words returned by the language model since it does not take into account enough information (syntactic, semantic etc.) to generate a replacement word that can be replaced perfectly for a word in a given sentence assuring the same semantic role. Thus, some linguistics filters are needed to improve the correctness and meaningfulness of the replacement.

Step 4. Replacing words in each sentence selected to be replaced by their replacement words that passed the filters, then we form a new unannotated set of sentences $\mathbf{S}_u$.

Step 5. Guessing semantic frames and their semantic labels for each sentence in $\mathbf{S}_u$ to have an annotated semantic frame set $\mathbf{S}_{nl}$.

In the following sections, we will present in more detail the language model used, some proposed filters, how to perform replacement and guess semantic role labels for the new sentences obtained by the replacement.

Algorithm 1 Generate novel training examples.

```
procedure GenerateNewExample($L$, $\mathbf{S}_l$, $\mathbf{V}$, $z$, $N$)
    $\mathbf{S}_u = \emptyset$, $\mathbf{S}_{nl} = \emptyset$, $\mathbf{S}_{temp} = \emptyset$;
    Select semantic frames that are used for the replacement from $\mathbf{S}_l$: $\mathbf{S}_{sl}$ = selected semantic frames
    for each sentence $s \in \mathbf{S}_l$ do
        for each word $w_i$ in $s$ do
            if $w_i$ is the predicate of a semantic frame $\in \mathbf{S}_{sl}$ then
                for each $nw_j \in \mathbf{V}$ do
                    $ReplacementScore(w_i, nw_j) = P(w_{i-z}, w_{i-z+1}, ..., w_{i-1}, nw_j, w_{i+1}, w_{i+2}, ..., w_{i+z})$ obtained by using $L$;
                end for
                Sort $nw_j$ according to $ReplacementScore$, then choose top $N$ words that have highest scores forming the ranked list $\mathbf{List}_i$;
                $\mathbf{S}_{temp} = \mathbf{S}_{temp} \cup (s, w_i, \mathbf{List}_i)$;
            end if
        end for
    end for
    for each $(s, w_i, \mathbf{List}_i)$ in $\mathbf{S}_{temp}$ do
        for each replacement word $nw_j$ in $\mathbf{List}_i$ do
            if $nw_j$ passes filters then
                for each semantic frame $f$ of $s$ that is in $\mathbf{S}_{sl}$ and receives $w_i$ as the predicate do
                    $s'$ = the sentence obtained by replacing $w_i$ by $nw_j$ in $s$;
                    $\mathbf{S}_u = \mathbf{S}_u \cup s'$;
                    $f'$= the semantic frame evoked by $nw_j$ in $s'$;
                    Guess semantic role labels of $f'$;
                    $\mathbf{S}_{nl} = \mathbf{S}_{nl} \cup f'$;
                end for
            end if
        end for
    end for
    Return $\mathbf{S}_{nl}$
end procedure
```

5.2 Language model

In this paper, we use the Recurrent Neural Network Language Model[9] (RNNLM) [10] [9] which is one of the most successful techniques for statistical language modeling. Unlike previous approaches in using artificial neural networks for modeling sequential data, recurrent neural networks are not trained with limited context size. By using recurrent connections, information (e.g., words from previous sentences in a discourse) can cycle inside these networks for a long time and have an influence on the final language model obtained. The architecture of RNNLM is shown in Figure 2. The input layer consists of a vector $\mathbf{w}(t)$ that represents the current word w_t encoded as 1 of V with V is the vocabulary (thus size of $\mathbf{w}(t)$ is equal to the size of the vocabulary), and of vector $\mathbf{s}(t-1)$ that represents output values in the hidden layer from the previous time step. After the network is trained, the output layer $\mathbf{y}(t)$ represents $P(w_{t+1}|w_t, \mathbf{s}(t-1))$. The network is trained by stochastic gradient descent using either usual backpropagation algorithm, or backpropagation through time [14]. The network is represented by input, hidden and output layers and corresponding weight matrices - matrices **U** and **W** between the input and the hidden layer, and matrix **V** between the hidden and the output layer. Output values in the layers are computed as follows:

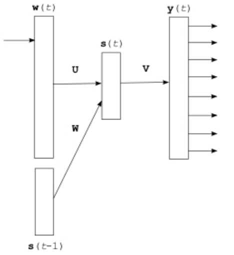


Fig. 2. Simple recurrent neural network.

$$s_j(t) = f(\sum_i w_i(t)u_{ji} + \sum_l s_l(t-1)w_{jl}) \tag{1}$$

$$y_k(t) = g(\sum_j s_j(t)v_{kj}) \tag{2}$$

where *f (z)* and *g(z)* are sigmoid and softmax activation functions:

$$f(z) = \frac{1}{1+e^{-z}}, g(z_m) = \frac{e^{z_m}}{\sum_k e^{z_k}} \tag{3}$$

The output layer y represents a probability distribution of the next word w_{t+1} given the history. The size of the hidden units is in our experiments set to 300. The standard backpropagation algorithm with stochastic gradient descent is used to train the model. In this research, we use the language model to calculate the probability of a word

[9] http://www.fit.vutbr.cz/ imikolov/rnnlm/

sequence $W = w_1w_2...w_m = W_1^m$. The language model probability of W is computed as follows:

$$P(W) = P(W_1^m) = \prod_{i=1}^{m} P(w_i|W_1^{i-1}) \tag{4}$$

Over the last few decades, an n-gram language model which assumes that the predicted word only depends on the previous n-1 words, is the most popular technique since it is simple and effective. Instead of using Equation 4, $P(W_1^m)$ is calculated in a more simple way, as $P(W_1^m) = \prod_{i=1}^{m} P(w_i|W_{i-n+1}^{i-1})$. However, an n-gram language model estimates its parameters in the discrete space, resulting in weak generalization capability on unknown data. In addition, the standard n-gram language model suffers from exponential growth of size, serious data fragmentation, and increased miss rate using longer context [18]. To overcome this problem, RNNLM, which has activation feedback with short-term memory and uses full history information instead of limiting context, can help us to calculate more accurately and efficiently $P(W_1^m)$ by Equation 4 using the output layer $y(t)$.

A language model tries to capture the properties of a language and is trained on a collection of unlabeled texts, so it can be considered as a text mining technique.

5.3 Filters

Because the list of top N replaceable words returned by the language model may contain a great deal of noise, we propose specific filters to improve the performance of the system.

Part-Of-Speech filter (POS filter): We keep replacement word nw_j for w_i if nw_j has the same POS tag as w_i, when replacing w_i in sentence s.

WordNet filter: We keep replacement word nw_j for w_i if nw_j and w_i are synonyms or have the same hypernym in WordNet. Here, we ignore the problem of word disambiguation. We only use the first word sense when finding the synonyms and the words that have the same hypernym.

For example, “January” has “Jan” as synonym, “February”, “March”, etc. are the words that have the same hypernym “Gregorian_calendar_month”.

Predicate filter: A suitable replacement word of a predicate should also evoke a frame with correct roles when it is placed in the sentence of the target. Our idea is to assign role labels to the new frame based on the role labels of the current frame, but it raises the problem of how to find a mapping between the role sets of the two frames, and detect the correct sense of the new frame. Based on this idea, one possibility is to select only replacement words for which the mappings between role sets are available. By using SemLink (see Section 3), we define a filter specifically for predicates: for each predicate w_p evoking a frame f, we keep replacement predicate word nw_j for w_p if f and one frame evoked by nw_j are mapped to the same VerbNet class and the mappings from those two frames to the VerbNet class are defined in SemLink.

5.4 Replacing words and guessing semantic labels

Replacement words that have passed filters are used to generate new training examples. Given a semantic frame f of a sentence s composed of n words $w_1, w_2, ..., w_n$, w_p ($p \in \{1, 2, ..., n\}$) is the predicate of f, and $w_{a1}, w_{a2}, ..., w_{am}$ ($a1, a2, ..., am \in \{1, 2, ..., n\}$)

are the heads of the m arguments of f with $r_1, r_2, ..., r_m$ as semantic role labels, respectively. After the filtering step, the list of replacement words of w_p, $List_p$, includes j words $\{nw_1, nw_2, ..., nw_j\}$. For each $nw_t \in List_p$, we replace w_p by nw_t in sentence s and obtain sentence s' composed of n words $w_1, w_2, ...w_{p-1}, nw_t, w_{p+1}, ..., w_n$. If nw_t has passed the Predicate filter - which we use as a default filter -, it can be a semantic predicate and the argument structure of the frame evoked by nw_t is similar to the argument structure of the frame evoked by w_p. Thus, we guess that nw_t also invokes a semantic frame in s' with $w_{a1}, w_{a2}, ..., w_{am}$ as arguments (the new semantic frame and f - the frame evoked by w_p - have the same argument words). In order to predict the sense and role labels of the new semantic frame, we use the mappings between PropBank semantic frames and VerbNet classes that can be found in SemLink. We first find a frame f' of nw_t so that both f' and f are mapped to a same VerbNet class c. The mappings from f and f' to c are denoted by m_1 and m_2, respectively. If the Predicate filter has been applied before, f' exists and can be found in this step. Then, we can guess that f' is the new frame evoked by nw_t in s'. As for semantic role labels of f', if in f, w_{ai} ($i \in \{1, 2, ..., m\}$) with semantic role label r_i, is a circumstance role AM-s, then its role does not change in f'. That means r_i is also the role label of w_{ai} in f'. Otherwise, if the role label of w_{ai} in f is r_i, then the role label of w_{ai} in f' is $m_2^{-1}(m_1(r_i))$. For example, the sentence “Rachel wore a hat in her room” has the frame “wear.01” (wore) with “Rachel” as A0, “hat” as A1, “in” (the head of the prepostion phrase “in her room”) as AM-LOC, and the predicate “wore” has “donned” as a replacement word. By replacing “wore” by “donned” in the sentence, we have a new sentence “Rachel donned a hat in her room” and “donned” evokes a new frame. In SemLink, we can find the VerbNet class “simple_dressing-41.3.1” linked to the Propbank frame “wear.01” and one PropBank frame of the predicate “don”, “don.01”. The role mapping between the VerbNet class and the two frames can be found in Table5. By applying our method, we have a new frame “don.01” with “Rachel” as A0 (mapped to the “Agent” VerbNet role), “hat” as A1 (mapped to the “Theme” VerbNet role), and “in” as AM-LOC (circumstance role) (See Figure 3).

Table 5. Role mapping of “simple_dressing-41.3.1” linked to both “wear.01” and “don.01”

Role of *simple_dressing-41.3.1*	Role of *wear.01*	Role of *don.01*
Agent	Arg0	Arg0
Theme	Arg1	Arg1

6 Case study

In the EU-EP7 MUSE project[10], in which KU Leuven is involved, we instantiate a virtual world with information extracted from children stories. Our fundamental goal is to introduce a new way of exploring and understanding information by “bringing text to life” through 3D interactive storytelling. Children stories will be taken initially as input, due to the relative simplicity of such stories and the relative ease of results evaluation,

[10] http://www.muse-project.eu/

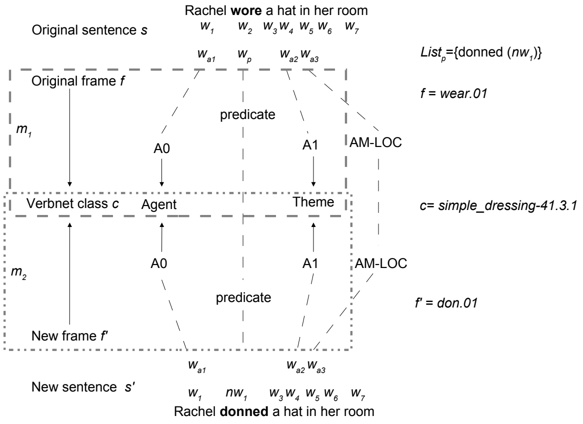

Rachel **wore** a hat in her room
Original sentence s
w_1 w_2 w_3 w_4 w_5 w_6 w_7
w_{a1} w_p w_{a2} w_{a3}
$List_p$={donned (nw_1)}
Original frame f
f = $wear.01$
m_1
predicate
A0
A1
AM-LOC
Verbnet class c
Agent
Theme
c= $simple\_dressing$-41.3.1
m_2
A0
A1
AM-LOC
predicate
f' = $don.01$
New frame f'
w_{a1} w_{a2} w_{a3}
New sentence s'
w_1 nw_1 w_3 w_4 w_5 w_6 w_7
Rachel **donned** a hat in her room

Fig. 3. An example of the replacement.

and will then be translated into formal knowledge that represents the actions, actors, plots and surrounding world. In a next step this formal knowledge is rendered as virtual 3D worlds in which the user can explore the text through interaction, re- enactment and guided game play. Finally, we will evaluate to which degree this procedure enhances the understanding by children of simple fantasy stories. In our project, NLP techniques are necessary to bring natural language closer to 3D immersive environments. Among the needed NLP techniques, SRL is one of the most important ones, since it labels the textual content of the story by semantic roles identifying actors and activities, that are then visualized in a 3D world. In this paper, we present our work on building a SRL system for the children story domain.

To create virtual 3D worlds, the location where the activities happen, the order of the activities, the tools used by the actors in the activities and the direction where the actors move toward, are very important information. Such kinds of information can be represented by the semantic roles AM-LOC, AM-TMP, AM-MNR and AM-DIR respectively in SRL. However, the performance of the current SRL systems on those roles is often very poor, especially when testing on a domain other than the one they are trained on. Therefore, in our work, we target to improve SRL system on those four roles: AM-LOC, AM-TMP, AM-MNR and AM-DIR.

6.1 Data

We select "Tuk the Hunter" story as the data for our project demonstration. The story (with some slight changes in the content) is the testing data in our case study. To evaluate the performance of SRL on our domain, we annotate semantic roles for the story following the PropBank annotation guideline and dependency head word labeling. The total number of annotated semantic frames in our testing data is 154. The detailed number of instances per role is given in Table 6.

Most of the annotated data available for semantic role labeling are in news domain. To

Table 6. Number of instances per role in the training and testing data

Data	AM-LOC	AM-TMP	AM-MNR	AM-DIR
Testing	19	28	32	10
Training	10387	23347	11837	1146

be used as our training data, we select CoNLL 2009[11] training dataset which contains parts of the Wall Street Journal corpus[12]. The detail number of instances per role in the training data is given in Table 6.

6.2 Expanding training data to children story domain

In our case study, we collect 252 children stories (mostly are fairy tales) to create the domain of children story. They are used together with the first 80 million words of the Reuters corpus to train the Recurrent Neutral Network Language Model[13], and the vocabulary of those stories is used to generate new training examples.

We realize that most of the instances of the four targeted roles are prepositions. It suggests us to choose semantic frames that contain at least one preposition argument in the CoNLL 2009 training data as the base for our training data expansion.

The SRL system is used in our experiment is the Lund university's semantic parsing [1], which is available freely, and one of the best systems in the CoNLL 2009 competition. In order to evaluate the effectiveness of our method, we compare the results obtained on the Tuk story by the Lund university's semantic parsing when training on our expanded training data and on the original training data. In our experiment setting, we use all of the three filters: POS filter, WordNet filter, Predicate filter. The maximum number of replacement words for each replacement position, $N = 500$, and the context window size, $z = 5$. Table 7 presents the results and the gains obtained on the four circumstance roles when using our expanded training data in precision, recall and F1 measures. From the table, it is clear that we obtain valuable recall, precision and F1 improvements (at least 7% for recall and F1) over all of the tested roles.

Table 7. Recall, precision, F1 results and gains (in %) per role when training on our expanded training data.

Role	Recall (Recall gain)	Precision (Precision gain)	F1 (F1 gain)
AM-LOC	47.37(+10.53)	60.00(+6.15)	52.94(+9.19)
AM-TMP	82.14(+7.14)	69.70(+7.93)	75.41(+7.67)
AM-MNR	56.25(+18.75)	85.71(+10.71)	67.92(+17.92)
AM-DIR	60.00(+10.00)	75.00(+3.57)	66.67(+7.84)

[11] http://ufal.mff.cuni.cz/conll2009-st/

[12] http://catalog.ldc.upenn.edu/LDC2012T04

[13] http://www.fit.vutbr.cz/ imikolov/rnnlm/

7 Conclusion

In this paper, we present a methodology of building an open-domain semantic role labeling. In our case study, we transfer the SRL model trained on the news domain to the children story domain, by collecting children stories to create the new domain, then replacing verbal predicates in the training data by the words of the new domain given a language model. We keep the precision score from dropping by using the occurrence probabilities and some linguistic filters to verify linguistic patterns obtained by the replacements. The valuable enhanced results over the four circumstance roles AM-LOC, AM-TMP, AM-MNR and AM-DIR show clearly the effectiveness of our methodology on this case study[14].

Acknowledgement

This work is funded by the EU ICT FP7 FET project "Machine Understanding for interactive StorytElling" (MUSE) `http://www.muse-project.eu/`.

References

1. Björkelund, A., Hafdell, L., Nugues, P.: Multilingual semantic role labeling. In: Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task. pp. 43–48. CoNLL '09, Association for Computational Linguistics, Stroudsburg, PA, USA (2009)
2. Chapelle, O., Scholkopf, B., Zien, A. (eds.): Semi-Supervised Learning. MIT Press (2006)
3. Collobert, R.: Deep learning for efficient discriminative parsing. In: AISTATS (2011)
4. Deschacht, K., De Belder, J., Moens, M.F.: The latent words language model. Computer Speech and Language 26(5), 384–409 (Oct 2012), `https://lirias.kuleuven.be/handle/123456789/344914`
5. Deschacht, K., Moens, M.F.: Semi-supervised semantic role labeling using the latent words language model. In: EMNLP. pp. 21–29. ACL (2009)
6. Fürstenau, H., Lapata, M.: Semi-supervised semantic role labeling via structural alignment. Comput. Linguist. 38(1), 135–171 (Mar 2012)
7. Huang, F., Ahuja, A., Downey, D., Yang, Y., Guo, Y., Yates, A.: Learning Representations for Weakly Supervised Natural Language Processing Tasks. Computational Linguistics 40, 85–120 (2013)
8. Levin, B.: English Verb Classes and Alternations A Preliminary Investigation. University of Chicago Press, Chicago and London (1993)
9. Mikolov, T.: Statistical Language Models Based on Neural Networks. Ph.D. thesis, Ph. D. thesis, Brno University of Technology (2012)
10. Mikolov, T., Karafit, M., Burget, L., Cernock, J., Khudanpur, S.: In: Kobayashi, T., Hirose, K., Nakamura, S. (eds.) INTERSPEECH. pp. 1045–1048. ISCA (2010)
11. Miller, G.A.: Wordnet: A lexical database for english. Commun. ACM 38(11), 39–41 (Nov 1995)

[14] An expanded version of this paper with comparable experimental results on the out-of-domain CoNLL 2009 testing data is submitted to the journal of IEEE/ACM Transactions on Audio, Speech and Language Processing.

12. Palmer, M., Gildea, D., Xue, N.: Semantic Role Labeling. Synthesis Lectures on Human Language Technologies, Morgan & Claypool Publishers (2010)
13. Palmer, M., Kingsbury, P., Gildea, D.: The proposition bank: An annotated corpus of semantic roles. Computational Linguistics 31(1), 71–106 (2005)
14. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. chap. Learning Internal Representations by Error Propagation, pp. 318–362. MIT Press, Cambridge, MA, USA (1986), http://dl.acm.org/citation.cfm?id=104279.104293
15. Schuler, K.K.: VerbNet: A Broad-Coverage, Comprehensive Verb Lexicon. Ph.D. thesis, University of Pennsylvania (2006)
16. Socher, R., Lin, C.C.Y., Ng, A.Y., Manning, C.D.: Parsing natural scenes and natural language with recursive neural networks. In: Getoor, L., Scheffer, T. (eds.) ICML. pp. 129–136. Omnipress (2011)
17. Weston, J., Ratle, F., Collobert, R.: Deep learning via semi-supervised embedding. In: Proceedings of the 25th International Conference on Machine Learning. pp. 1168–1175. ICML '08, ACM, New York, NY, USA (2008), http://doi.acm.org/10.1145/1390156.1390303
18. Yujing Si, Zhen Zhang, T.L.J.P., Yan, Y.: Enhanced word classing for recurrent neural network language model. In: JICS: Journal of Information and Computational Science, Vol. 10. pp. 3595–3604 (2013)

Learning Semantically Coherent Rules

Alexander Gabriel[1], Heiko Paulheim[2], and Frederik Janssen[3]

[1] agabriel@mayanna.org
Technische Universität Darmstadt, Germany
[2] heiko@informatik.uni-mannheim.de
Research Group Data and Web Science
University of Mannheim, Germany
[3] janssen@ke.tu-darmstadt.de
Knowledge Engineering Group
Technische Universität Darmstadt, Germany

Abstract. The capability of building a model that can be understood and interpreted by humans is one of the main selling points of *symbolic* machine learning algorithms, such as rule or decision tree learners. However, those algorithms are most often optimized w.r.t. classification accuracy, but not the understandability of the resulting model. In this paper, we focus on a particular aspect of understandability, i.e., *semantic coherence*. We introduce a variant of a separate-and-conquer rule learning algorithm using a WordNet-based heuristic to learn rules that are semantically coherent. In an evaluation on different datasets, we show that the approach learns rules that are significantly more semantically coherent, without losing accuracy.

Keywords: Rule Learning, Semantic Coherence, Interpretability, Rule Learning Heuristics

1 Introduction

Symbolic machine learning approaches, such as rule or decision tree induction, have the advantage of creating a model that can be understood and interpreted by human domain experts – unlike statistical models such as Support Vector Machines. In particular, rule learning is one of the oldest and most intensively researched fields of machine learning [14].

Despite this advantage, the actual understandability of a learned model has received only little attention so far. Most learning algorithms are optimized w.r.t. the classification accuracy, but not understandability. Most often the latter is measured rather naively by, e.g., the average number of rules and/or conditions without paying any attention to the relation among them.

The understandability of a rule model comprises different dimensions. One of those dimensions is *semantic coherence*, i.e., the semantic proximity of the different conditions in a rule (or across the entire ruleset). Prior experiments have shown that this coherence has a major impact on the reception of a rule

In: P. Cellier, T. Charnois, A. Hotho, S. Matwin, M.-F. Moens, Y. Toussaint (Eds.): Proceedings of DMNLP, Workshop at ECML/PKDD, Nancy, France, 2014.

model. This notion is similar to the notion of semantic coherence of texts, which is a key factor to understanding those texts [20].

In a previous user study, we showed different rules describing the quality of living in cities to users. The experiments showed that semantically coherent rules – such as *Cities with medium temperatures and low precipitation* – are favored over incoherent rules, such as *Cities with medium temperatures where many music albums have been recorded* [27].

In this paper, we discuss how separate-and-conquer rule learning algorithms [12] can be extended to support the learning of more coherent rules. We introduce a new heuristic function that combines a standard heuristic (such as *Accuracy* or *m-Estimate*) with a semantic one, and allows for adjusting the weight of each component. With that weight, we are able to control the trade-off between classification accuracy and semantic coherence.

The rest of this paper is structured as follows. We begin by briefly introducing separate-and-conquer rule learning. Next, our approach to learning semantically coherent rules is detailed. In the following evaluation, we introduce the datasets and show the results. Here, also some exemplary rules are given, indeed indicating semantic coherence between the conditions of the rules. After that, related work is captured. Then, the paper is concluded and future work is shown.

2 Separate-and-Conquer Rule Learning

Separate-and-conquer rule learning is still amongst the most popular strategies to induce a set of rules that can be used to classify unseen examples, i.e., correctly map them on their respective classes. How exactly this strategy is implemented varies among the different algorithms but most of them fit into the framework of separate-and-conquer. This led to the development of the so-called SeCo suite [18], a versatile framework that allows for most existing algorithms to be configured properly. Based on the flexibility and the convenient way to implement new functions or extensions, we chose this framework for our experiments.

In essence, a separate-and-conquer rule learner proceeds in two major steps: First, a single rule, that fulfills certain quality criteria, is learned from the data (this is called the conquer step of the algorithm). Then, all (positive) examples that are covered by this rule are removed from the dataset (the separate step) and the algorithm proceeds by learning the next rule until all examples are covered.

Certainly, this strategy is only usable for binary data as a notion of positive and negative example is mandatory but then, if desired, it can guarantee that every positive example is covered (*completeness*) and no negative one is covered (*consistency*). There are different strategies to convert multi-class datasets to binary ones. However, in this paper we used an ordered binarization as implemented in the SeCo framework. Therefore, the classes of the dataset are ordered by their class-frequency and the smallest class is defined to be the positive one whereas the other ones are treated as negative examples. After the necessary number of rules to cover the smallest class is learned, all examples from it are removed and the next smallest one is defined to be positive while again the rest of the examples are negative. The algorithm proceeds in this manner until all classes expect the largest one are covered. The resulting ruleset is a so-called decision list where for each example that is to be classified the rules are tested from top to bottom and the first one that covers the example is used for prediction. If, however, no rule covers the example, a default rule at the end of the list assigns it to the largest class in the dataset.

A single rule is learned in a top-down fashion meaning that it is initialized as an empty rule and conditions are greedily added one by one until no more negative examples are covered. Then, the best rule encountered during this process is heuristically determined and returned as best rule. Note that this has not to be the last rule covering no negative example, i.e., *consistency* due to reasons of overfitting is not assured. A heuristic, in one way or another, maximizes the covered positive examples while trying to cover as few negative ones as possible. The literature shows a wide variety of different heuristics [13]. For the experiments conducted in this paper we had to make a selection and chose three well known heuristics namely *Accuracy*, *Laplace Estimate*, and the *m-Estimate*, as defined later. We are aware of the restrictions that come with our selection but we are confident that our findings regarding the semantic coherence are not subject to a certain type of heuristic but rather are universally valid.

To keep it simple, we used the default algorithm implemented in the SeCo framework. Namely, the configuration uses a top-down hill-climbing search (a beam size of one) that refines a rule as long as negative examples are covered. The learning of rules stops when the best rule covers more negative than positive examples and the conditions of a rule test for equality (nominal conditions) or use $<$ and $\geq$ for numerical conditions. No special pruning or post-processing of rules is employed. For the *m-Estimate*, the parameter was set to 22.466 as suggested in [17].

3 Enabling Semantic Coherence

The key idea of this paper is to enrich the heuristic used for finding the best condition with a semantic component that additionally to the goal of maximizing positive examples while minimizing negatives, will incorporate that the selected condition will also be as semantically coherent as possible. In essence, we have two components now:

- The classic heuristic (selects conditions based on statistical properties of the data) and
- the semantic heuristic (selects conditions based on their semantic coherence with previous conditions).

Hence, the new heuristic *WH* offers the possibility to trade-off between statistical validity (the classic heuristic *CH*) and the semantic part (a semantic heuristic *SH*). This is enabled by a parameter α that weights the two objectives:

$$WH(Rule) = \alpha \cdot SH(Rule) + (1 - \alpha) \cdot CH(Rule), \quad \alpha \in [0, 1] \qquad (1)$$

A higher value α gives more weight to semantic coherence, while a value of $\alpha = 0$ is equivalent to classic rule learning using only the standard heuristic. We expect that higher values of α lead to a decrease in predictive accuracy because the rule learning algorithm focuses less on the quality of the rule and more on choosing conditions that are semantically coherent (which are likely not to have a strong correlation with the rule's accuracy). At the same time, higher values of α should lead to more coherent rules.

When learning rules, the first condition is selected by using the classic heuristic *CH* only (since a rule with only one condition is always coherent in itself). Then, while growing the rule, the *WH* heuristic is used, which leads to conditions being added that result in both a coherent and an accurate rule according to the trade-off specified by α.

3.1 WordNet Similarity

There are different possibilities to measure the semantic relatedness between two conditions. In this paper, we use an external source of linguistic information, i.e., *WordNet* [8]. WordNet organizes words in so-called *synsets*, i.e., sets of synonym words. Those synsets are linked by homonym and hyperonym relations, among others. Using those relations, the semantic distance between words in different synsets can be computed.

In the first step, we map each feature that can be used in a rule to one or more synsets in WordNet[4]. To do so, we search WordNet for the feature name. In the following, we will consider the case of measuring the semantic coherence of two features named *smartphone vendor* and *desktop*.

The search for synsets returns a list of synsets, ordered by relevance. The search result for *smartphone vendor* is empty $\{\}$, the search result for *desktop* is $\{desktop\#n\#1, desktop\#n\#2\}$ where *desktop#n#1* describes a tabletop and *desktop#n#2* describes a desktop computer.[5]

If the list is not empty, we add it to the attribute label's list of synset lists. If otherwise the list is empty, we check whether the attribute label is a compound of multiple tokens and restart the search for each of the individual tokens. We then add all non-empty synset lists that are returned to the list of synset lists of the attribute label. The result for *smartphone vendor* is then $\{\{smartphone\#n\#1\}, \{vendor\#n\#1\}\}$ while the result for desktop is $\{\{desktop\#n\#1, desktop\#n\#2\}\}$.

In the second step, we calculate the distance between two synsets using the LIN [21] metric. We chose this metric as it performs well in comparison with other metrics [3], and it outputs a score normalized to $[0, 1]$.

[4] Note that at the moment, we only use the names of the features to measure semantic coherence, but not the nominal or numeric feature values that are used to build a condition.

[5] The 'n' indicates that the synsets are describing nouns.

The LIN metric is based on the Information Content (IC) metric [29], a measure for the particularity of a concept. The IC of a concept c is calculated as the negative of the log likelihood, simpler put: the negative of the logarithm of the probability to encounter concept c:

$$IC(c) = -\log p(c) \tag{2}$$

Higher values denote less abstract, more general concepts, while lower values denote more abstract, less general concepts. The body of text used for the calculation of the $p(c)$ values in this work is the SemCor [23] corpus, a collection of 100 passages from the Brown corpus which were semantically tagged "based on the WordNet word sense definition" and thus provide the exact frequency distribution of each synset, which covers roughly 25% of the synsets in WordNet [19].

The LIN metric is calculated by dividing the Information Content (IC) of the least common synset of the two synsets by the sum of their Information Content, and multiplying the result with two:[6]

$$lin(syn_1, syn_2) = 2 \cdot \frac{IC(lcs)}{IC(syn_1) + IC(syn_2)} \tag{3}$$

Information Content. For each pair of synsets associated with two attributes, we calculate the LIN metric. In our example, the corresponding values are

$$\begin{aligned} lin(smartphone\#n\#1, desktop\#n\#1) &= 0.0, \\ lin(smartphone\#n\#1, desktop\#n\#2) &= 0.5, \\ lin(vendor\#n\#1, desktop\#n\#1) &= 0.0, \text{ and} \\ lin(vendor\#n\#1, desktop\#n\#2) &= 0.0. \end{aligned}$$

In the third step, we choose the maximum value for each pair of synset lists (syn) so that we end up with the maximum similarity value per pair of tokens. The overall semantic similarity of two attributes (att) is then computed as the mean of those similarities across the tokens t:

$$SH(att_1, att_2) = \underset{\substack{\forall t_1 \in att_1 \\ \forall t_2 \in att_2}}{avg} \; \underset{\substack{\forall syn_1 \in t_1 \\ \forall syn_2 \in t_2}}{max} \; lin(syn_1, syn_2) \tag{4}$$

This assigns each word pair the similarity value of the synset combination that is most similar among all the synset combinations that arise from the two lists of possible synsets for the two words. Thus, in our example, the SH value assigned to *smartphone vendor* and *desktop* would be 0.25.

To compute the semantic coherence of a rule given the pairwise SH scores for the attributes used in the rule, we use the mean of those pairwise scores to assign a final score to the rule.[7]

[6] This metric limits the similarity calculation to synsets of the same POS and works only with nouns and verbs. Our implementation returns a similarity value of 0 in all other cases.

[7] All experiments were carried out with minimum and maximum as well, but using the mean turned out to give the best results.

Table 1. Datasets used in the experiments

Dataset	#Attributes	Found in WordNet
hepatitis	19	68%
primary-tumor	17	71%
bridges2	11	73%
zoo	17	94%
flag	27	100%
auto-mpg	7	100%
balloons	4	100%
glass	9	100%

4 Evaluation

We have conducted experiments with different classic heuristics on a number of datasets from the UCI machine learning repository[8] shown in Table 1. The table depicts the overall number of attributes and the percentage of attributes for which at least one matching synset was found in WordNet. For classic heuristics *CH*, we chose *Accuracy*, *m-Estimate*, and *Laplace Estimate*, which are defined as follows:

$$\mathit{Accuracy} := p - n \equiv \frac{p + (N - n)}{P + N} \tag{5}$$

$$\mathit{Laplace\ Estimate} := \frac{p + 1}{p + n + 2} \tag{6}$$

$$m\text{-}\mathit{Estimate} := \frac{p + m \cdot \frac{P}{P+N}}{p + n + m} \tag{7}$$

where p, n denote the positive/negative examples covered by the rule and P, N stand for the total positive/negative examples. Please see [17] for more details on these heuristics.

In addition, we used the semantic heuristic *SH* based on WordNet as defined above. For each experiment, we report the accuracy (single run of a ten fold cross validation) and the average semantic coherence of all the rules in the ruleset (measured by *SH*), as well as the average rule length and the overall number of conditions and rules in the ruleset.

As datasets, we had to pick some that have attribute labels that carry semantics, i.e., the attributes have to have speaking names instead of, e.g., names from `att1` to `att20` (which unfortunately is the case for the majority of datasets in the UCI repository). We searched for datasets where we could map at least two thirds of the attributes to at least one synset WordNet. This led to the eight datasets used for the experiments in this paper which are listed in Table 1.

Table 2. Macro average accuracy of the learned rulesets on the eight datasets. Statistically significant deviations ($p > 0.05$) from $\alpha = 0$ are marked in bold.

Classic	α									
Heuristic	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Accuracy	0.649	0.667	0.668	0.668	0.669	0.669	0.668	0.668	0.668	0.668	**0.465**
m-Estimate	0.670	0.673	0.672	0.671	0.671	0.670	0.670	0.673	0.673	0.674	**0.474**
Laplace	0.673	0.680	0.679	0.682	0.681	0.680	0.681	0.679	0.679	0.681	**0.476**

Table 3. Average semantic coherence of the learned rulesets on the eight datasets. Statistically significant deviations ($p > 0.05$) from $\alpha = 0$ are marked in bold.

