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

Interactions between Data Mining and Natural Language Processing

2nd International Workshop, DMNLP 2015
Porto, Portugal, September 2015
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 2015 workshop: DMNLP'15, held on September 7, 2015 in Porto. DMNLP'15 (Workshop on Interactions between Data Mining and Natural Language Processing) is the second edition of a 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.

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 2015. and the ECML/PKDD 2015 workshop chairs Bernhard Pfahringer andLuis Torgo.

September 2015

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 Crmilleux	Universit de Caen, France
Batrice Daille	Laboratoire d'Informatique de Nantes Atlantique, France
Luigi Di Caro	University of Torino, Italy
Pierre Geurts	University of Lige, Belgium
Francois Jacquenet	Laboratoire Hubert Curien, France
Jiri Klma	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
Koichi Takeuchi	Okayama University, Japan
Isabelle Tellier	Lattice, Paris, France
Xifeng Yan	University of California at Santa Barbara, USA
Pierre Zweigenbaum	LIMSI-CNRS, Paris, France

Table of Contents

Annotated suffix tree similarity measure for text summarization

Maxim Yakovlev and Ekaterina Chernyak

National Research University – Higher School of Economics
Moscow, Russia
myakovlev,echernyak@hse.ru

Abstract. The paper describes an attempt to improve the TextRank algorithm. TextRank is an algorithm for unsupervised text summarisation. It has two main stages: first stage is representing a text as a weighted directed graph, where nodes stand for single sentences, and edges are weighted with sentence similarity and connect sequential sentences. The second stage is applying the PageRank algorithm [1] as is to the graph. The nodes that get the highest ranks form the summary of the text.
We focus on the first stage, especially on measuring the sentence similarity. Mihalcea and Tarau [4] suggest to employ the common scheme: use the Vector space model (VSM), so that every text is a vector in the space of words or stems, and compute cosine similarity between these vectors. Our idea is to replace this scheme by using the annotated suffix trees (AST) [5] model for sentence representation. The AST overcomes several limitations of the VSM model, such as being dependent on the size of vocabulary, the length of sentences and demanding stemming or lemmatisation. This is achieved by taking all fuzzy matches between sentences into account and computing probabilities of matched coocurrencies.
More specifically we develop an algorithm for common subtree construction and annotation. The common subtrees are used to score the similarity between two sentences. Using this algorithm allows us to achieve slight improvements according to cosine baseline on our own collection of Russian newspaper texts. The AST measure gained around 0.05 points of precision more than the cosine measure. This is a great figure for natural language processing task, taking into account how low the baseline precision of the cosine measure is. The fact that the precision is so low can be explained by some lack of consistency in the constructed collection: the authors of the articles use different strategies to highlight the important sentences. The text collection is heterogeneous: in some articles there are 10 or more sentences highlighted, in some only the first one. Unfortunately, there is no other test collection for text summarisation in Russian. For further experiments we might need to exclude some articles, so that the size of summary would be more stable. Another issue of our test collection is the selection of sentences that form summaries. When the test collections are constructed manually, summaries are chosen to common principles. But we can not be sure that the sentences are not highlighted randomly.
Although the AST technique is rather slow, it is not a big issue for the text summarisation problem. The summarisation problem is not that

In: P. Cellier, T. Charnois, A. Hotho, S. Matwin, M.-F. Moens, Y. Toussaint (Eds.): Proceedings of DMNLP, Workshop at ECML/PKDD, Porto, Portugal, 2015.

kind of problems where on-line algorithms are required. Hence the precision plays more significant part than time characteristics.
There are several directions of future work. First of all, we have to conduct experiments on the standard DUC (Document Understanding Conference [2]) collections in English. Second, we are going to develop different methods for construction and scoring of common subtrees and compare it to each other. Finally, we may use some external and more efficient implementation of the AST method, such as EAST Python library by Mikhail Dubov [3], which uses annotated suffix arrays. More details on this work can be found in [6].
Keywords: TextRank, annotated suffix tree

References

1. Brin S., Page L. : The anatomy of a large-scale hypertextual Web search engine. In: Proceedings of the seventh international conference on World Wide Web 7, 107-117 (1998)
2. Document Understanding Conference, `http://www-nlpir.nist.gov/`
3. Enhanced Annotated Suffix Tree, `https://pypi.python.org/pypi/EAST/0.2.2/`
4. Mihalcea R., Tarau P. : TextRank: bringing order into text. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, 404-411 (2004)
5. Pampapathi R., Mirkin B., Levene M. (2008): A suffix tree approach to anti-spam email filtering. In: Machine Learning, 65(1), 309-338 (2008)
6. Yakovlev M., Chernyak E. (2015): Using annotated suffix tree similarity measure for text summarization. In: Proccedings of European Conferenece on Data Analysis, 2015.

Narrative Generation from Extracted Associations

Pierre-Luc Vaudry and Guy Lapalme

Université de Montréal, Montréal, Canada
{vaudrypl,lapalme}@iro.umontreal.ca

Keywords. Narrative. Natural Language Generation. Association rule discovery. Activity of Daily Living. Data-to-text. Rhetorical relations. Coherence.

In [1], we study how causal relations may be used to improve narrative generation from real-life temporal data. We describe a method for extracting potential causal relations from temporal data and for structuring a generated report. The method is applied to the generation of reports highlighting unusual combinations of events in the Activity of Daily Living (ADL) domain.

Our experiment applies association rules discovery techniques in [2] for selecting candidate associations based on three properties: frequency, confidence and significance. We assume that temporal proximity and temporal precedence are indicators of potential causality.

The generation of a report from the ADL data for a given period follows a pipeline architecture. The first stage is data interpretation, which consists of finding instances of the previously selected association rules in the input. For each of those, one or more semantic relations are introduced as part of a hypothetic interpretation of the input data. Next those relations are used to plan the document as a whole in the document planning stage. The output is a rhetorical structure which is then pruned to keep only the most important events and relations. Follows a microplanning stage that plans the phrases and lexical units expressing the events and rhetorical relations. This produces a lexico-syntactic specification that is realised as natural language text in the last stage: surface realisation.

After analysing the results, the extracted relations seem to be useful to locally link activities with explicit rhetorical relations. However, further work is needed to better exploit them for improving coherence at the global level.

References

1. Vaudry, P.-L., Lapalme, G.: Narrative Generation from Extracted Associations. In: Proceedings of the 15th European Workshop on Natural Language Generation., Brighton, United Kingdom (Sept 2015).
2. Hamalainen, W., Nykanen, M.: Efficient Discovery of Statistically Significant Association Rules. In: ICDM '08 Proceedings of the Eighth IEEE International Conference on Data Mining. pp. 203–212 (2008).

Some Thoughts on Using Annotated Suffix Trees for Natural Language Processing

Ekaterina Chernyak

National Research University – Higher School of Economics
Moscow, Russia
echernyak@hse.ru

Abstract. The paper defines an annotated suffix tree (AST) - a data structure used to calculate and store the frequencies of all the fragments of the given string or a collection of strings. The AST is associated with a string to text scoring, which takes all fuzzy matches into account. We show how the AST and the AST scoring can be used for Natural Language Processing tasks.
Keywords: text representation, annotated suffix tree, text summarization, text categorization

1 Introduction

Natural Language Processing tasks require a text being represented by a sort of a formal structure to be processed by a computer. The most popular text representation is the Vector Space Model (VSM), designed by Salton [1]. The idea of the VSM is simple: given a collection of texts, represent every text as a vector in a space of terms. A term is a word itself or a lemmatized word or the stem of a word or any other meaningful part of the word. The VSM is widely used in any kind of Natural Language Processing tasks. The few exceptions are machine translation or text generation, when word order is important, while the VSM completely loses it. For these purposes Ponte and Croft introduced the language model [2], which is based on calculating the probability of the sequence of n words or characters, so-called n-grams. There is one more approach to text representation, which is based on suffix trees and suffix arrays. Originally the suffix tree was developed for fuzzy string matching and indexing [3]. However there appear to be several application of suffix trees to Natural Language Processing. One of them is document clustering, presented in [4]. When some sort of probability estimators of the paths in the suffix tree are introduced, it can be used as a language model for machine translation [5] and information retrieval [6].

In this paper we are going to concentrate on the so-called annotated suffix tree (AST), introduced in [8]. We will present the data structure itself and several Natural Language Processing tasks where the AST representation is successfully used. We are not going to make any comparisons to other text representation models, but will show that using the AST approach helps to overcome some exciting problems. The paper is organized as follows: the Section 2 presents the definition of the AST and the algorithm for the AST construction, Sections from 3 to 7 present exciting applications of the AST (almost all developed with author's contribution), Section 8 lists some future application, Section 9 suggests how to compare the AST scoring to other approaches, Section 10 is devoted to the AST scoring implementation. Section 11 concludes.

The project is being developed by the "Methods of web corpus collection, analysis and visualisation" research and study group under guidance of prof. B. Mirkin (grant 15 - 05 - 0041 of Academic Fund Program).

2 Annotated suffix tree

2.1 Definition

The suffix tree is a data structure used for storing of and searching for strings of characters and their fragments [3]. When the suffix tree representation is used, the text is considered as a set of strings, where a string may be any significant part of the text, like a word, a word or character n-gram or even a whole sentence. An annotated suffix tree (AST) is a suffix tree whose nodes (not edges!) are annotated by the frequencies of the strings fragments.

An annotated suffix tree (see Figure 1)[7] is a data structure used for computing and storing all fragments of the text and their frequencies. It is a rooted tree in which:

- Every node corresponds to one character
- Every node is labeled by the frequency of the text fragment encoded by the path from the root to the node.

2.2 AST construction

Our algorithm for constructing an AST is a modification of the well-known algorithm for constructing suffix trees [3]. The algorithm is based on finding suffixes and prefixes of a string. Formally, the i-th suffix of the sting is the substring, which starts at i-th character of the string. The i-th prefix of the string is the substring, that ends on the i-th character of the string. The AST is built in an iterative way. For each string, its suffixes are added to the AST one-by-one starting from an empty set representing the root. To add a suffix to the AST, first check, whether there is already a match, that is, a path in the AST that encodes / reads the whole suffix or its prefix. If such a match exists, we add 1 to all the frequencies in the match and append new nodes with frequencies 1 to the last node in the match, if it does not cover the whole suffix. If there is no match, we create a new chain of nodes in the AST from the root with the frequencies 1.

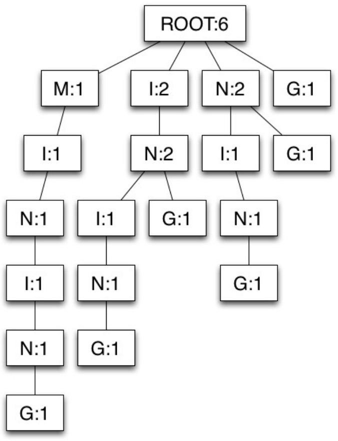

Fig. 1. An AST for string "mining".

2.3 AST relevance measure

To use an AST to score the string to text relevance we first build an AST for a text. Next we match the string to the AST to estimate the relevance.

A procedure for computing string-to-text relevance score:
Input: string and AST for a given text.
Output: the AST scoring.

1. The string is represented by the set of its suffixes;
2. Every suffix is matched to the AST starting from the root. To estimate the match we use the average conditional probability of the next symbol:

$$score(match(suffix, ast)) = \sum_{node \in match} \phi\left(\frac{\frac{f(node)}{f(parent(node))}}{|suffix|}\right) ,$$

where $f(node)$ is the frequency of the matching node, $f(parent(node))$ is it's parent frequency, and $|suffix|$ is the length of the suffix;

3. The relevance of the string is evaluated by averaging the scores of all suffixes:

$$relevance(string, text) = SCORE(string, ast) = \\ = \frac{\sum_{suffix} score(match(suffix, ast))}{|string|},$$

where $|string|$ is the length of the string.

Note, that "score" is found by applying a scaling function to convert a match score into the relevance evaluation. There are three useful scaling functions, according to experiments in [8] for spam classification:

– Identity function: $\phi(x) = x$
– Logit function:

$$\phi(x) = \log \frac{x}{1-x} = \log x - \log(1-x)$$

– Root function $\phi(x) = \sqrt{x}$

The identity scaling stands for the conditional probability of characters averaged over matching fragments (CPAMF).

Consider an example to illustrate the described method. Let us construct an the for the string “mining”. This string has six suffixes: “mining”, “ining”, “ning”, “ing”, “ng”, and “g’ . We start with the first suffix and add it to the empty AST as a chain of nodes with the frequencies equal to unity. To add the next suffix, we need to check whether there is any match, i.e. whether there is such a path in the AST starting at its root that encodes / reads a prefix of “ining”. Since there is no match between existing nodes and the second suffix, we add it to the root as a chain of nodes with the frequencies equal to unity. We repeat this step until a match is found: a prefix of the fourth suffix “ing” matches the second suffix “ining”: two first letters, “in”, coincide. Hence we add 1 to the frequency of each of these nodes and add a new child node “g” to the leaf node “n” (see Figure 1). The next suffix “ng” matches the third suffix and we repeat the same actions: increase the frequency of the matched nodes and add a new child node that does not match. The last suffix does not match any path in the AST, so again we add it to the AST’s root as a single node with its frequency equal to unity. Now let us calculate the relevance score for string “dining” using the AST in Figure 1. There are six suffixes of the string “dining”: ‘dining”, “ining”, “ning”, “ing”, “ng”, and “g’ . Each of them is aligned with an AST path starting from the root. The scorings of the suffixes are presented in Table 1.

Table 1. Computing the string “dining” score

Suffix	Match	Score
“dining”	None	0
“ining”	“ining”	$\frac{1/1+1/1+1/2+2/2+2/6}{5} = 0.76$
“ning”	“ning”	$\frac{1/1+1/1+1/2+2/6}{4} = 0.71$
“ing”	“ing”	$\frac{1/2+2/2+2/6}{3} = 0.61$
“ng”	“ng”	$\frac{1/2+2/6}{2} = 0.41$
“g”	“g”	$\frac{1/6}{1} = 0.16$

We have used the identity scaling function to score all 6 suffixes of the string “dining”. Now, to get the final CPAMF relevance value we sum and average them:

$$relevance(dining, mining) = \frac{0 + 0.76 + 0.71 + 0.61 + 0.41 + 0.16}{6} = \\ = \frac{2.65}{6} = 0.44$$

In spite of the fact that "dining" differs from "mining" by just one character, the total score, 0.44, is less than unity. This is not only because the trivial suffix "dining" contributes 0 to the sum, but also because conditional probabilities get smaller for the shorter suffixes.

3 Spam filtering

The definition of the AST presented above was for first time introduced by Pampapathi, Mirkin and Levene in [7] for spam filtering. The AST was used as a representation tool for every class (spam and ham). By introducing a procedure for scoring the class AST they developed a classifier that beats the Naive Bayes classifier in a series of experiments on standard datasets. The success of ASTs in domain of email filtering was due to the notion of match permutation normalization, which allowed to take into account some intentional typos developed by spamers to pass over spam filters. Match permutation normalization is in a a sense analogous to the edit distance [10] that if frequently implemented in spam filters [11].

4 Research paper categorization

The problem of text categorization is formulated as follows. Given a collection of documents and a domain taxonomy, annotate a document with relevant taxonomy topics. A taxonomy is a rooted tree, such that every node corresponds to a (taxonomy) topic of the domain. The taxonomy generalizes the relation "is – a" or "is a part of".