Classic	α										
Heuristic	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Accuracy	0.146	**0.212**	0.222	**0.241**	**0.235**	**0.235**	**0.237**	**0.239**	**0.233**	**0.233**	–
m-Estimate	0.165	0.195	0.199	0.204	0.207	0.209	0.211	0.217	0.222	0.232	–
Laplace	0.156	**0.206**	**0.223**	**0.228**	**0.231**	**0.228**	**0.227**	**0.227**	**0.227**	**0.226**	–

4.1 Results of the Experiments

Table 2 shows the macro average accuracy across the eight datasets for different values of α. It can be observed that, except for $\alpha = 1$, the accuracy does not change significantly. This is an encouraging result, as it shows that a weight of up to 0.9 can be assigned to the semantic heuristic without the learning model losing much accuracy. How much exactly the coherence can be enforced has to be examined by a more detailed inspection of the parameter values in between 0.9 and 1.0. Interestingly, the trade-off between coherence and accuracy seems to occur rather at the edge at high parameter values. Clearly, a study of these parameters would yield more insights, but, however, ensuring such high coherence without a noticeable effect on accuracy already is a remarkable effect and seems to be sufficient for our purposes. Only when assigning all weight to the semantic heuristic (and none to the classic heuristic), the accuracy drops significantly, which is the expected result. In most of these cases, no rules are learned at all, but only a default rule is created, assigning all examples to the majority class.

In Table 3, we report the macro average semantic coherence of the learned rulesets across the eight datasets. The results have to be seen in context with Table 2 as our primary goal was to increase semantic coherence while not losing too much accuracy. Clearly, the higher the values of α will be, the more semantic coherence will be achieved anyway. This is because the heuristic component uses the same measure for semantic coherence as is reported in the evaluation in Table 3. However, as confirmation, it can be observed that the semantic coherence is indeed increased in all cases, whereas, when using *m-Estimate* as a classic heuristic, the increase is not statistically significant. As stated above, no

[8] http://archive.ics.uci.edu/ml/

Table 4. Two rules learned for primary-tumor

$\alpha = 0.0$	peritoneum = yes, skin = yes, histologic-type = adeno	→ class = ovary
$\alpha = 0.8$	peritoneum = yes, **skin = yes**, **pleura = no**, **brain = no**	→ class = ovary

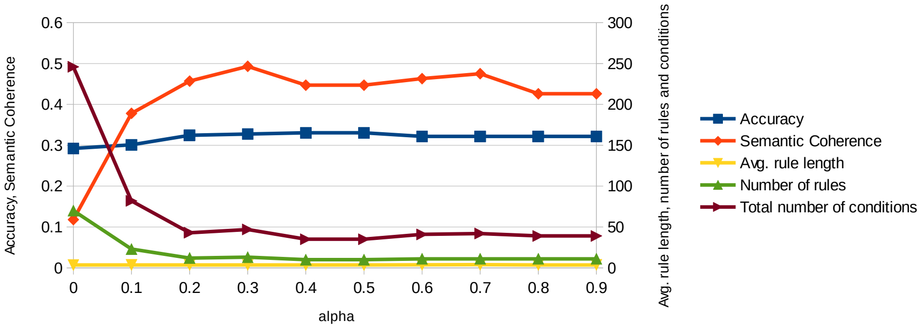


Fig. 1. Detailed results on the primary-tumor dataset, using *Accuracy* as a classic heuristic

rules are learned in many cases for $\alpha = 1$, so the semantic coherence cannot be computed there.

These results support our main claim, i.e., that it is possible to learn more coherent rules without losing classification accuracy. What is surprising is that even for $\alpha = 0.9$, the accuracy does not drop. This may be explained by the selection of the first condition in a rule, which is picked according only to the classic heuristic and thus leads to growing a rule that has at least a moderate accuracy. Furthermore, in many cases, there may be a larger number of possible variants for growing a rule the learning algorithm can choose from, each leading to a comparable value according to the classic heuristic, so adding weight to the semantic heuristic still can lead to a reasonable rule.

4.2 Analysis of the Models

The two rules learned for the primary-tumor dataset shown in Table 4 illustrate the difference between rules with and without semantic coherence. Both rules cover two positive and no negative example, i.e., according to any classic heuristic, they are equally good. However, the second one can be considered to be semantically more coherent, since three out of four attributes refer to body parts (skin, pleura, and brain), and are thus semantically related.

In order to further investigate the influence of the semantic heuristic on general properties of the learned ruleset, we also looked at the average rule length, the total number of rules, and the total number of conditions in a ruleset. The results are depicted in Tables 5 and 6.

In Table 5 we observe a mostly constant and sometimes increasing number of rules for all but the last three datasets. This exception to the overall trend is

Table 5. An overview of the number of rules and conditions in the learned rulesets for selected values of α for all datasets. Datasets where a drop occurred are shown at the end of the table.

Dataset		*Accuracy*		*m-Estimate*		*Laplace Estimate*	
	α	# rules	# conditions	# rules	# conditions	# rules	# conditions
auto-mpg	0.0	47	120	14	50	48	114
	0.5	48	127	14	46	47	110
	0.9	48	127	14	46	48	110
balloons	0.0	2	4	2	4	4	12
	0.5	2	4	2	4	4	12
	0.9	2	4	2	4	4	12
bridges2	0.0	27	59	10	25	30	65
	0.5	27	61	10	25	29	65
	0.9	27	61	10	25	29	65
flag	0.0	24	78	21	51	52	106
	0.5	38	90	24	63	54	113
	0.9	38	90	24	63	54	113
zoo	0.0	12	14	11	15	19	19
	0.5	16	18	12	16	19	19
	0.9	16	18	13	19	19	19
glass	0.0	40	90	16	52	50	102
	0.5	35	82	17	55	36	74
	0.9	35	82	17	55	36	74
hepatitis	0.0	10	22	6	22	13	26
	0.5	3	6	6	18	5	8
	0.9	3	6	6	18	5	8
primary-tumor	0.0	70	246	26	119	81	324
	0.5	10	35	12	56	75	331
	0.9	11	39	11	46	16	68

analyzed more closely in case of the primary-tumor dataset. The values for this dataset are depicted in Fig. 1.

When looking at the rulesets learned on the primary-tumor dataset, it can be observed that many very special rules for small classes, covering only a few examples, are missing when increasing the value for α. A possible explanation is that as long as there are many examples for a class, there are enough degrees of freedom for the rule learner to respect semantic coherence. If, on the other hand, the number of examples drops (e.g., for small classes), it becomes harder to learn meaningful semantic rules, which leads the rule learner to ignore those small example sets. Since only a small number of examples is concerned by this, the accuracy remains stable – or it even rises slightly, as ignoring those small sets may eventually reduce the risk of overfitting.

Note that a similar trend could be observed for the other two datasets (hepatitis and glass, depicted at the lower part of Table 5). While the changes are not so intense for the *m-Estimate*, certainly those for the other two heuristics are significant. Interestingly, most often the rules in the beginning of the decision list

Table 6. Average rule length of the learned rulesets on the eight datasets. Statistically significant deviations ($p > 0.05$) from $\alpha = 0$ are marked in bold.

Classic	α										
Heuristic	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Accuracy	2.270	2.295	2.296	2.296	2.281	2.281	2.310	2.321	2.288	2.288	**0.250**
m-Estimate	2.329	2.295	2.304	2.334	2.348	2.342	2.356	2.337	2.365	2.322	**0.250**
Laplace	2.920	2.851	2.862	2.811	2.811	2.833	2.828	2.821	2.796	2.788	**0.375**

are similar and at a certain point, no rules are learned any more. Thus, similar to the effect noticeable at the dataset primary-tumor, the following low coverage rules are not induced any more.

However, when looking at the average rule length (cf. Table 6), the only significant change occurs when all weight is given to the semantic component. The reason is that most often no rule is learned at all in this case.

4.3 Semantic Coherent Rules in Relation to Characteristic Rules

When we inspected the rule sets and the behavior of our separate-and-conquer learner in more detail, we found that semantically coherent rules interestingly have a connection to so-called characteristic rules [22, 4]. Where a discriminant rule tries to use as few conditions as possible with the goal of separating the example(s) of a certain class versus all the other ones, a characteristic rule has as much as possible conditions that actually describe the example(s) at hand. For instance, if the example to be described would be an elephant, a discriminant rule would concentrate on the single attribute(s) an elephant has and no other animal shows such as, e.g., its trunk, its gray color, or its huge ears. Instead, a characteristic rule would list all attributes that indicate an elephant such as four legs, a tail, thick skin etc. In essence, a discriminant rule has only conditions that discriminate elephants from all other animals whereas a characteristic rule rather describes the elephant without the need to be discriminant, i.e., to use only features no other animal has.

Not surprisingly, a semantically coherent rule tends to show the same properties. Often the induced rules consist of conditions that are not necessarily important to discriminate the examples, but rather are semantically coherent with the conditions located at earlier positions in these rules. This becomes obvious when we take a look at the above example of the two rules where the rule without semantic influence has a condition less albeit both of them have the same coverage.

However, the number of rules is strongly dependent on the attribute's semantics. For most of the datasets where actually less rules are induced with our approach, semantic coherence is hard to measure. The glass database contains of descriptions of chemicals, in the hepatitis dataset biochemical components are used as features and in primary-tumor we have simply considerably more classes. A detailed examination of this phenomenon remains subject to future work.

5 Related Work

Most of the work concerned with the trade-off between interpretability and accuracy stems from the fuzzy rules community. Here, this trade-off is well-known and there are a number of papers that addressed this problem [30]. There are several ways to deal with it, either by using (evolutionary) multiobjective optimization [16], context adaptation, hierarchical fuzzy modeling as well as fuzzy partitioning, membership functions, rules, rule bases or similar. However, most often comprehensibility of fuzzy rules is measured by means such as the transparency of the fuzzy partitions, the number of fuzzy rules and conditions or the complexity of reasoning, i.e., defuzzification and inference mechanisms. As we use classification rules in this paper, most of these techniques are not applicable.

There are also some papers about comprehensibility in general. For example, [33] deals with the means of dimensionality reduction and with presenting statistical models in a way that the user can grasp them better, e.g., with the help of graphical representations or similar. The interpretability of different model classes is discussed in [10]. The advantages and disadvantages of decision trees, classification rules, decision tables, nearest neighbor, and Bayesian networks are shown. Arguments are given why using model size on its own for measuring comprehensibility is not the best choice and directives are demonstrated how user-given constraints such as monotonicity constraints can be incorporated into the classification model. For a general discussion of comprehensibility this is very interesting, however, as single conditions of a rule are not compared against each other, the scope is somewhat different than in our work.

A lot of papers try to induce a ruleset that has high accuracy as well as good comprehensibility by employing genetic, evolutionary, or ant colony optimization algorithms. Given the right measure for relating single conditions of a rule or even whole rules in a complete ruleset, this seems to be a promising direction. Unfortunately, most of the fitness functions do not take this into account. For example, in [25] an extension of a ant colony algorithm was derived to induce unordered rulesets. They introduced a new measure for comprehensibility of rules, namely the *prediction-explanation size*. In essence this measure is oriented more strongly on the actual prediction hence the average number of conditions that have to be checked for predicting the class value. Therefore, not the total number of conditions or rules is measured as usual measures often do but for an unordered ruleset exactly those that are actually used for classifying the example at hand. For ordered rulesets also rules are counted that are before the classifying rule in the decision list as they have to be also checked at prediction time. Other algorithms are capable of multi-target learning [24] and define interestingness as those rules that cover example of infrequent classes in the dataset. Also, some papers deal with interpretability rather as a side effect [2], while here no optimization of this objective is done during learning time. In contrast, [7] uses a simple combination of accuracy maximization and size minimization in the fitness function of the genetic algorithm.

Some research is focused on specific problems where consequently rather unique properties are taken into account [31]. In this bioinformatic domain, only the presence of an attribute (value) is of interest whereas the absence is of no concern. The contribution are two new versions of CN2 [6] and Ant-Miner [26] which are able to incorporate this constraint.

Another thread is concerned with the measures themselves. For example, [9] surveyed objective measures (data-driven) for interestingness and defined a new objective, namely attribute surprisingness *AttSurp*, i.e., by arguing that a user is mostly interested in a rule that has high prediction performance but many single attributes with a low information gain, the authors define AttSurp as one divided by the information gain of all attributes in the rule. In [11] it is argued that small disjuncts (i.e., rules that cover only a very small number of positive examples) are indeed surprising while most often not unfolding good generalization or predictive quality. Here, also AttSurp is used which is different to most other interestingness measures in the sense that not the whole rule body is taken into account but single attributes which one can also see as a property of our algorithm. Interestingly, surprisingness also is related to Simpson's Paradox.

6 Conclusions and Future Work

In this paper, we have examined an approach to increase the understandability of a rule model by learning rules that are in themselves semantically coherent. To do so, we have introduced a method for combining classic heuristics, tailored at learning *correct* rule models, with semantic heuristics, tailored at learning *coherent* rules. While we have only looked at the coherence of *single* rules, adding means to control the coherence across a set of rules would be an interesting extension for future work.

An experiment with eight datasets from the UCI repository has shown that it is possible to learn rules that are significantly more coherent, while not being significantly less accurate. In fact, the accuracy of the learned model has stayed constant in all cases, even if adjusting the influence of the semantic heuristic to 90% of the overall heuristic. These results show that, even at a very preliminary stage, the proposed approach actually works.

Furthermore, we have observed that in some cases, adding the semantic heuristic may lead to more compact rule sets, which are still as accurate as the original ones. Although we have a possible explanation, i.e., that it is difficult for semantically enhanced heuristics to learn rules for small sets of examples, we do not have statistically significant results here. An evaluation with synthetic datasets may lead to more insights into the characteristics of datasets for which this property holds, and help us to confirm or reject that hypothesis.

Although we have evidence from previous research that semantically coherent rules are perceived to be better understandable, e.g. in [27], we would like to strengthen that argument by additional user studies. These may also help revealing other characteristics a ruleset should have beyond *coherence*, e.g., minimum or maximum length. For example, the experiments in [27] have indicated that less accurate rules (e.g., *Countries with a high HDI are less corrupt*) are preferred over more accurate ones (e.g., *Countries with a HDI higher than 6.243 are less corrupt*).

In this paper, we have only looked into one method of measuring semantic coherence, i.e., a similarity metric based on WordNet. There are more possible WordNet-based metrics, e.g., the LESK [1] and the HSO [15] metrics, which both work with adjectives and adverbs in addition to nouns and verbs and they support arbitrary pairing of the POS classes. Furthermore, there is a number of alternatives beyond WordNet, e.g., the use of Wikipedia [32] or a web search engine [5]. Furthermore, in the realm of Linked Open Data, there are various means to determine the relatedness of two concepts [28].

The approach so far only uses the classical heuristic to select the first rule, which sometimes lead to rules that are not too coherent w.r.t. that attribute, e.g., if there are no other attributes that match the first one well semantically. Here, it may help to introduce a semantic part in the selection of the first condition as well, e.g., the average semantic distance of all other attributes to the one selected. However, the impact of that variation on accuracy has to be carefully investigated.

Another possible point for improvement is the selection of the final rule from one refinement process. So far, we use the same combined heuristic for the refinement and the selection, but it might make sense to use a different weight here, or even entirely remove the semantic heuristic from that step, since the coherence has already been assured by the selection of the conditions.

In summary, we have introduced an approach that is able to explicitly trade off semantic coherence and accuracy in rule learning, and we have shown that it is possible to learn more coherent rules without losing accuracy. However, it remains an open question whether or not our results are generalizable to other types of rule learning algorithms that do not rely on a separate-and-conquer strategy. We will inspect the impact on other rule learners in the near future.

References

1. Banerjee, S., Pedersen, T.: An Adapted Lesk Algorithm for Word Sense Disambiguation Using WordNet. In: Computational linguistics and intelligent text processing, pp. 136–145. Springer Berlin Heidelberg (2002)
2. Bojarczuk, C.C., Lopes, H.S., Freitas, A.A.: Discovering comprehensible classification rules by using genetic programming: a case study in a medical domain. In: Banzhaf, W., Daida, J., Eiben, A.E., Garzon, M.H., Honavar, V., Jakiela, M., Smith, R.E. (eds.) Proceedings of the Genetic and Evolutionary Computation Conference. vol. 2, pp. 953–958. Morgan Kaufmann, Orlando, Florida, USA (1999)
3. Budanitsky, A., Hirst, G.: Evaluating WordNet-based Measures of Lexical Semantic Relatedness. Computational Linguistics 32(1), 13–47 (2006)
4. Cai, Y., Cercone, N., Han, J.: Attribute-oriented induction in relational databases. In: Knowledge Discovery in Databases, pp. 213–228. AAAI/MIT Press (1991)
5. Cilibrasi, R., Vitányi, P.M.B.: The google similarity distance. CoRR abs/cs/0412098 (2004)
6. Clark, P., Niblett, T.: The CN2 Induction Algorithm. Machine Learning 3(4), 261–283 (1989)

7. Falco, I.D., Cioppa, A.D., Tarantino, E.: Discovering interesting classification rules with genetic programming. Applied Soft Computing 1(4), 257 – 269 (2002)
8. Fellbaum, C.: WordNet. Wiley Online Library (1999)
9. Freitas, A.: On rule interestingness measures. Knowledge-Based Systems 12(56), 309 – 315 (1999)
10. Freitas, A.A.: Comprehensible classification models: A position paper. SIGKDD Explor. Newsl. 15(1), 1–10 (Mar 2014)
11. Freitas, A.A.: On objective measures of rule surprisingness. In: Proceedings of the Second European Symposium on Principles of Data Mining and Knowledge Discovery. pp. 1–9. PKDD '98, Springer-Verlag, London, UK, UK (1998)
12. Fürnkranz, J.: Separate-and-Conquer Rule Learning. Artificial Intelligence Review 13(1), 3–54 (1999)
13. Fürnkranz, J., Flach, P.A.: ROC 'n' Rule Learning - Towards a Better Understanding of Covering Algorithms. Machine Learning 58(1), 39–77 (January 2005)
14. Fürnkranz, J., Gamberger, D., Lavrač, N.: Foundations of Rule Learning. Springer Berlin Heidelberg (2012)
15. Hirst, G., St-Onge, D.: Lexical Chains as Representations of Context for the Detection and Correction of Malapropisms. In: Fellbaum, C. (ed.) WordNet: An Electronic Lexical Database, pp. 305–332. MIT Press (1995)
16. Ishibuchi, H., Nojima, Y.: Analysis of interpretability-accuracy tradeoff of fuzzy systems by multiobjective fuzzy genetics-based machine learning. International Journal of Approximate Reasoning 44(1), 4–31 (Jan 2007)
17. Janssen, F., Fürnkranz, J.: On the quest for optimal rule learning heuristics. Machine Learning 78(3), 343–379 (Mar 2010)
18. Janssen, F., Zopf, M.: The SeCo-framework for rule learning. In: Proceedings of the German Workshop on Lernen, Wissen, Adaptivität - LWA2012 (2012)
19. Jiang, J.J., Conrath, D.W.: Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy. In: Proceedings of International Conference on Research in Computational Linguistics (ROCLING X). pp. 19–33. No. Rocling X (1997)
20. Kintsch, W., Van Dijk, T.A.: Toward a model of text comprehension and production. Psychological review 85(5), 363 (1978)
21. Lin, D.: An Information-Theoretic Definition of Similarity. In: ICML. pp. 296–304 (1989)
22. Michalski, R.S.: A theory and methodology of inductive learning. Artificial Intelligence 20(2), 111–162 (1983)
23. Miller, G.a., Leacock, C., Tengi, R., Bunker, R.T.: A Semantic Concordance. In: Proceedings of the workshop on Human Language Technology. pp. 303–308. Association for Computational Linguistics, Morristown, NJ, USA (1993)
24. Noda, E., Freitas, A., Lopes, H.: Discovering interesting prediction rules with a genetic algorithm. In: Proceedings of the 1999 Congress on Evolutionary Computation. pp. 1322–1329. IEEE (1999)
25. Otero, F.E., Freitas, A.A.: Improving the interpretability of classification rules discovered by an ant colony algorithm. In: Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation. pp. 73–80. GECCO '13, ACM, New York, NY, USA (2013)
26. Parpinelli, R.S., Lopes, H.S., Freitas, A.A.: Data mining with an ant colony optimization algorithm. IEEE Transactions on Evolutionary Computation 6(4), 321–332 (August 2002)
27. Paulheim, H.: Generating possible interpretations for statistics from linked open data. In: 9th Extended Semantic Web Conference (ESWC) (2012)

28. Paulheim, H.: Dbpedianyd – a silver standard benchmark dataset for semantic relatedness in dbpedia. In: Workshop on NLP & DBpedia (2013)
29. Resnik, P.: Using Information Content to Evaluate Semantic Similarity in a Taxonomy. In: Proceedings of the 14th International Joint Conference on Artificial Intelligence. vol. 1 (1995)
30. Shukla, P.K., Tripathi, S.P.: A survey on interpretability-accuracy (i-a) trade-off in evolutionary fuzzy systems. In: Watada, J., Chung, P.C., Lin, J.M., Shieh, C.S., Pan, J.S. (eds.) 5th International Conference on Genetic and Evolutionary Computing. pp. 97–101. IEEE (2011)
31. Smaldon, J., Freitas, A.A.: Improving the interpretability of classification rules in sparse bioinformatics datasets. In: Proceedings of AI-2006, the Twenty-sixth SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence. pp. 377–381. Research and Development in Intelligent Systems XXIII, Springer London (2007)
32. Strube, M., Ponzetto, S.P.: WikiRelate! Computing semantic relatedness using Wikipedia. In: In Proceedings of the 21st National Conference on Artificial Intelligence. pp. 1419–1424. No. February, AAAI Press (2006)
33. Vellido, A., Martn-Guerrero, J.D., Lisboa, P.J.G.: Making machine learning models interpretable. In: 20th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (2012)

Text Classification Using Association Rules, Dependency Pruning and Hyperonymization

Yannis Haralambous and Philippe Lenca

Institut Mines Telecom, Telecom Bretagne, UMR CNRS 6285 Lab-STICC
Technopôle Brest Iroise CS 83818, 29238 Brest Cedex 3, France
`<surname>.<name>@telecom-bretagne.eu`

Abstract. We present new methods for pruning and enhancing itemsets for text classification via association rule mining. Pruning methods are based on dependency syntax and enhancing methods are based on replacing words by their hyperonyms of various orders. We discuss the impact of these methods, compared to pruning based on tfidf rank of words.

Introduction

Automatic text classification is an important text mining task, due to the huge number of text documents that we have to manage daily. Text classification has a wide variety of applications such as Web document and email classification. Indeed, most of the Web news services daily provide a large number of articles making them impossible to be organized manually [14]. Automatic subject classification [9] and SPAM filtering [18] are two additional examples of the interest of automatic text classification.

Automatic text classification can be defined as below. Given a set of documents such that each document is labeled with a class value, learn a model that assigns a document with unknown class to one or more particular classes. This can also be done by assigning a probability value to each class or by ranking the classes.

A wide variety of classical machine learning techniques have been used for text classification. Indeed, texts may be represented by word frequencies vectors, and thus most of the quantitative data methods can be used directly on the notorious "bag-of-words" model (cf. [27,3]).

Choosing a classifier is a multicriteria problem. In particular one has often to make a trade-off between accuracy and comprehensibility. In this paper, we are interested in both criteria with a deeper interest in comprehensibility. We are thus interested in rule-based approaches and especially in class association rules algorithms. Several studies have already successfully considered association rule-based approaches in text mining (e.g., [4], [29], [7], [25]). This framework is suitable for considering some statistical characteristics (e.g., high-dimensionality, sparsity...) of the bag-of-words model where a document is represented as a set of words with their associated frequency in the document.

In: P. Cellier, T. Charnois, A. Hotho, S. Matwin, M.-F. Moens, Y. Toussaint (Eds.): Proceedings of DMNLP, Workshop at ECML/PKDD, Nancy, France, 2014.

However a text is more than a set of words and their frequencies. Enhancing the bag-of-words approach with linguistic features has also attracted several works (e.g., [12,11,13,22], [23,16,10], [21,6]).

We here propose a class association rules based approach enriched by linguistic knowledge. The paper is organized as follows: after introducing the techniques we are going to use (class association rules § 1.1, dependencies § 1.2, hyperonymization § 1.3) we describe our main algorithms (for training § 2.1, classifying § 2.2 and evaluating § 2.3); follows the experimental section, where we give results obtained by tfidf pruning § 3.2, dependency-based pruning § 3.3 and hyperonymization § 3.4, and, finally, we end up by a conclusion and perspectives for future work § 4.

1 Proposed model for text classification

Let a *corpus* be a set $\mathbf{C} = \{D_1, \ldots, D_n\}$ of documents. Let $\mathcal{C}$ be a set of classes. An *annotated corpus* is a pair $(\mathbf{C}, \mathrm{class})$ where $\mathrm{class} : \mathbf{C} \to \mathcal{C}$ is a function that maps each document D_i to a (predefined) class of $\mathcal{C}$.

A document $D \in \mathbf{C}$ is a set of sentences S. The corpus $\mathbf{C}$ can be considered as a set of sentences $\mathbf{S} = \{S_1, \ldots, S_m\}$ if we go through the forgetful functor (which forgets the document to which the sentence belongs). Repeated sentences in the same document, or identical sentences in different documents are considered as distinct, i.e., there is a function $\iota \colon \mathbf{S} \to \mathbf{C}$ which restores the forgotten information. We extend the class function to $\mathbf{S}$ by $\mathrm{class}(S) := \mathrm{class}(\iota(S))$.

A sentence S is a sequence of words w (sometimes we will consider S simply as a set, without changing the notation). Let $\mathcal{W} = \bigcup_{S\in\mathbf{S}} \bigcup_{w\in S}\{w\}$ be the set of all words of $\mathbf{C}$.

1.1 Class association rules and text classification

Let $\mathcal{I}$ be a set of objects called *items* and $\mathcal{C}$ a set of classes. A *transaction* T is a pair $(\{i_1, \ldots, i_n\}, c)$, where $\{i_1, \ldots, i_n\} \subseteq \mathcal{I}$ and $c \in \mathcal{C}$. We denote by $\mathcal{T}$ the set of transactions, by $\mathrm{items}(T)$ the set of items (or "itemset") of T and by $\mathrm{class}(T)$ the class of T.

Let I be an itemset. The *support* of I is defined by

$$\mathrm{supp}(I) := \tfrac{\#\{T\in\mathcal{T} \mid I\subseteq\mathrm{items}(T)\}}{\#\mathcal{T}}.$$

Let $\sigma \in [0, 1]$ be a value called *minimum support*. An itemset I is called *frequent* if its *support* exceeds σ.

The *confidence* of a transaction t is defined as

$$\mathrm{conf}(t) := \tfrac{\#\{T\in\mathcal{T} \mid \mathrm{items}(t)\subseteq\mathrm{items}(T)\wedge\mathrm{class}(t)=\mathrm{class}(T)\}}{\#\{T\in\mathcal{T} \mid \mathrm{items}(t)\subseteq\mathrm{items}(T)\}}.$$

Let $\kappa \in [0, 1]$ be a value called *minimum confidence*. A *class association rule* (or "CAR") $r = (\{i_1, \ldots, i_n\}, c)$ [15] is a transaction with frequent itemset and a *confidence* exceeding κ.

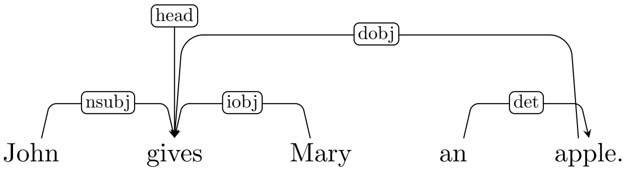
To classify text with CARs, we consider words as being items, documents as being itemsets and pairs of documents and classes as being transactions. The advantage of this technique is that CARs can be easily understood and hence potentially improved by the user, especially if the classifier is tuned so that it produces humanly reasonable number of rules. Once the classifier is trained, to classify a new sentence we first find all CARs whose items are contained in the sentence, and then use an aggregation technique to choose a predominant class among those of the CARs we found.

An important issue of CARs is that the complexity is exponential with respect to the itemset size, and hence we need to keep it bounded in specific ranges, independently of the size of documents to classify. Using entire documents as transactions is computationally out of reach, therefore pruning techniques play an important rôle. Our approach consists in (a) restricting CARs to the sentence level, (b) prune sentences by using morphosyntactic information (cf. § 1.2) and modifying itemsets using semantic information (cf. § 1.3).

1.2 Itemset pruning using dependencies

One can prune sentences either by using word frequencies (cf. § 3.2) or by using information obtained by morphosyntactic parsing (cf. § 3.3). In this paper we introduce the latter approach, in the frame of dependency grammar.

Dependency grammar [28,19] is a syntactic theory, alternative to *phrase-structure analysis* [8] which is traditionally taught in primary and secondary education. In phrase-structure syntax, trees are built by grouping words into "phrases" (with the use of intermediate nodes NP, VP, etc.), so that the root of the tree represents the entire sentence and its leaves are the actual words. In dependency grammar, trees are built using solely words as nodes (without introducing any additional "abstract" nodes). A single word in every sentence becomes the *root* (or *head*) of the tree. An oriented edge between two words is a *dependency* and is tagged by a representation of some (syntactic, morphological, semantic, prosodic, etc.) relation between the words. For example in the sentence "John gives Mary an apple," the word "gives" is the head of the sentence and we have the following four dependencies:



where tags nsubj, dobj, iobj, det denote "noun subject," "direct object," "indirect object" and "determinant."

Let S be a sentence and $\mathcal{D}$ be the set of dependency tags: {nsubj, ccomp, prep, dobj, . . . } A *dependency* is a triple (w_1, w_2, d) where $w_1, w_2 \in S$ and $d \in \mathcal{D}$. Let Dep(S) denote the set of dependencies of S and root(S) the head of S.

Pruning will consist in defining a *morphosyntactic constraint* ϕ i.e. a condition on dependencies (and POS tags) of words, the fulfillment of which is necessary for the word to be included in the itemset.

But before describing pruning algorithms and strategies, let us first present a second technique used for optimizing itemsets. This time we use semantic information. We propose to replace words by their hyperonyms, expecting that the frequencies of the latter in the itemsets will be higher than those of the former, and hence will improve the classification process.

1.3 Hyperonymization

The WordNet lexical database [20] contains sets of words sharing a common meaning, called *synsets*, as well as semantic relations between synsets, which we will use to fulfill our goal. More specifically, we will use the relations of *hyperonymy* and of *hyperonymic instance*. The graph having synsets as nodes, and hyperonymic relations as edges, is connected and rooted: starting with an arbitrary synset, one can iterate these two relations until attaining a sink. Note that in the case of nouns it will invariably be the synset `00001740` {entity} while for verbs there are approx. 550 different verb sinks.

Let $\mathbb{W}$ be the WordNet lexical database, $s \in \mathbb{W}$ a synset and $h : \mathbb{W} \to 2^{\mathbb{W}}$ the hyperonymic or hyperonymic instance relation. We define an *hyperonymic chain* $\mathrm{CH}(s)$ as a sequence $(s_i)_{i\geq 0}$ where $s_0 = s$ and $s_i \in h(s_{i-1})$, for all $i \geq 1$. Hyperonymic chains are not unique since a given synset can have many hyperonyms. To replace a word by the most pertinent hyperonym, we have to identify the most significant hyperonymic chains of it.

The *wn-similarity* project [24] has released synset frequency calculations based on various corpora. Let $\mathrm{lf}(s)$ denote the logarithmic frequency of synset s in the BNC English language corpus [2] and let us arbitrarily add infinitesimally small values to the frequencies so that they become unique ($s \neq s' \Rightarrow \mathrm{lf}(s) \neq \mathrm{lf}(s')$). We use frequency as the criterion for selecting a single hyperonymic chain to represent a given synset, and hence define the *most significant hyperonymic chain* $\mathrm{MSCH}(s)$ as the hyperonymic chain $(s_i)_{i\geq 0}$ of s such that $s_i = \arg\max_{s\in h(s_{i-1})} \mathrm{lf}(s)$, for all $i \geq 1$. The chain $\mathrm{MSCH}(s)$ is unique thanks to the uniqueness of synset frequencies.

Our CARs are based on words, not synsets. Hence we need to extend MSCHs to words. Let w be a lemmatized word. We denote by $\mathrm{Synsets}(w) \subset \mathbb{W}$ the set of synsets containing w. If the cardinal $\#(\mathrm{Synsets}(w)) > 1$ then we apply a standard disambiguation algorithm to find the most appropriate synset s_w for w in the given context. Then we take $(s_i)_i = \mathrm{MSCH}(s_w)$ and for each synset s_i in this chain we define $h_i(w) = \mathrm{proj}_1(s_i)$ $(i > 0)$, that is the projection of s_i to its first element, which by WordNet convention is the most frequent word in the synset. The function vector $h_* : \mathcal{W} \to \mathcal{W}$ (with $h_0 \equiv \mathrm{Id}$) is called *hyperonymization*, and $h_i(w)$ is the *i-th order hyperonym of* w.

```
Algorithm 1: Training
  Data: An annotated corpus $\mathbf{C}$, values of minimum support $\sigma$ and minimum
        confidence $\kappa$
  Result: A set of CARs $\mathcal{R} = (\{R_1, \ldots, R_N\}, \text{conf})$ where $\text{items}(R_i) \subset \mathcal{W}$,
          $\text{class}(R_i) \in \mathcal{C}$, and $\text{conf}(R_i)$ is the confidence of rule $R_i$
  Train($\mathbf{C}$, $\sigma$, $\kappa$):
      $\mathbf{S}$ := forgetful($\mathbf{C}$); $\mathbf{S}'$ := $\emptyset$;
      for $S \in \mathbf{S}$ do
          $S'$ := Hyperonymize (Prune (Lemmatize ($S$)));
          class($S'$) := class($\iota(S)$);
          $\mathbf{S}'$ := $\mathbf{S}' \cup \{S'\}$;
      end
      $\mathcal{R}$ := Apriori ($\mathbf{S}', \sigma, \kappa$);
  end
```

2 Operational implementations for document classification

Our text classifier operates by first training the classifier on sentences and then classifying the documents by aggregating sentence classification. These two procedures are described in Sections 2.1 and 2.2 respectively. Specific evaluation procedure is presented in Section 2.3.