There are two basic approaches to the problem of text categorization: supervised and unsupervised. Supervised approaches give high precision values when applied to web document categorization [12], but may fail when applied to research paper categorization, since the research taxonomies, such as ACM Computing Classification System [13], are seldom revised and the supervised techniques may overfit [14]. The unsupervised approaches to text categorization are based on information retrieval – like idea: given the set of taxonomy topics, let us find those research papers that are relevant to every topic. The question for researcher is the following: what kind of the relevance model and measure to choose? In [15] we experimentally compared cosine relevance function, which measures the cosine between $tf - idf$ vectors in Vector Space Model [1], BM25, based on the probabilistic relevance framework, and the AST scoring, introduced above. These three relevance measures where applied to a relatively small dataset of 244 articles, published in ACM journals and the current version of ACM Computing Classification System. The AST scoring outperforms cosine and BM25 measures, by being more robust and taking not crisp but fuzzy measures into account. The next step in this research direction would be testing the AST scoring versus w-shingling procedure [17], which is also a fuzzy matching technique that requires text preprocessing, such stemming or lemmatization. However there is no need in stemming or lemmatization to apply the AST scoring.

5 Taxonomy refinement

Taxonomies are widely used to represent, maintain and store domain knowledge, see, for example SNOMED [18] or ACM CCS [13]. Domain taxonomy construction is a difficult task and a number of researchers have come out with idea of taxonomy refinement. The idea of taxonomy refinement is the following: having one taxonomy or upper levers of taxonomy refine it with topics extracted from additional sources such as other taxonomies, web search or Wikipedia. We followed this strategy and developed a two-step approach to taxonomy refinement, presented in more details in [21]. We concentrated on taxonomies of probability theory and mathematical statistics (PTMS) and numerical mathematics (NM), both in Russian. On a first step an expert sets manually the upper layers of taxonomy. On the second step these upper layers are refined by Wikipedia category tree and the articles, belonging to this tree, from the same domain. In this study the AST scoring is used several times:

- To clear the Wikipedia data from noise;
- To assign the remaining Wikipedia categories to the taxonomy topics;
- To form the intermediate layers of the taxonomy by using Wikipedia subcategories;
- To use Wikipedia articles in each of the added category nodes as its leaves.

The Wikipedia data is rather noisy: there some articles that are stubs or irrelevant to parental categories (the categories, they belong to) and the more so there are subcategories (of a category) that are irrelevant to the parental categories. For example, we found the article "ROC curve" be irrelevant to the category "Regression analysis" and the category "Accidentally killed" to the category "Randomness". To define what article is irrelevant we exploit the AST scoring twice:

- We scored the title of the article to the text of the article to detect stubs;
- We scored the title of the parental category to the text of the article to detect irrelevant category.

If the value of the scoring function is less than a threshold we decided that the article is irrelevant. Usually we set the threshold at 0.2. To assign the remaining Wikipedia categories to the taxonomy topics we score the taxonomy topics to all the articles in the category merged into one text. Next we found the maximum value of the scoring function and assigned the category to the corresponding taxonomy topic. Finally, we score the title of parental categories to the articles of the subcategories, merged into one. If the subcategory to category scoring is higher than the subcategory to taxonomy topic, the subcategory remains on the intermediate layer of the refined taxonomy tree under its parental category. Finally, the articles left after clearing from noise became leaves in the refined taxonomy tree. The quality of achieved PTMS and NM taxonomies is difficult to evaluate computationally, so the design of the user study is an open question.

6 Text summarization

Automatic text summarisation is one of the key tasks in natural language processing. There are two main approaches to text summarisation, called abstractive and extractive approaches [22].

According to the abstractive approach, the summary of a text is another text, but much shorter, generated automatically to make the semantic representation of the text. According to extractive approach, the summary of a text is nothing else, but some important parts of the given text, such as a set of important sentences.

The extractive summarisation problem can be formulated in the following way. Given a text T that is a sequence of sentences S that consists of words V, select a subset of the sentences S^* that are important in T. Therefore we need to define:

- what importance of a sentence is;
- how to measure importance of the sentence; Hence we need to introduce a function, $importance(s)$, which measures the importance of a sentence. The higher *importance* is, the better. Next step is to build the summary. Let us rank all the sentences according the values of *importance*. Suppose we look for the summary that consists of five sentence. Hence we take the five sentences with the highest values of *importance* and call them top-5 sentences according to *importance*. Generally, the summary of the text are the top-N sentences according to *importance* and N is set manually.

The best results for this statement of the problem are achieved by Mihalcea and Tarau [23], where $importance(s)$ is introduced as PageRank type function [24] without any kind of additional grammar, syntax or semantic information. The main idea of the suggested TextRank algorithm is to represent a text as a directed graph, where nodes stand for sentences and edges connect sequential sentences. The edges are weighted with sentence similarity. When PageRank is applied to this graph, every node receives its rank that is to be interpreted as the importance of the sentence, so that $importance(s) = PageRank(s_{node})$, where s_{node} is the node corresponding to sentence s.

To measure similarity of the sentences the authors of TextRank algorithm suggest to use the basic VSM (Vector Space Model) scheme. First every sentence is represented as a vector in space of words or stems. Next cosine similarity between those vectors is computed. We can use the AST scoring as well for scoring the similarity between two sentences. To do this we have to introduce the common tree technique.

6.1 Constructing common subtree for two ASTs

To estimate the similarity between two sentences we find the common subtree of the corresponding ASTs. We do the depth-first search for the common chains of nodes that start from the root of the both ASTs. After the common subtree is constructed we need to annotate and score it. We annotate every node of the common subtree with the averaged frequency of the corresponding nodes in initial ASTs. Consider for example two ASTs for strings "mining" and "dinner" (see Fig. 1 and Fig. 2, correspondingly). There are two common chains: "I N" and "N", the first one consists of two nodes, the second one consists of a single node. Both this chains form the common subtree. Let us annotate it. The frequency of the node "I" is equal to 2 in the first AST and to 1 in the second. Hence, the frequency of this node in the common subtree equals to $\frac{2+1}{2} = 1.5$. In the same way we annotate the node "N" that follows after the node "I" with $\frac{2+1}{2} = 1.5$ and the node "N" on the first level with $\frac{2+2}{2} = 2$. The root is annotated with the sum of the frequencies of the first level nodes that is $1.5 + 2 = 3.5$.

6.2 Scoring common subtree

The score of the subtree is the sum of scores of every chain of nodes. The score of the path is the averaged sum of the conditional probabilities of the nodes, where conditional probability of the node is the frequency of the node divided by the frequency of its parent. For example, the conditional probability of the node "G:1" on the third level of the AST on Fig. 1 is 1/2. Let us continue with the example of "mining" and "dinner". There are two chains in their common subtree: "I N" and "N". The score of "I N" chain is $(1.5/1.5 + 1.5/3.5)/2 = 0.71$, since there are 2 nodes in the chain. The score of one node chain "N" is $1.5/3.5 = 0.42$. The score of the whole subtree is $(0.71 + 0.42) = 1.13$.

The collection for experiments was made of 400 articles from Russian news portal called Gazeta.ru. The articles were marked up in a special way, so that some of sentences were highlighted because of being more important. This highlighting was done either by the author of the article or by the editor on the basis of their own ideas. In our experiments we considered those sentences as the summary of the article. We tried to reproduce these summaries using TextRank with cosine similarity measure and AST scoring.

Using this algorithm allowed us to gain around 0.05 points of precision according to cosine baseline on our own collection of Russian newspaper texts. This is a great figure for Natural Language Processing task, taking into account that the baseline precision of the cosine measure was very low. The fact that the precision is so low can be explained by some lack of consistency in the constructed collection: the authors of the articles use different strategies to highlight the important sentences. The text collection is heterogeneous: in some articles

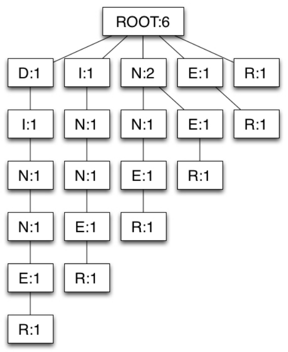

Fig. 2. An AST for string “dinner”.

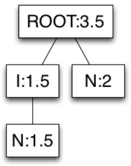

Fig. 3. Common subtree of ASTs for stings “mining” and “dinner”.

there are 10 or more sentences highlighted, in some only the first one. More details of this experiment are presented in [25].

7 Association rule extraction

Several research group develop different approaches to extraction and visualization of association rules from text collections [26, 27]. Association rule is a rule $X \implies Y$, where both X and Y are sets of concepts, possibly a singleton, and the implication means some sort of co-occurrence relation. An association rule has two important features, called support and confidence. When the rule is extracted from the text collection, the support of the set X $support(X)$ usually stands for the proportion of the documents where concepts X occur and the confidence of the association rule $confidence(X \implies Y)$ stands for conditional probability of Y given X. The majority of approaches to association rule extraction share the following idea in common: the concepts should be extracted from the text collection. Using the fuzzy AST scoring we can diminish this limitation and produce the rules on the set of concepts provided by a user. In [28] we presented a so-called “conceptual map”, which is a graph of association rules $X \implies Y$. To make the visualization easy we restricted ourselves only to single item sets, so that $|X| = |Y| = 1$. We analyzed a collection of Russian language newspaper articles on business and the concepts were provided by a domain expert. We used the AST scoring to score every concept k_i to every text from the collection. Next we formed $F(k_i)$ the set of articles, to which the concept k_i relevant (i.e. the scoring is higher than a threshold, usually 0.2). Finally, there was a rule $k_i \implies k_j$ if the ratio $\frac{F(k_i) \cap F(k_j)}{F(k_i)}$ was higher than the predefined confidence threshold. An example of conceptual map (translated into English) can be found on Fig. 4.

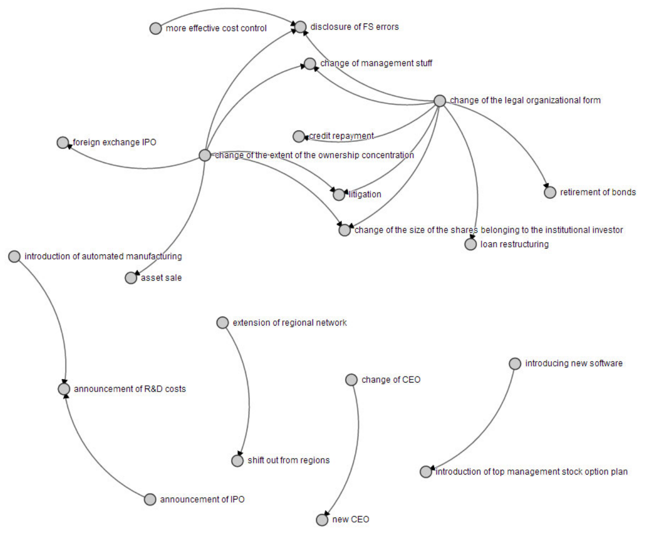
more effective cost control
disclosure of FS errors
change of management stuff
change of the legal organizational form
foreign exchange IPO
credit repayment
change of the extent of the ownership concentration
litigation
retirement of bonds
change of the size of the shares belonging to the institutional investor
loan restructuring
introduction of automated manufacturing
asset sale
extension of regional network
introducing new software
change of CEO
announcement of R&D costs
shift out from regions
introduction of top management stock option plan
announcement of IPO
new CEO

Fig. 4. A conceptual map

This conceptual map may serve as a tool for text analysis: it reveals some hidden relations between concepts and it can be easy visualized as a graph. Of course, to estimate the power of conceptual maps we have to conduct an user study.

8 Future work

In the following sections we will briefly present some Natural Language Processing tasks, where AST scoring might be used.

8.1 Plagiarism detection

Ordinary suffix trees are widely used for plagiarism detection [29]. The common subtree technique can also be used in this case. Suppose we have two texts, construct two individual ASTs and the common AST. The size of the common AST will show how much these texts share in come. Scoring the common AST allows to measure how significant coinciding parts are. With no doubts, the common AST can be used for indexing of coinciding parts of the texts. Hence, it inherits advantages of ordinary suffix trees with some additional functionality.

8.2 Compound splitting

Splitting compounds, such as German compounds, is necessary for machine translation and information retrieval. The splitting is usually conducted according to some morphological or probabilistic models [30]. We have a hypothesis that scoring prefixes of compound words to the AST, constructed from the collection of simple words, will allow to split compounds without using additional morphological knowledge. The main research in this direction is the design of the collection of simple words.

8.3 Profanity filtering

The Russian profanity language is rich and complex and has a complex derivation, usually based on adding prefixes (such as “za”, “pro”, “vy”, etc). New words appear almost every month, so it is difficult to maintain a profanity dictionary. Profanity filtering is an important part of Russian Text or Web mining, specially since some special limitations on using profanity were introduced. The task is to find words in a text that are profane and, for example, to replace them with star symbols “***”. Note, that Russian derivative includes also a variety of endings, so lematization or stemming should be used. Since Porter stemmer [31] does not cope with prefixes, it can be easily replaced by some sort of the AST-scoring.

9 Comparison to other approaches

Cosine measure on $tfidf$ vectors is a traditional baseline in majority of Natural Language Processing tasks and is easily overcame by any sort of more robust and fuzzy similarity or relevance measure, such as w-shingling [17], super shingles [32], mega shingles [33] and character n-grams [34]. The main future research concentrates on drawing comparison between these fuzzy measure and AST scoring.

10 Implementation

Mikhail Dubov’s implementation of AST construction and scoring is based on suffix arrays, which makes it space and time efficient. It is available at `https://github.com/msdubov/AST-text-analysis`. It can be used as console utility or as a Python library.

11 Conclusion

In this paper the notion of annotated suffix tree is defined. The annotated suffix trees are used by several research groups and in the paper several finished, running or future projects are presented. The annotated suffix tree is a simple but powerful tool for scoring different types of relevance or similarity. This paper may sound light weighted and to make it more theoretical, we will conclude by provided some insights on probabilistic or morphological origins of ASTs. From one point of view, we have a strong feeling that it can proved that the AST or the common AST is a string kernel, thus it can be used to generate features for text classification / categorization or to measure similarity. From another point of view, the AST is a sort of supervised stemmer, that can be used to generate terms more efficiently than model-based stemmers.

12 Acknowledgments

I am deeply grateful to my supervisor Dr. Boris Mirkin for an oppurtunity to learn from him and to work under his guidance for so long and to my colleagues Mikhail Dubov, Maxim Yakovlev, Vera Provotorova and Dmitry Ilvovsky for collaboration.

References

1. G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. Information Processing and Management, Vol.2, no 5, pp. 513-523, 1998.
2. Ponte, J. M., and Croft B.W.. A language modeling approach to information retrieval. In Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval, pp. 275-281. ACM, 1998.
3. Gusfield D., Algorithms on Strings, Trees, and Sequences, Cambridge University Press, 1997.
4. Zamir O., Etzioni, O. Web document clustering: A feasibility demonstration. Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval, pp. 46-54. ACM, 1998.
5. Kennington C.R., Kay M. , Friedrich. A.. Suffix Trees as Language Models. In LREC, pp. 446-453. 2012.
6. Huang J.H., Powers D.. Suffix tree based approach for chinese information retrieval. Intelligent Systems Design and Applications, 2008. ISDA'08. Eighth International Conference on, vol. 3, pp. 393-397. IEEE, 2008.
7. Pampapathi R., Mirkin B., Levene M., A suffix tree approach to anti-spam email filtering, Machine Learning, 2006, Vol. 65, no.1, pp. 309-338.
8. Chernyak E.L., Chugunova O.N., Mirkin B.G., Annotated suffix tree method for measuring degree of string to text belongingness, Business Informatics, 2012. Vol. 21, no.3, pp. 31-41 (in Russian).
9. Chernyak E.L., Chugunova O.N., Askarova J.A., Nascimento S., Mirkin B.G., Abstracting concepts from text documents by using an ontology, in Proceedings of the 1st International Workshop on Concept Discovery in Unstructured Data. 2011, pp. 21-31.