2.1 Training

The `Train` algorithm (cf. Alg. 1) takes as input an annotated corpus **C** and values of minimum support σ and minimum confidence κ. It returns a set of CARs together with their confidence values.

The first part of the algorithm consists in processing the corpus, to obtain efficient and reasonably sized transactions. Three functions are applied to every sentence:

1. `Lemmatize` is standard lemmatization: let $\mathcal{P}$ be the set of POS tags of the *TreeTagger* system [26] (for example, NP stands for "proper noun, singular", VVD stands for "verb, past tense", etc.), and let $\mathcal{W}'$ be the set of lemmatized forms of $\mathcal{W}$ (for example, "say" is the lemmatized form of "said"); then we define $\lambda : \mathcal{W} \to (\mathcal{W} \cup \mathcal{W}') \times \mathcal{P}$, which sends a word w to the pair (w', p) where w' is the lemmatized form of w (or w itself, if the word is unknown to *TreeTagger*) and p is its POS tag.
2. `Prune` is a function which prunes the lemmatized sentence so that only a small number of (lemmatized) words (and POS tags) remains. Several sentence pruning strategies will be proposed and compared (cf. § 3.2 and 3.3).
3. `Hyperonymize` is a function which takes the words in the pruned itemset and replaces them by the members of their most significant hyperonymic chains. Several strategies will also be proposed and compared (cf. § 3.4).

Algorithm 2: Classification

```
Data: A set of CARs $\mathcal{R}$, a document $D_0$
Result: The predicted class predclass($D_0$), variety $\beta$, dispersion $\varDelta$
Classify($\mathcal{R}$, $D_0$):
    for $S \in D_0$ do
        if $\exists r \in \mathcal{R}$ such that items($r$) $\subset S$ then
            $R_S := \operatorname*{arg\,max}_{r \in \mathcal{R} \wedge \mathrm{items}(r) \subset S} \mathrm{conf}(r)$;
        end
    end
    $\mathrm{predclass}(D_0) := \operatorname*{arg\,max}_{c \in \mathcal{C}} \sum_{\substack{S \in D_0 \\ \mathrm{class}(R_S)=c}} \mathrm{conf}(R_S)$;
    $\beta := \#\{c \in \mathcal{C} \mid (\mathrm{class}(R_S) = c) \wedge (\mathrm{conf}(R_S) > 0)\}$;
    $\varDelta := \max_{c \in \mathcal{C}} \sum_{\substack{S_i \in D_0 \\ \mathrm{class}(R_{S_i})=c}} \mathrm{conf}(R_{S_i}) - \min_{c \in \mathcal{C}} \sum_{\substack{S_i \in D_0 \\ \mathrm{class}(R_{S_i})=c}} \mathrm{conf}(R_{S_i})$;
end
```

The second part of Alg. 1 uses the *apriori* algorithm [5] with the given values of minimum support and minimum confidence and output restrictions so as to generate only rules with item $c \in \mathcal{C}$ in the consequent. It returns a set $\mathcal{R}$ of CARs and their confidence.

It should be noted that this algorithm operates on individual sentences, hereby ignoring the document level.

2.2 Classification

The `Classify` algorithm (cf. Alg. 2) uses the set of CARs produced by `Train` to predict the class of a new document D_0 and furthermore provides two values measuring the quality of this prediction: variety β and dispersion $\varDelta$.

The first part of the algorithm takes each sentence S of the document D_0 and finds the most confident CAR that can be applied to it (i.e., such that the itemset of the rule is entirely contained in the itemset of the sentence). At this stage we have, for every sentence: a rule, its predicted class and its confidence.

Our basic unit of text in `Train` is sentence, therefore CARs generated by Alg. 1 produce a class for each sentence of D_0. An aggregation procedure is thus needed in order to classify the document. This is done by taking class by class the sum of confidence of rules and selecting the class with the highest sum.

Although this simple class-weighted sum decision strategy is reasonable, it is not perfect and may lead to wrong classification. This strategy will be *optimally* sure and robust if (a) the number of classes is minimal, and (b) the values when summing up confidence of rules are sufficiently spread apart. The degree of fulfillment of these two conditions is given by the parameters *variety* β (the number of classes for which we have rules), and *dispersion* $\varDelta$ (the gap between the most confident class and least confident one). These parameters will contribute to comparison among the different approaches we will investigate.

Algorithm 3: Evaluation

```
Data: An annotated corpus $\mathbf{C}$, initial values of minimal support $\sigma_0$ and
      confidence $\kappa_0$, standard number of rules $\rho_0$
Result: Values of average precision $\overline{P}$, recall $\overline{R}$, F-measure $\overline{F}$. Values of average
        number of rules $\rho$, variety $\beta$ and dispersion $\Delta$
SingleEvaluate($\mathbf{C}$, $\sigma$, $\kappa$):
    $(\mathbf{C}_1, \dots, \mathbf{C}_{10})$ := Partition (Shuffle ($\mathbf{C}$),10);
    /* tenfold cross validation                                          */
    for $I \in \{1, \dots, 10\}$ do
        $(\mathcal{R}_I, \beta_I, \Delta_I)$ := Train ($\mathbf{C} \setminus \mathbf{C}_1, \sigma, \kappa$);
        for $D \in \mathbf{C}_I$ do
            predclass($D$) := Classify ($\mathcal{R}_I$, $D$);
        end
        for $c \in \mathcal{C}$ do
            $R_I(c)$ := $\frac{\#\{d \in \mathbf{C}_I | (\mathrm{predclass}(d)=c) \wedge (\mathrm{class}(d)=c)\}}{\#\{d \in \mathbf{C}_I | \mathrm{class}(d)=c\}}$;
            $P_I(c)$ := $\frac{\#\{d \in \mathbf{C}_I | (\mathrm{predclass}(d)=c) \wedge (\mathrm{class}(d)=c)\}}{\#\{d \in \mathbf{C}_I | \mathrm{predclass}(d)=c\}}$; $F_I(c)$ := $\frac{2R_I(c)P_I(c)}{R_I(c)+P_I(c)}$;
        end
    end
    for $c \in \mathcal{C}$ do
        $(R(c), P(c), F(c))$ := $\frac{1}{10}\sum_{I=1}^{10}(R_I(c), P_I(c), F_I(c))$;
    end
    $(\rho, \beta, \Delta)$ := $\frac{1}{10}\sum_{I=1}^{10}(\#\mathcal{R}_I, \beta_I, \Delta_I)$;
    $(\overline{R}, \overline{P}, \overline{F})$ := $\frac{1}{\#\mathcal{C}}\sum_{c \in \mathcal{C}}(R(c), P(c), F(c))$;
end
Evaluate($\mathbf{C}$, $\sigma_0$, $\kappa_0$, $\rho_0$):
    $(\sigma, \kappa)$ := FindOptimal($\mathbf{C}, \sigma_0, \kappa_0, \rho_0$);
    $(\overline{R}, \overline{P}, \overline{F}, \rho, \beta, \Delta)$ := SingleEvaluate($\mathbf{C}, \sigma, \kappa$);
end
```

2.3 Evaluation

We evaluate the classifier (Alg. 3), by using 10-fold cross validation to obtain average values of recall, precision, F-measure, variety and dispersion. This is done by algorithm `SingleEvaluate`, once we specify values of minimal support and minimal confidence.

Comparing rule-based classification methods is problematic because one can always increase F-measure performance by increasing the number of rules, which results in overfitting them. To avoid this phenomenon and compare methods in a fair way, we fix a number of rules ρ_0 (we have chosen $\rho_0 = 1{,}000$ in order to produce a humanly reasonably readable set of rules) and find values of minimal support and confidence so that F-measure is maximal under this constraint.

Function `FindOptimal` will launch `SingleEvaluate` as many times as necessary on a dynamic grid of values (σ, κ) (starting with initial values (σ_0, κ_0)), so that, at the end, the number of rules produced by `Train` is as close as possible to ρ_0 (we have used $\#\mathcal{R} \in [\rho_0 - 2, \rho_0 + 2]$) and $\overline{F}$ is maximal.

Algorithm 4: Tfidf-based corpus pruning

```
Data: An annotated corpus (considered as a set of sentences) $\mathbf{S}$
Result: The pruned corpus $\mathbf{S}'$
Prune($\mathbf{S}$, $N$):
    $\mathbf{S}' := \emptyset$;
    for $S \in \mathbf{S}$ do
        for $w \in S$ do
            $\mathtt{Tfidf}_S(w) := \mathrm{freq}_S(w) \cdot \log\left(\frac{\#\{S \in \mathbf{C}\}}{\#\{S \in \mathbf{C} | w \in S\}}\right)$
        end
        $S' := \emptyset$; $S_0 := S$;
        for $i \in \{1, \ldots, N\}$ do
            $w' := \arg\max_{S_0} \mathtt{Tfidf}_S(w)$;
            $S_0 := S_0 \setminus \{w'\}$; $S' := S' \cup \{w'\}$;
        end
        $\mathbf{S}' := \mathbf{S}' \cup \{S'\}$;
    end
end
```

3 Experimental results on Reuters corpus

In this section, we investigate three methods: (a) pruning through a purely frequentist method, based on tfidf measure (§ 3.2); (b) pruning using dependencies (§ 3.3); (c) pruning using dependencies followed by hyperonymic extension (§ 3.4).

3.1 Preliminaries

In the Reuters [1] corpus we have chosen the 7 most popular topics (GSPO = sports, E12 = monetary/economic, GPOL = domestic politics, GVIO = war, civil war, GDIP = international relations, GCRIM = crime, law enforcement, GJOB = labor issues) and extracted the 1,000 longest texts of each.

The experimental document set is thus a corpus of 7,000 texts of length between 120 and 3,961 words (mean 398.84, standard variation 169.05). The texts have been analyzed with the Stanford Dependency Parser [17] in collapsed mode with propagation of conjunct dependencies.

3.2 Tfidf-based corpus pruning

Tfidf-based corpus pruning consists in using a *classical* `Prune` function as defined in Alg. 4. It will be our baseline for measuring performance of dependency- and hyperonymy-based methods.

Note that this definition of the tfidf measure diverges from the legacy one by the fact that we consider not documents but sentences as basic text units. This is because we compare tfidf-generated CARs to those using syntactic information, and syntax is limited to the sentence level. Therefore, in order to obtain a fair comparison, we have limited term frequency to the sentence level and our "document frequency" is in fact a sentence frequency.

Having calculated $\texttt{Tfidf}_S(w)$ for every $w \in S \in \mathbf{S}$, we take N words from each sentence with the highest tfidf values, and use them as transaction items. The performance of this method depends on the value of N. On Fig. 1 the reader can see the values of three quantities as functions of N:

1. F-measure: we see that F-measure increases steadily and reaches a maximum value of 83.99 for $N = 10$. Building transactions of more than 10 words (in decreasing tfidf order) deteriorates performance, in terms of F-measure;
2. variety: the number of predicted classes for sentences of the same document progressively increases but globally remains relatively low, around 3.1, except for $N = 12$ and $N = 13$ where it reaches 4.17;
3. dispersion: it increases steadily, with again a small outlier for $N = 12$, probably due to the higher variety obtained for that value of N.

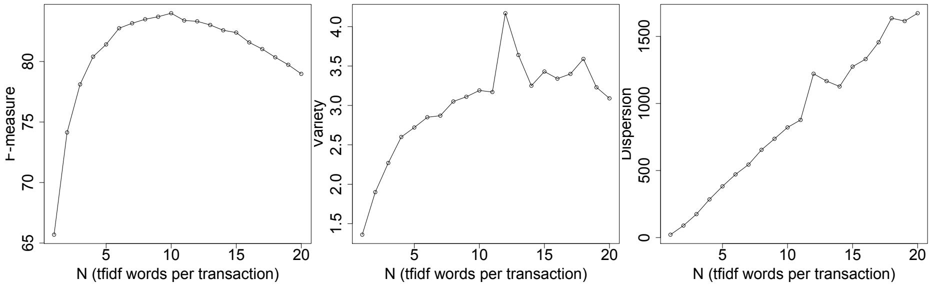


Fig. 1. F-measure, variety and dispersion of tfidf-based pruning methods as a function of the number of words kept in the corpus

Furthermore, each investigated method will generate transactions of various sizes. It is fair to compare them with tfidf-based methods with similar transactions sizes. Therefore we will use the results displayed in Fig. 1 to compare the performance of subsequent methods with the one of the tfidf-based method of similar transaction size. Table 1 presents the results obtained by applying the tdfif-based pruning method, with a single word per transaction ($N = 1$).

Table 1. Tfidf-based pruning, keeping a single word per transaction

	E12	GCRIM	GDIP	GJOB	GPOL	GSPO	GVIO	AVG
Recall	69.30	44.48	55.44	45.75	52.54	82.90	67.98	59.77
Precision	70.09	77.81	71.25	79.76	71.62	80.78	73.35	74.95
F-measure	69.69	56.60	62.36	58.15	60.61	81.83	70.56	65.69

MinSupp=0.006, MinConf=67.6, Var.=1.36, Disp.=21.53, AvgTransSize=1.00

3.3 Methods based on dependencies

In this section we investigate several strategies using the dependency structure of sentences. Our general approach (cf. Alg. 5) keeps only words of S that fulfill

Algorithm 5: Dependency-based corpus pruning

```
Data: An annotated corpus $\mathbf{S}$ and a morphosyntactic contraint
      $\phi\colon S \to \{\text{true}, \text{false}\}$
Result: The pruned corpus $\mathbf{S}'$
Prune($\mathbf{S}$, $\phi$):
    $\mathbf{S}'$ := $\emptyset$;
    for $S \in \mathbf{S}$ do
        $S'$ := $\emptyset$;
        for $w \in S$ do
            if $\phi(w) = \text{true}$ then
                $S'$ := $S' \cup \{w\}$
            end
        end
        $\mathbf{S}'$ := $\mathbf{S}' \cup \{S'\}$;
    end
end
```

Table 2a. Strategy I_0: Pruning by keeping only heads of sentences

	E12	GCRIM	GDIP	GJOB	GPOL	GSPO	GVIO	AVG
Recall	38.54	57.46	26.88	17.43	31.06	88.49	51.90	44.54
Precision	49.13	66.18	49.61	65.73	39.07	42.18	60.31	53.17
F-measure	43.19	61.51	34.87	27.55	34.61	57.13	55.79	44.95

MinSupp=0.004, MinConf=36.6, Var.=2.14, Disp.=47.84, AvgTransSize=1.00

Table 2b. Strategy I_1: Pruning by keeping only nsubj → head dependencies

	E12	GCRIM	GDIP	GJOB	GPOL	GSPO	GVIO	AVG
Recall	70.55	60.44	66.97	58.27	63.22	78.76	69.92	66.88
Precision	73.05	80.84	72.87	86.98	71.71	85.29	76.29	78.15
F-measure	71.78	69.17	69.80	69.79	67.19	81.89	72.97	71.80

MinSupp=0.007, MinConf=60.4, Var.=1.43, Disp.=41.71, AvgTransSize=1.04

a given morphosyntactic constraint ϕ. The following strategies correspond to various definitions of ϕ.

Strategy $\mathbf{I_0}$ Our first strategy will be to keep only the head of each sentence (which, incidentally, is a verb in 85.37% of sentences of our corpus). This corresponds to the constraint $\phi(w) \equiv (w = \text{root}(S))$. Results are given on Table 2a.

Although the recall of GSPO is quite high (a possible interpretation could be that sports use very specific verbs), F-measure is quite low when we compare it to the one of the tfidf-based method of the same average itemset length, namely 65.69%.

Strategy $\mathbf{I_1}$ The second strategy consists in keeping words connected to the head by a (single) dependency of type nsubj (= nominal subject). This occurs in

Table 2c. Strategy I'_1: Pruning by keeping only ccomp → head dependencies

	E12	GCRIM	GDIP	GJOB	GPOL	GSPO	GVIO	AVG
Recall	57.33	33.82	25.31	16.96	21.74	47.62	59.60	37.48
Precision	37.83	47.55	38.50	42.06	34.62	57.05	54.17	44.54
F-measure	45.59	39.53	30.54	24.17	26.71	51.91	56.75	39.31

MinSupp=0.008, MinConf=34.4, Var.=1.97, Disp.=19.59, AvgTransSize=1.15

Table 2d. Strategy I_2: Pruning by keeping only nouns at distance 1 from head

	E12	GCRIM	GDIP	GJOB	GPOL	GSPO	GVIO	AVG
Recall	80.75	75.92	73.24	68.59	70.59	95.55	77.96	77.51
Precision	73.21	83.51	75.35	89.86	73.67	80.52	77.36	79.07
F-measure	76.80	79.53	74.28	77.80	72.09	87.39	77.66	77.94

MinSupp=0.016, MinConf=51.6, Var.=2.43, Disp.=244.82, AvgTransSize=2.70

Table 2e. Strategy I'_2: Pruning by keeping only verbs at distance 1 from head

	E12	GCRIM	GDIP	GJOB	GPOL	GSPO	GVIO	AVG
Recall	54.32	62.50	44.37	20.41	27.58	91.39	67.68	52.61
Precision	49.58	65.98	48.44	78.39	46.69	43.57	63.84	56.64
F-measure	51.84	64.19	46.32	32.39	34.68	59.01	65.70	50.59

MinSupp=0.019, MinConf=30, Var.=4.00, Disp.=175.41, AvgTransSize=2.01

79.84% of sentences of our corpus. The constraint is then $\phi(w) \equiv (\exists(w, \mathrm{root}(S), \mathrm{nsubj}) \in \mathrm{Dep}(S))$. Results are given on Table 2b.

Note that the slightly higher than 1 transaction size is probably due to the rare cases where there are more than one nsubj dependencies pointing to the head. The scores rise dramatically when compared to those of the strategy based only on the head of the sentence. The average F-measure (71.80%) is significantly higher than the tfidf-based performance for the same average transaction size (65.69%). *This shows that using a dependency property to select a word is a better choice than the one provided by the frequentist tfidf-based method.* Note that the case of nsubj is unique: if we take ccomp (= clausal complement) instead of nsubj, the performance falls even below the level of strategy I_0 (Table 2c).

Strategy $\mathbf{I_2}$ The third strategy considers all nouns (POS tags starting with `N`) at distance 1 from the head in the dependency graph. Such dependencies occur in 59.24% of the sentences of our corpus. This corresponds to $\phi(x) \equiv ((\exists(x, \mathrm{root}(S), d) \in \mathrm{Dep}(S)) \wedge (\mathrm{POS}(x) = \mathtt{N*}))$. Results are given on Table 2d.

The result seems better than the one of strategy I_1 (Table 2b). However, if we take transaction size into account, it is in fact merely equivalent to—and hence not better than, as it was the case for I_1—the tfidf-based method with the same transaction size. Once again we see a very high recall rate for the sports category.

Algorithm 6: Corpus hyperonymization

```
Data: A dependency-pruned corpus $\mathbf{S}'$, an hyperonymic function
      MSCH: $\mathcal{W} \to \mathcal{W}^{\mathbb{N}}$, the hyperonymic order $N$
Result: The hyperonymically extended corpus $\mathbf{S}''$
Hyperonymize($\mathbf{S}'$, MSCH, $N$):
    $\mathbf{S}'' := \emptyset$;
    for $S' \in \mathbf{S}'$ do
        $S'' := \emptyset$;
        for $w \in S'$ do
            if $\mathrm{proj}_N(\mathrm{MSCH}(w)) \neq \emptyset$ then
                $S'' := S'' \cup \{\mathrm{proj}_N(\mathrm{MSCH}(w))\}$
            else
                $S'' := S'' \cup \{w\}$
            end
        end
        $\mathbf{S}'' := \mathbf{S}'' \cup \{S''\}$
    end
end
```

One could be tempted to check the performance of taking verbs (instead of nouns) at distance 1 from the head. Indeed, verbs at that position are more frequent than nouns: they occur in 62.94% of the sentences of our corpus. Nevertheless, the results are not as good (Table 2e). This shows that despite their high frequency, verbs contain less pertinent information than nouns at the same distance from the head.

3.4 Methods based on dependencies and hyperonyms

In this section we add semantic information by the means of hyperonyms, using the hyperonymization function h (§ 1.3). The preprocessing is done by Alg. 6: $h_i(w)$ is an N-th order hyperonym of w, if it exists in WordNet. In case there is no N-th order hyperonym, the word remains unchanged. We call N the *hyperonymic factor* of our itemset transformation.

Strategy $\mathbf{II_1}$ This strategy considers hyperonymic factor $N = 1$. We thus first apply strategy I_1 and then hyperonymization h_1. Results are presented on Table 3a.

The performance is globally inferior to the one of Strategy I_1 (in which, F-measure attained 71.80%). It is interesting to note that the recall of class GJOB has decreased significantly (48.96% vs. 58.27%): in other words, using hyperonyms when dealing with labor issues results into failure to recognize 9.31% of the documents as belonging to the domain; one could say that terms used in GJOB lose their “labor specificity” already at first-order hyperonymization. On the other hand, the (already high in I_1) recall of GSPO has increased even more, compared to I_1 (from 78.76% to 82.20%): it seems that sports terminology remains in the domain even after hyperonymization, and replacing specific terms

Table 3a. Strategy II_1: I_1 followed by first-order hyperonymization

	E12	GCRIM	GDIP	GJOB	GPOL	GSPO	GVIO	AVG
Recall	72.39	56.04	71.32	48.96	59.02	82.20	70.42	65.76
Precision	66.21	75.42	64.33	82.21	67.71	73.20	70.73	71.40
F-measure	69.16	64.30	67.64	61.37	63.07	77.44	70.57	67.65

MinSupp=0.010, MinConf=44.8, Var.=1.92, Disp.=78.47, AvgTransSize=1.04

Table 3b. Strategy II_2: I_1 followed by second-order hyperonymization

	E12	GCRIM	GDIP	GJOB	GPOL	GSPO	GVIO	AVG
Recall	69.02	52.78	71.97	47.77	54.24	80.80	65.67	63.18
Precision	64.94	74.57	61.23	80.64	65.81	69.96	72.33	69.93
F-measure	66.92	61.81	66.16	60.00	59.47	74.99	68.84	65.46

MinSupp=0.008, MinConf=44.8, Var.=1.85, Disp.=70.51, AvgTransSize=1.04

by more general ones has increased their frequency as items, and hence improved recall. We have the same phenomenon with the recall of GDIP (which increased from 66.97% to 71.32%), and also slightly with the recalls of E12 and GVIO.

Strategy II_2 This strategy is similar to strategy II_1 but uses hyperonymic factor $N = 2$. Results are presented on Table 3b.

The performance is globally inferior to the one of II_1 (where we used first-order hyperonyms), with two minor exceptions: the recall of GDIP that increased by 0.65% and the precision of GVIO that increased by 1.6%. What is noteworthy however, is the fact that the recalls of GDIP and GSPO are still higher than the ones of strategy I_1 (no hyperonyms).

To better understand the behavior of the system when climbing the hyperonymic chain by replacing words by hyperonyms of increasingly higher order (and returning to the original word when there are no hyperonyms left) we calculated the performance for N-th order hyperonyms for $1 \leq N \leq 12$. Note that when $N > 12$ the amount of remaining hyperonyms is negligible and the strategy is similar to strategy I_1 (no hyperonyms). On Fig. 2, the reader can see the evolution of recall (black), precision (red) and F-measure (blue) for the average of all class, and then specifically for GSPO and for GDIP. Dashed lines represent the recall, precision and F-measure of strategy I_1.

In the average case, the effect of hyperonymization of orders 1–4 is to decrease performance. After $N = 5$, the global number of hyperonyms available in WordNet rapidly decreases so that the situation gradually returns to the one of I_1 (no hyperonyms) and we see curves asymptotically converging to I_1 lines from underneath.

Not so for GSPO, the GSPO recall curve of which is above the I_1 value for most N ($N = 1$, 2, 6–8 and 10–12).

The phenomenon is even better illustrated in the case of GDIP: as the reader can see on the figure, the *complete GDIP recall curve is located above the I_1 one.* It seems that in these two cases (GDIP and, to a lesser extent, GSPO),

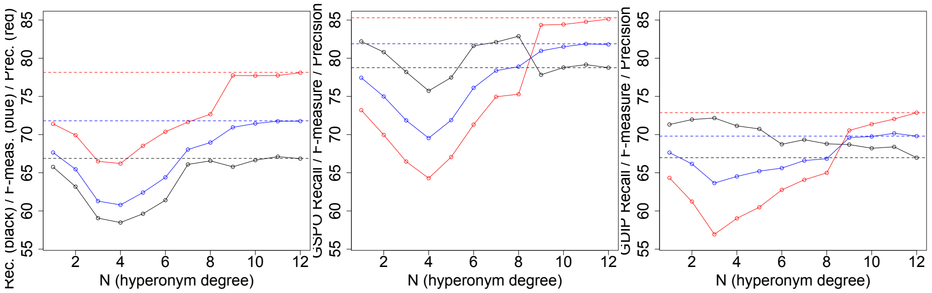


Fig. 2. F-1 measure for hyperonymization of orders $1 \leq N \leq 12$: the average case, class GSPO, class GDIP

hyperonyms of all orders have a positive impact on the classifier. Unfortunately this impact only concerns recall and is compensated by bad precision, so that F-measure is still inferior to the I_1 case.

4 Conclusion and future work

In this paper we have investigated the use of association rules for text classification by applying two new techniques: (a) we reduce the number of word features through the use of morphosyntactic criteria in the framework of dependency syntax; for that we keep words dependent from the head by specific dependencies and/or having specific POS tags (b) we replace words by their hyperonyms of different orders, which we have calculated out of WordNet using frequencies and, in some cases, disambiguation. We have obtained positive results for case (a), in particular when we compare dependency-based single-item rules with tfidf-based ones. In case (b) the results we share in this paper are less efficient but still interesting, especially we found classes for which hyperonymization significantly improves recall.

This work opens several perspectives, among which:

— examine why these particular classes are favorable to hyperonymization, whether this is related to the structure of WordNet or to linguistic properties of the domain;

— explore *partial hyperonymization* i.e., is it possible to hyperonymize only specific items according to the needs of the classifier?[1] How do we choose, on the word level, if we should rather keep the original word (to increase precision) or switch to some hyperonym (to increase recall)?

— we have used only recall and precision as quality measures of our rules, and our evaluation is strongly dependent on these measures since the selection of the 1,000 rules we keep is entirely based upon them. There are other quality

[1] Indeed, by hyperonymizing all words one wins on one side and loses on the other: for example “dalmatian” and “poodle” will both be replaced by “dog”, but “dog” occurrences will be replaced by “canid”. It would be more preferable to keep the word “dog” in the second case, so that we have a real increase in frequency.

measures available, how do they apply and how can they be compared and combined? How robust are the results?

— and finally: how can we optimize the distinctive feature of association rules, namely the fact of being intelligible by the user? How can the user's experience (and linguistic knowledge) be incorporated in the enhancement of rules to obtain the best possible result from his/her point of view?

References

1. Reuters corpus, volume 1, english language, 1996-08-20 to 1997-08-19, `http://about.reuters.com/researchandstandards/corpus/statistics/index.asp`
2. British National Corpus (1994), `http://www.natcorp.ox.ac.uk`
3. Aggarwal, C.C., Zhai, C.: A Survey of Text Classification Algorithms, chap. 6, pp. 163–222. Mining Text Data, Springer
4. Ahonen, H., Heinonen, O., Klemettinen, M., Verkamo, A.I.: Applying data mining techniques in text analysis. TR C-1997-23, Department of Computer Science, University of Helsinki (1997)
5. Borgelt, C.: Efficient implementations of apriori and eclat. In: Workshop of Frequent Item Set Mining Implementations (FIMI 2003), Melbourne, FL, USA (2003)
6. Ferrer i Cancho, R., Solé, R.V., Köhler, R.: Patterns in syntactic dependency networks. Physical Review E 69, 1–8 (2004)
7. Cherfi, H., Napoli, A., Toussaint, Y.: A Conformity Measure Using Background Knowledge for Association Rules: Application to Text Mining, pp. 100–115. IGI Global (2009)
8. Chomsky, N.: Syntactic structures. Mouton (1957)
9. Cohen, W.W.: Learning rules that classify e-mail. In: AAAI Spring Symposium on ML and IR (1996)
10. Curran, J.R., Moens, M.: Scaling context space. In: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics. pp. 231–238 (2002)
11. Do, T.N.Q.: A graph model for text analysis and text mining. Master Thesis, Université de Lorraine, (2012)
12. Jaillet, S., Laurent, A., Teisseire, M.: Sequential patterns for text categorization. Intell. Data Anal. 10(3), 199–214 (2006)
13. Kovacs, L., Baksa-Varga, E.: Dependency-based mapping between symbolic language and Extended Conceptual Graph. In: 6th International Symposium on Intelligent Systems and Informatics. pp. 1–6 (2008)
14. Lang, K.: Newsweeder: Learning to filter netnews. In: International Conference on Machine Learning. pp. 331–339. Morgan Kaufmann (1995)
15. Liu, B., Hsu, W., Ma, Y.: Integrating classification and association rule mining. In: Proc. of the Int. Conf. on Knowledge Discovery and Data Mining (New York). pp. 80–86 (1998)
16. Lowe, W.: Towards a theory of semantic space. In: Proceedings of the Twenty-Third Annual Conference of the Cognitive Science Society. pp. 576–581 (2001)
17. de Marneffe, M.C., MacCartney, B., Manning, C.D.: Generating typed dependency parses from phrase structure parses. In: LREC 2006. pp. 449–454 (2006)
18. Mehran Sahami, Susan Dumais, D.H., Horvitz, E.: A bayesian approach to filtering junk email. In: AAAI Workshop on Learning for Text Categorization. AAAI Technical Report WS-98-05 (1998)

19. Mel′čuk, I.A.: Dependency syntax : theory and practice. Albany: State University Press of New York (1987)
20. Miller, G.A.: WordNet: A lexical database for English. Communications of the ACM 38(11), 39–41 (1995)
21. Nivre, J.: Dependency Grammar and Dependency Parsing. MSI report 05133. School of Mathematics and Systems Engineering, Växjö University, (2005)
22. Ordoñez-Salinas, S., Gelbukh, A.: Information Retrieval with a Simplified Conceptual Graph-Like Representation 6437(Chapter 9), 92–104 (2010)
23. Padó, S., Lapata, M.: Dependency-based construction of semantic space models. Computational Linguistics 33(2), 161–199 (2007)
24. Pedersen, T.: Information content measures of semantic similarity perform better without sense-tagged text. In: Proceedings of the 11th Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL HLT 2010). pp. 329–332 (2010)
25. Roche, M., Azé, J., Matte-Tailliez, O., Kodratoff, Y.: Mining texts by association rules discovery in a technical corpus. In: Intelligent Information Processing and Web Mining. pp. 89–98. Advances in Soft Computing, Springer Verlag (2004)
26. Schmid, H.: Probabilistic part-of-speech tagging using decision trees. In: Proceedings of International Conference on New Methods in Language Processing, Manchester, UK (1994)
27. Sebastiani, F.: Machine learning in automated text categorization. ACM Comput. Surv. 34(1), 1–47 (2002)
28. Tesnière, L.: Éléments de syntaxe structurale. Klincksieck (1959)
29. Zaïane, O.R., Antonie, M.L.: Classifying text documents by associating terms with text categories. In: Australasian Database Conference. CRPIT, vol. 5. Australian Computer Society (2002)

Learning Subgraph Patterns from text for Extracting *Disease–Symptom* Relationships

Mohsen Hassan, Adrien Coulet, and Yannick Toussaint

LORIA (CNRS, Inria NGE, Université de Lorraine),
Campus scientifique, Vandoeuvre-lès-Nancy, F-54506, France
{mohsen.sayed,adrien.coulet,yannick.toussaint}@loria.fr

Abstract. To some extent, texts can be represented in the form of graphs, such as *dependency graphs* in which nodes represent words and edges represent grammatical dependencies between words. Graph representation of texts is an interesting alternative to string representation because it provides an additional level of abstraction over the syntax that is sometime easier to compute. In this paper, we study the use of graph mining methods on texts represented as dependency graphs, for extracting relationships between pairs of annotated entities. We propose a three step approach that includes *(1)* the transformation of texts in a collection of dependency graphs; *(2)* the selection of frequent subgraphs, named hereafter *patterns*, on the basis of positive sentences; and *(3)* the extraction of relationships by searching for occurrences of patterns in novel sentences. Our method has been experimented by extracting *disease–symptom* relationships from a corpus of 51,292 PubMed abstracts (428,491 sentences) related to 50 rare diseases. The extraction of correct *disease–symptom* relationships has been evaluated on 565 sentences, showing a precision of 0.91 and a recall of 0.49 (F-Meaure is 0.63). These preliminary experiments show the feasibility of extracting good quality relationships using frequent subgraph mining.

1 Introduction

In many domains such as biomedical research, text is a major source of information; unfortunately text corpora are frequently too large to be fully considered manually [1]. We focus here on the task of Relation Extraction (RE), which consists in identifying and qualifying valid relationships between entities already recognized in the text. Figure 1 illustrates the process of RE with an example of relation between a disease and a symptom. First, Named Entity Recognition (NER) identifies the interesting entities in the text and annotate them with the corrected category. Second step identifies if named entities are involved in a relationship (and may qualify the type of the relationship) or not.

Texts may be represented at different levels: words, bag of words, sequences of words, syntactic trees, graphs (dependency graphs); and they may be enriched by some linguistic features: part of speech, syntactic or semantic features. In this paper we study how text, represented in the form of graphs, can be processed with simple graph mining methods, to perform RE.

In: P. Cellier, T. Charnois, A. Hotho, S. Matwin, M.-F. Moens, Y. Toussaint (Eds.): Proceedings of DMNLP, Workshop at ECML/PKDD, Nancy, France, 2014.