10. Levenshtein, V. I., Binary codes capable of correcting deletions, insertions, and reversal. Soviet Physics Doklady Vol.10, no 8, pp. 707710.
11. Tretyakov K., Machine learning techniques in spam filtering. Data Mining Problem-oriented Seminar, MTAT, vol. 3, no. 177, pp. 60-79. 2004.
12. M. Ceci and D. Malerba Classifying web documents in a hierarchy of categories: a comprehensive study. Journal of Intelligent Information Systems, Vol. 28, no. 1, pp. 37-78, 2007.
13. ACM Computing Classification System (ACM CCS), 1998, available at: http://www.acm.org/about/class/ccs98-html
14. A.P. Santos and F. Rodrigues. Multi-label hierarchical text vlassification using the ACM taxonomy Proceedings of 14th Portuguese Conference on Artificial Intelligence, pages 553 - 564, Aveiro, Portugal, 2010.
15. Chernyak E. L. An approach to the problem of annotation of research publications, Proceedings of The Eighth International Conference on Web Search and Data Mining, pp. 429-434.
16. S. Robertson and H. Zaragoza. The probabilistic relevance gramework: BM25 and beyond. Journal Foundations and Trends in Information Retrieval, Vol.25, no 4., pp. 333-389, 2009
17. Manber, Udi. Finding Similar Files in a Large File System. Usenix Winter, vol. 94, pp. 1-10. 1994.
18. SNOMED CT - Systematized Nomenclature of Medicine Clinincal Terms, www.ihtsdo.org/snomed-ct/, visited 09.25.14.
19. Van Hage W.R., Katrenko S., Schreiber G., A Method to Combine Linguistic Ontology-Mapping Techniques, in Proceedings of 4th International Semantic Web Conference, 2005, pp. 34-39.
20. Grau B.C., Parsia B., Sirin E. Working with Multiple Ontologies on the Semantic Web, in Proceedings of the 3d International Semantic Web Conference, 2004, pp. 620-634.
21. Chernyak E. L., Mirkin B. G. Refining a Taxonomy by Using Annotated Suffix Trees and Wikipedia Resources. Annals of Data Science. Vol. 2. No. 1. P. 61-82, 2015.
22. Hahn U., Mani I. The challenges of automatic summarization, Computer, Vol.33, no.11, pp. 29-36, 2000
23. Mihalcea R., Tarau P. TextRank: bringing order into text. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, pp. 404-411, 2004
24. Brin S., Page L. The anatomy of a large-scale hypertextual Web search engine. Proceedings of the seventh international conference on World Wide Web 7, 107-117, 1998
25. , Chernyak E.L., Yakovlev M.S., Using annotated suffix tree similarity measure for text summarization (under revision)
26. Pak Chung W., Whitney P., Thomas J.. Visualizing association rules for text mining. Information Visualization, 1999. Proceedings. 1999 IEEE Symposium on, pp. 120-123. IEEE, 1999.
27. Mahgoub, H., Rsner, D., Ismail, N., Torkey, F.. A text mining technique using association rules extraction. International journal of computational intelligence 4, no. 1, pp. 21-28, 2008.
28. Morenko, E. N., Chernyak E.L., Mirkin B.G.. Conceptual Maps: Construction Over a Text Collection and Analysis. In Analysis of Images, Social Networks and Texts, pp. 163-168. Springer International Publishing, 2014.

29. Krisztin M., Zaslavsky A., Schmidt, H.. Document overlap detection system for distributed digital libraries. Proceedings of the fifth ACM conference on Digital libraries. ACM, 2000.
30. Koehn P., Knight K.. Empirical methods for compound splitting. Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics-Volume 1. Association for Computational Linguistics, 2003.
31. Porter, M. F. An algorithm for suffix stripping. Program Vol. 14, no. 3, pp, 130-137 (1980).
32. Chowdhury A., Frieder O., Grossman D., McCabe M.C.."Collection statistics for fast duplicate document detection. ACM Transactions on Information Systems (TOIS) Vol. 20, no. 2 ,pp. 171-191 (2002).
33. Conrad J. G., Schriber C.P.. Managing dj vu: Collection building for the identification of nonidentical duplicate documents. Journal of the American Society for Information Science and Technology Vol 57, no. 7 pp. 921-932 (2006).
34. Damashek M.. Gauging similarity with n-grams: Language-independent categorization of text. Science Vol. 267, no. 5199, pp 843-848 (1995).

MIL: Automatic Metaphor Identification by Statistical Learning

Yosef Ben Shlomo and Mark Last

Department of Information Systems Engineering
Ben-Gurion University of the Negev
Beer-Sheva 84105, Israel
E-mail: bshyossi@gmail.com, mlast@bgu.ac.il

Abstract. Metaphor identification in text is an open problem in natural language processing. In this paper, we present a new, supervised learning approach called MIL (Metaphor Identification by Learning), for identifying three major types of metaphoric expressions without using any knowledge resources or handcrafted rules. We derive a set of statistical features from a corpus representing a given domain (e.g., news articles published by Reuters). We also use an annotated set of sentences, which contain candidate expressions labelled as 'metaphoric' or 'literal' by native English speakers. Then we induce a metaphor identification model for each expression type by applying a classification algorithm to the set of annotated expressions. The proposed approach is evaluated on a set of annotated sentences extracted from a corpus of Reuters articles. We show a significant improvement vs. a state-of-the-art learning-based algorithm and comparable results to a recently presented rule-based approach.

Keywords: Metaphor Identification, Natural Language Processing, Supervised Learning.

1 Introduction

A metaphor is defined in previous works (such as Krishnakumaran & Zhu, 2007) as the use of terms related to one concept in order to describe a term, which is related to a different concept. For example, in the metaphoric expression “fertile imagination”, the word “imagination”, which is related to the concept / domain “cognition”, is described by the word “fertile”, which is usually related to the concept / domain, “land / soil”. Metaphor identification can be useful in numerous applications that require understanding of the natural language such as machine translation, information extraction, and automatic text summarization. For example, the word “жесткая” in Russian may have a metaphorical meaning of “tough” when applied to the word “политика” (policy) or a literal meaning of “hard” when applied to the word “кровать” (bed).

Metaphoric expressions have a variety of syntactic structures. In this paper, we focus on three major types of syntactic structures discussed in the work of (Krishnakumaran and Zhu, 2007). In type 1 expression, a subject noun is related to an object noun by a form of the copula verb “to be” (e.g., “God is a father”). In type 2 expression, the subject noun is associated with a metaphorically used verb and an object noun (e.g., "The actor painted his relationship with…"). Type 3 expression is an adjective-noun phrase (e.g., "sweet child"). An expression of one of the three structures above may or may not be a metaphor. Therefore, in this paper we treat the problem of metaphor identification as a binary classification problem of labelling a given expression as metaphoric or literal. The first step in our method is choosing a corpus representative of a given domain and building an annotated data set of labelled expressions from that corpus. The second step is feature extraction from the domain corpus, which is the main subject of our work. Then we induce a classification model for each expression type, using a set of annotated sentences, and evaluate its accuracy on a hold-out set.

This paper is organized as follows. Section 2 covers the state-of-the-art approaches to automated metaphor detection. Section 3 presents MIL (Metaphor Identification by Learning), a novel algorithm for metaphor detection. The proposed algorithm is empirically evaluated in Section 4. In Section 5, we conclude the paper with some insights and directions for future research.

2 Related Work

The work (Birke & Sarkar, 2006) focused on classifying the uses of verbs in a sentence as either literal or non-literal (type 2 expressions). They adopted the work of (Karov & Edelman, 1998), who worked on word sense disambiguation of words within their contexts (i.e., sentences).

Krishnakumaran & Zhu (2007) suggested three algorithms for distinguishing between live and dead metaphors. A dead metaphor is a metaphor that is already assimilated and familiar in the spoken language (e.g., "fell in love") and a live metaphor is a less familiar metaphor, which is not yet assimilated. The methodology of Krishnakumaran & Zhu 2007's work is based on conditional probabilities combined with WordNet (Fellbaum, 1999), which represents the language ontology.

The authors of (Neuman, et al., 2013) present three rule-based algorithms for metaphor identification in three expression types (1, 2, and 3). They suggest identifying metaphors by negating literalness. They define a set of rules for detecting whether a particular expression is literal, and if it is not literal, it is assumed to be a metaphor. Following the work of Turney, et al. (2011), they define as literal an expression comprised of words (e.g., verb and noun), which have in common at least one concrete category. Examples of concrete categories include *physical objects*, like "table", or *body parts*, like "hair". The ruleset defined by (Neuman, et al., 2013) builds upon multiple knowledge resources (such as Wiktionary and WordNet).