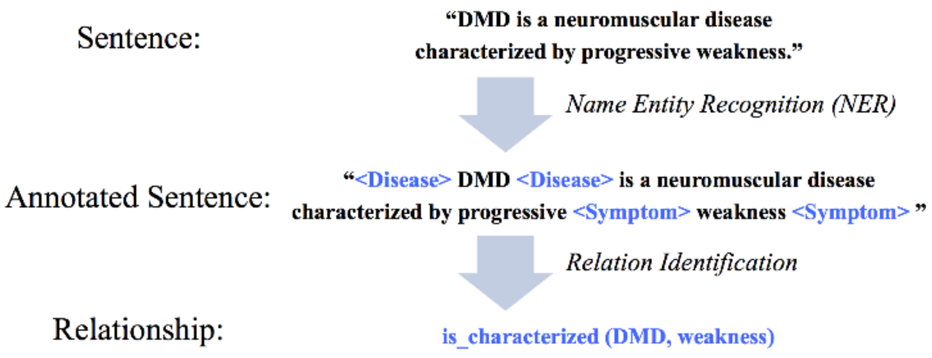

Sentence:
"DMD is a neuromuscular disease characterized by progressive weakness."
Name Entity Recognition (NER)
Annotated Sentence:
"<Disease> DMD <Disease> is a neuromuscular disease characterized by progressive <Symptom> weakness <Symptom> "
Relation Identification
Relationship:
is_characterized (DMD, weakness)

Fig. 1: The process of Relation Extraction (RE)

Frequent Subgraph Mining (FSM) is a graph mining method that extracts frequently occurring subgraphs either from a single graph or a set of graphs [2]. We propose in this paper to extract relationships from text through a three step method, based on FSM. The first step concerns data preparation and consists in transforming texts into graphs and recognizing name entities. The second step relies on the identification of labeled and oriented subgraphs, named hereafter *patterns*, that are connecting frequently two imposed typed entities, *e.g.*, subgraphs connecting one disease to one of its symptom. The third step uses generated patterns for extracting relationships between these entities.

The paper is organized as follows: Section 2 presents background elements regarding graph mining. Section 3 introduces our three step method. Section 4 reports experiments of our method on the extraction of *disease–symptom* relationships. Section 5 presents related works and Section 6 discusses the interest of using graph mining for RE.

2 Graph Mining

A graph is defined as a pair $G = (V, E)$ where V is a set of vertices (or nodes) and E is a set of edges connecting vertices such as $E \subseteq V \times V$. A graph is a *directed graph* when edges are oriented pairs of vertices. A graph is a *labeled graph* when vertices and edges are associated with labels.

2.1 Frequent Subgraph Mining

$S = (SV, SE)$ is a subgraph of G if $SV \subseteq V$ and $SE \subseteq E$. Given a graph collection $\mathcal{G} = \{G_1, G_2, ..., G_k\}$, with $G_i = (V_i, G_i)$, and a minimum support *min_sup*, the Frequent Subgraph Mining task (denoted FSM) extracts the collection of subgraphs $\mathcal{S} = \{S_1, ..., S_n\}$, with $S_i = (SV_i, SG_i)$ that occur in $\mathcal{G}$ with a support greater than *min_sup*. The support of a subgraph S_i is the number of its occurrences in $\mathcal{G}$[1].

[1] The relative support of S_i is $\frac{|S_i|}{|\mathcal{G}|}$

FSM algorithms are mainly based on two distinct approaches: *Apriori*-based and *pattern growth*-based approaches. *Apriori*-based graph mining algorithms share similarities with Apriori-based frequent itemset mining algorithms [3]. In their case, the search for frequent subgraphs starts with graphs with no edge. At each iteration, the size of the newly discovered frequent substructures is increased by one by joining two subgraphs from the previous iteration. AGM, FSG and FFSM are examples of Apriori-based algorithms [2,4,5]. The *pattern-growth* mining algorithms extend a frequent graph by trying to add successively a new edge to every possible position. If the new graph is frequent, a new frequent graph can be expended; if it is not frequent a new edge is tried to be added. gSpan [6], CloseGraph [7]and Gaston [8] are examples of pattern-growth algorithms.

2.2 gSpan

gSpan is a FSM algorithm that processes undirected labeled graphs. Given a collection of such graphs, gSpan returns the set of frequent subgraphs and their support. To generate this result, gSpan generates a Tree Search Space (TSS) that is composed of all trees and subtrees that rely in the collection of graphs. gSpan represents each tree of the TSS using a specific encoding, named *minimum Depth-First Search (DFS) Code*. This code is unique for each tree because it is constructed following the unique DFS traversal that follows the lexicographic order of vertex labels.

gSpan follows a pattern-growth mining approach, *i.e.,* expends at each iteration a frequent graph with a new edge, trying every potential position. An issue with this approach is that the same graph can be discovered several times from different frequent graphs. gSpan avoids this problem by introducing a *right-most extension technique*, where edge extensions only takes place on a specific position determined by DFS Codes.

3 Relationship Extraction using Frequent Subgraph Mining

We propose an original method based on FSM to extract relationships from text. Figure 2 depicts an overview of this three step method. Each step is detailed in next subsections.

3.1 Data Preparation

This step aims at transforming a collection of texts into a collection of Dependency Graphs (DG). To achieve this, texts are submitted to the following tasks: Sentence Splitting, NER and Named Entity (NE) Substitution, Sentence Filtering, Dependency Parsing and lemmatization. First, texts are split into sentences. Then, NEs are recognized. We focused on relation between diseases and symptoms. Thus, we replaced each occurrence of these entities by the corresponding generic word “DISEASE” or “SYMPTOM”. Sentences are filtered to keep those

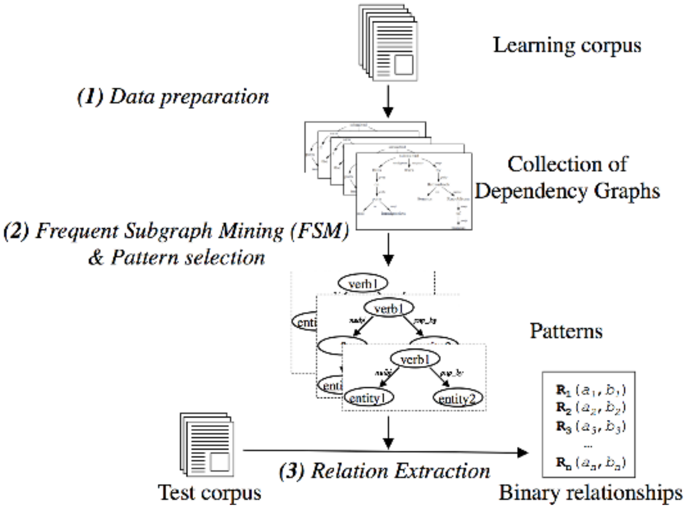


Fig. 2: Overview of our Relation Extraction (RE) method

involving at least two entities of interest. Dependency parsing produces for each sentence one labeled directed graph, named DG. Such DG is made of vertices that represent words and edges that are grammatical dependencies between words. Figure 3 shows the dependency graph of the sentence "*DMD is a neuromuscular disease characterized by progressive weakness.*".

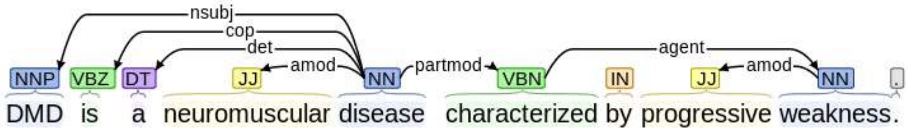

nsubj
cop
det
amod
partmod
agent
amod
NNP
VBZ
DT
JJ
NN
VBN
IN
JJ
NN
.
DMD is a neuromuscular disease characterized by progressive weakness.

Fig. 3: Example of Dependency Graph (DG) processed by the Stanford Parser and drawn with Brat

Finally, words in DG are replaced by their lemmas, by a more general form that is more likely to appear in other graphs. Figure 4 shows an example of DG resulting from the data preparation step.

The collection of formatted DG is the input to FSM for mining the most frequent subgraph patterns that preserve the relations between two named entities.

3.2 Pattern Extraction

Frequent Subgraph Mining (FSM) aims at extracting useful patterns for the relation extraction process. Given a set of DGs and the support threshold, gSpan

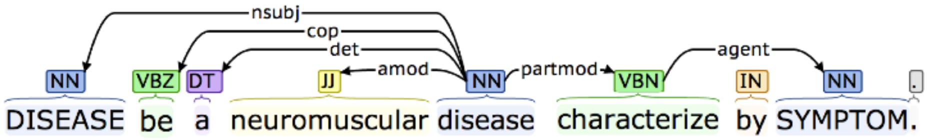


Fig. 4: Example of Dependency Graph after replacement of recognized entities (diseases and symptoms) by generic words (DISEASE and SYMPTOM) and lemmatization

extracts an undirected subgraph patterns. These patterns give the relationships between interesting annotated entities. Figure 5 shows an example of such pattern, extracted from graph in Figure 4. This subgraph pattern gives the relation based on grammatical dependencies between the disease "DMD" and the symptom "weakness".

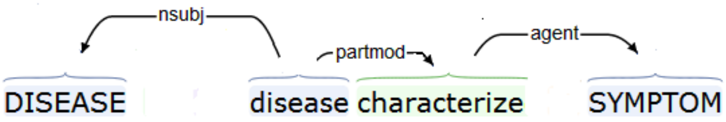


Fig. 5: Example of extracted pattern

Then, the patterns that contain the following are excluded: *(1)* conj_and or conj_or dependency relation between any two nodes; *(2)* The dependency path for DISEASE is equal to the dependency path for SYMPTOM, this means that DISEASE and SYMPTOM have the same semantic role in the sentence and this might be an error from NER; *(3)* no node for DISEASE or SYMPTOM (at least one disease and one symptom must be exist). Figure 6 shows an example of such excluded patterns. These patterns can be discovered from a sentence like "this disease is characterized by DISEASE and SYMPTOM"[2].

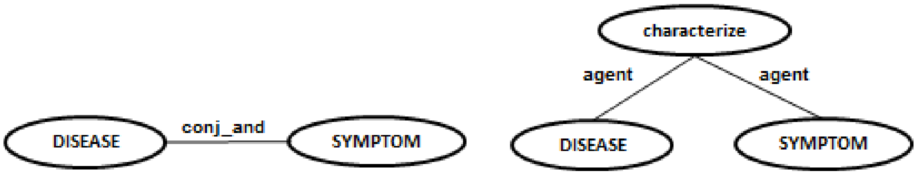


Fig. 6: Two examples of excluded patterns

[2] The uppercase words are the generic words for NEs

Bunescu and Mooney proposed a kernel method that used the shortest path between the two entities in the undirected version of the dependency graph [9]. We proposed similarly to compute the shortest path, but from directed dependency graph, which is useful for expressing the direction of relation between entities and consequently gives more precise relations. Two paths with the same sequence of syntactic dependency labels are similar if the direction of the syntactic dependencies are the same. Hence, the shortest path method (SPM) for extracting a smaller set of patterns than gSpan patterns has been used [9]. It consists in extracting the shortest path between two entities (*e.g.,* disease and symptom) in a dependency graph. Bunescu and Mooney used words and POS for expressing their pattern, but in SPM we consider the whole subgraph.

Figure 7 shows the shortest path between the two entities DISEASE and SYMPTOM in the dependency graph.

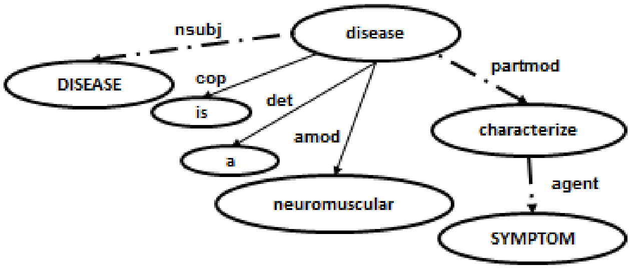


Fig. 7: The shortest path between DISEASE and SYMPTOM

Given the following two annotated sentences “DISEASE is a disease characterized by SYMPTOM” and “DISEASE is anomaly accompanied by SYMPTOM”. First, SPM get the graph of each sentence as shown in figure 8. Then, SPM compute the common shortest path from the graphs of the two sentences. If the values of the nodes in the pattern are different, their values are replaced by “*” and keeping a list of all possible values for each node. Hence, two graphs patterns can be merged and represented in one generalized pattern [3]. The support of the new generalized pattern is less than or equal the sum of the supports of the two patterns. The support of the generalized pattern is automatically computed when executing the generalization operation which makes the process run faster.

Figure 9 shows two examples of pattern, one resulting from SPM patterns and the other resulting from gSpan. SPM method checks every node in the subgraph pattern to contain all possible values. This makes the pattern more general than gSpan pattern and increases the frequency value of the pattern. This has two advantages: first it produces a smaller set of patterns than gSpan patterns which is easier for analysis and evaluation purposes; second it leads to

[3] There is no redundancy because all redundant patterns are merged into one pattern

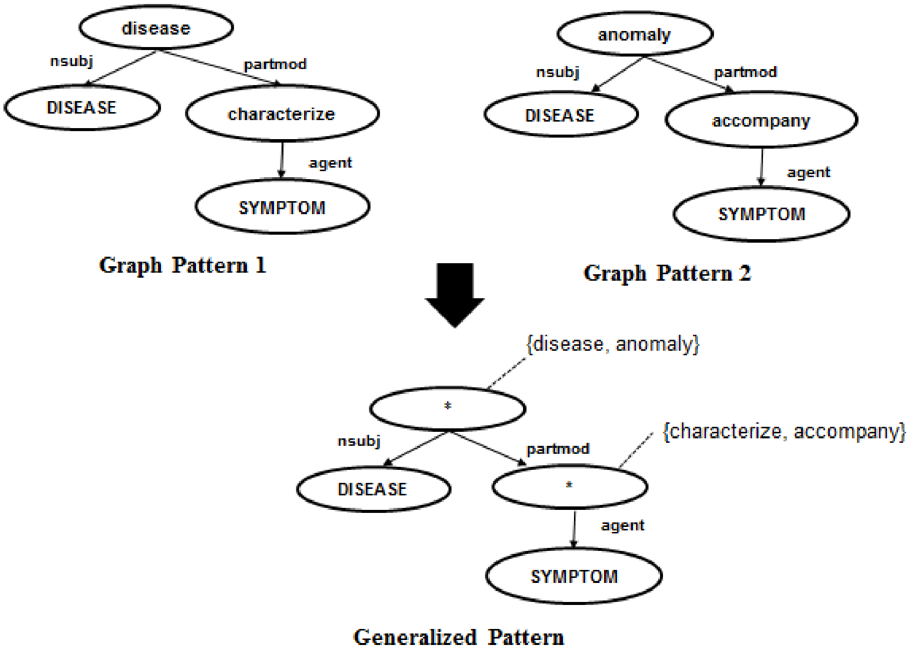


Fig. 8: Example of our method pattern

a higher coverage than gSpan when the pattern may not be extracted by gSpan because of its low frequency. On the other side, SPM did not use POS tags as a single feature as proposed in [9], what makes the pattern more generic and increases the coverage of patterns but induces a lower precision.

When SPM extends every node with all possible values, some values don't represent a correct relation between the annotated entities. Figure 10 shows rejected patterns that should be removed by the generalization operation to increase the quality of patterns.

The extracted patterns are classified into two classes: positive and negative patterns. The classification is based on pattern support and quality. The quality Q of a pattern is computed by the following formula

$$Q = \frac{T}{S} \tag{1}$$

where T is the number of all correct sentences in the pattern extension and S is the support of the pattern. A sentence is correct if it contains the pattern and the relation identified by the pattern is correct. For example, the pattern in Figure 8 has support 23. This means that the number of sentences that contain this pattern is 23. All disease-symptom relationships provided in these 23 sentences are correct. Then, T=23 and Q=23/23. Hence, the quality of this pattern is 1.

The pattern is a positive pattern if its support is higher than a minimum support (min_sup) threshold and its quality is higher than a minimum quality

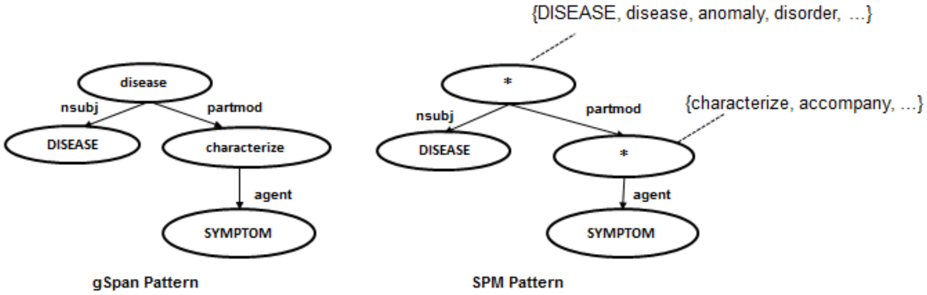


Fig. 9: Examples of patterns generated from gSpan and SPM

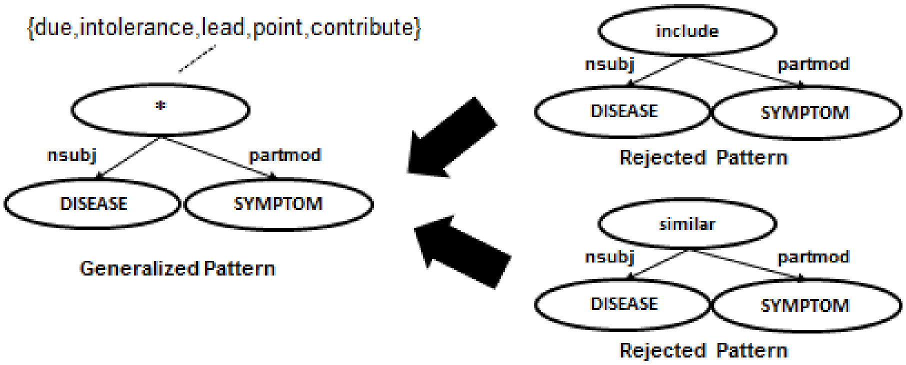


Fig. 10: Example of rejected patterns

threshold. Only positive patterns are considered for extracting new relationships from a new corpus.

3.3 Relationship Extraction using Patterns

Positive patterns previously selected are used to discover new relationships between entities mentioned in a new corpus. Similarly to the learning process, a set of dependency graphs are generated from the new corpus exactly as the data preparation step (sentences splitting and NER are also required before dependency parsing). Then, a pattern matching for the selected patterns with the dependency graphs is done to extract the binary relationships between the interesting entities. A value that expresses the quality of each new extracted relation is also returned (accordingly to the quality of the pattern used in the extraction process).

4 Experiment

We build-up experiments on the basis of a medical corpus related to rare diseases. This corpus is explored to extract relationships between diseases and symptoms. Figure 11 presents the process of our experiments and its evaluation. Details are provided in the following subsections.

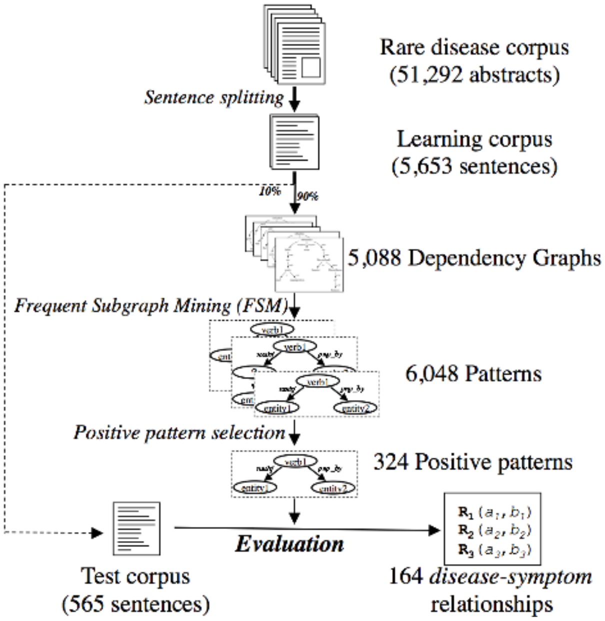


Fig. 11: Overview of our experiment for the extraction of disease-symptom relationships

4.1 Rare Disease Corpus

Our rare disease corpus is composed of 51,292 PubMed abstracts related to 50 Orphanet[4] rare diseases[5]. Abstracts are obtained by querying manually PubMed, using its web user interface. The query submitted to PubMed has the following form: “($disease_{1,pref\_name}$ `or` $disease_{1,syn_1}$ `or...or` $disease_{1,syn_n}$) `or...` `or` ($disease_{k,pref\_name}$ `or` $disease_{k,syn_1}$ `or...or` $disease_{k,syn_m}$)”

where $disease_{i,pref\_name}$ and $disease_{i,syn_j}$ are respectively referring to the preferred name and the j^{th} synonym of disease i according to the Orphanet Rare Disease Ontology[6].

[4] `http://www.orpha.net`

[5] The 50 diseases are listed at: `http://www.loria.fr/~msayed/50RareDiseases`

[6] `http://www.bioportal.bioontology.org/ontologies/ORDO`

4.2 Building a Dependency Graph Dataset

51,292 abstracts are split in 428,941 sentences using LingPipe[7], and subsequently submitted to disease and symptom NER. We use MetaMap to annotate each sentence of the corpus using UMLS semantic types "Disease or Syndrome" and "Sign or Symptom" [10]. MetaMap annotations of some very general words like "disorder" or "symptoms" have been removed to avoid noise in the rest of the process. Annotated sentences are divided into a *learning corpus*, made of 90% of sentences randomly selected, and a *test corpus*, made of the 10% left. In the learning corpus, each recognized disease or symptom is replaced by the generic string DISEASE or SYMPTOM (which are indexed when several disease or symptom are recognized in one sentence). Sentences that does not contain at least one disease and one symptom are filtered out, what reduces their number to 5,653.

The Stanford Parser is used to build the Dependency Graph (DG) of each sentence [11]. It is set to collapsed dependencies with propagation of conjunct dependencies option. As a result, conjunctions are propagating the dependencies that involves the conjuncts. For example, in the sentence "DISEASE is characterized by SYMPTOM1 and SYMPTOM2" this option guarantees that the same pattern will be observed between DISEASE and SYMPTOM1, and between DISEASE and SYMPTOM2. Finally, lemmatization is achieved using the Stanford CoreNLP suite.

4.3 Frequent Subgraph Mining

From prepared dependency graphs, gSpan extracts all frequent subgraphs. Those are filtered to keep only subgraphs that contain one disease and one symptom. This guarantees that only patterns that describe the dependency relation between disease and symptom are kept. We applied our program to gSpan subgraphs to identify in each case the shortest path between the nodes DISEASE and SYMPTOM. When several diseases or symptoms are in a unique sentence, one shortest path is computed for each pair (DISEASEx–SYMTOMy). This resulted in 6,048 subgraph patterns, a smaller set compared to gSpan result, consequently easier to evaluate. Because we think that shortest paths represent the most significance part of subgraph, we focused on these reduced graphs.

4.4 Selection of Positive Patterns

First, patterns with a $frequency \geq 2$ are selected from the 6,048. Accordingly, 615 patterns are frequent, covering 2,535 sentences from all 5,653 (44.84%). Second, patterns with a *pattern quality* ≥ 0.5, our quality threshold, are selected (see formula 1). It results 324 patterns (that cover 1,329 sentences or 23.51%), which are considered as *positive patterns* and are aiming at extracting new relationships from text.

[7] http://alias-i.com/lingpipe

4.5 Evaluation

Finally, we evaluate the ability of positive patterns to identify disease-symptom relationships in the test corpus. The evaluation process can be divided in three tasks. *(i)* For each sentence in the test corpus, two lists of disease-symptom pairs are composed: the list *i-a* of all possible disease-symptom pairs found in the sentence; the list *i-b* of pairs extracted by our method, *i.e.,* when a positive pattern matches the DG of a test sentence. Obviously, list *(i-b)* is a subset of *(i-a)*. *(ii)* Each pair of list *i-a* is marked manually as Correct if it corresponds to a relation actually mentioned in the sentence, or Incorrect, if it is not. *(iii)* Pairs that are marked as Correct and are extracted by our method (*i.e.,* in list *i-b*) are True Positive (TP); pairs that are marked as Incorrect and are not extracted by our method are True Negative (TN); pairs that are marked as Incorrect and are extracted by our method (*i.e.,* in list *i-b*) are False Positive (FP); pairs that are marked as Correct and are not extracted by our method are False Negative (FN).

Table 1: Confusion matrix corresponding to the evaluation of RE method on a corpus of 565 sentences. Because several relationships can be extracted from one sentence, TP+TN+FP+FN is higher than the number of sentences.

		Pattern-based extraction	
		Extracted	Not extracted
Manual extraction	Correct	TP=149	FN=172
	Incorrect	FP=15	TN=441

Table 1 shows the confusion matrix resulting form the evaluation process. It enables to compute the precision (P)[8] , recall (R)[9], F-measure [10], accuracy (Acc) [11] and specificity (S)[12]. Evaluation shows high precision (0.90) and specificity (0.96); reasonable F-measure (0.76) and accuracy (0.75); and a low recall (0.46).

5 Related Works

5.1 Mining Text as Set of words

Co-occurrence is the simplest method to identify relationships between two entities that co-occur in the same sentence. This approach is based on the hypothesis that if two entities are mentioned frequently together, it is likely that these

[8] P=TP/(TP+FP)

[9] R=TP/(TP+FN

[10] F-measure=2*P*R/(P+R)

[11] Acc=(TP+TN)/(P+N)

[12] S=TN/(FP+TN)

two entities are related. Co-occurrence methods have been successfully applied to the automated construction of networks of biomolecules such as protein-protein or gene-disease networks [12,13]. Co-occurrence approach tends to achieve a good recall but low precision. This can be balanced when one is mining very large corpus. Another issue with such approaches is that the type of relationships and their direction are unknown.

Bags of words are artificial constructs where one textual document is represented as an unordered set of the words it contains, *i.e.*, the *bag.* In this set, each word is usually associated with its frequency of occurrence in the document, then enabling to weight words within the bag. This is used to classify documents with similar words and words frequency profiles. Indeed, when associated with a proper dictionary, a document represented as a bag can be encoded as a simple vector of integers. This is a compact representation that enables to work with large corpora of documents. It suffers from low precision.

Sequence of words It consists of a partial order (*i.e.,* the sequence) of words, POS tags, general POS tags, entity or chunk type, etc. These features are used to build patterns or rules that assert a relationships between entities. Blohm *et al.* presented method based on a taxonomic sequential pattern for RE which extends a sequential mining algorithm to take into account a taxonomy of morpho-syntactic and lexico-semantic features [14]. It allows generalization or specialization among patterns, which affects the precision and the recall of the patterns. Quiniou *et al.* studied how to use the sequence mining to identify more generic linguistic patterns and show that sequence mining is more powerful than n-grams to express the linguistic patterns [15]. Béchet *et al.* provided a sequential pattern mining algorithm which discover the relations between genes and rare diseases in biomedical corpus [16]. The proposed algorithm extracts expressive linguistics patterns more efficient than patterns extracted with itemsets. Sequence mining tends to generate a very high number of patterns what makes difficult the analysis and evaluation tasks. Consequently filter are usually applied to reduce the number of extracted patterns.

5.2 Mining Trees

Parse Tree is an ordered, rooted tree that represents the syntactic structure of a sentence. Some works have been proposed to use such syntactic structure for extracting relations between entities. Galitsky introduced the operation of the syntactic generalization which take a pair of syntactic parse trees and find the maximal common subtrees [17]. Galitsky employed the nearest neighbour learning method to find the maximal common subtrees. Zhang *et al.* proposed a kernel approach that uses the syntactic tree representation of sentences for RE. They studied how to capture the syntactic structure by using a convolution tree kernel and support vector machines [18]. Zelenko *et al.* also proposed a tree kernel method, but using shallow parse tree representations [19]. The same tree kernel approach has been used by Culotta and Sorensen, but allowed feature weighting and used additional features such as Wordnet, POS, entity types [20]. In both approaches, a relation instance is defined by the smallest subtree in the parse or dependency tree that includes interesting entities. The tree kernel approaches achieve good results but they are hard to implement and computationally complex. Note that trees are specific type of graphs and mining trees can be easily adapted to graphs.

5.3 Mining Graphs

Many RE methods based on DG have been proposed [21,22]. Chowdhury et Lavelli proposed a hybrid kernel approach, which is based on different features: dependency patterns, regex patterns, path enclosed and shallow linguistic kernels [23]. In this case, dependency patterns are reduced graphs, which are subgraphs from dependency graphs. The reduced graph extends the shortest path (smallest common subgraph) of the dependency by adding *(a)* dependent nodes (when exist) of nodes in the shortest path; *(b)* the immediate governor(s) (when exist) of the least common governor. For sake of simplicity, we choose in this paper to consider only the shortest path with no dependents. Bunescu *et al.* proposed a RE approach similarly based on the shortest path between two entities in undirected dependency graphs [9].

Chowdhury et Lavelli also proposed to use a reduced graph pattern that is a set of syntactic dependences of the corresponding reduced graph. For example, the reduced graph pattern of the graph represented Figure 7 is $\langle nsuj, cop, det, jj, partmod, agent\rangle$. Note that, in this case, reduced graph patterns are undirected.

Adolphs *et al.* developed a rule learning algorithm to learn graph rules which identify subgraphs in arbitrary graphs [24]. First, subgraphs are extracted. Then, subgraph generalization is done to form rules by underspecifying the nodes and introducing place-holders labeled with the role for the argument nodes.

6 Discussion and Conclusion

6.1 Discussion

The proposed hybrid kernel approach of Chowdhury et Lavelli [23] is evaluated on 5 different corpora for the extraction of the protein-protein relationship and the results varied from corpus to another. Considering pattern features and corpora used, our method shows a good precision (0.91) and low recall (0.49). This illustrates that graph mining can produce precise patterns for RE but additional work is required, such as adding features (*e.g.*, similar to those proposed in Chowdhury's work). Béchet *et al.* [16] use sequential mining patterns for extracting gene-disease relationships. The method gives the best precision 0.68 (recall 0.36) when using min_sup = 50 while the best recall is 0.65 (precision is 0.66) when using min_sup = 5. While in our method we achieve the best precision

0.94 (recall 0.33) and the best recall 0.67 (precision 0.41). In addition, the huge number of patterns produced by sequence mining; makes the interpretation task hard.

In our experiments, we fixed min_sup=2. When min_sup=2, the number of extracted patterns is 615. When increasing the min_sup threshold, the number of extracted patterns and recall decrease. For example, if min_sup=3, then the number of extracted patterns is 268. When decreasing the min_sup threshold, the number of extracted patterns and recall increase. For example, If min_sup=1, then the number of extracted patterns is 6048. This number of patterns is large for analysis and patterns with support=1 may be not important because they are rare patterns.

Figure 12 shows the relation between the precision and pattern quality threshold and between recall and pattern quality threshold. Precision increases and recall decreases when the pattern quality threshold increases. The best precision value is 0.94 when the quality threshold is 100 and the best recall value is 0.67 when the quality threshold is 0. The trade-off between the precision and recall is required according to the purpose of the application.

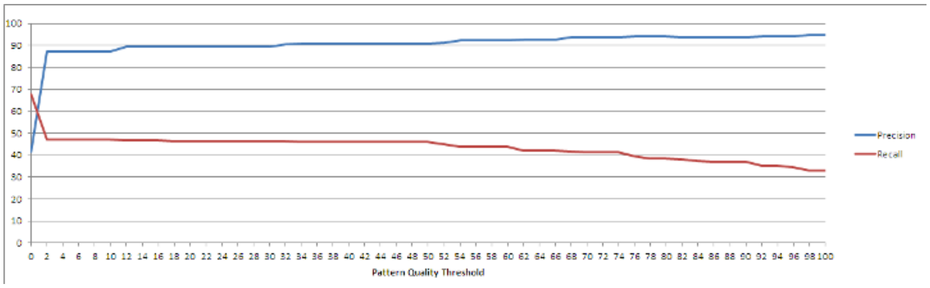


Fig. 12: The relation between precision and recall with the quality threshold

The study of FN and FP relations is necessary for improving the recall and precision respectively. In the following sentence "In areas in which transmission is occurring, WNV infection should be considered in patients with acute flaccid paralysis.", the relation between disease "WNV" and symptom "flaccid paralysis" is marked as FN relation (because we didn't generate a positive pattern that describes this relation). A possible solution for this problem is to consider patterns with low frequency (rare patterns), another solution is to enlarge the learning corpus. These produce a larger patterns set which reduces FN relations and increases the recall.

On the other side, to increase the precision, the number of FP relations needs to be reduced. The following sentence "Muscular dystrophy is a nosology for a group of hereditary muscle disorders characterized by progressive wasting and weakness of skeletal muscle, where degeneration of muscle fibers is detected by pathological examination" generates a FP relation between "hereditary muscle" and “weakness”. One solution is to consider only patterns with high quality by increasing the quality threshold to ensure that the extracted patterns are precise enough.

Finally, Unlike gSpan and Chowdhury’s work, SPM doesn’t able to keep other features such as negation relation. issues like this must be token in consideration for further improvements and extensions to SPM.

6.2 Conclusion

This paper illustrates how graph mining can be used for RE. We propose a simple method based on FSM to extract relationship from a corpus of text represented as DGs. FSM enables to identify subgraph patterns that are filtered based on their frequency and quality. Selected patterns are in turn used to extract relationships form novel sentences. Our evaluation on a corpus related to rare diseases showed a very high precision of 0.91.

In the future, the recall of the FSM-based method may be enhanced by improving its ability to identify FN relations. Also, thanks of the readability of the extracted patterns, studying and adding new features or constraints to improve the quality of these patterns is possible and may increase the recall and precision values. Combining features of sequences, syntax trees and dependency graphs may introduce more precise patterns with higher recall.

References

1. Larsen, P.O., von Ins, M.: The rate of growth in scientific publication and the decline in coverage provided by science citation index. Scientometrics (2010) 575–603
2. Kuramochi, M., Karypis, G.: Frequent subgraph discovery. In: Proceedings of the 2001 IEEE International Conference on Data Mining. ICDM ’01, Washington, DC, USA, IEEE Computer Society (2001) 313–320
3. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: Proceedings of the 20th International Conference on Very Large Data Bases. VLDB ’94, San Francisco, CA, USA, Morgan Kaufmann Publishers Inc. (1994) 487–499
4. Inokuchi, A., Washio, T., Motoda, H.: An apriori-based algorithm for mining frequent substructures from graph data. In: Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery. PKDD ’00, London, UK, UK, Springer-Verlag (2000) 13–23
5. Huan, J., Wang, W., Prins, J.: Efficient mining of frequent subgraphs in the presence of isomorphism. In: Proceedings of the Third IEEE International Conference on Data Mining. ICDM ’03, Washington, DC, USA, IEEE Computer Society (2003) 549–
6. Yan, X., Han, J.: gspan: Graph-based substructure pattern mining. In: Proceedings of the 2002 IEEE International Conference on Data Mining. ICDM ’02, Washington, DC, USA, IEEE Computer Society (2002) 721–
7. Yan, X., Han, J.: Closegraph: Mining closed frequent graph patterns. In: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD ’03, New York, NY, USA, ACM (2003) 286–295

8. Nijssen, S., Kok, J.N.: The gaston tool for frequent subgraph mining. Electr. Notes Theor. Comput. Sci. **127**(1) (2005) 77–87
9. Bunescu, R.C., Mooney, R.J.: A shortest path dependency kernel for relation extraction. In: Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing. HLT '05, Stroudsburg, PA, USA, Association for Computational Linguistics (2005) 724–731
10. Aronson, A.R.: Effective mapping of biomedical text to the umls metathesaurus: the metamap program. Proc AMIA Symp (2001) 17–21
11. de Marneffe, M.C., Manning, C.D.: The stanford typed dependencies representation. In: Coling 2008: Proceedings of the Workshop on Cross-Framework and Cross-Domain Parser Evaluation. CrossParser '08, Stroudsburg, PA, USA, Association for Computational Linguistics (2008) 1–8
12. Šarić, J., Jensen, L.J., Ouzounova, R., Rojas, I., Bork, P.: Extraction of regulatory gene/protein networks from medline. Bioinformatics **22**(6) (March 2006) 645–650
13. Friedman, C., Kra, P., Yu, H., Krauthammer, M., Rzhetsky, A.: Genies: a natural-language processing system for the extraction of molecular pathways from journal articles. Comput. Appl. Biosci. **17**(suppl_1) (June 2001) S74–82
14. Blohm, S., Buza, K., Cimiano, P., Schmidt-Thieme, L. Applied Semantic Web Technologies. In: Relation Extraction for the Semantic Web with Taxonomic Sequential Patterns. Taylor and Francis Group (2011) 185–209
15. Quiniou, S., Cellier, P., Charnois, T., Legallois, D.: What about sequential data mining techniques to identify linguistic patterns for stylistics? In Gelbukh, A.F., ed.: CICLing (1). Volume 7181 of Lecture Notes in Computer Science., Springer (2012) 166–177
16. Béchet, N., Cellier, P., Charnois, T., Crémilleux, B., Jaulent, M.C.: Sequential pattern mining to discover relations between genes and rare diseases. In Soda, P., Tortorella, F., Antani, S., Pechenizkiy, M., Cannataro, M., Tsymbai, A., eds.: CBMS, IEEE (2012) 1–6
17. Galitsky, B.: Machine learning of syntactic parse trees for search and classification of text. Engineering Applications of Artificial Intelligence **26**(3) (2013) 1072 – 1091
18. Zhang, M., Zhou, G., Aw, A.: Exploring syntactic structured features over parse trees for relation extraction using kernel methods. Inf. Process. Manage. **44**(2) (March 2008) 687–701
19. Zelenko, D., Aone, C., Richardella, A.: Kernel methods for relation extraction. J. Mach. Learn. Res. **3** (March 2003) 1083–1106
20. Culotta, A., Sorensen, J.: Dependency tree kernels for relation extraction. In: Proceedings of the 42Nd Annual Meeting on Association for Computational Linguistics. ACL '04, Stroudsburg, PA, USA, Association for Computational Linguistics (2004)
21. Fundel, K., Küffner, R., Zimmer, R.: Relex—relation extraction using dependency parse trees. Bioinformatics **23**(3) (January 2007) 365–371
22. Coulet, A., Shah, N.H., Garten, Y., Musen, M.A., Altman, R.B.: Using text to build semantic networks for pharmacogenomics. Journal of Biomedical Informatics **43**(6) (2010) 1009–1019
23. Chowdhury, M.F.M., Lavelli, A.: Combining tree structures, flat features and patterns for biomedical relation extraction. In: EACL. (2012) 420–429
24. Adolphs, P., Xu, F., Li, H., Uszkoreit, H.: Dependency graphs as a generic interface between parsers and relation extraction rule learning. In: Proceedings of the 34th Annual German Conference on Advances in Artificial Intelligence. KI'11, Berlin, Heidelberg, Springer-Verlag (2011) 50–62

Interpretable Low-rank Document Representations with Label-dependent Sparsity Patterns

Ivan Ivek

Rudjer Boskovic Institute, Bijenicka 54, 10000 Zagreb, Croatia

Abstract. In context of document classification, where in a corpus of documents their label tags are readily known, an opportunity lies in utilizing label information to learn document representation spaces with better discriminative properties. To this end, in this paper application of a Variational Bayesian Supervised Nonnegative Matrix Factorization (supervised vbNMF) with label-driven sparsity structure of coefficients is proposed for learning of discriminative nonsubtractive latent semantic components occuring in TF-IDF document representations. Constraints are such that the components pursued are made to be frequently occurring in a small set of labels only, making it possible to yield document representations with distinctive label-specific sparse activation patterns. A simple measure of quality of this kind of sparsity structure, dubbed inter-label sparsity, is introduced and experimentally brought into tight connection with classification performance. Representing a great practical convenience, inter-label sparsity is shown to be easily controlled in supervised vbNMF by a single parameter.

Keywords: Document Categorization, Latent Semantic Analysis, Supervised Sparse Nonnegative Matrix Factorization, Variational Bayes

1 Introduction

As an essential step in machine learning applications which both efficiency and quality of learning depend on, dimensionality reduction has become a well covered subject of research [MPH09] which produced archetipal linear methods with low-rank assumptions such as Principal Component Analysis (PCA) [Jol02] and Nonnegative Matrix Factorization (NMF) [LS99], as well as their kernelized and generally non-linear variants, to touch upon some. Originally they have been formulated as entirely unsupervised methods. However, in supervised and semi-supervised learning applications, where labels of learning samples are readily available, it may be appealing to use this information to obtain lower-dimensional representations of data which not only attempt to preserve the original variance in the data, but also promise to deliver representation spaces with better discriminative properties. A well known representative which incorporates this desideratum is Fisher's Linear Discriminant Analysis (FLD) [MK01].

In: P. Cellier, T. Charnois, A. Hotho, S. Matwin, M.-F. Moens, Y. Toussaint (Eds.): Proceedings of DMNLP, Workshop at ECML/PKDD, Nancy, France, 2014.

In recent relevant literature there is a pronounced trend of using probabilistic generative models for this purpose. Probabilistic approaches to learning lie on a well developed mathematical apparatus which offers flexible enough modeling of prior knowledge in form of graphical models, supported by well-known meta-algorithms for estimating model parameters. Of this family of algorithms, along with Probabilistic Latent Semantic Analysis (pLSA) [Hof99], a common probabilistically formulated baseline algorithm in text mining is Latent Dirichlet Allocation (LDA) [BNJ12] with its more recent discriminative modifications [BM07][LJSJ08], as well as probabilistic formulations of sparse NMF [Cem09] and their supervised counterparts [Ive14].

Sparse coding is known to result in efficient and robust representations which have proven suitable for applications such as data compression, denoising and missing data imputting [Mal99]. On the other hand, representations obtained discriminatively are suitable for classification purposes. Combining those two properties, the basis of this work is a probabilistically formulated method for sparse additive representations of data using nonnegative latent components which are of sparsity structure additionally driven by data labeling [Ive14]. In context of document classification, the decomposition is suitable for finding interpretable patterns of semantically related terms, with high discriminative potential.

1.1 Document Feature Spaces

Disregarding syntactic and semantic interrelations of words, the simplest and most often used intermediate form for document representation is bag-of-words; after tokenization, purification and stemming, frequency of relevant terms is determined for each document resulting in representations of documents as frequencies of particular terms. Models such as LDA have a natural interpretation when decomposing bag-of-words representations, while other approaches may benefit from TF-IDF weighting [RU11] which heuristically measures the importance of a term for a particular document in a specific corpus of documents. For a term with index τ in ν-th document, as a product of two measures,

$$tfidf_{\nu\tau} = tf_{\nu\tau} * idf_{\tau}, \tag{1}$$

TF-IDF score is proportional to (normalized) frequency of a particular term in a document,

$$tf_{\nu\tau} = \frac{\#_{\nu\tau}}{\max_t(\#_{\nu t})}, \tag{2}$$

but stunted by a measure of how rare this term occurs in the entire corpus,

$$idf_{\tau} = \ln \frac{N}{n_{\tau}}, \tag{3}$$

where the number of occurences of term τ in ν-th document is denoted by $\#_{\nu\tau}$, the number of documents in the corpus by N and the number of documents which contain term τ at least once by n_{τ}.

1.2 NMF as a Tool for Latent Semantic Analysis

Bag-of-words-based approaches to text mining are known to suffer from problems of polysemy and synonymy of terms. These problems can be alleviated by representing documents in spaces of patterns of frequencies of semantically related terms rather than in the original space of term frequencies [DDF+90]. Luckily, algorithms for learning of such representations exist, of which perhaps the best known are pLSA formulations. Also assuming inherent nonnegativity in the data, NMF decompositions can be interpreted the same way as pLSA, revealing patterns of semantically related terms underlying the data. Furthermore, a specific connection worth mentioning is that a NMF formulation based on generalized KL-divergence minimizes the exactly same objective function as the original pLSA formulation does [GG05].
Nonnegativity is a reasonable assumption and a desireable bias when modeling either term frequencies or derived intermediate document representations such as TF-IDF. In general, NMF aims at decompositions in form of $\boldsymbol{X}{\approx}\boldsymbol{T}\boldsymbol{V}$, where $\boldsymbol{X}$, $\boldsymbol{T}$ and $\boldsymbol{V}$ are all nonnegative matrices. Although the decomposition is nonunique in general, to some extent nonuniqueness may be compensated for by adding additional bias in the model, of which most prominent is sparsity of solution [LS99]. Sparsity is enforced in divergence-based NMF by different sparsity promoting regularizers, e.g. [Hoy04], and in probabilistic formulations by imposing sparse prior distributions on the coefficients [Cem09].
Throughout this paper, in context of document representation for categorization purposes, $\boldsymbol{X}$ will be regarded as a collection of documents organized columnwise and represented by TF-IDF features, $\boldsymbol{T}$ as a low-rank collection of latent semantic components organized columnwise, and $\boldsymbol{V}$ as matrix of coefficients when $\boldsymbol{X}$ is projected onto the space of latent semantic components $\boldsymbol{T}$. In other words, each document is modeled as a strict superposition of the nonnegative latent semantic components.

2 Methodology

2.1 Supervised NMF Model

The generative model [Ive14] assumes that each column of data $x_{:\tau}$ is a result of latent components $t_{:i}$ consisting of independent gamma-distributed variables,

$$p\left(t_{\nu i}\middle|a^t_{\nu i}, b^t_{\nu i}\right) = \mathcal{G}\left(t_{\nu i}\middle|a^t_{\nu i}, b^t_{\nu i}\right), \tag{4}$$

interacting through linear mixing with coefficients $v_{i\tau}$ under Poissonian noise:

$$p\left(s_{\nu i\tau}|t_{\nu i}, v_{i\tau}\right) = \mathcal{P}\left(s_{\nu i\tau}|t_{\nu i}, v_{i\tau}\right) \tag{5}$$

$$p\left(x_{\nu\tau}|s_{\nu:\tau}\right) = \delta\left(x_{\nu\tau} - \sum_i s_{\nu i\tau}\right). \tag{6}$$

Mixing coefficients $v_{i\tau}$ are assumed to be exponentially distributed with different scale parameters for different selections of label indicators $z_\tau \in \mathcal{L}$, formulated as mixtures of variables λ_{il} with z_τ as discrete numerical mixture selection variables,

$$p\left(v_{i\tau}|z_\tau, \lambda_{i:}\right) = \mathcal{G}\left(v_{i\tau}\middle|1, \sum_{l\in\mathcal{L}} \delta(z_\tau - l)\lambda_{il}^{-1}\right) \tag{7}$$

Note that label indicators z_τ are elements of a discrete set of (integer) numbers $\mathcal{L}$ for convenience of notation. Variables λ_{il}^{-1}, representing expectations of magnitudes of coefficient components i for all samples labeled as l are constrained by inverse-gamma priors,

$$p\left(\lambda_{il}\middle|a_{il}^{\lambda}, b_{il}^{\lambda}\right) = \mathcal{G}\left(\lambda_{il}\middle|a_{il}^{\lambda}, b_{il}^{\lambda}\right). \tag{8}$$

Because inverse-gamma is a heavy-tailed distribution, by setting the probability mass to be concentrated around some small value, significantly larger values of λ_{il}^{-1} will occur rarely. Thus, such a prior imposes an additional bias to produce models having only a minority of indicators λ_{il}^{-1} with significantly large mean values on average, which, hierarchically propagating to activation coefficients $v_{i\tau}$, constrain samples having the same label to have only a small shared subgroup of latent patterns significantly active.
Using compact notation

$$\begin{aligned}
p\left(\boldsymbol{X}|\boldsymbol{S}\right) &= \prod_{\nu,\tau} p\left(x_{\nu\tau}|s_{\nu:\tau}\right)\\
p\left(\boldsymbol{S}|\boldsymbol{T},\boldsymbol{V}\right) &= \prod_{\nu,\tau} p\left(s_{\nu:\tau}|t_{\nu i}, v_{i\tau}\right)\\
p\left(\boldsymbol{T}\middle|\boldsymbol{A^t},\boldsymbol{B^t}\right) &= \prod_{\nu,i} p(t_{\nu i}|a_{\nu i}^{t}, b_{\nu i}^{t})\\
p\left(\boldsymbol{V}\middle|\boldsymbol{\mathit{\Lambda}},\vec{z}\right) &= \prod_{i,\tau} p\left(v_{i\tau}|\lambda_{i:}, z_\tau\right)\\
p\left(\boldsymbol{\mathit{\Lambda}}\middle|\boldsymbol{A^{\lambda}},\boldsymbol{B^{\lambda}}\right) &= \prod_{i,l} p\left(\lambda_{il}\middle|a_{il}^{\lambda}, a_{il}^{\lambda}\right),
\end{aligned}$$

joint distribution of the supervised NMF model can be written as

$$\begin{aligned}
&p\left(\boldsymbol{X},\boldsymbol{S},\boldsymbol{T},\boldsymbol{V},\boldsymbol{\mathit{\Lambda}}\middle|\boldsymbol{A^t},\boldsymbol{B^t},\boldsymbol{A^{\lambda}},\boldsymbol{B^{\lambda}},\vec{z}\right)\\
&\quad= p\left(\boldsymbol{X}|\boldsymbol{S}\right)p\left(\boldsymbol{S}|\boldsymbol{T},\boldsymbol{V}\right)p\left(\boldsymbol{T}\middle|\boldsymbol{A^t},\boldsymbol{B^t}\right)p\left(\boldsymbol{V}\middle|\boldsymbol{\mathit{\Lambda}},\vec{z}\right)p\left(\boldsymbol{\mathit{\Lambda}}\middle|\boldsymbol{A^{\lambda}},\boldsymbol{B^{\lambda}}\right).
\end{aligned} \tag{9}$$

Linear mixing as described by (4), (5) and (6) is the same as in Poisson-gamma NMF [Cem09]. Equations (7) and (8) additionally formulate a sparsity structure abstracted from the level of data samples to the level of labels, making it possible to pursue decompositions with recognizable sparsity patterns characteristic of data samples which share the same label tag.

2.2 Variational Bayesian Learning Algorithm

To give a concise outline of general treatment of learning by VB, let the observed variables be denoted by $\boldsymbol{D}$, the hyperparameters of a model by $\boldsymbol{H}$ and both the unobserved variables and the model parameters by $\boldsymbol{\Theta}$. Minimization of discrepancy between posterior $p\left(\boldsymbol{\Theta}|\boldsymbol{D},\boldsymbol{H}\right)$ (which is in general difficult to optimize directly, especially in a fully Bayesian manner) and an introduced instrumental approximation $q(\boldsymbol{\Theta})$ measured by Kullback-Liebler divergence gives rise to a lower bound on the posterior,

$$\mathcal{L} = \left\langle \ln p\left(\boldsymbol{D},\boldsymbol{\Theta}|\boldsymbol{H}\right)\right\rangle_{q(\boldsymbol{\Theta})} + \mathcal{H}\left[q(\boldsymbol{\Theta})\right], \tag{10}$$

where entropy of the probability density function in the argument is denoted by $\mathcal{H}\left[.\right]$. Supposing that $q(\boldsymbol{\Theta})$ is of factorized form $q(\boldsymbol{\Theta}) = \prod_{\alpha\in C} q(\boldsymbol{\Theta}_{\boldsymbol{\alpha}})$, it can be shown that the iterative local updates at iteration (n+1) alternating over C in form of

$$q(\boldsymbol{\Theta}_{\boldsymbol{\alpha}})^{(n+1)} \propto \exp\left(\left\langle \ln p\left(\boldsymbol{D},\boldsymbol{\Theta}|\boldsymbol{H}\right)\right\rangle_{\frac{q(\boldsymbol{\Theta})^{(n)}}{q(\boldsymbol{\Theta}_{\boldsymbol{\alpha}})^{(n)}}}\right) \tag{11}$$

improve the lower bound (10) monotonically. Moreover, should the model be conjugate-exponential, for a fully factorized approximation, expressions in (11) neccessarily assume analytical forms [Win03].

To obtain convenient analytical forms of update equations of supervised vb-NMF model defined by (9) the approximative distribution is chosen to be factorized as $q\left(\boldsymbol{S},\boldsymbol{T},\boldsymbol{V},\boldsymbol{\Lambda}\right) = q\left(\boldsymbol{S}\right)q\left(\boldsymbol{T}\right)q\left(\boldsymbol{V}\right)q\left(\boldsymbol{\Lambda}\right)$, with shorthand notation $q\left(\boldsymbol{S}\right) = \prod_{\nu,\tau} q\left(s_{\nu:\tau}\right)$, $q\left(\boldsymbol{T}\right) = \prod_{\nu,i} q\left(t_{\nu i}\right)$, $q\left(\boldsymbol{V}\right) = \prod_{i,\tau} q\left(v_{i\tau}\right)$ and $q\left(\boldsymbol{\Lambda}\right) = \prod_{i,l} q\left(\lambda_{il}\right)$. The variational Bayesian update expressions are obtained by specifying

$$p\left(\boldsymbol{D},\boldsymbol{\Theta}|\boldsymbol{H}\right) = p\left(\boldsymbol{X},\boldsymbol{S},\boldsymbol{T},\boldsymbol{V},\boldsymbol{\Lambda}\left|\boldsymbol{A}^{\boldsymbol{t}}\right.,\boldsymbol{B}^{\boldsymbol{t}},\boldsymbol{A}^{\boldsymbol{v}},\boldsymbol{B}^{\boldsymbol{v}}\right)$$

together with

$$q\left(\boldsymbol{\Theta}\right) = q\left(\boldsymbol{S},\boldsymbol{T},\boldsymbol{V},\boldsymbol{\Lambda}\right)$$

in (11). For computational convenience, optimization is done with respect to the lower bound (10) additionally relaxed using Jensen's inequality, to finally yield the update expressions summarized in Table A1., where the iterative learning algorithm is presented using efficient matrix operations; the accompanying matrix forms of the variables, hyperparameters and variational parameters of the model are listed in the first column of the table. A more detailed treatment of the learning algorithm can be found in [Ive14].

3 Experiments

All experiments have been performed on *20Newsgroups*[1] dataset, *bydate* version split into training and test sets; rather than estimating the generalization error of classification by crossvalidation techniques, the underlying ambition is merely to explore peak potentials of classification using different representation spaces evaluated on a single train-test split in same conditions.

[1] Available from Jason Rennie's web Page, http://qwone.com/~jason/20Newsgroups/

3.1 Dataset

Experiments have been performed on *20Newsgroups* dataset sorted by date wih duplicates and headers removed, with documents having multiple labels left out and preprocessed to obtain a bag-of-words representation. The dataset is split into a training set and a test set. To alleviate computational load, the set of features has been heuristically reduced to 10000 terms, based on maximum TF-IDF score accross all documents.

3.2 Experimental Setup

Representation spaces in which consequently classification takes place which are taken under consideration are the ones obtained by PCA, Poisson-gamma unsupervised vbNMF [Cem09], and the supervised vbNMF, all decomposing the matrix of TF-IDF scores of the training set only. Having learned a specific space of reduced dimensionality, representation of the test set in this space is found by projection on the vector basis in case of PCA or by optimizing the matrix of coefficients only using Poisson-gamma vbNMF formulation (i.e. the matrix of latent components is fixed to what has been learned in the training step) in case of both unsupervised and supervised vbNMF methods.

For Poisson-gamma vbNMF sparse decompositions have been pursued by fixing shape hyperparameters of the gamma distributed coefficients to a value less than or equal to 1 throughout the entire run, while other hyperparameters (constrained to be the same for all elements of matrices $\boldsymbol{T}$ and $\boldsymbol{V}$, a single one for each of the matrices) have been optimized automatically by maximization of the lower bound directly in a non-Bayesian manner [Cem09]. For supervised vbNMF, hyperparameters aLambda have been fixed and varied, while other hyperparameters have been left to the algorithm to optimize, by direct optimization as in [Cem09]. Specifically, following initialization, λ parameters are chosen to be all equal and fixed for a burn-in period of 10 iterations, not until after which they start to get optimized according to the algorithm in Table A1.

To accentuate the predictive potentials of the considered representation spaces by themselves, rather than in conjunction with a strong classifier, the classifier of choice is k-NN using cosine similarity metric, with k chosen heuristically as the square root of the cardinality of the training set.

Dimension of space of latent components has been varied as a parameter for all decomposition methods. Because at each run the NMF algorithms converge to some local minimum, to explore these local minima, for each parameter set they have been run 10 times with random initializations.

3.3 Evaluation

Metrics of classification performance used in the experiments are micro-averaged accuracy, defined as

$$a^{micro} = \frac{\sum_l N_l^{correct}}{\sum_l N_l^{all}},$$

and macro-averaged accuracy,

$$a^{macro} = \frac{1}{L} \sum_{l} \frac{N_l^{correct}}{N_l^{all}},$$

where the number of correctly classified documents belonging to the l-th label is denoted by $N_l^{correct}$, the number of documents belonging to l-th label in the test split by N_l^{all} and the number of labels by L. By averaging the accuracies calculated separatedly for each of the labels, macro-averaged accuracy compensates for label-imbalance of test datasets.

As a measure of sparsity, Hoyer's measure [Hoy04], based on ratio of l1 and l2 norms and originally introduced in context of NMF penalization, will be used. For a vector $\vec{x} = [x_1, ..., x_n]^T$ it is defined as

$$sparsity\left(\vec{x}\right) = \frac{1}{\sqrt{n}-1}\left(\sqrt{n} - \frac{\sum_i |x_i|}{\sqrt{(\sum_i x_i^2)}}\right), \tag{12}$$

taking value of 1 in case only a single element is non-zero (maximum sparsity), and a value of 0 if all elements are equal (minimum sparsity). For the purpose of this paper, when referring to sparsity of matrices, matrix is assumed to be vectorized first by appending its columns, then treating it as a vector according to (12).

If labels in a document corpus are meaningfully assigned based on topics of documents, then meaningful discovered latent semantic components are expected to have specific patterns of occurence for documents belonging to a specific label. Using supervised vbNMF, those patterns are modeled as patterns in sparsity of coefficients (i.e. in patterns of support of sparse coefficients) that documents labeled the same have in common. To measure the consistency of occurence of sparsity patterns in labels, let a representation by coefficients of N documents in I dimensional space be denoted by $\boldsymbol{V} \in \mathbb{R}^{IxN}$, i.e. n-th document is represented by coefficient vector $[\boldsymbol{V}]_{:n}$, and let sums of coefficient sets which share the same label be accumulated in matrix $\boldsymbol{L} \in \mathbb{R}^{IxL}$, where L is number of labels as

$$[\boldsymbol{L}]_{:l} = \sum_{n \in N_l} [\boldsymbol{V}]_{:n}, \tag{13}$$

where n iterates over subset of document indices with the same label, N_l.

Now, inter-label sparsity can be introduced, defined as sparsity of matrix $\boldsymbol{L}$. The motivation behind (13) is that $l0$ norm of a sum of vectors with the same sparsity pattern (same support) is the same as the exclusive $l0$ norm of such vectors by themselves, and, the more those vectors deviate from the pattern (i.e. when the vectors have differing supports), the larger the $l0$ norm of the sum will be. Note that the latter rationale holds exactly for $l0$ definition of sparsity, while for more relaxed definitions of sparsity such as (12) the behavior will be only qualitatively similar. For the purpose of this paper, sparsity of (13) will be measured as Hoyer's sparsity (12).

3.4 Results and Discussion

For comparison, as a baseline, classification results of PCA are plotted against the dimension of representation space on Fig. 1. For unsupervised vbNMF, micro-averaged accuracies averaged accross random initializations for different shape parameters with varying number of latent semantic components are shown in Fig. 2.

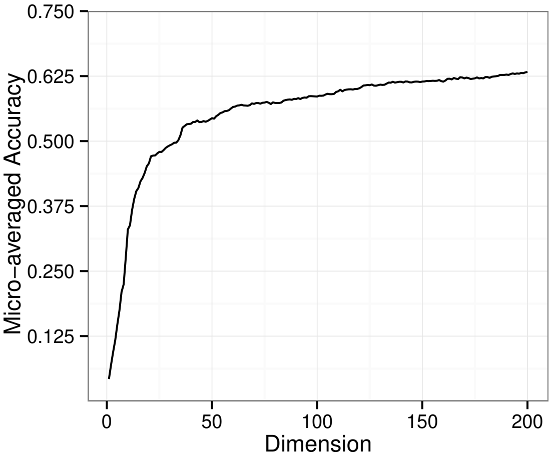


Fig. 1. Classification results using PCA.

Compared to PCA, even with a larger dimension of representation space, vbNMF with sparsity constraints did not bring improvements on average, regardless of the degree of sparsity penalization. The explanation is that, even though sparse representation spaces may be good for clustering, natural clusters may differ greatly from labeling and consequently even be detrimental to classification applications [BM07] when compared to dense representations such as PCA. Better representations for classification purposes are expected to be found by introducing label information to the model, which in spaces obtained by supervised vbNMF (Fig. 3.) indeed manifested as a boost in classification performance.
Both unsupervised vbNMF and supervised vbNMF consistently resulted in sparse decompositions. However, label-driven structure present in supervised vbNMF decompositions (engineered as to be the sole difference in the experiments) is to be accounted for the beneficial effect observed. Examples of sparsities accross labels according to (13) are visualized on on Fig. 4. for the sparse unsupervised vbNMF decomposition which produced peak micro-averaged accuracy of 0.5796 and on Fig. 5. for an arbitrarily chosen supervised variant with matching dimension. The supervised variant produced distinctive sparsity patterns accross labels, which is also reflected quantitatively on inter-label sparsity of the decomposition.

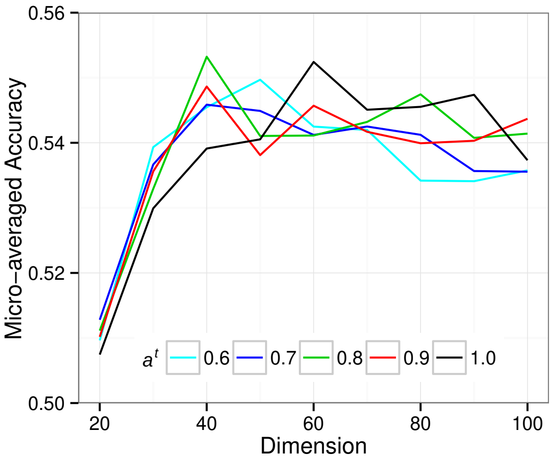


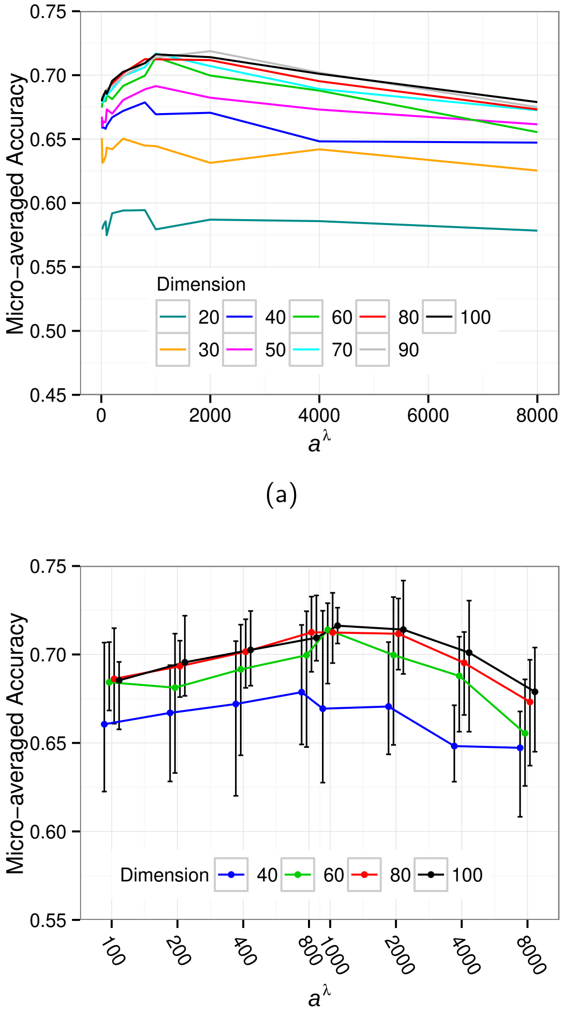
Fig. 2. Classification results of unsupervised vbNMF (averaged accross 10 random initializations) with varying level of sparsity penalization.

The connection between sparsity on the level of labels and classification performance is further explored using Fig. 6., showing data for all supervised representations obtained in the experiments. Variance of the scatter plot becomes tighter with increasing dimension of representation space, meaning that for a sufficiently large dimension of the decomposition, inter-label sparsity is indeed a good predictor for classification quality on this dataset.

Equally importantly, experiments show that in case of supervised vbNMF, inter-label sparsity can elegantly be controlled by a single parameter a^{λ} alone, regardless of dimension: as illustrated by Fig. 7., a logarithmic increase of a^{λ} is accompanied by a trend of growth of inter-label sparsity, only to be broken by too extreme regularizations, when tails of the prior have little mass. On the other hand, unsupervised vbNMF resulted in moderate levels of inter-label sparsity because sparsity structure is supported by the structure of data features only, with somewhat higher values in cases of very strong sparsity regularizations and an impractically small number of latent patterns.

Classification results using k-NN classifier with heuristically chosen k on representation spaces obtained by the three methods are summarized in Table 1., reporting peak value of its micro- and macro-averaged accuracies; for the set of parameters which yielded the peak performance, corresponding accuracies averaged accross the 10 random initializations together with minimal achieved accuracies are reported. On Fig. 3.b), showing smooth dependence of micro-averaged accuracy (averaged across random initializations) on an interesting range of a^{λ} for a selection of dimensions, peak performance as entered in Table 1. can be seen marked.

To conclude the remarks on the experiments, it is worth mentioning that, if sparse representations are pursued, care is advised when choosing and optimiz-



(a)

(b)

Fig. 3. Supervised vbNMF classification results, averaged over 10 random initializations. a) Dependence on level of sparsity penalization, varying dimensions of representation spaces. b) Dependence on level of sparsity penalization, varying dimensions of representation spaces. Error bars represent maximum and minimum values among the random initializations. x-axis is shown on logarithmic scale.

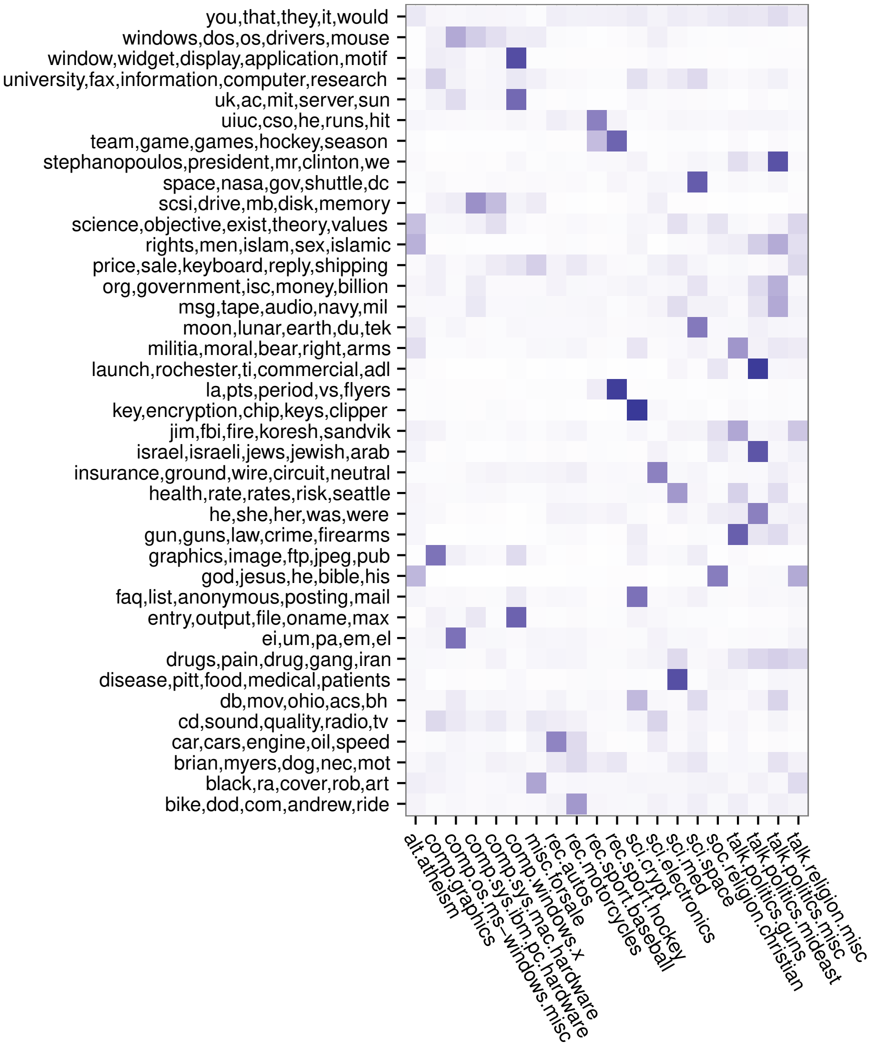


Fig. 4. Sparsity accross labels according to (13) for the unsupervised vbNMF decomposition with best classification performance; the more intense the tone of blue, the higher the sparsity. Each of the 40 latent semantic components is represented by its 5 most significant terms. Coefficient sparsity is 0.6862, inter-label sparsity is 0.5784.