The work of (Shutova, et al. 2010) focuses on *semi-supervised learning* for identification of type 2 metaphors. The metaphor identification process is based on the principle of clustering by association, i.e., the clustering of words by their associative language neighborhood using the verb's subject as an anchor. A domain-independent approach to type 2 metaphor identification is presented in (Shutova, et al., 2013). The authors report a high precision of 0.79, but no information about the system recall and F-measure is provided.

Turney, et al. (2011) provide a solution for the type 3 metaphor classification problem called the *Concrete-Abstract* algorithm. In contrast to previous works that treated this issue as a sub-problem of the Word Sense Disambiguation problem, Turney, et al. (2011) suggest identifying metaphors by considering the abstractness level of words in a given expression. They found that in a wide range of type 3 metaphoric expressions, nouns tend to be abstractive while adjectives tend to be concrete. Their methodology builds upon a unique algorithm for ranking the abstractness level of a given word by comparing it to 20 abstract words and 20 concrete words that are used as paradigms of abstractness and concreteness. The main limitation of this approach is that it uses a single feature (abstractness level) that covers a limited range of expressions. The classification model used by Turney, et al. (2011) is logistic regression.

The authors of (Hovy, et al., 2013) use SVMs with tree kernels for supervised metaphor classification based on a vector representation of the semantic aspects of each word and different tree representations of each sentence. They report the best F1 measure of 0.75 without specifying the distribution of metaphor types in their dataset. Very similar results (F-measure = 0.76 for English) are reported for type 2 and type 3 expressions by (Tsvetkov, et al., 2014) who use three categories of features: 1) abstractness and imageability, (2) word supersenses (extracted from WordNet), and (3) unsupervised vector-space word representations. Using translation dictionaries, the authors apply a trained English model to three other languages (Spanish, Russian, and Farsi). The English dataset used by (Tsvetkov, et al., 2014) included 3,737 annotated sentences from the *Wall Street Journal* domain.

3 MIL (Metaphor Identification by Learning)

3.1 Overview

The first step in our proposed methodology is choosing a domain corpus, which represents a particular domain (area) of text documents (e.g., Reuter's news articles). The next step is the extraction of candidate expressions that satisfy one of the three syntactic structures discussed above (types 1, 2, and 3). This can be done using existing natural language processing tools. We also use the domain corpus to construct a word-context matrix to be used in the feature extraction phase. Then we manually annotate a selected subset of candidate expressions, since in a large corpus, it is not feasible to annotate them all. Each selected expression is labeled as 'literal' or metaphoric' by native language speakers.

Then we perform feature extraction, the largest and most important task in our work. The goal of feature extraction is to compute statistical features that may differentiate between metaphor and literal expressions. After the feature extraction is complete, a feature selection process must be carried out for choosing the features that are most relevant to the classification task.

The next step is inducing a classification model by running a supervised learning algorithm (e.g., C4.5). The model is built separately for each syntactic type. The resulting classification model can be used for classification of new expressions as literal or metaphoric. The model performance can be evaluated by such measures as precision and recall using cross-validation.

3.2 Domain Corpus Pre-processing

The basic actions on a domain corpus are parsing the corpus into sentences, tokenization, stemming the tokens by the Porter Stemmer algorithm (Porter, 1980), removing stopwords, and then calculating the frequency of each token. Then we build a co-occurrence matrix for words with frequency of 100 and higher (like in Turney et al., 2011). The co-occurrence matrix is used for calculating the abstractness level of a given word. The idea behind this matrix is that the co-occurring words are more likely to share an identical concept. We denote the co-occurrence matrix by *F*. Then we calculate the Positive Pointwise Mutual Information (PPMI) of *F*. The purpose of PPMI is to prevent the bias that can be caused by highly frequent words. We calculate PPMI as described in (Turney and Pantel, 2010) and get a new matrix denoted as *X*.

The X matrix is smoothed with truncated Singular Value Decomposition (SVD) (Deerwester, et al., 1990), which decomposes X into a multiplication of three matrices $X \approx U_k S_k V_k^T$. The matrix U_k represents the left singular vectors, the matrix S_k is a diagonal matrix of the singular values and the matrix V_k^T represents the right singular vectors. The idea behind using SVD is that a conceptual relation between two words can be indicated by a third word called a "latent factor" (e.g., the relation between *soccer* and *basketball* can be established by the word *sport* even if that word does not occur in text).

SVD calculation has two parameters. The first one is k, which represents the number of latent factors. We manually calibrated k starting from the value of 1000 as used in the work of (Turney, et al., 2011) and reduced it by 100 at each iteration in order to reduce the running time (more latent factors means longer running time), without decreasing the quality of results. We stopped when there was a substantial decrease in the results accuracy. In this manner, we have set the best value of k to 300. The other parameter is p, which adjusts the weights of the latent factors as in (Caron, 2001). We adopted its value from the work of (Turney, et al., 2011) and set it to 0.5. The second matrix we use is the multiplication of $U_k S_k^p$. We use this matrix for computing the *semantic similarity* of two words as a cosine similarity of two matrix rows (Turney and Pantel, 2010). We annotate $U_k S_k V_k^T$ as *matA* and $U_k S_k^p$ as *matB*.

Finally, we represent each of the corpus documents by calculating the average vector of the frequent words the document contains. The idea behind that is that in our view, each document represents some concept of its own and it can contribute to the identification of the concept transition.

3.3 Feature Extraction

The proposed feature set contains four types of features: features, which apply to every single word in a candidate expression (two features for type 1 and type 3 expressions, three features for type 2), features, which apply to every word pair in a candidate expression (one feature for type 1 and 3, three features for type 2), features, which apply to the sentence containing the candidate expression, and features, which apply to the candidate expression itself.

The first set of statistical-based features builds upon the idea of concrete-abstract mapping (Turney et al., 2011). They present a measure of the abstractness level, which can be calculated for each word in a candidate expression. We are using this measure to define the following features:

- *Abstract Scale*. The abstractness level of a word according to the algorithm presented in (Turney et al., 2011) is calculated as:

$$\sum_{k=1}^{20} pCor(word, abstract\ paradigm) - \sum_{k=1}^{20} pCor(word, concrete\ paradigm) \quad (1)$$

where *word* is a semantic vector of the target word, *abstract paradigm* is a semantic vector of a very abstractive word (e.g., sense), *concrete paradigm* is a vector of a very concrete word (e.g., donut) and $pCor$ is the Pearson correlation. The abstract scale rank is normalized between zero and one. The semantic vectors are taken from a pre-processed matrix based on words' co-occurrences. The full list of 20 abstract paradigm words and 20 concrete paradigm words is given in (Turney et al., 2011). This feature is applied to every word in a given expression.

- *Abstract Scale difference*. The absolute difference between the abstractness levels of every two words in an expression.
- *Abstract Scale Average / Variance*. The average / variance of the abstractness levels of all frequent words in the sentence that contains the candidate expression.

We also define a set of statistical-based features that can indicate a conceptual mapping between different word categories. These features include word-level features, document-level features, and domain-level features. Word-level features are features which are associated with the semantic meaning of a given word. Document-level features are features which are associated with the entire document's meaning (document which contains the expression) and domain-level features are features, which are associated with the entire domain corpus. The statistical-based features are defined below:

Word-level features

- *Semantic Relation*. The semantic relation value between every two words in a given expression. This value is taken from the associated entry of this words pair in the pre-processed *matA* (the large NxN matrix). Low values are an indication for low semantic relation between two words and thus imply metaphoric behavior (the conceptual mapping definition) and vice versa.
- *Semantic Relation Average / Variance*. The average / variance of the semantic relation values between every two words in the sentence that contains the candidate expression (including the expression itself).