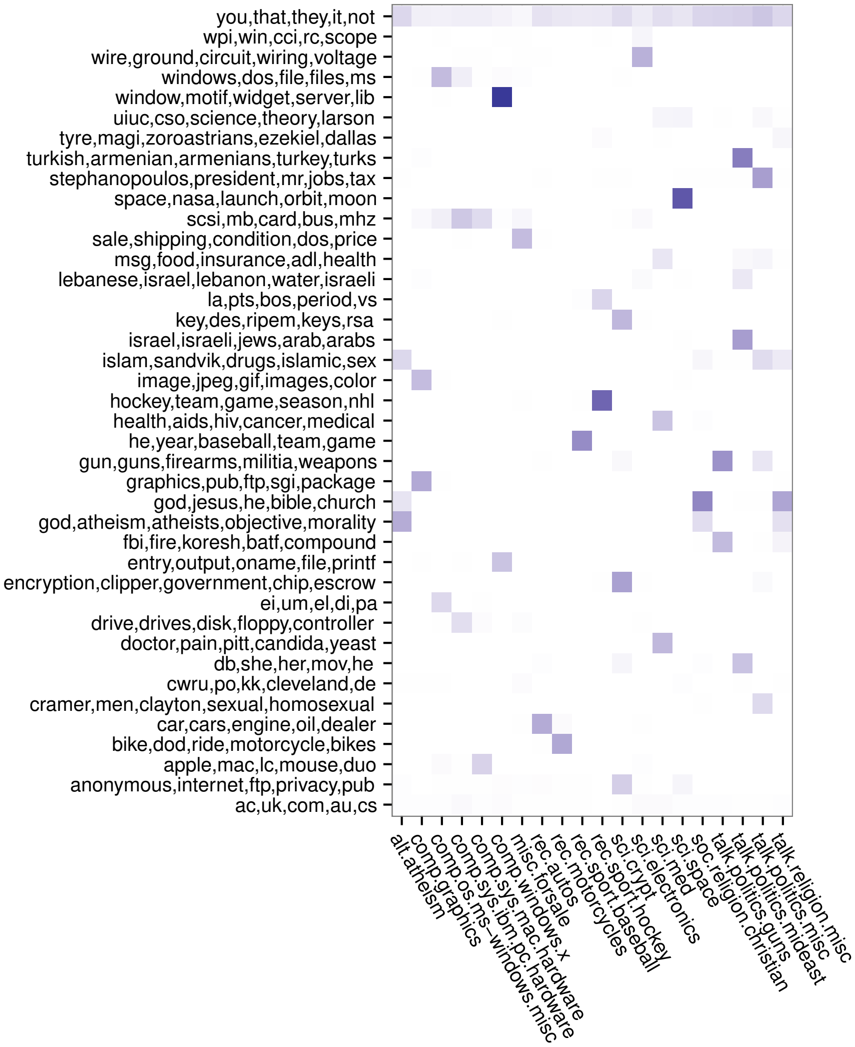


Fig. 5. Sparsity accross labels according to (13) for an arbitrarily chosen supervised vbNMF decomposition with 40 latent semantic components, each represented by its 5 most significant terms. Coefficient sparsity is 0.8752, inter-label sparsity is 0.8578.

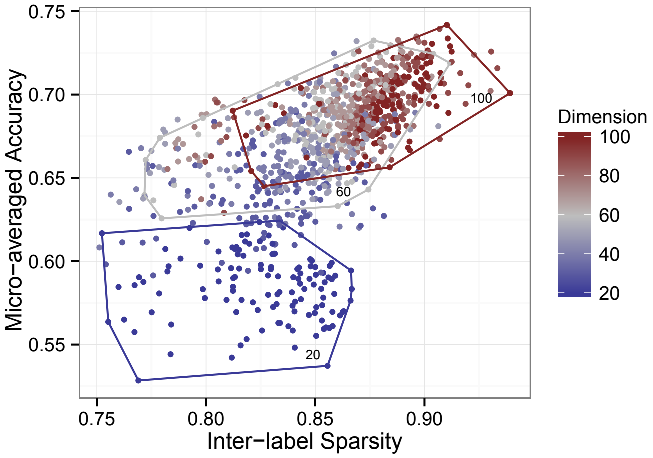


Fig. 6. Correlation between micro-averaged accuracy and inter-label sparsity. Each point in the scatter plot represents a single supervised NMF decomposition. Convex hulls contain points corresponding to choices of dimensions of 20, 60 and 100.

ing hyperparameters by non-Bayesian minimization of bound. Because sparse constraints both on matrices $\boldsymbol{T}$ and $\boldsymbol{V}$ act as two competing penalizations, useful decompositions are obtained more easily by constraining only one of the matrices to be sparse - either the matrix of latent components to obtain a parts-based representation, or the matrix of coefficients to obtain a sparse representation of data. So, when optimizing the shape parameter of one of the matrices in such a manner next to a fixed hyperparameter of the other matrix which is to be made sparse, due to the automated (and, equally importantly, non-Bayesian) nature of the optimization the former may also come to describe a sparse distribution and in effect impede the desired bias toward the desired type of sparsity.

	Algorithm	PCA	Unsupervised vbNMF	Supervised NMF
Micro-averaged Accuracy	[Min,Max]		[0.5190,0.5796]	[0.6890,**0.7418**]
	Mean	0.6330	0.5532	**0.7141**
	Dimension	200	40	100
Macro-averaged Accuracy	[Min,Max]		[0.5033,0.5655]	[0.6758,**0.7277**]
	Mean	0.6179	0.5393	**0.6997**
	Dimension	200	40	100

Table 1. Summary of experimental results

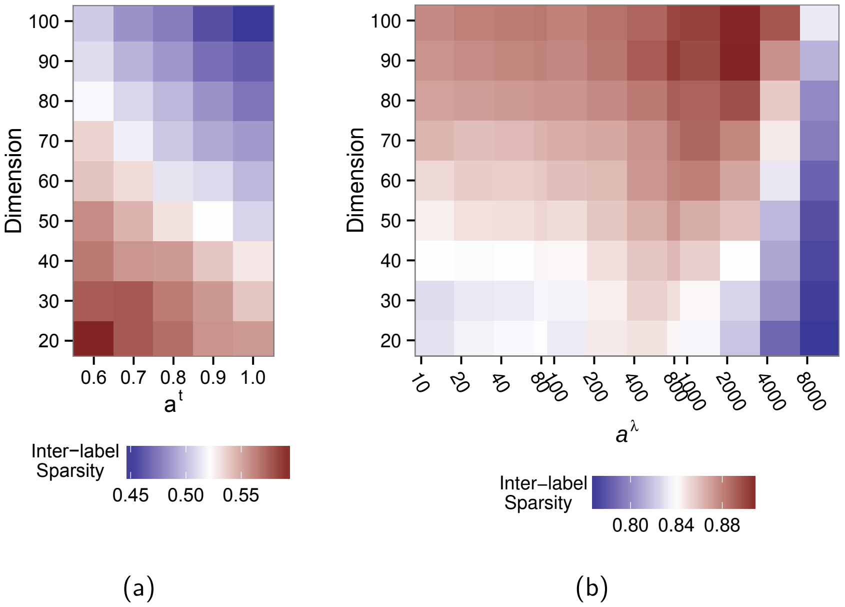


Fig. 7. Dependence of inter-label sparsity (averaged over 10 random initializations) on dimension of representation space and parameters which control sparsity. a) Unsupervised vbNMF with sparsity constraints. b) Supervised vbNMF; inter-label sparsity can be controlled by a^λ.

4 Conclusion

It has been well documented that using label information in low-rank representation learning is vital to obtain representations with good discriminative properties. In this context, applied to classification of a document corpus, a probabilistic learning algorithm which combines sparse coding and supervised learning has been presented.

To characterize advantages of using label information, two extreme cases have been juxtaposed, the presented supervised model and a fully unsupervised one, belonging to the same family, having the same noise model and using the same metaalgorithm for parameter learning.

A qualitative inspection motivated the introduction of the notion of inter-label sparsity, abstracting sparsity of coefficients on the level of documents to sparsity on the level of document labels. Experiments point to a strong connection between the inter-label sparsity of the representation and the classification performance metrics. Furthermore, inter-label sparsity of decompositions obtained by supervised vbNMF can elegantly be controlled by a single parameter. However, even though sparsity and nonnegativity constraints intuitively seem appropriate and result in compact and interpretable document representations, a question

remains whether there is any actual advantage in using sparse representations over dense ones as classification precursors.
As quality of representation spaces has been primarily addressed in this work, little regard has been given to quality of the classifier *per se*. Because it is reasonable to expect that a stronger classifier would result in even better classification results, it would be interesting to compare a well-tuned classifier in the representation spaces obtained by supervised vbNMF to state-of-the-art aproaches in the field, on benchmark datasets. Future work based on semi-supervised modifications of the model is considered, to make the model more flexible and applicable in more commonly occuring, semi-supervised, scenarios.

Acknowledgments This work was supported by the Croatian Ministry of Science, Education and Sports through the project "Computational Intelligence Methods in Measurement Systems", No. 098-0982560-2565.

References

[BM07] David M Blei and J McAuliffe. Supervised Topic Models. *Neural Information Processing Systems*, 21, 2007.

[BNJ12] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022, 2012.

[Cem09] Ali Taylan Cemgil. Bayesian inference for nonnegative matrix factorisation models. *Computational Intelligence and Neuroscience*, 2009, January 2009.

[DDF$^+$90] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41:391–407, 1990.

[GG05] Cyril Goutte and Eric Gaussier. Relation between PLSA and NMF and implications. *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 601–602, 2005.

[Hof99] Thomas Hofmann. Probabilistic Latent Semantic Analysis. In *Uncertainity in Artifitial Intelligence - UAI'99*, page 8, 1999.

[Hoy04] Patrik O Hoyer. Non-negative matrix factorization with sparseness constraints. *Journal of Machine Learning Research*, 5:1457–1469, 2004.

[Ive14] Ivan Ivek. Supervised Dictionary Learning by a Variational Bayesian Group Sparse Nonnegative Matrix Factorization. May 2014.

[Jol02] I T Jolliffe. *Principal Component Analysis*, volume 98. 2002.

[LJSJ08] S Lacoste-Julien, F Sha, and MI Jordan. DiscLDA: Discriminative learning for dimensionality reduction and classification. *NIPS*, pages 897–904, 2008.

[LS99] D D Lee and H S Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401:788–791, 1999.

[Mal99] Stéphane Mallat. *A Wavelet Tour of Signal Processing*. 1999.

[MK01] Aleix M. Martinez and Avinash C. Kak. PCA versus LDA. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23:228–233, 2001.

[MPH09] L J P van der Maaten, E O Postma, and H J van den Herik. Dimensionality Reduction: A Comparative Review. *Journal of Machine Learning Research*, 10:1–41, 2009.

[RU11] Anand Rajaraman and Jeffrey D Ullman. Mining of Massive Datasets. *Lecture Notes for Stanford CS345A Web Mining*, 67:328, 2011.

[Win03] John M Winn. Variational message passing and its applications. *Ph.D. thesis, Department of Physics, University of Cambridge*, 2003.

A Summary of the Learning Algorithm

Observed variables:	Initialize variational parameters. $t = 0$.
$\left[\boldsymbol{X}\right]_{\nu\tau} = x_{\nu\tau}$	Loop:
$\left[\boldsymbol{\Delta}\right]_{\tau l} = \delta\left(z_\tau - l\right)$	$\boldsymbol{\Xi} = \boldsymbol{X}. \Big/ \left(\left(\exp \boldsymbol{L}_{\boldsymbol{t}}^{(t)}\right) * \left(\exp \boldsymbol{L}_{\boldsymbol{v}}^{(t)}\right)\right)$
Hyperparameters:	$\boldsymbol{\Sigma}_{\boldsymbol{v}}^{(t+1)} = \exp \boldsymbol{L}_{\boldsymbol{v}}^{(t)} . * \left(\left(\exp \boldsymbol{L}_{\boldsymbol{t}}^{(t)}\right)^T * \boldsymbol{\Xi}\right)$
$\left[\boldsymbol{A}^{\boldsymbol{t}}\right]_{\nu i} = a_{\nu i}^t$	$\boldsymbol{\Sigma}_{\boldsymbol{t}}^{(t+1)} = \exp \boldsymbol{L}_{\boldsymbol{t}}^{(t)} . * \left(\boldsymbol{\Xi} * \left(\exp \boldsymbol{L}_{\boldsymbol{v}}^{(t)}\right)^T\right)$
$\left[\boldsymbol{B}^{\boldsymbol{t}}\right]_{\nu i} = b_{\nu i}^t$	$\boldsymbol{A}_{\boldsymbol{t}} = \boldsymbol{A}^{\boldsymbol{t}} + \boldsymbol{\Sigma}_{\boldsymbol{t}}^{(t+1)}$
$\left[\boldsymbol{A}^{\boldsymbol{\lambda}}\right]_{il} = a_{il}^\lambda$	$\boldsymbol{B}_{\boldsymbol{t}} = 1. \Big/ \left(1. / \boldsymbol{B}^{\boldsymbol{t}} + \boldsymbol{1} * \left(\boldsymbol{E}_{\boldsymbol{v}}^{(t)}\right)^T\right)$
$\left[\boldsymbol{B}^{\boldsymbol{\lambda}}\right]_{il} = b_{il}^\lambda$	$\boldsymbol{E}_{\boldsymbol{t}}^{(t+1)} = \boldsymbol{A}_{\boldsymbol{t}} . * \boldsymbol{B}_{\boldsymbol{t}}$
Variational parameters:	$\boldsymbol{L}_{\boldsymbol{t}}^{(t+1)} = \Psi\left(\boldsymbol{A}_{\boldsymbol{t}}\right) + \ln \boldsymbol{B}_{\boldsymbol{t}}$
$\left[\boldsymbol{E}_{\boldsymbol{t}}^{(t)}\right]_{\nu i} = \left\langle t_{\nu i}\right\rangle^{(t)}$	$\boldsymbol{A}_{\boldsymbol{v}} = \boldsymbol{1} + \boldsymbol{\Sigma}_{\boldsymbol{v}}^{(t+1)}$
$\left[\boldsymbol{L}_{\boldsymbol{t}}^{(t)}\right]_{\nu i} = \left\langle \ln t_{\nu i}\right\rangle^{(t)}$	$\boldsymbol{B}_{\boldsymbol{v}} = 1. \Big/ \left(\boldsymbol{E}_{\boldsymbol{\lambda}}^{(t)} * \boldsymbol{\Delta} + \left(\boldsymbol{E}_{\boldsymbol{t}}^{(t)}\right)^T * \boldsymbol{1}\right)$
$\left[\boldsymbol{E}_{\boldsymbol{v}}^{(t)}\right]_{i\tau} = \left\langle v_{i\tau}\right\rangle^{(t)}$	$\boldsymbol{E}_{\boldsymbol{v}}^{(t+1)} = \boldsymbol{A}_{\boldsymbol{v}} . * \boldsymbol{B}_{\boldsymbol{v}}$
$\left[\boldsymbol{L}_{\boldsymbol{v}}^{(t)}\right]_{i\tau} = \left\langle \ln v_{i\tau}\right\rangle^{(t)}$	$\boldsymbol{L}_{\boldsymbol{v}}^{(t+1)} = \Psi\left(\boldsymbol{A}_{\boldsymbol{v}}\right) + \ln \boldsymbol{B}_{\boldsymbol{v}}$
$\left[\boldsymbol{\Sigma}_{\boldsymbol{t}}^{(t)}\right]_{\nu i} = \sum_{\tau} \left\langle s_{\nu i\tau}\right\rangle^{(t)}$	$\boldsymbol{A}_{\boldsymbol{\lambda}} = \boldsymbol{A}^{\boldsymbol{\lambda}} + \boldsymbol{\Delta} * \boldsymbol{1}$
$\left[\boldsymbol{\Sigma}_{\boldsymbol{v}}^{(t)}\right]_{i\tau} = \sum_{\nu} \left\langle s_{\nu i\tau}\right\rangle^{(t)}$	$\boldsymbol{B}_{\boldsymbol{\lambda}} = 1. \Big/ \left(1. \Big/ \boldsymbol{B}^{\boldsymbol{\lambda}} + \boldsymbol{E}_{\boldsymbol{v}}^{(t+1)} . * \boldsymbol{\Delta}\right)$
$\left[\boldsymbol{E}_{\boldsymbol{\lambda}}^{(t)}\right]_{il} = \left\langle \lambda_{il}\right\rangle^{(t)}$	$\boldsymbol{E}_{\boldsymbol{\lambda}}^{(t+1)} = \boldsymbol{A}_{\boldsymbol{\lambda}} . * \boldsymbol{B}_{\boldsymbol{\lambda}}$
$\left[\boldsymbol{L}_{\boldsymbol{\lambda}}^{(t)}\right]_{il} = \left\langle \ln \lambda_{il}\right\rangle^{(t)}$	$\boldsymbol{L}_{\boldsymbol{\lambda}}^{(t+1)} = \Psi\left(\boldsymbol{A}_{\boldsymbol{\lambda}}\right) + \ln \boldsymbol{B}_{\boldsymbol{\lambda}}$
	Optimize hyperparameters (nonbayesian)
	End loop

Table A1. The learning algorithm in matrix form. Left column: observed variables, hyperparameters and variational parameters organized as matrices. Right column: the algorithm; by $.*$ and $./$ elementwise matrix product and elementwise matrix division are denoted, respectively, and by $\mathbf{1}$ matrix of ones of appropriate dimensions.

TagMiner: A Semisupervised Associative POS Tagger Effective for Resource Poor Languages

Pratibha Rani, Vikram Pudi, and Dipti Misra Sharma

International Institute of Information Technology, Hyderabad, India
pratibha_rani@research.iiit.ac.in,{vikram, dipti}@iiit.ac.in

Abstract. We present here, TagMiner, a data mining approach for part-of-speech (POS) tagging, an important Natural language processing (NLP) classification task. It is a semi-supervised associative classification method for POS tagging. Existing methods for building POS taggers require extensive domain and linguistic knowledge and resources. Our method uses combination of a small POS tagged corpus and a raw untagged text data as training data to build the classifier model using association rules. Our tagger works well with very little training data also. The use of semi-supervised learning provides the advantage of not requiring a large high quality tagged corpus. These properties make it especially suitable for resource poor languages. Our experiments on various resource-rich, resource-moderate and resource-poor languages show good performance without using any language specific linguistic information. We note that inclusion of such features in our method may further improve the performance. Results also show that for smaller training data sizes our tagger performs better than state-of-the-art CRF tagger using same features as our tagger.

Keywords: Part-of-Speech Tagging, Associative Classification, Association Rules, Semi-supervised Classification, NLP

1 Introduction

Part-of-speech (POS) tagging is an important NLP classification task that takes a word or a sentence as input, assigns a POS tag or other lexical class marker to a word or to each word in the sentence, and produces the tagged text as output. For this task several rule based [7, 8], stochastic supervised [6, 15, 30], and unsupervised [2, 5, 16] approaches are available for a number of languages. All of these approaches (including the state-of-the-art taggers) require training data and linguistic resources like dictionaries in *large* quantities. These taggers do not perform well for languages which do not have much resources and training data, referred to as *resource poor languages*.

The creation of linguistic resources is a time consuming expensive process which requires expert linguistic knowledge. So, there is a need to develop semi-supervised and generic POS tagging methods which take advantage of raw untagged corpus and require less or no lexical resources. A few such available

In: P. Cellier, T. Charnois, A. Hotho, S. Matwin, M.-F. Moens, Y. Toussaint (Eds.): Proceedings of DMNLP, Workshop at ECML/PKDD, Nancy, France, 2014.

techniques are mentioned in Sect. 2. In order to perform well, these techniques require a *large* raw untagged corpus. Unfortunately, for many resource poor languages, even obtaining this is hard.

This motivates us to explore data mining methods to build generic POS tagger. Data mining, being composed of data driven techniques, is a promising direction to explore or to develop language/domain independent POS tagging methods. However, direct application of data mining concepts for this task is not feasible and requires handling various challenges like 1) mapping POS tagging task to association rule mining problem, 2) developing semi-supervised methods to extract association rules from training set of *tagged* and *raw untagged* data combined and 3) handling challenges of POS tagging task (discussed in Sect. 4.2), like class imbalance, data sparsity and phrase boundary problems.

Associative classification [28] is a well known data mining based classification approach which uses association rules [1] to build the classifier model. In this work, we apply associative classification for POS tagging and present Tag-Miner, a generic semi-supervised method for POS tagging. Our method uses a combination of a *small* POS tagged corpus and a raw untagged text data as training data to build a classifier model using a new concept of *context based association rule mining*. These association rules work as context based tagging rules. Our Experiments demonstrate that it gives good performance even without using any linguistic resources—except for a small POS tagged corpus—for resource-rich English, resource-moderate Hindi and resource-poor Telugu, Tamil and Bengali languages.

Our method is generic in two aspects: (1) it does not use any language specific linguistic information such as morphological features and there is ample scope to improve further by including such features, (2) it does not require a large, high quality, tagged corpus and uses the POS tags of the tagged corpus only to calculate scores of "context based lists" which are used to form association rules. This can be easily adapted for various languages. Also, as an additional benefit model made by our tagger is human understandable since it is based on association rules.

Our algorithm has following advantages, especially suitable for resource poor languages, arising due to the use of raw untagged data: (1) it tags unknown words without using smoothing techniques, (2) the coverage of words present in the classifier model is increased which in turn increases tagging accuracy and (3) it creates additional linguistic resources from raw untagged data in the form of word clusters.

Remainder of this paper is as follows: Section 2 surveys related work. Section 3 formally presents the problem. Section 4, 5 and 6 present details of our proposed approach. Section 7 gives details of the datasets, various experiments and discusses the performance. Section 8 concludes our work.

2 Related Work

Associative classifiers use association rules to build a classifier model. They have been successfully applied for various classification tasks, for example, [34] presents an associative classifier for mammography image classification and [26] uses it for predictive analysis in health care data mining. Some of the associative classifiers worth mentioning are CBA [21] which integrates association rules and classification by finding class association rules, CMAR [20] uses concept of multiple class-association rules, CPAR [33] is based on predictive association rules and ACME [29] exploits maximum entropy principle. A good review of various associative classifiers and the detailed analysis of this method can be found in [28]. In some other association rule based approaches [18] uses association rules in a hybrid system of Naive Bayes and genetic classifier for text classification and [23] presents a supervised *language specific* hybrid algorithm of statistical method and association rule mining to increase the POS tagging accuracy of Chinese text. To the best of our knowledge no semi-supervised method exists for association rule mining from training set of tagged and raw untagged data combined.

For POS tagging, one of the first semi-supervised methods was proposed by [10] which uses raw untagged corpus by incorporating features obtained from a small fraction of untagged data along with features obtained from a large tagged data. A good overview of the existing semi-supervised POS tagging methods and discussion on their limitations is provided by [27], which uses graph as a smoothness regularizer to train CRFs [19] in a semi-supervised manner from a large untagged data and a small tagged data. In another approach [25] presents a condensed nearest neighbor method for semi-supervised POS tagging and report 97.5% accuracy on WSJ dataset of English. Most of the existing semi-supervised POS tagging methods use a combination of complex learning methods and existing supervised tagging methods to learn from large untagged data and moderate sized tagged data. All these methods have been developed for resource rich English and other European languages.

To the best of our knowledge no semi-supervised tagging method has been employed for resource moderate Hindi and resource poor Telugu and Tamil languages. Also to the best of our knowledge no fully data mining based generic POS tagger exists for any language. Baseline POS taggers for various languages are discussed below. We note that all the reported accuracy values were obtained for very small sized test sets. All the mentioned POS taggers use linguistic (especially morphological) knowledge in some or the other form, while our approach uses only the POS tags of the tagged set in an indirect form and learns from the raw untagged data.

For Hindi language, [22] proposes a CRF model with Transformation Based Learning (TBL) with morphological features and reports 78.67% accuracy on SPSAL corpus. [14] reports 92.36% accuracy on ISPC corpus using special linguistic features in a HMM model. [24] proposes an HMM model with morphological features and reports 93.05% accuracy. For Telugu language, [22] applies Transformation Based Learning (TBL) on top of a CRF model and reports 77.37% accuracy on SPSAL corpus. [14] uses various special linguistic features in a HMM model and reports 91.23% accuracy on ISPC corpus.

For Bengali language, [11] presents various supervised and semi-supervised Maximum Entropy and HMM models using morphological features and report 87.9% accuracy for semi-supervised HMM model on CIIL corpus. [13] reports 92.35% accuracy using a voted approach among various models. For Tamil language, [31] presents a linear programming based SVM model and reports 95.63% accuracy.

3 Problem Definition

Automated POS tagging is a classification task which takes a word or a sentence as input, assigns a POS tag or other lexical class marker to a word or to each word in the sentence, and produces the tagged text as output. In semi-supervised paradigm the POS tagger is built from a corpus of untagged sentences and a set of tagged sentences. The POS tagging classification problem is formally defined as follows:

Given a set of tags $\mathit{\Gamma} = \{T_1, T_2, \ldots, T_n\}$, an **annotated set** of tagged sentences $AS = \{St_1, St_2, \ldots St_N\}$, where $St_i = \langle W_1/T_i, W_2/T_j \ldots W_n/T_k\rangle$ (where W_i is a word and T_i is a tag from $\mathit{\Gamma}$) and a **raw untagged** training corpus of sentences $D = \{S_1, S_2 \ldots S_M\}$, where $S_i = \langle W_1 W_2 \ldots W_m\rangle$, the goal is to build a classifier model $\mathit{\Phi}$ which outputs the best tag sequence $\langle T_1 T_2 \ldots T_l\rangle$ for an input sequence of words $\langle W_1 W_2 \ldots W_l\rangle$.

4 TagMiner

4.1 Mapping POS tagging task to Association Rule Mining problem

According to the *one sense per collocation* [32] hypothesis, the sense of a word in a document is effectively determined by its *context*. The notion of context has been used in various methods of POS tagging [2, 30]. A context can occur in multiple places in the text. We refer to the list of occurrences of a context as a *context based list*. We use this idea for building TagMiner. In our method, we mine context based association rules from training data containing both tagged and untagged text. Our method works as follows:

- We collect all possible words occurring in the same context from the raw untagged data into a list called *context based list* (formally defined later). In this way we are able to find groups of words of *similar categories* from the raw untagged data.
- Using the annotated set and the tag finding algorithm (in Fig. 1), we find association rules of the form: $Context \Rightarrow Tag$ for the context based lists. Each rule maps a context based list to a suitable POS tag. These association rules work as the context based classification rules.
- Lastly, we group these context based association rules according to their POS tags to form clusters. This set of clusters is used as the classifier model to tag words using the method described in Sect. 6 and Fig. 2.

By experimenting with two varieties of bi-gram (one with preceding word and the other with succeeding word as context) and trigram as possible contexts we found that trigram works best for our method. For a word instance W_i, we fix its context as a trigram containing W_i in the middle and we use this context to find the *context based list*. Any other notion of context can be used as long as it fits into the formalism given below.

Context Based List: If Ψ is a function mapping from a word instance W_i in the data to its context $\Psi(W_i)$, then $\Psi^{-1}(\Psi(W_i))$ is a list of words instances sharing the same context. We refer to this list as *context based list* of $\Psi(W_i)$. It denotes words of similar category or type as W_i in a specific context and can store multiple instances of a word. For a given trigram $(W_{i-1}\ W_i\ W_{i+1})$ of words, $\Psi(W_i) = (W_{i-1}, W_{i+1})$. The preceding word W_{i-1} and succeeding word W_{i+1} are called *context words* and $\Psi(W_i)$ is called the *context word pair* of W_i.

Context Based Association Rule: For each context based list L, our approach finds association rule of the form $L \Rightarrow T$. This rule maps the context based list L to a POS tag T with *support* and *confidence* parameters defined below. Since each list L is obtained from a unique context word pair, so each association rule uniquely associates a context to a POS tag and works as the context based tagging rule.

In the following definitions and formulas we develop the intuition and the method to compute the interestingness measures of the significant association rules. The complexity in defining support is due to the presence of raw untagged training data required for semi-supervised learning. The support is the frequency (count) of occurrences of the context in the dataset. Context based lists are made from raw untagged data D and we are interested in the words of this list for which we know the tag in annotated set AS. Hence, we define Support of a context as follows:

AllTagContextSupport: Number of unique words of a context based list L whose tags are available (in annotated set AS) is denoted as $AllTagContextSupport(L)$. This measure gives the number of tagged words of L.

ContextSupport: For a list of words L in which duplicates may be present, $ContextSupport(L)$ is defined as the set of unique words present in L.

Coverage: For a *context based list* L,

$$Coverage(L) = \frac{AllTagContextSupport(L)}{|ContextSupport(L)|} \tag{1}$$

This measure represents the confidence that enough number of tagged samples are present in L.

ContextTagSupport: Number of unique words of a *context based list* L present in annotated set AS with a particular tag T is denoted as $ContextTagSupport(L, T)$.

Confidence: For a *context based list* L and tag T,

$$Confidence(L, T) = \frac{ContextTagSupport(L, T)}{|ContextSupport(L)|} \tag{2}$$

This measure represents the confidence that considerable number of words in list L have a particular tag T and leads to rules of the form $Context \Rightarrow Tag$.

WordTagSupport: Frequency of tag T for a word W in the annotated set AS is denoted as $WordTagSupport(T, W)$.
WordTagScore: For a word W and tag T, $WordTagScore$ is defined as:

$$WordTagScore(W, T) = \frac{WordTagSupport(T, W)}{\max\limits_{T_i \in \Gamma} WordTagSupport(T_i, W)} \tag{3}$$

This represents how good the tag fits the word on a scale of 0 to 1.
ListTagScore: For a tag T in *context based list* L, *ListTagScore* is defined as:

$$ListTagScore(L, T) = \frac{\sum\limits_{W_i \in ContextSupport(L)} WordTagScore(W_i, T)}{|\{W_i \in ContextSupport(L) : W_i/T \in AS\}|} \tag{4}$$

Where, **AS** is the annotated set. This formula represents the average frequency of tag T in *context based list* L. Intuitively, it represents how good the tag fits the list. Unfortunately, this is not always indicative of the correct tag for the list. For example, if a tag is overall very frequent, it can bias this score. Therefore, we compare this with the following score, inspired by the notion of *Conviction* [9].
BackgroundTagScore: For a tag T in annotated set AS, *BackgroundTagScore* is defined as:

$$BackgroundTagScore(T) = \frac{\sum\limits_{W_i \in ContextSupport(AS)} WordTagScore(W_i, T)}{|\{W_i \in ContextSupport(AS) : W_i/T \in AS\}|} \tag{5}$$

This represents the average frequency of tag T in annotated set AS.

4.2 POS Tagging Challenges

POS tagging, especially for resource poor languages, involves three major challenges listed below. In our approach we handle each of them explicitly.

1. **Data sparsity problem**: Some POS tag classes are present in the annotated set with very few representations. This is not enough to derive statistical information about them. In our approach, the use of raw untagged data reduces this problem (shown in Sect. 7.4).
2. **Class imbalance problem**: POS tag classes are highly imbalanced in their occurrence frequency. While selecting a tag this may lead to biasing towards the most frequent tags. Existing solutions of class imbalance problem typically favor rare classes [12]. However, while tagging the *context based lists*, we need to find POS tags for them in such a way that we neither favor frequent tags nor rare tags. We tackle this problem using a novel *Minmax* approach to find the best preferred POS tag instead of the most frequent one (described in Sect. 5.2).
3. **Phrase boundary problem**: Some lists are formed at phrase boundaries where the context comes from two different phrases. We need to filter out those *context based lists* which do not contain words of similar categories. In this case, the context of a word instance need not represent strong context and so the context based list may contain unrelated words. We use suitable parameters to handle this problem (explained in Sect. 5.3).

```
1. for each tag $T_i \in \varGamma$ present in annotated set $AS$ do:
2.     Find $BackgroundTagScore(T_i)$        // Use Equation (5)
3. for context based list $L$ do:
4.     Find $Coverage(L)$        // Use Equation (1)
5.     if $Coverage(L) \geq MinCoverage$:
6.         $ContextTagSupport(L, T_{max}) = \max_{T_i \in \varGamma} ContextTagSupport(L, T_i)$
7.         $Maxconf = Confidence(L, T_{max})$        // Use Equation (2)
8.         if $Maxconf > MinConfidence$:
9.             $MaxTset = \{T_i \mid ContextTagSupport(L, T_i) == ContextTagSupport(L, T_{max})\}$
10.            $BestPrefTag = FindBestPrefTag(L, MaxTset)$
11.            Return $BestPrefTag$
12.        else: Return NOTVALIST
13.    else: Return NOTVALIST

14. $FindBestPrefTag(L, MaxTset)$:
15.     Initialize $PrefTagset = \{\}$
16.     for each word $W$ of $ContextSupport(L)$ present in $AS$ do:
17.         $Tagset(W) = \{T_i \mid W$ has tag $T_i$ in $AS\}$
18.         $UnqTagset = Tagset(W) \cap MaxTset$
19.         Find $MaxWTag \mid WordTagSupport(MaxWTag, W) ==$
            $\max_{T_j \in UnqTagset} WordTagSupport(T_j, W)$
20.         $PrefTagset = PrefTagset \cup MaxWTag$
21.     Find $MinTag \in PrefTagset \mid \exists W_{min} \in ContextSupport(L)$ with
$WordTagSupport(MinTag, W_{min}) == \min_{W_i \in ContextSupport(L)} WordTagSupport(MinTag, W_i)$
22.     Find $ListTagScore(L, MinTag)$        // Use Equation (4)
23.     if $ListTagScore(L, MinTag) \geq BackgroundTagScore(MinTag)$: Return $MinTag$
24.     else: Return NOTVALIST
```

Figure 1: Algorithm to find POS tag for a *context based list*.