- *Cosine-similarity*. The cosine similarity between every two words in a given expression. The cosine similarity is between the two vectors associated with the given words pair, taken from the pre-processed *matB* (the low dimensionality matrix). Low values indicate low conceptual relation between two words and thus imply metaphoric behavior.
- *Cosine-similarity Average / Variance*. The average / variance of the cosine similarity between every two words in a given expression. Low values indicate low conceptual relation between two words and thus imply metaphoric behavior.
- *First k Words*. For each two words in a given expression, we first find the *k* most similar words to the noun (if both words are nouns then the subject noun is selected). We then calculate the average cosine similarity of the second word to those *k* words and use it as a feature. A similarity between two words is computed as the cosine similarity of their associated vectors in *matB*. The value *k* = 30 was chosen based on the quality of classification results.

Document-level features

- *First k Documents*. Same as *First k Words*, based on document vectors rather than word vectors. Considering a document as a topic/s oriented, its representation by its word vectors average is actually the semantic representation of its topic/s. Here we also set *k* to 30 after manual calibration considering the results quality (F-measure).
- *Documents Jaccard Similarity*. It is calculated between semantically related neighborhoods of each two words in a given expression. The word neighborhood is defined as a window of ± 5 words surrounding a given word in the same sentence. In the previous features, we detect a conceptual mapping between two words by enriching *one* of the words with its semantically related neighborhood. In this feature, we take this idea one-step further and enrich *both words* with their semantically related neighborhoods. The feature is the Jaccard similarity of these two neighborhoods, calculated as follows:

$$\frac{SDT(w1,T) \cap SDT(w2,T)}{SDT(w1,T) \cup SDT(w2,T)}. (2)$$

Where *SDT(x)* is a group of documents that are related to word *x* according to a predefined threshold *T*. In our experiments, we manually set *T* to 0.35 after trying different values.

Domain-level features

- *Domain Corpus Frequency*. The normalized frequency of a word in the domain corpus. This feature is extracted for every word in the candidate expression. A low word's frequency in a given corpus can indicate metaphoric behavior since the word is not strongly related to the given domain and thus is used there as a metaphor.
- *Domain Corpus Frequency Difference*. The absolute value of the difference between the frequencies of every two words in a given expression.
- *Positive Point-wise Mutual Information*. This feature (based on Turney and Pantel, 2010) is applied to every two words in a given expression.

3.4 Feature and Model Selection

We used the Wrapper method of (Kohavi & John, 1997) to select the best features for each expression type. This method gets as input a classifier (e.g., decision tree) and a criterion to maximize (e.g., accuracy), and finds the subset of features that maximizes this criterion. We applied this method to each expression type. The criterion we used was the F-measure. We applied the following classifiers: Logistic Regression, Naïve Bayes, K-Nearest Neighbors (KNN), Voting Features Intervals (VFI), Random Forest, Random Tree, and J-48. We also combined each one of them with the AdaBoost algorithm.

All algorithms were run on Weka (Hall, et al., 2009), an open source implementation of Machine Learning algorithms. For each domain and expression type, we used the Wrapper method with 10-fold cross-validation to choose the algorithm and the feature set that provided the maximum average F-measure value over the ten splits of the dataset.

4 Experimental Results

4.1 Corpora

Our results in this paper are based on the Reuters Corpus, which was previously used as a domain corpus in (Neuman, et al., 2013). It consists of 342,000 English documents, which include 3.9 million sentences.

We used the same annotated corpus as in (Neuman, et al., 2013). The annotated corpus was constructed by extracting from the domain corpus sentences containing one of the five target nouns related to the concepts of "government" and "governance" in both literal and metaphorical sense: Father, God, Governance, Government, and Mother. Every selected sentence was parsed with the Stanford Part-of-Speech Tagger (de Marneffe & Manning, 2008). Candidate expressions having one of the three syntactic structures were independently denoted as metaphoric or literal by four human judges who were given the following definition of a metaphoric expression:

Literal is the most direct or specific meaning of a word or expression. Metaphorical is the meaning suggested by the word that goes beyond its literal sense.

The annotators were also given examples of literal and metaphorical expressions of types 1, 2, and 3. Inter-annotator agreement, measured in terms of Cronbach's alpha, was 0.78, 0.80, and 0.82 for type I, II, and III, respectively (Neuman, et al., 2013).

Finally, an expression was labeled as metaphoric if at least three judges out of four considered it as such; otherwise, it was labelled as literal. **Table 1** shows the distribution of expressions by their type and label (Literal / Metaphorical) in the set of annotated sentences.

Table 1. Candidate Expressions, Reuters Annotated Set (Neuman, et al., 2013)

Annotators Decision\ Type	Type 1	Type 2	Type 3
Literal	40 (33.3%)	242 (48.6%)	576 (75.8%)
Metaphorical	80 (66.7%)	256 (51.4%)	184 (24.2%)
Total	120	498	760

4.2 Comparative Results

In this section, we present the comparative results of MIL vs. two state-of-the-art algorithms - Concrete-Abstract (Conc-Abs) of Turney, et al. (2011) and CCO of Neuman, et al. (2013), which so far have reported the best performance, in terms of the F-measure on all three types of metaphoric expressions. The following performance measures were calculated using 10-fold cross-validation: precision, recall, and F-measure. The comparative results are shown in Tables 2-4 for expressions of type 1, 2, and 3, respectively. The Conc-Abs and CCO results are replicated from (Neuman, et al., 2013) and they refer to the same domain (Reuters) and the same set of annotated sentences.

Table 2. Comparative results type 1, Reuters

	MIL	Conc-Abs	CCO
Precision	86	76.5	83.9
Recall	92.5	76.5	97.5
F-measure	89.2	76.5	90.1

Table 3. Comparative results type 2, Reuters

	MIL	Conc-Abs	CCO
Precision	65.2	63.9	76.1
Recall	77	67.2	82
F-measure	70.6	65.4	78.9

Table 4. Comparative results type 3, Reuters

	MIL	Conc-Abs	CCO
Precision	46.8	0	54.4
Recall	39.7	0	43.5
F-measure	42.9	0	48.3

The MIL algorithm has clearly outperformed the Concrete-Abstract algorithm in terms of F-measure with an advantage of 12.7%, 5.2%, and 42.9% for expression types

1, 2, and 3, respectively. Moreover, all type 3 expressions were identified by the Concrete-Abstract algorithm as literal leading to the F-measure of zero. The most likely reason for that is that our dataset rarely contains expressions where the noun is an abstract noun (e.g., the noun "thoughts" in the expression "dark thoughts" is an abstract noun) and most metaphoric expressions contain concrete nouns (e.g., the noun "heart" in the expression "broken heart" is a concrete noun). Since Conc-Abs relies on a single feature, which is the noun's abstractness level, it cannot detect a metaphor in these cases. However, CCO outperformed MIL, especially in type 2 and type 3 expressions. Unlike MIL, which is a supervised learning approach, CCO is a rule-based method, requiring for each new language and domain a significant amount of manual expert labor along with multiple high-quality knowledge resources, which are unavailable for most human languages. The F-measure results reported by (Tsvetkov, et al., 2014) for type 2 and type 3 expressions (76%) are also better than the results reached by MIL, but their system is dependent upon a massive knowledge resource –WordNet.

Table 5 shows the list of features and the classifier selected by the Wrapper method of (Kohavi & John, 1997) for each expression type. We can conclude from the selected feature list that the feature *First k Documents* is a general feature, since it has been selected in all three expression types. The following features have been selected for two expression types out of three: *Domain Corpus Frequency* (Types 1 and 3), *Cosine-similarity Variance* (Types 1 and 3), and *Cosine-similarity Average* (Types 2 and 3). These results imply that for detecting conceptual mapping between two words in a given expression, it can be useful to consider the semantic neighborhood of each word as well as its frequency in the domain corpus.