5 Building Classifier Model from Context Based Lists

5.1 Finding Association Rule for a Context Based List

The first step in our classifier model building method is to compute *context based lists* from an untagged training corpus D. It may be noted that a context based list can store multiple instances of a word. We use a sliding window of size three to collect the context based lists from D, in a single iteration, taking care of sentence boundaries.

In the next step we use the algorithm shown in Fig. 1 to find association rules for all the context based lists. In this algorithm, $BackgroundTagScore$ of all the POS tags present in the annotated set AS (lines 1-2) are computed first. Then for a context based list satisfying the threshold values of $Coverage$ and $Confidence$ (lines 3-9), function $FindBestPrefTag$ (described in Sect. 5.2) finds the best preferred tag (lines 10-11, 14-24) from the set of tags with maximum $ContextTagSupport$ (lines 7-9).

For a context based list L present as antecedent in association rule $L \Rightarrow T$, tag T returned by this algorithm becomes the consequent. This algorithm outputs best preferred tags for all the context based lists and hence finds association rules for all of them.

5.2 Handling Class Imbalance Problem

We handle the *class imbalance problem* by using a novel *Minmax* approach in the function $FindBestPrefTag$ (lines 14-24 in Fig. 1) and parameters $ListTagScore$ and $BackgroundTagScore$. In *Minmax* approach the preferred tag T_i for *context based list* L, is the one which has maximum $ContextTagSupport(L, T_i)$ but minimum $WordTagSupport(T_i, W)$ among those words of list L which have tag T_i as the best tag in AS. This takes care that the selected tag is supported by majority of the words in the list and is not biased by the most frequent tag of the annotated set.

To find the best preferred tag in function $FindBestPrefTag$, from the set of all the tags with maximum $ContextTagSupport$ value (line 9), at first we found those tags which were best tags (having maximum $WordTagSupport$ value) for the words of list L in AS (lines 15-20). Next, from this set of preferred tags we find the tag with minimum $WordTagSupport$ value (line 21). Then criteria $ListTagScore(L, T_i) \geq BackgroundTagScore(T_i)$ (lines 22-23) ensures that the selected tag has above average support in the annotated set and the context based list, both. If none of the tags satisfy this criteria, then we tag the list as "NOTVALIST" (line 24).

5.3 Handling Phrase Boundary Problem

To filter out context based lists with the *phrase boundary problem* (see Sect. 4.2) we use two suitable threshold values for parameters *Confidence* and *Coverage*. *Coverage* takes care of the fact that a context based list has considerable number of words to map it to a tag and *Confidence* ensures that the tag found for the list is the one which is supported by majority of the words in the list.

If context based list L has *Coverage* and *Confidence* values less than the corresponding threshold values $MinCoverage$ and $MinConfidence$, we tag L as "NOTVALIST" (lines 3-8, 12, 13 in Fig. 1). If L satisfies both of the threshold values then only we find the set of all the tags which have maximum value of $ContextTagSupport(L, T_i)$ and use this set (lines 9-10) to find the best preferred tag for the list (lines 14-24).

5.4 POS tag wise grouping of Association Rules to form Clusters

In the last step, we group context based lists according to their POS tags to get clusters of context based lists as classifier model. We exclude context based lists with tag "NOTVALIST" from the grouping process. Then we process these clusters to store word frequencies, corresponding context word pairs and their frequencies in each cluster. We represent the set of clusters as *Clustset*.

Since we are highly confident about the tags of the words present in the annotated set AS so, to improve cluster quality we apply a pruning strategy on the words of the clusters present in AS and remove those words from each cluster which do not have a matching cluster tag in AS. Finally, we get a set of clusters in which each cluster has a set of words with their frequencies and a set of associated context word pairs with their frequencies. Each cluster has a unique POS tag. These clusters are overlapping in nature and words can belong to multiple clusters.

6 POS tagging Method

To tag the words of a test sentence we make use of the test word's context word pair, preceding word and the word frequency in a cluster to decide the tag of the word (see Fig. 2). When a test word is found in only one cluster then we output the cluster tag as the tag of the test word. But when a test word is found in many clusters, then to select the suitable clusters following priority order is followed:

1. **Criteria 1:** Highest priority is given to the presence of matching context word pair of the test word in the clusters.
2. **Criteria 2:** Second highest priority is given to the presence of matching preceding word of the test word as first word of the context word pairs in clusters.
3. **Criteria 3:** Last priority is given to the frequency of the test word in the clusters.

For test words not present in any cluster we use criterion 1 and 2 to select appropriate clusters. Based on the priority order, only one of the criterion is used to select the suitable clusters. If we are not able to find any suitable cluster then we return "NOTAG" as the tag of the test word.

Even when we find suitable clusters, to increase precision, our method finds POS tags only for those cases where it is confident. It avoids to wrongly classify non confident cases and returns "NOTAG" for them. This is especially useful when the cost of misclassifying (false positive) is high. This also gives opportunity to integrate other language/domain specific POS taggers as they can be used for the non-confident cases.

After selecting the suitable clusters we need to make sure that we have enough confidence in the highest probability tag obtained from the clusters. To ensure this we use the parameter *TagProbDif*, which gives the fractional difference between the highest and the second highest cluster tag probabilities and is defined as follows:

$$TagProbDif = \frac{TagProb(C_{max}) - TagProb(C_{secmax})}{TagProb(C_{max})} \tag{6}$$

Where, C_{max} is the cluster with highest $TagProb(C_i)$ value and C_{secmax} is the cluster with second highest $TagProb(C_i)$ value. $TagProb(C_i)$ of a cluster is defined as follows:

$$TagProb(C_i) = \frac{\text{Frequency of X in } C_i}{\sum\limits_{\forall C_j \in Clustset} \text{Frequency of X in } C_j} \tag{7}$$

Where, X is set as follows: If the test word is present in cluster C_i then $X =$ test word. For test word not present in any cluster, if the clusters are selected based on the presence of the context word pair of the test word then $X =$ context

```
for each word $Wmid$ in sentence $S$ with context word pair $CW_p$ and $CW_s$ do:

1. Initialize $PredClustset = \{\}$
2. if $\exists$ cluster $C_i \in Clustset \mid Wmid \in C_i$:
   (a) Find $PClustset = \{C_i \mid Wmid \in C_i\}$
   (b) if $\exists$ cluster $C_j \in PClustset \mid CW_p$ and $CW_s$ pair is present as context word pair in
       cluster $C_j$:
         Find all such clusters from $PClustset$ and append to $PredClustset$    #Criteria 1
   (c) else:
         if $\exists$ cluster $C_j \in PClustset \mid CW_p$ is present as preceding word in a context word
           pair in cluster $C_j$:
             Find all such clusters from $PClustset$ and append to $PredClustset$   #Crit. 2
         else: Append $PredClustset = PredClustset \cup PClustset$      #Criteria 3
3. else:
   (a) if $\exists$ cluster $C_i \in Clustset \mid CW_p$ and $CW_s$ pair is present as context word pair in
       cluster $C_i$:
         Find all such clusters from $Clustset$ and append to $PredClustset$    #Criteria 1
   (b) else:
         if $\exists$ cluster $C_i \in Clustset \mid CW_p$ is present as preceding word in a context word
           pair in cluster $C_i$:
             Find all such clusters from $Clustset$ and append to $PredClustset$   #Crit. 2
         else: Return NOTAG
4. $\forall C_i \in PredClustset$ Find $TagProb(C_i)$          // Use Equation 7
5. Find $C_{max} =$ cluster with highest $TagProb(C_i)$ value in $PredClustset$
6. Find $C_{secmax} =$ cluster with second highest $TagProb(C_j)$ value in $PredClustset$
7. Find $TagProbDif$        // Use Equation 6
8. if $TagProbDif \geq Minprobdif$: Return $PredTag =$ POS tag label of cluster $C_{max}$
9. else: Return NOTAG
```

Figure 2: Method to tag words of a sentence using set of clusters $Clustset$.

word pair. If the clusters are selected based on the presence of the preceding word of the test word as first word of the context word pairs in clusters then X = preceding word of the test word. In this way we are able to tag some *unseen/unknown* words also which are not present in the training data. This, in a way, acts as an alternative of smoothing technique for them.

After selecting the clusters (based on priority order) we compute their $TagProb$ values using (7) and then compute $TagProbDif$ using (6). For $TagProbDif$ value above a suitable threshold value $Minprobdif$ we output the tag of cluster with highest $TagProb$ value as the tag of the test word, otherwise we return "NOTAG"(see Fig. 2).

7 Experiments, Results and Observations

7.1 Dataset Details

We have done our experiments on resource-rich English[1](uses Biber tag set [17]), resource-moderate Hindi [3, 4] and resource-poor Telugu[2] [3], Tamil[3] and Ben-

[1] New York Times dataset of American National Corpus available at `http://americannationalcorpus.org/FirstRelease/contents.html`

[2] Provided by IIIT Hyderabad, data is part of IL-ILMT project sponsored by MC&IT, Govt. of India Reference No: 11(10)/2006-HCC(TDIL)

[3] Available at `http://sanskrit.jnu.ac.in/ilci/index.jsp`

gali[4] languages. Table 1 gives details of all the language datasets. All the five language datasets have flat tag sets present in annotated training and test sets without any hierarchy and considerable number of lexical ambiguities are also present. We note that except English all the other four languages are morphologically rich and have free word-order property. The POS tag data distribution in the resource-moderate and resource-poor language datasets are highly imbalanced and sparse.

Table 1. Details of all language datasets with accuracy values obtained by **TagMiner**.

	Hindi	**Telugu**	**Tamil**	**Bengali**	**English**
No. of Words in Raw Untagged Training set	393303	104281	169705	85796	1293388
No. of Words in Annotated Training set	282548	83442	20207	21561	629532
No. of POS Tags in Annotated Training set	35	28	28	27	109
No. of Words in Test set	70811	20854	22352	20618	471977
No. of POS Tags in Test set	32	24	27	29	105
No. of Test Words tagged as NOTAG by TagMiner	1916	1634	2647	3448	9385
Average Accuracy (%) (Equation 8)	**87.8**	**87.6**	**83.46**	**76.17**	**88.5**
Resource Type	Moderate	Poor	Poor	Poor	Rich

7.2 Performance Analysis and Observations

We observed that following set of threshold values $MinConfidence = 60\%$, $MinCoverage = 60\%$ and $MinprobDif = 30\%$ for the three parameters gives best $AverageAcuracy$ (defined below) values for all the five languages. Tables 1 and 2 show the results for this set of parameter values.

$$AverageAccuracy = \frac{\text{Number of correctly tagged test words}}{|\text{Test set}| - \text{No. of test words tagged as NOTAG}} \tag{8}$$

Where, |Test set| = No. of words in the test set.

For both known and unknown test words, for all the five languages, maximum number of correct tagging was done by giving highest priority to presence of context word pair in the cluster. Here, *known* words means test set words which are present in untagged training set and *unknown* word means unseen test set words which are not present in the untagged training set. Note that words of annotated set are not included in the classifier model, only their tags

[4] Available at `http://sanskrit.jnu.ac.in/ilci/index.jsp`

are used indirectly while building the model. In the results shown in Table 1, around 46% unknown English words, 60% unknown Hindi words, 67% unknown Telugu words, 52% unknown Bengali words and 57% unknown Tamil words were correctly tagged using their context word pair. This shows the strength of our tagger to tag unknown words without using any smoothing technique used by other POS taggers.

In Table 2, we compare our results with a supervised CRF[5] tagger [19]. This tagger uses words, their POS tag and context word pair information from annotated data, while our tagger uses words and their context word pair information from untagged data and POS tag information from annotated data. We observe that for annotated data size $\leq 25K$ words, our tagger gives better *AverageAccuracy* than CRF tagger. Our tagger also gives better POS tag precisions and better tagging accuracies than CRF tagger for unknown words and performance improves by increasing the untagged data size up to a certain size. This shows that our tagger can be a better choice for resource poor languages. Also, as an additional benefit model made by our tagger is more human understandable than model made by CRF tagger.

7.3 Effect of Annotated (POS tagged) Data Size

We varied the size of annotated set of Tamil (see Table 3) while keeping the raw untagged set constant and observed that the coverage of words by the clusters in the classifier model increases with the increase in the size of annotated data, the tagging accuracy increases while the number of words missed by the model (tagged as "NOTAG") decreases. For all languages we observed that increasing the annotated training data size improves cluster quality which increases the *AverageAcuracy* values but only up to a certain size. We also observed that there is only a slight decrease in *AverageAcuracy* value with decrease in annotated set size, so performance does not decrease drastically when the annotated set is made smaller. Our tagger gives above 70% *AverageAcuracy* for annotated data size as low as 5K and raw untagged data size 10K on all the languages. This justifies the use of small annotated set to build a semi-supervised POS tagging model for resource poor languages.

7.4 Effect of Raw Untagged Data Size

In Tables 1, 2 and 4 , we observe that increasing the raw untagged training data size initially increases word coverage of clusters which in turn increases the *AverageAcuracy* values but stabilizes after a certain size. For all languages we observed that the coverage of words by the clusters in the classifier model increases with the increase in the size of untagged data (while keeping the size of annotated set constant). This accounts for the increase in tagging accuracy and decrease in the number of words missed by the model (tagged as NOTAG). Other interesting observation is that *AverageAccuracy* does not vary much as

[5] Available at `http://crfpp.googlecode.com/svn/trunk/doc/index.html`, CRF model outputs tag for all test words. So, for CRF tagger AverageAccuracy = (No. of correctly tagged test words)/(No. of test words).

Table 2. Average Accuracy values for all languages obtained by CRF tagger and our tagger **TagMiner** for various annotated training set sizes (≤ 25000 words).

Lang.	Test set size	Annotated Training set size	CRF	TagMiner		
			Average Accuracy (%)	Average Accuracy (%)	No. of NOTAG Words	Untagged Training set size
Hindi	20227	5730	74.6	**79.1**	3195	10025
		10030	78.4	**79.05**	2740	10025
		15771	81.3	**82.1**	2116	25020
		25591	84.7	**85.0**	1903	50019
Telugu	20854	4994	59.4	**81.4**	5617	9994
		9994	67.1	**82.6**	3897	23435
		14995	71.2	**84.4**	3240	43434
		23435	75.3	**84.3**	2419	104281
Tamil	22352	5006	48.9	**75.0**	6957	40988
		9941	59.9	**79.7**	4357	80004
		15007	65.9	**82.1**	3778	80004
		20207	69.4	**83.1**	3495	80004
Bengali	20618	5010	47.3	**73.5**	7081	49997
		10003	56.2	**74.6**	5332	85796
		15009	59.3	**75.2**	4269	85796
		21561	63.0	**77.8**	4170	85796
English	24952	10671	70.4	**79.7**	3774	50444
		15298	72.9	**82.3**	3424	50444
		24825	76.4	**82.5**	2574	93679

Table 3. Effect of annotated data size on classifier model on 22352 Tamil test set words for $MinCoverage = 60\%$, $MinConfidence = 60\%$, $Minprobdif = 30\%$ and 169705 raw untagged words.

No. of Words in Annotated set	No. of Clusters in model (No. of unique words)	No. of NOTAG test Words	Average Accuracy (%)
5006	22 (2021)	5317	72.2
9941	25 (3553)	3575	79.0
15007	26 (4842)	2940	82.09
20207	26 (5774)	2647	83.46

the untagged data size varies, so our algorithm is able to perform well even with a small sized untagged data.

7.5 Effect of Various Parameters

We made the following observations about the effect of parameter values: (1) Increasing threshold values of $MinConfidence$ for parameter *Confidence*, it in-

Table 4. Effect of raw untagged data size on classifier model on 70811 Hindi test set words and 20854 Telugu test set words for $MinCoverage = 60\%$, $MinConfidence = 60\%$ and $Minprobdif = 30\%$ with 282548 annotated Hindi words and 104281 annotated Telugu words.

Language	**No. of Words in Raw set**	**No. of Clusters in model (No. of unique words)**	**No. of NOTAG test Words**	**Average Accuracy (%)**
Hindi	**50019**	25 (4366)	4714	**87.3**
	98331	28 (6081)	3664	87.7
	128329	28 (6865)	3220	87.9
	158337	29 (7546)	2890	87.9
	188326	29 (8112)	2793	88.0
	196659	29 (8260)	2748	88.0
	282554	30 (9517)	2484	88.0
	294979	30 (9663)	2450	**88.1**
	393303	30 (10817)	1916	87.8
Telugu	**23435**	23 (3600)	3079	**86.1**
	43434	23 (5091)	2276	87.4
	63436	23 (6221)	1828	87.4
	83442	23 (7198)	1749	**88.2**

creases the quality of clusters but at the same time it also increases the number of *context based lists* tagged as "NOTVALIST" which decreases the word coverage of clusters. (2) Decreasing threshold values of $MinCoverage$ for parameters *Coverage* although decreases the quality of clusters but at the same time it increases the word coverage of clusters by decreasing the number of *context based lists* tagged as "NOTVALIST". (3) By varying the threshold value of *Minprobdif* from 5% to 30% for parameter *TagProbDif* we found that increasing the threshold value increases the precision values of POS tags but slightly decreases their recall because the number of words tagged as "NOTAG" increases. Practical advantage of this parameter is that it ensures that tagging of ambiguous and non-confident cases is avoided. (4) The number of POS tag clusters obtained in the classifier model is almost independent of the selected threshold values of the parameters. For the datasets given in Table 1 and for the range of threshold values $MinConfidence = 60\%$ to 90% and $MinCoverage = 0\%$ to 75%, number of POS tag clusters found for English was 100 to 101, for Hindi was 29 to 31, for Tamil was 22 to 26, for Bengali was 25 and for Telugu was 23. We noted that the POS tags missing from the set of clusters were the rare POS tags having very low frequencies.

8 Conclusions and Future Work

In this work we developed TagMiner, a semi-supervised associative classification method for POS tagging. We used the concept of context based list and context based association rule mining. We developed a method to find interestingness measures required to find the association rules in a semi-supervised manner from a training set of tagged and raw untagged data combined. We showed that TagMiner gives good performance for resource rich as well as resource poor languages without using extensive linguistic knowledge. It works well even with less tagged training data and less untagged training data. It can also tag unknown words. To some extent, it handles class imbalance and data sparsity problems using the untagged data and a special method to find interestingness measures. It handles phrase boundary problem using a set of parameters. These advantages make it very suitable for resource poor languages and can be used as an initial POS tagger while developing linguistic resources for them.

Future work includes (1) using other contexts instead of trigram, (2) finding methods to include linguistic features in the current approach, (3) mining tagging patterns from the clusters to find tag of a test word and (4) using this approach for other lexical item classification tasks.

References

1. Agrawal, R., Imieliński, T., Swami, A.: Mining Association Rules Between Sets of Items in Large Databases. In: Proc. of SIGMOD. pp. 207–216 (1993)
2. Banko, M., Moore, R.C.: Part-of-Speech Tagging in Context. In: Proc. of COLING (2004)
3. Bharati, A., Misra Sharma, D., Bai, L., Sangal, R.: AnnCorra : Annotating Corpora Guidelines For POS And Chunk Annotation For Indian Languages. Tech. Rep. TR-LTRC-31, Language Technologies Research Centre, IIIT, Hyderabad (2006)
4. Bhatt, R., Narasimhan, B., Palmer, M., Rambow, O., Sharma, D.M., Xia, F.: A multi-representational and multi-layered treebank for Hindi/Urdu. In: Proc. of the Third Linguistic Annotation Workshop. pp. 186–189 (2009)
5. Biemann, C.: Unsupervised Part-of-Speech Tagging Employing Efficient Graph Clustering. In: Proc. of ACL (2006)
6. Brants, T.: TnT: a statistical part-of-speech tagger. In: Proc. of ANLP. pp. 224–231 (2000)
7. Brill, E.: A Simple Rule-Based Part of Speech Tagger. In: Proc. of ANLP. pp. 152–155 (1992)
8. Brill, E.: Transformation-Based Error-Driven Learning and Natural Language Processing: A Case Study in Part-of-Speech Tagging. Comput. Linguist. 21(4), 543–565 (1995)
9. Brin, S., Motwani, R., Ullman, J.D., Tsur, S.: Dynamic Itemset Counting and Implication Rules for Market Basket Data. In: Proc. of SIGMOD. pp. 255–264 (1997)
10. Cutting, D., Kupiec, J., Pedersen, J., Sibun, P.: A practical part-of-speech tagger. In: Proc. of the third conference on ANLP. pp. 133–140 (1992)
11. Dandapat, S., Sarkar, S., Basu, A.: Automatic Part-of-speech Tagging for Bengali: An Approach for Morphologically Rich Languages in a Poor Resource Scenario. In: Proc. of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions. pp. 221–224 (2007)
12. Dubey, H., Pudi, V.: Class Based Weighted K-Nearest Neighbor over Imbalance Dataset. In: Proc. of PAKDD (2). pp. 305–316 (2013)

13. Ekbal, A., Hasanuzzaman, M., Bandyopadhyay, S.: Voted Approach for Part of Speech Tagging in Bengali. In: Proc. of PACLIC. pp. 120–129 (2009)
14. Gadde, P., Yeleti, M.V.: Improving statistical POS tagging using Linguistic feature for Hindi and Telugu. In: Proc. of ICON (2008)
15. Gimenez, J., Marquez, L.: Svmtool: A general pos tagger generator based on support vector machines. In: Proc. of LREC. pp. 43–46 (2004)
16. Goldwater, S., Griffiths, T.: A fully Bayesian approach to unsupervised part-of-speech tagging. In: Proc. of the 45th Annual Meeting of the ACL. pp. 744–751 (June 2007)
17. Ide, N., Suderman, K.: The American National Corpus First Release. In: Proc. of LREC. pp. 1681–1684 (2004)
18. Kamruzzaman, S.M., Haider, F., Hasan, A.R.: Text Classification using Association Rule with a Hybrid Concept of Naive Bayes Classifier and Genetic Algorithm. CoRR abs/1009.4976 (2010)
19. Lafferty, J.D., McCallum, A., Pereira, F.C.N.: Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In: Proc. of ICML. pp. 282–289 (2001)
20. Li, W., Han, J., Pei, J.: CMAR: Accurate and Efficient Classification Based on Multiple Class-Association Rules. In: Proc. of ICDM. pp. 369–376 (2001)
21. Liu, B., Hsu, W., Ma, Y.: Integrating Classification and Association Rule Mining. In: Proc. of KDD. pp. 80–86 (1998)
22. P.V.S., A., G., K.: Part Of Speech Tagging Using Conditional Random Fields and Transformation Based Learning. In: Proc. of IJCAI Workshop SPSAL. pp. 21–24 (2007)
23. Shaohong, Y., Guidan, F.: Research of POS Tagging Rules Mining Algorithm. Applied Mechanics and Materials 347-350, 2836–2840 (August 2013)
24. Shrivastava, M., Bhattacharyya, P.: Hindi POS Tagger Using Naive Stemming: Harnessing Morphological Information Without Extensive Linguistic Knowledge. In: Proc. of ICON (2008)
25. Søgaard, A.: Semisupervised condensed nearest neighbor for part-of-speech tagging. In: Proc. of ACL HLT: short papers - Volume 2. pp. 48–52 (2011)
26. Soni, S., Vyas, O.: Using Associative Classifiers For Predictive Analysis In Health Care Data Mining. Int. Journal Of Computer Application 4(5), 33–37 (July 2010)
27. Subramanya, A., Petrov, S., Pereira, F.: Efficient graph-based semi-supervised learning of structured tagging models. In: Proc. of EMNLP. pp. 167–176 (2010)
28. Thabtah, F.: A Review of Associative Classification Mining. The Knowledge Engineering Review 22(1), 37–65 (Mar 2007)
29. Thonangi, R., Pudi, V.: ACME: An Associative Classifier Based on Maximum Entropy Principle. In: Proc. of ALT. pp. 122–134 (2005)
30. Toutanova, K., Klein, D., Manning, C.D., Singer, Y.: Feature-rich part-of-speech tagging with a cyclic dependency network. In: Proc. of NAACL HLT'03 - Volume 1. pp. 173–180 (2003)
31. V., D., Kumar, A., G., S., P., S.K., S., R.: Tamil POS Tagging using Linear Programming. Int. Journal of Recent Trends in Engineering 1(2), 166–169 (May 2009)
32. Yarowsky, D.: One sense per collocation. In: Proc. of the workshop on Human Language Technology. pp. 266–271 (1993)
33. Yin, X., Han, J.: CPAR: Classification based on Predictive Association Rules. In: Proc. of SDM. pp. 331–335 (2003)
34. Zaïane, O.R., Antonie, M.L., Coman, A.: Mammography Classification By an Association Rule-based Classifier. In: Proc. of MDM/KDD. pp. 62–69 (2002)

Sequential Patterns of POS Labels Help to Characterize Language Acquisition

Isabelle Tellier[1,2], Zineb Makhlouf[1], Yoann Dupont[1]

(1) Lattice, CNRS - UMR 8094, (2) University Paris 3 - Sorbonne Nouvelle

Abstract. In this paper, we try to characterize various steps of the syntax acquisition of their native language by children with emerging sequential patterns of Part Of Speech (POS) labels. To achieve this goal, we first build a set of corpora from the French part of the CHILDES database. Then, we study the linguistic utterances of the children of various ages with tools coming from Natural Language Processing (morpho-syntactic labels obtained by supervised machine learning) and sequential Data Mining (emerging patterns among the sequences of morpho-syntactc labels). This work thus illustrates the interest of combining both approaches. We show that the distinct ages can be characterized by variations of proportions of morpho-syntactic labels, which are also clearly visible inside the emerging patterns.

Keywords. language acquisition, POS labeling, CRF, Sequential Data Mining, emerging patterns

1 Introduction

The acquisition of their native language by children, especially how grammatical constructions are gradually mastered, is a process which largely remains mysterious. Some researches address this issue within a Natural Language Processing framework, for example by implementing programs trying to mimic the learning process [CM06,Ali10]. Our approach in this paper is different: we do not target to reproduce, but to *mine* children productions, from a morphosyntactic point of view. More precisely, we study the linguistic utterances of children of various ages, seen as sequences of part-of-speech (POS) labels, with sequential data mining tools.

Sequential data mining can be applied to any kind of data following an order relation. This relation is often related to time; for texts, it is only the linear order of words in sentences. Sequential data mining allows to extract sequential patterns, that is sequences or sub-sequences of itemsets that repeatedly occur in the data. This domain has given rise to many works [AS95,SA96,Zak01,NR07]. If the extracted sequences are contiguous portions of texts, patterns coincides with the older notion of repeated segments [Sal86].

When data are composed of natural language texts, the itemsets are not necessarily reduced to words: lemmas and POS labels can also be taken into account. The use of sequential data mining technics in such a linguistic context has recently been tested for the extraction of Named Entities [NAFS13], the discovery

In: P. Cellier, T. Charnois, A. Hotho, S. Matwin, M.-F. Moens, Y. Toussaint (Eds.): Proceedings of DMNLP, Workshop at ECML/PKDD, Nancy, France, 2014.

of relations between entities in the biological field [CPRC09,CCP10,BCCC12] or the study of stylistic differences between textual genres [QCCL12]. As we look at the emergence of grammatical constructions in children, we are mainly interested here in patterns of morpho-syntactic labels. As a matter of fact, they are more general than words or lemmas and provide more abstract characterizations of a given age. We seek in particular to exhibit specific *emerging patterns* for different age groups.

The remaining of the article is as follows. First, we present the way our corpora of children's productions of different age groups have been collected. Then, we explain how we processed their morpho-syntactic analysis. Observing that usual POS taggers available for French made many mistakes on our data, we have built a new one, by training a machine learning device (a CRF model) on a reduced set of manually corrected data. We show that, despite this reduced set of manual corrections, the new tagger obtained behaves far better than the previous one on our data. Finally, the last part of the paper describes the technique used for the extraction of n-grams of morpho-syntactic labels of each specific age group and provides quantitative and qualitative analyses of the corresponding emerging patterns.

2 Corpora

2.1 The CHILDES Corpus

Several resources collecting children's productions exist online, as those available in the CNRTL[1]. But the best known and most widely used database is CHILDES[2] [Elm01], a multilingual corpus of transcriptions of recorded interactions between adults and children. In this article, we are only interested in the French part of these data. The recordings of a child cover several months or years, the age of the children may therefore vary from one record to another. Relying on the transcription manual[3] which explicits the meta-data associated with the corpus, we created six different sub-corpora corresponding to six age groups: from the "1-2 years" to the "6-7 years".

2.2 Pretreatments

In this corpus, children and parents communicate by speech turns. Each speech turn is transcribed and delimited by a period. In the following, we consider that each line corresponds to a "sentence". The transcriptions are annotated and are often followed by additional information in a (semi-)standard format allowing to describe elements of the situation (e.g. objects which are in the scene). We performed a preprocessing step to focus only on linguistic productions. We have

[1] Centre National des Ressources Textuelles et Linguistiques (http://www.cnrtl.fr for children's production): see Traitement de Corpus Oraux en Français (TCOF) corpus

[2] http://childes.psy.cmu.edu/

[3] http://childes.psy.cmu.edu/manuals/CHAT.pdf

removed all special characters related to standards of transcription, as well as all information of phonetic nature, which are not relevant for the analysis of syntactic constructions and prevent the use of a tagger. We have also eliminated from our data all adult utterances.

The characteristics of each of our initial sub-corpora are presented in the table of Figure 1. There are differences between them: the corpus for the age of "6-7 years" is the smallest one. To balance the corpora of the different age groups, we have sampled them according to the *number of words*: this feature is more reliable than the number of sentences, because the length of the sentences is a key factor which significantly varies from one age to another (see the following). To have comparable sub-corpora, the number of words is thus more reliable than the number of sentences.

corpus	number of sentences	number of words	nb of distinct words	average length of the sentences
1-2 years	41786	63810	3019	1.23
2-3 years	115114	324341	8414	2.15
3-4 years	60317	243244	8479	4.62
4-5 years	16747	74719	4465	4.71
5-6 years	4542	29422	938	6.96
6-7 years	3383	21477	841	6.88

Fig. 1. Characteristics of the initial sub-corpora

2.3 Sampling

The smallest corpus in terms of words (the one of "6-7 years") is the reference sample for the other age groups. So, we chose to take 20,000 words per corpus, with a rate of 0.01% tolerance. To build our new corpora from the initial ones, we sampled sentences randomly until the sum of all words in all sentences reaches this size. After the sampling, we have six new corpora, whose properties are given in the table of Figure 2.

The corpora now have comparable size in terms of words. The number of sentences in each corpus have of course decreased, but we note that the average lengths of the sentences follow the same evolution than in the initial corpora. This is crucial because, as long as the children grow up, they tend to produce longer sentences. This is a well-known key feature of language acquisition [Bro73,MC81]. To go further in our exploration, we will now label the productions of the children with morpho-syntactic labels.

corpus	number of sentences	number of words	nb of distinct words	average length of the sentences
1-2 years	14284	20348	1086	1.42
2-3 years	9075	20504	1427	2.26
3-4 years	5043	21051	1575	4.17
4-5 years	4433	20949	1806	4.73
5-6 years	3047	20514	805	6.73
6-7 years	3147	20525	819	6.52

Fig. 2. Characteristics of the sampled sub-corpora

3 POS labeling

3.1 Use of an existing tagger

As we want to characterize the acquisition of syntactic constructions, we need more information than simple transcriptions of words. Our experiments in this article rely on a morpho-syntactic tagging of children's productions: we must thus assign to each word in the sub-corpora a label corresponding to its grammatical category. Several tools are available to annotate plain text in French with "Part of Speech" (POS) labels, such as TreeTagger [Sch94]. In our work, we have used SEM[4] [TDE$^+$12], which was obtained by training a linear CRF (Conditional Random Fields) model on the French Treebank [ACT03]. The set of labels adopted in SEM, similar to the one of [CC08], includes 30 different categories among which the main important ones for the following are: NC (for common nouns), V (for verbs), DET (for determiners), P (for prepositions), I (for interjections) and CLS (for subject clitic). SEM also integrates the external lexical resource Lefff [CSL04] to help achieve a better labeling.

SEM has been learned with labeled sentences extracted from the French newspaper "Le Monde". Our texts of children productions have very different properties, and we therefore expect many annotation errors. Indeed, the corpus CHILDES is composed of oral transcription, whose conventions differ from those of writing (especially concerning punctuations). Furthermore, children utterances are often far from standard French. It has already been observed that, even if SEM is supposed to reach 97% accuracy on texts similar to those on which it has been learned, it reaches 95.6% accuracy on more casual written texts from blogs, and only 81.6% on oral productions of adults.

To assess the quality of SEM on our data, we have randomly selected 200 sentences from each of our six corpora, tagged them with SEM and manually corrected the labeling errors, following the annotation conventions of the French Treebank. The accuracy of SEM on these samples (see table of Figure 4) ranges from 70% (2-3 years) to 87% (6-7 years). The detailed F-measures of the main categories for each age group can also be seen in the table of Figure 3: the label interjection (I), very rare in the French Treebank but very frequent in our

[4] http://www.lattice.cnrs.fr/sites/itellier/SEM.html

corpora, are particularly not well recognized by SEM (the F-measures goes from 33.33 for the "1-2 years" age group to 0 for the the "6-7 years" one).

3.2 Learning a New tagger

As we want to perform statistical measures on the morpho-syntactic labels, labeling errors must be reduced as much as possible. In [TDEW13], it has been shown that to learn a good tagger by supervised machine learning, it is more efficient to have a small annotated corpus similar to the target data than to have a large too different training set. So, we decided to use the labelled sentences which have been manually corrected for the evaluation of SEM as training data to learn a new tagger adapted to our corpora.

For this, we have used the same tools as those used to learn SEM, that is CRFs (Conditional Random Fields), introduced by [LMP01] and implemented in the software Wapiti [LCY10]. CRFs are graphical models that have proven their effectiveness in the field of automatic annotation by supervised machine learning [TTA09,TDE^{+}12]. They allow to assign the best sequence of annotations y to an observable sequence x. For us, the elements of x are words enriched with endogenous attributes (presence of caps, digits, etc.) or exogenous ones (e.g. associated properties in Lefff), while y is the corresponding sequence of morpho-syntactic labels.

We trained our new tagger thanks to $200 * 6 = 1200$ annotated and manually corrected sentences (which is a very small number to learn a POS tagger), and we tested it on $50 * 6 = 300$ other independent sentences, equally sampled from the 6 distinct sub-corpora. The table of Figure 3 gives the F-measures of the main labels obtained by SEM and by the re-learned tagger for each age group, while the accuracy of both taggers are provided in the table of Figure 4.

corpus	CLS	DET	I	NC	P	V
1-2 years	100/100	80/100	33.33/57.14	76.92/84.21	0/0	80/100
2-3 years	71.43/93.33	66.67/54.55	12.5/90.91	71.43/80	40/33.33	71.43/63.64
3-4 years	77.42/100	80/78.26	13.33/88.89	88.89/94.74	71.43/71.43	83.87/94.74
4-5 years	89.8/94.55	80.95/89.36	8.7/97.78	75.76/93.15	90.91/80	88.89/95.89
5-6 years	81.08/97.56	91.18/93.15	0/94.74	86.32/96.08	78.05/88.89	92.96/90.14
6-7 years	96.55/100	87.88/97.14	0/80	90/92.13	89.47/87.8	93.88/89.36

Fig. 3. F-measures of the main distinct labels before (with SEM) /after the re-learning

We observe that the relearning leads to a significant improvement of the accuracy of about 10% in average. SEM is better for only 4 cells out of 36 in the table of Figure 3, probably thanks to its better vocabulary exposure: the French Treebank on which SEM was learned was about ten times larger than our training corpus. The improvement brought by relearning is larger for oral-specific labels such as I. It is therefore very beneficial, despite a very small training corpus. This

corpus	SEM	re-trained tagger
1-2 years	82%	85%
2-3 years	70%	80%
3-4 years	73%	88%
4-5 years	75%	90%
5-6 years	80%	92%
6-7 years	87%	90%
average	**77.83%**	**87.5%**

Fig. 4. Impact of the re-learning on the accuracy of the distinct age groups

can be explained by the fact that the vocabulary used in our texts is relatively limited and redundant: few data are therefore sufficient to obtain a tagger which is effective on our corpus, even if it is not uniformly better than SEM on every label (it would obviously be much less effective on other types of data). In the following, we systematically use the new version of the tagger.

3.3 Analysis of POS labels

Figure 5 shows the distribution of the main morpho-syntactic categories in the different age groups. For example, we see that the curve of the label I (interjection) is decreasing (except for the 4-5 years age group): it seems that children use fewer and fewer interjections in their productions as long as grow up. In contrast, the label P (preposition) is strictly increasing, which is consistent with an acquisition of increasingly sophisticated syntactic constructions. Curves for the labels CLS (subject clitic) and V (verb) follow very similar variations, probably because they are often used together: they increase till the age of 4, then decrease from 4 to 6, and finally stabilize at the age of 6. Observing labels DET (determiner) and NC (common nouns), we notice that until the age of 4 years, NC is the most common label, but not yet being systematically associated with a DET. It is only at the age of 4 that both curves become parallel (most probably when most NC is preceded by a DET). We finally note that from the age of 5 years, the proportions of different labels stabilize.

The residual errors of the tagger (there is more than 10% remaining labeling errors) lead us to be prudent with these observations. But it is clear that some of the phenomena observed here would not have been possible without re-learning: interjections, for example, were the words most poorly recognized by the original SEM, because they are very rare in newspaper articles. However, their production appears to be an important indicator of the child's age group. Example sentences like "ah maman" ("ah mom") or "heu voilà " ("uh there") were respectively labeled as "ADJ NC" and "ADV V" with the original SEM tagger. After the re-learning, the labels became "I NC " and "I V", which is at least more correct.

Although we can already draw some interesting conclusions from these curves, we cannot characterize the syntactic acquisition of children from single isolated

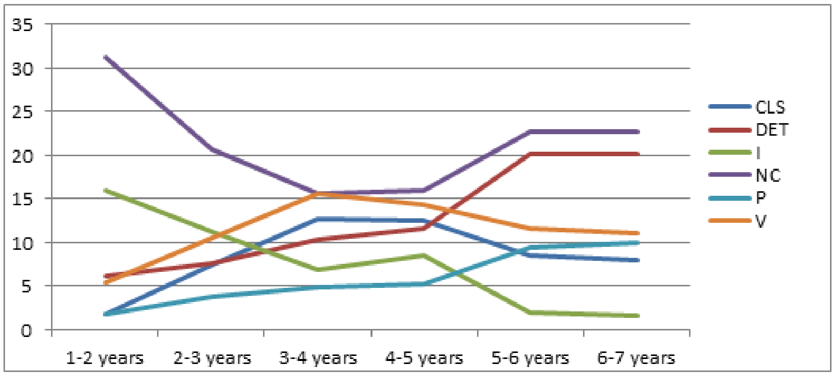


Fig. 5. Proportions of each label for each age group

categories. We thus decided to use sequential data mining techniques on our data to explore them further.

4 Sequential Patterns Extraction

4.1 General Definitions

Many studies have focused on the analysis of texts seen as sequential data. For example, the notion of *repeated segment* is used in textometrics [Sal86] to characterize a contiguous sequence of items appearing several times in a text. Sequential data mining [AS95] generalizes such concept, with notions like sequential patterns of itemsets. In our case, itemsets can be composed of words and POS labels. A sequence of itemsets is an ordered list of itemsets. An order relation can be defined on such sequences: a sequence $S_1 = \langle I_1, I_2, ..., I_n \rangle$ is included into a sequence $S_2 = \langle I'_1, I'_2, ..., I'_m \rangle$, which is noted $S_1 \subseteq S_2$, if there exist integers $1 \leq j_1 \leq j_2 \leq ... \leq j_n \leq m$ such that $I_1 \subseteq I'_{j_1}, I_2 \subseteq I'_{j_2}, ..., I_n \subseteq I'_{j_n}$ (in the classical sense of itemset inclusion). The table of Figure 6 provides examples of sequences of itemsets found in our corpus labelled with the re-trained tagger.

The support of a sequence S, denoted $sup(S)$, is equal to the number of sentences of the corpus containing S. For example, in the table of Figure 6, $sup(\langle(\text{ADJ})\ (\text{NC})\rangle) = 2$. The *relative support* of a sequence S is the proportion of sequences containing S in the base of initial sequences. It is worth $\frac{1}{2}$ for the sequence in our example, because this sequence is present in 2 out of the 4 sequences of the database. Algorithms mining sequential patterns are based on a minimum threshold for extracting frequent patterns. A *frequent pattern* is thus a sequence for which the support is greater than or equal to this threshold. Other concepts are also useful to limit the number of extracted patterns.

seq. id	sequence
1	⟨(le, DET) (petit, ADJ) (chat, NC)⟩ ("the little cat")
2	⟨(le, DET) (grand, ADJ) (arbre, NC)⟩ ("the big tree")
3	⟨(le, DET) (chat, NC)⟩ ("the cat")
4	⟨(tombé, VPP) (et, CC) (cassé, VPP)⟩ ("fallen and broken")

Fig. 6. Examples of sequences of itemsets (word, POS label)

4.2 Extraction of Sequential Patterns under constraints

In [YHA03], was introduced the notion of closed patterns that allows to eliminate redundancies without loss of information. A frequent pattern S is closed, if there is no other frequent pattern S' such $S \subseteq S'$ and $sup(S) = sup(S')$. In our example, if we fix minsup=2, the frequent pattern ⟨(DET) (NC)⟩, extracted from Figure 6, is not closed because it is included in the pattern ⟨(le, DET) (NC)⟩ and they both have a support equals to 3. But the pattern ⟨(DET) (small, ADJ) (NC)⟩ is closed. A length constraint can also be used. It defines the minimum and maximum number of items contained in a pattern [BCCC12].

4.3 Algorithm

There are several available tools for extracting sequential patterns such as GSP [SA96] and SPADE [Zak01]. CloSpan [YHA03] and BIDE [WH04] are able to extract frequent closed sequential patterns. SDMC[5], used here, is a tool based on the method proposed in [PHMA$^+$01]. It extracts several types of sequential patterns, where items can correspond to simple words, lemma and/or their morpho-syntactic category (the tagger is parameterized, which allowed us to use our tagger). In this work, we wanted to characterize grammatical constructions, and we thus focused only on sequences of POS labels. The algorithm of SDMC implements the *pattern growth* technic; it is briefly discussed in [BCCC12]. It allows to extract sequential patterns under several constraints.

4.4 Emerging Patterns

[DL99] introduced the concept of *emerging pattern*. A frequent sequential pattern is called *emerging* if its relative support in a set of data set is significantly higher than in another set of data. Formally, a sequential pattern P of a set of data D_1 is emerging relatively to another set of data D_2 if $GrowthRate(P) \geq \rho$,

[5] https://sdmc.greyc.fr, login and password to be asked

with $\rho > 1$. The growth rate function is defined by:

$$\begin{cases} \infty & if\ support_{D_2}(P) = 0 \\ \frac{supp_{D_1}(P)}{supp_{D_2}(P)} & otherwise \end{cases}$$

where $supp_{D_1}(P)$ (respectively $supp_{D_2}(P)$) is the relative support of the pattern P in D_1 (respectively D_2). Any pattern P whose support is zero in a set is neglected.

5 Experiments

5.1 Parameters

The corpora used in our experiments are those described in section 2.3. We are interested here in sequences of itemsets restricted to POS labels without any gap (thus corresponding to n-grams, or repeated segments of labels), under some constraints (such as having a support strictly greater than a given threshold or pruning non-closed patterns), to limit their number. To set the lengths of sequences, we took account of the average size of sentences. So, we have decided to select patterns of length between 1 and 10. The minsup threshold is set to 2 and $\rho = 1.001$. To find the emerging patterns of a certain age group, we do as [QCCL12] did for literary genres: each age group (D_1) is compared to the set of every other age groups (D_2).

5.2 Quantitative Results

Figure 7 shows the number of frequent and emerging patterns obtained under our constraints for each age group. For example, for the age of 4-5 years, there are 1933 frequent patterns but only 842 emerging ones (42.6%). A serious reduction has occurred, which will make the observation easier. The number of emerging patterns is relatively stable across ages from 3-4 years and is important in each age group. As these emerging patterns are defined relatively to every other age group, this suggests the existence for each age group of characteristic phases of grammatical acquisitions.

Figure 8 shows the average size of the frequent and emerging patterns for each age group. The curves are very similar, suggesting that emerging patterns have properties which are similar to frequent patterns. In both cases, the length is increasing and reaches its maximum at the age of 5-6 years old. This parameter seems very correlated to the one of sentence length (see Figure 1): not only utterances become longer as the children grow up, but also the grammatical patterns they instantiate.

Figures 9 and 10 show the distributions of the main morpho-syntactic labels in frequent and emerging patterns respectively for each age group. These results are consistent with those obtained on the entire corpus (cf. Figure 5).

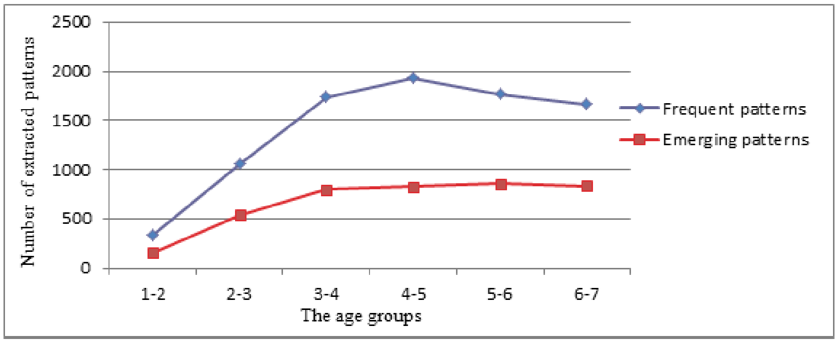


Fig. 7. Number of frequent versus emerging patterns for each age group

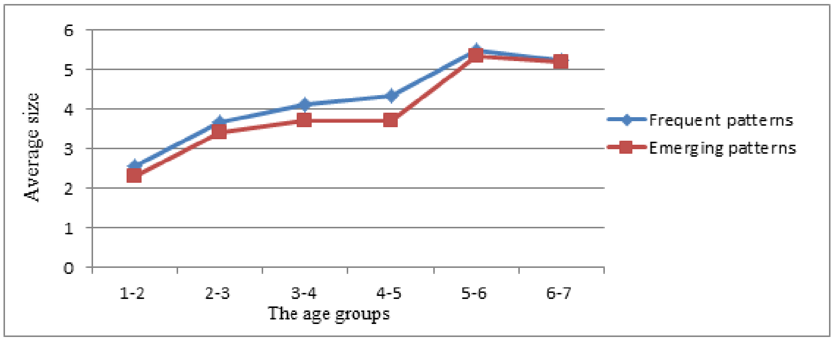


Fig. 8. Average length of frequent versus emerging patterns for each age group

The proportion of interjections still regularly decreases, while the one of prepositions increases, which is consistent with syntactic constructions of increasing complexity. We also note that the CLS and V curves are parallel and that, before the age of 4 years, the NC label is very frequent without being associated with the label DET. These curves show that the proportions of labels in the frequent and emerging patterns of each age group are similar to those of the corpus. In this sense, these patterns seem to be *representative* of the different age groups.

5.3 Qualitative Results

The table of Figure 11 provides examples of emerging patterns of each age group, and some corresponding sentences. These examples show that a single pattern can correspond to various sentences, and that they have increasing complexity. We note that even before the age of 2, children can produce sentences with a CN preceded by a DET. We also note, for example, that the patterns "(DET) (NC)" and "(DET) (NC) (CLS) (V) (VINF)" respectively extracted of the age

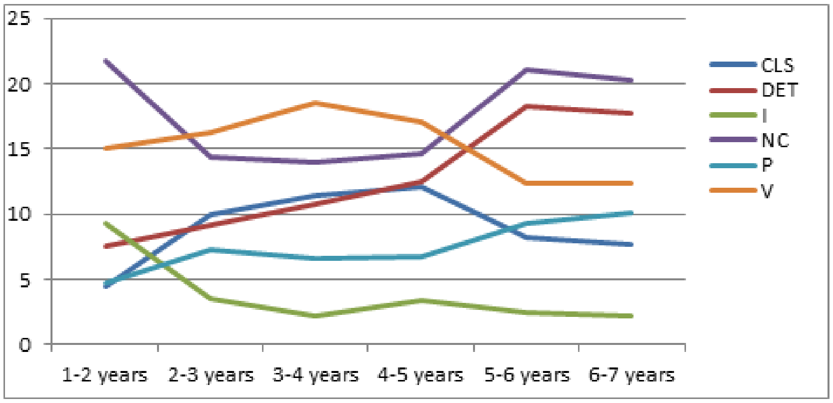


Fig. 9. Proportions of distinct labels in frequent patterns for each age group

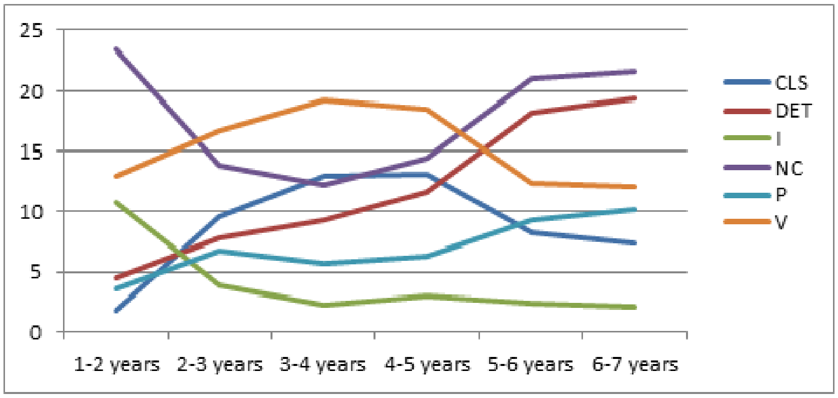


Fig. 10. Proportions of distinct labels in emerging patterns for each age group

"1-2 years" and "4-5 years are included in "(P) (DET) (NC) "and" (DET) (NC) (CLS) (V) (VINF) (DET) (NC)" respectively, of the following age group. This is consistent with a gradual acquisition of complex syntactic constructions.

6 Conclusion

In this article, we have applied techniques from Natural Language Processing, machine learning and sequential Data Mining to study the evolution of children's utterances of different ages. The phase of morpho-syntactic labeling required the learning of a specific tagger, adapted to our data. It was a necessity, considering that current available taggers do not properly handle oral transcriptions, and even less those of children: interjections, for example, which are very specific of

1-2 years	(P) (NC)	- **à maman** ("to mom") - sac **à dos** ("backpack")
	(DET) (NC)	- **le ballon** ("the ball") - **des abeilles** ("some bees")
2-3 years	(P) (DET) (NC)	- **de la tarte** ("some pie") - poissons **dans l’eau** ("fishes in the water")
	(ADVWH) (CLS) (V)	- **où il est ?** ("where it is ?") - **comment il marche ?** ("how it works ?")
3-4 years	(ADV) (CLS) (V)	- **non il est** par terre (" no it is on the floor") - **ici il pourra** passer ("here it will be able to pass")
4-5 years	(ADV) (CLS) (CLO) (V)	- **alors tu m’as** vue ? ("so you saw me ?") - **oui j’en fais** souvent ("yes I do some often")
	(DET) (NC) (CLS) (V) (VINF)	- **les lapins ils vont rentrer** ("the rabbits they will come in") - **le chat il veut attraper** l’oiseau ("the cat it wants to catch the bird")
5-6 years	(DET) (NC) (CLS) (V) (VINF) (DET) (NC)	- **l’enfant il va chercher le chat** ("the child he goes and fetch the cat") - **le monsieur il va chercher les cerises** ("the man he goes and catch the cherries")
	(CC) (DET) (NC) (CLS) (V) (DET) (NC)	- la maman **et le papa ils regardaient le garçon** ("the mommy and the daddy they watched the boy") - **et le chat il mange les cerises** ("and the cat it eats the cherries")
6-7 years	(P) (VINF) (DET) (NC)	- les oiseaux les aident **à ramasser les cerises** ("the birds help them to pick up the cherries") - il y a un chat qui essaie **de chasser des oiseaux** ("there is a cat trying to catch birds")
	(DET) (NC) (PROPEL) (V) (DET) (NC) (P) (DET) (NC)	- il y a **un chat qui suit la fille avec son panier** ("there is a cat which follows the girl with a basket") - et aussi **un monsieur qui ramasse des cerises dans un arbre** ("and a man picking up cherries in a tree")

Fig. 11. Examples of emerging patterns in each age group

oral productions, would have been poorly recognized without re-learning. This is crucial, as the curves of label proportions show that their frequency appears as an important way to characterize a child’s age group.

We currently restricted our research to n-grams of POS labels but further work could use richer itemsets of the type (word, lemma, POS tag). Our exploration seems to confirm that the extracted emerging patterns are representative of the age group in which they arise. The provided examples further confirm the intuition that (at least some of) the patterns of increasing age groups are included into each other, going in the direction of a grammatical sophistication.

As far as we know, these kinds of analyses had never been performed before. Of course, a detailed analysis of the patterns obtained remains to be done by specialists of language acquisition. They could for example allow to characterize typical evolutions of grammatical knowledge, or help to diagnose pathological evolution of a child's productions. We hope that they will provide valuable tools for the study of language acquisition phases.

7 Aknowlegment

This work is supported by a public grant overseen by the French National Research Agency (ANR) as part of the "Investissements d'Avenir" program (reference: ANR-10-LABX-0083).
The authors acknowledge Christophe Parisse, for his advice.

References

[ACT03] A. Abeillé, L. Clément, and F. Toussenel. Building a treebank for french. In A. Abeillé, editor, *Treebanks*. Kluwer, Dordrecht, 2003.

[Ali10] A. Alishahi. *Computational modeling of human language acquisition (Synthesis lectures on human language technologies)*. San Rafael: Morgan and Claypool Publisher, 2010.

[AS95] R. Agrawal and R. Srikant. Mining sequential patterns. In *Int. Conf. Data Engineering: IEEE*, 1995.

[BCCC12] N. Béchet, P. Cellier, T. Charnois, and B. Crémilleux. Discovering linguistic patterns using sequence mining. In *proceedings of CICLing'2012*, pages 154–165, 2012.

[Bro73] R. W. Brown. *A first language: the early stages. Cambridge, Mass.* Harvard University Press, Cambridge, Massashusetts, 1973.

[CC08] B. Crabbé and M. H. Candito. Expériences d'analyse syntaxique statistique du français. In *Actes de TALN'08*, 2008.

[CCP10] P. Cellier, T. Charnois, and M. Plantevit. Sequential patterns to discover and characterise biological relations. In A. Gelbukh, editor, *CICLing 2010. LNCS, vol. 6008*, pages 537–548, 2010.

[CM06] N. Chater and C. D. Manning. Probabilistic models of language processing and acquisition. In *Trends in Cognitive Science, 10(7)*, pages 335–344, 2006.

[CPRC09] T. Charnois, M. Plantevit, C. Rigotti, and B. Crémilleux. Fouille de données séquentielles pour l'extraction d'information. In *Traitement Automatique des Langues, 50(3)*, 2009.

[CSL04] L. Clément, B. Sagot, and B. Lang. Morphology based automatic acquisition of large-coverage lexica. In *LREC 2004, Lisbonne*, 2004.

[DL99] G. Dong and J. Li. Efficient mining of emerging patterns: Discovering trends and differences. In *Proc. of SIGKDD'99*, 1999.

[Elm01] J. Elman. Connectionism and language acquisition. In *Essential readings in language acquisition. In Oxford : Blackwell*, 2001.

[LCY10] Thomas Lavergne, Olivier Cappé, and François Yvon. Practical very large scale CRFs. In *Proceedings of ACL'2010*, pages 504–513. Association for Computational Linguistics, July 2010.

[LMP01] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML)*, pages 282–289, 2001.

[MC81] J. F. Miller and R. S. Chapman. The relation between age and mean length of utterance in morphemes. In *Journal of Speech and Hearing Research, 24*, pages 154–161, 1981.

[NAFS13] D. Nouvel, J-Y. Antoine, N. Friburger, and A. Soulet. Fouille de règles d'annotation partielles pour la reconnaissance d'entités nommées. In *TALN'13*, pages 421–434, 2013.

[NR07] M. Nanni and C. Rigotti. Extracting trees of quantitative serial episodes. In *Proc. of KDID'07*, pages 170–188, 2007.

[PHMA$^+$01] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M. Hsu. Prefixspan: Mining sequential patterns by prefix-projected growth. In *ICDE, IEEE Computer Society*, pages 215–224, 2001.

[QCCL12] S. Quiniou, P. Cellier, T. Charnois, and D. Legallois. Fouille de données pour la stylistique : cas des motifs séquentiels émergents. In *Proceedings of the 11th International Conference on the Statistical Analysis of Textual Data, Liege*, pages 821–833, 2012.

[SA96] R. Srikant and R. Agrawal. Mining sequential patterns: Generalizations and performance improvements. In *EDBT 1996. LNCS, vol. 1057*, pages 3–17, 1996.

[Sal86] A. Salem. Segments répétés et analyse statistique des données textuelles. In *Histoire & Mesure volume 1 - numéro 2*, pages 5–28, 1986.

[Sch94] Helmut Schmid. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of International Conference on New Methods in Language Processing*, pages 44–49, 1994.

[TDE$^+$12] I. Tellier, D. Duchier, I. Eshkol, A. Courmet, and M. Martinet. Apprentissage automatique d'un chunker pour le français. In *Actes de TALN'12, papier court (poster)*, 2012.

[TDEW13] I. Tellier, Y. Dupont, I. Eshkol, and I. Wang. Adapt a text-oriented chunker for oral data: How much manual effort is necessary? In *The 14th International Conference on Intelligent Data Engineering and Automated Learning (IDEAL'2013), Special Session on Text Data Learning, LNAI, Hefei (Chine)*, 2013.

[TTA09] Y. Tsuruoka, J. Tsujii, and S. Ananiadou. Fast full parsing by linear-chain conditional random fields. In *Proceedings of EACL 2009*, pages 790–798, 2009.

[WH04] J. Wang and J. Han. Bide: Efficient mining of frequent closed sequences. In *ICDE, IEEE Computer Society*, pages 79–90, 2004.

[YHA03] X. Yan, J. Han, and R. Afshar. Mining closed sequential patterns in large databases. In *SDM SIAM*, 2003.

[Zak01] M. J. Zaki. Spade: An efficient algorithm for mining frequent sequences. In *Machine Learning Journal 42(1/2)*, pages 31–60, 2001.

RegExpMiner: Automatically discovering frequently matching regular expressions

Julien Rabatel[1], Jérôme Azé[1], Pascal Poncelet[1], and Mathieu Roche[1,2]

[1] LIRMM, CNRS UMR 5506, Univ. Montpellier 2, 34095 Montpellier, France
[2] UMR TETIS - Cirad, Irstea, AgroParisTech, 34093 Montpellier, France

Regular expressions (REs) are a very powerful and popular tool to manipulate string data in a variety of applications. They are used as search templates to look for the occurrences of a given piece of text in a document, or to define how a given piece of text should be formatted in order to be valid (e.g., to check that the value entered in an email field of a Web form is correctly formatted), or even to help solving more complex NLP tasks [NS07]. Their popularity in those various application domains arises from several reasons. First, they are easy to understand and manipulate for common usages, despite their wide expressiveness and power of abstraction. Second, they are natively usable within a large variety of programming languages, hence making them suitable to be integrated into every project addressing text processing tasks. Their usage often relies on a very limited amount of hand-crafted REs. It is indeed difficult to automatically obtain the REs matching with a given set of strings for which no *a priori* knowledge about their underlying formatting rules is given. Such an automatic discovery of REs would nonetheless offer some very interesting prospects. Regular expressions indeed have an interesting abstraction power as they are able to provide information about how textual content is formatted, rather than focusing on the actual sequences of characters. Having a more abstract description space for describing textual content then offers new insights. For instance, an application scenario consists in data cleaning problems. Given a database containing some textual content about entities (e.g., addresses, names, phone numbers, etc.), one may be interested in finding values contained in the database that are mistakes from the people who entered them. Such typos and formatting mistakes can easily be highlighted if they result in strings that do not match the same regular expressions as the majority of the other strings.

While regular expressions can be seen as interesting descriptors of textual data for various NLP and machine learning tasks, they are hard to obtain. The literature does not offer fully relevant solutions when one wishes to enumerate some REs to describe a given set of strings. Regular Expression learning [Fer05], for instance, consists in building a single regular expression matching with a given set of positive string examples. Such approaches typically do not allow exceptions w.r.t. the set of strings to be matched, hence losing their interest as soon as input data are noisy. Additionally, only one RE is learned while one can expect to obtain several REs reflecting the different templates that co-exist in the data. E.g., one cannot expect all the values of a list of international ZIP codes to respond to only one template, as each country may use a different one. Constructing one single RE matching with all of them will of-

In: P. Cellier, T. Charnois, A. Hotho, S. Matwin, M.-F. Moens, Y. Toussaint (Eds.): Proceedings of DMNLP, Workshop at ECML/PKDD, Nancy, France, 2014.

ten lead to an over-generalization of the underlying templates that would make the obtained RE irrelevant in practical applications. On the other hand, the sequence mining literature, when applied to string data, offers the possibility to discover more various templates via frequent patterns, i.e., data fragments occurring in a sufficient amount of strings. While this general principle answers the problems above-mentioned for RE learning approaches, the type of extracted patterns (e.g., sequential patterns [AS95], episodes [MTV97]) is typically much less expressive than REs. Some efforts have however been put in allowing the generalization of sequence elements [PLL^{+}10] but extracted sequential patterns have little commonality with REs, as they only aim at discovering sequence elements that are frequently found in the same order.

We propose an approach for extracting regular expressions under the form of frequent patterns in textual data. To this end, we define a relevant pattern language that offers some interesting algorithmic properties. While we do not aim at exploiting all the characteristics and expressiveness of the RE language, we focus on providing a preliminary approach by keeping some of its main features. In particular, we fully consider the problem of allowing the generalization of characters via the use of predefined *character classes*, commonly used in REs[3]. Another aspect that this approach takes into account is the repetition of some characters in strings. For instance, we assume that the strings "*012*" and "*9876543*", should both be generalizable to the RE /[0−9]+/, i.e., a list of consecutive digit characters, even if they do not contain the same digits nor the same amount of digits. We define the *frequent regular expression pattern* mining problem by providing a theoretical framework linking together the RE and sequence mining worlds, and highlight some properties that, while inspired from known properties in sequence mining, are specific to the problem we consider study and employs them to design the `RegExpMiner` algorithm to mine such patterns.

References

[AS95] Rakesh Agrawal and Ramakrishnan Srikant. Mining sequential patterns. In *Data Engineering, 1995. Proceedings of the Eleventh International Conference on*, pages 3–14. IEEE, 1995.

[Fer05] Henning Fernau. Algorithms for learning regular expressions. In *Algorithmic Learning Theory*, pages 297–311. Springer, 2005.

[MTV97] Heikki Mannila, Hannu Toivonen, and A Inkeri Verkamo. Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1(3):259–289, 1997.

[NS07] David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26, 2007.

[PLL^{+}10] Marc Plantevit, Anne Laurent, Dominique Laurent, Maguelonne Teisseire, and Yeow Wei Choong. Mining multidimensional and multilevel sequential patterns. *ACM Transactions on Knowledge Discovery from Data*, 4(1), 2010.

[3] Character classes are sets of characters. When tested against a string, a character class matches with any of the characters it contains. For instance, the character class [0−9] contains all the digit characters $0, 1, \cdots, 9$, which allows it to match with strings such as "3" or "8", but not with "A".

NLP-based Feature Extraction for Automated Tweet Classification

Anna Stavrianou, Caroline Brun, Tomi Silander, Claude Roux

Xerox Research Centre Europe, Meylan, France

Name.surname@xrce.xerox.com

1 Introduction

Traditional NLP techniques cannot alone deal with twitter text that often does not follow basic syntactic rules. We show that hybrid methods could result in a more efficient analysis of twitter posts. Tweets regarding politicians have been annotated with two categories: the opinion polarity and the topic (10 predefined topics). Our contributions are on automated tweet classification of political tweets.

2 Combination of NLP and Machine Learning Techniques

Initially we used our syntactic parser [1] which has given high results on opinion mining when applied to product reviews [2] or the Semeval 2014 Sentiment Analysis Task [3]. However, when applied to Twitter posts, results were not satisfactory. Thus, we use a hybrid method and combine knowledge given by our parser with learning.

Linguistic information has been extracted from every annotated tweet. We have used features such as bag of words, bigrams, decomposed hashtags, negation, opinions, etc. The“liblinear” library (http://www.csie.ntu.edu.tw/~cjlin/liblinear/) was used to classify tweets. We used logistic regression classifier (with L2-regularization), where each class c has a separate vector w_c of weights for all the input features. More formally, $P(c|x; w_c) \propto e^{\sum_{i=1}^{d} w_{ci} x_i}$, where x_i is the ith feature and the w_{ci} is its weight in class *c*. When learning the model, we try to find the vectors of weight w_c that maximize the product of the class probabilities in the training data.

Our objective has been to identify the optimal combination of features that yields good prediction results, while avoiding overfitting. Some features used are: Snippets: during annotation, we kept track of the snippets that explained why the annotator tagged the post with a specific topic or polarity, Hashtags: decomposition techniques have been applied to hashtags, and they are analyzed by an opinion detection system that extracts the semantic information they carry [4].

We have selected the models using a 10-fold cross validation in the training data and evaluated them by their accuracy in the test data. For the topic-category task, (6,142 tweets, 80% used for training), the annotation had <0.4 inter-annotator agreement, which shows the difficulty of the task. Table 1. shows the results when NLP features are used, as well as when some semantic merging of classes takes place.

In: P. Cellier, T. Charnois, A. Hotho, S. Matwin, M.-F. Moens, Y. Toussaint (Eds.): Proceedings of DMNLP, Workshop at ECML/PKDD, Nancy, France, 2014.

Table 1. Cross-validation (2nd col) and prediction (3rd col) results for topic classification.

NLP features	44.38	29.37
NLP features + merging	48.91	34.17

Binary classification was applied to improve the results. We selected the class with the highest distribution and annotated the dataset with CLASS1 and NOT_CLASS1 tags. We created a model for the prediction of CLASS1, the prediction of CLASS2 and a model for the prediction of the rest of the 8 classes. Merging these models gave an accuracy of 40.03%, higher than the max accuracy of Table 1.

Table 2. Binary classification results (2nd col: cross-validation, 3rd col: prediction) for topic.

CLASS1/NOT_CLASS1	85.28	62.57
CLASS2/NOT_CLASS2 (removal of CLASS1)	92.10	68.42
The rest of the classes (removal of CLASS2)	49.58	38.24

For the opinion polarity task (5,754 tweets, 80% used for training), the inter-annotator agreement was higher (~ 0.8). As Table 3. shows, we have used not only NLP features from the tweet but also from the 'snippet'. The "syntactic analysis" is the opinion tag given from our opinion analyser.

Table 3. Cross-validation (2nd col) and prediction (3rd col) results for the opinion polarities.

NLP features (syntactic analysis of opinion)	61.28 (62.13)	56.77 (56.6)
NLP features of snippet (syntactic analysis)	66.41(67.99)	61.2 (61.46)

As a conclusion, in this paper we provide a model that predicts opinions and topics for a tweet in the political context. More research around feature analysis will be carried out. We also plan to add more features yielded by our syntactic analyzer such as POS tags, or tense. We should also consider a multiple-class labelling.

3 Acknowledgements

This work was partially funded by the project ImagiWeb ANR-2012-CORD-002-01.

4 References

1. Ait-Mokthar, S., Chanod, J.P.: Robustness beyond Shallowness: Incremental Dependency Parsing. NLE Journal, 2002.
2. Brun, C.: Learning opinionated patterns for contextual opinion detection. COLING 2012.
3. Brun, C., Popa, D., Roux, C.: XRCE: Hybrid Classification for Aspect-based Sentiment Analysis. In International Workshop on Semantic Evaluation (SemEval), 2014 (to appear).
4. Brun, C., Roux, C.: Décomposition des « hash tags » pour l'amélioration de la classification en polarité des « tweets ». In TALN, July, 2014.

Author Index