Table 5. Features and classifier selected by the Wrapper method, Reuters

Expression Type	Selected Features	Selected Classifier
Type 1	Semantic Relation First *k* Documents Domain Corpus Frequency Abstract Scale Average Cosine-similarity Variance	Random Forest
Type 2	Abstract Scale First *k* Documents First *k* Words Semantic Relation Average Cosine-similarity Average	AdaBoost with Naïve Base
Type 3	Cosine-similarity First k Documents Abstract Scale difference Documents' Jaccard Similarity Domain Corpus Frequency Domain Corpus Frequency Difference Cosine-similarity Average Cosine-similarity Variance	AdaBoost with VFI

5 Conclusion

In this paper, we have presented a novel supervised learning approach for automatic metaphor identification in three syntactic structure types. We have extended the single feature set used by Turney, et al. (2011) with a large amount of statistical features. We have shown a significant improvement vs. a learning-based algorithm (Concrete-Abstract). However, MIL was outperformed by a rule-based algorithm (CCO), which applies a set of rules to a candidate expression in order to determine if it is a literal or not. In CCO, the rules are generated separately for each of the three expression types, and if a candidate expression satisfies all of them, it is labelled as literal. Otherwise, it is labeled as a metaphor. Although CCO outperformed MIL, it has some major disadvantages. One of the major disadvantages is that the rules are based on a relatively large amount of linguistic resources, including COCA (Corpus of Contemporary American English http://www.ngrams.info/), ConceptNet (http://conceptnet5.media.mit.edu/), WordNet (https://wordnet.princeton.edu/), and Wiktionary (https://en.wiktionary.org/wiki/English).
Future research on using statistical features for metaphor detection may include experimentation with additional predictive features, domains, and languages. Transfer learning across different domains may also be explored.

References

Krishnakumaran, S., & Zhu, X. (2007). Hunting Elusive Metaphors Using Lexical Resources. *Proceedings of the Workshop on Computational Approaches to Figurative Language*, (pp. 13-20). Stroudsburg, PA, USA.

Birke , J., & Sarkar, A. (2006). A Clustering Approach for the Nearly Unsupervised Recognition of Nonliteral Language. *In Proceedings of EACL-06*, (pp. 329–336).

Caron, J. (2001). Experiments with LSA scoring: Optimal rank and basis. *Proceedings of the SIAM Computational Information Retrieval Workshop*.

de Marneffe, M.-C., & Manning, C. D. (2008). The Stanford typed dependencies representation. *Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation* (pp. 1-8). Association for Computational Linguistics.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. (1990). Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science*, 391-407.

Fellbaum, C. (1999). *WordNet: An Electronic Lexical Database*. Blackwell Publishing Ltd.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The WEKA Data Mining Software: An Update. *SIGKDD Explorations, 11*(1).

Hovy, D., Srivastava, S., Jauhar, S. K., Sachan, M., Goyal, K., Li, H., . . . Hovy, E. (2013). Identifying metaphorical word use with tree kernels. *Proceedings of the First Workshop on Metaphor in NLP*, (pp. 52-57).

Karov, Y., & Edelman, S. (1998). Similarity-based word sense disambiguation. *Computational Linguistics*, 41-59.

Kohavi, R., & John, G. H. (1997). Wrappers for feature subset selection. *Artificial intelligence, 97*(1-2), 273–324.

Neuman, Y., Assaf, D., Cohen, Y., Last, M., Argamon, S., Howard, N., & Frieder, O. (2013). Metaphor Identification in Large Texts Corpora. *PLOS ONE*, 1-9.

Shutova, E., Korhonen, A., & Teufel, S. (2013). Statistical Metaphor Processing. *Computational Linguistics*, 301-353.

Shutova, E., Sun , L., & Korhonen, A. (2010). Metaphor Identification Using Verb and Noun Clustering. *COLING '10 Proceedings of the 23rd International Conference on Computational Linguistics*, (pp. 1002-1010). Beijing.

Tsvetkov, Y., Boytsov, L., Gershman, A., Nyberg, E., & Dyer, C. (2014). Metaphor Detection with Cross-Lingual Model Transfer. *ACL-2014*, (pp. 248-258).

Turney, P. D., & Pantel, P. (2010). From Frequency to Meaning : Vector Space Models of Semantics. *Journal of artificial intelligence research*, 1-48.

Turney, P., Neuman, Y., Assaf, D., & Cohen, Y. (2011). Literal and Metaphorical Sense Identification through Concrete and Abstract Context. *Proceedings of the 2011 Conference on the Empirical Methods in Natural Language Processing*, (pp. 680-690).

A Peculiarity-based Exploration of Syntactical Patterns: a Computational Study of Stylistics

Mohamed-Amine Boukhaled, Francesca Frontini, Jean-Gabriel Ganascia

LIP6 (Laboratoire d'Informatique de Paris 6), Université Pierre et Marie Curie and CNRS (UMR7606), ACASA Team, 4, place Jussieu,
75252-PARIS Cedex 05 (France)
{mohamed.boukhaled, francesca.frontini, jean-gabriel.ganascia}@lip6.fr

Abstract. In this contribution, we present a computational stylistic study and comparison of classic French literary texts based on a data-driven approach where discovering interesting linguistic patterns is done without any prior knowledge. We propose an objective measure capable of capturing and extracting meaningful stylistic syntactic patterns from a given author's work. Our hypothesis is based on the fact that the most relevant syntactic patterns should significantly reflect the author's stylistic choice and thus they should exhibit some kind of peculiar overrepresentation behavior controlled by the author's purpose with respect to a linguistic norm. The analyzed results show the effectiveness in extracting interesting syntactic patterns from novels, and seem particularly promising for the analysis of such particular texts.

Keywords: Computational Stylistics, Interestingness Measure, Sequential Pattern Mining, Syntactic Style

1 Introduction

Computational stylistics is a subdomain of computational linguistics located at the intersection of several research areas such as natural language processing, literary analysis and data mining. The goal of computational stylistics is to extract style patterns characterizing a particular type of texts using computational and automatic methods (Craig 2004). When investigating the writing style of a particular author, the task will automatically explore linguistic forms of his style, which is not only distinguishing features, but also the deliberate overuse of certain structures by the author compared to a linguistic norm (Mahlberg 2012). However, the notion of style in the context of computational stylistics appears to be wide enough, and is manifested on several linguistic levels: lexicon, syntax, semantics and pragmatics. Each level has its own markers of styles and its own linguistic units that characterize it.

Many works have been done in the literature to analyze the stylistic traits on these different linguistic levels (Biber 2006, Biber & Conrad 2009, Ramsay 2011, Frontini et al. 2014; see Siemens & Schreibman, 2013 for a discussion and overview). In this contribution, syntactic style will be targeted.
In their study Quiniou et al. (2012) have shown the interest of using sequential data mining methods for the stylistic analysis of large texts. They have shown that relevant and understandable patterns that are characteristic of a specific type of text can be extracted using sequential data mining techniques such as sequential pattern mining.
However, the process of extracting textual patterns is known by its property of producing a large amount of patterns, even from a relatively small sample of text. Thus, a measure of interest is to be applied to identify the most important and relevant patterns for the characterization of the text's style in question.
In this paper, we present a computational stylistic study of classic texts of French literature based on a data-driven approach where the discovery of interesting linguistic forms is done without any prior knowledge. Specifically, the proposed method is based on the assessment of the peculiar overrepresentation of syntactic patterns extracted using sequential data mining technique from texts with respect to a norm corpus. This method is intended to quantitatively support a textual analysis by focusing on the verification of the degree of importance of each syntactic pattern (syntagmatic segments with potential gaps), and by extracting the syntactic patterns that characterize the syntactical style of a work by a particular author.

2 Approach for extracting relevant syntactic patterns

Our method consists of two steps. First, a sequential pattern mining algorithm is applied to the texts in order to extract recurrent syntactic patterns. Second, a peculiarity-based interestingness measure that evaluates of the overrepresentation (in terms of frequency of occurrence with respect to a norm corpus) is applied to the set of extracted syntactic patterns. Thus, each syntactic pattern will be assigned an interestingness value indicating its importance and its relevance for the characterization of text's syntactic style. In what follows, we present in section 2.1 the corpus used in our experience, and its dividing protocol into two parts: text to analyze and text used as norm. Then, section 2.2 introduces some elements necessary to understand the process of extracting sequential syntactic patterns. Finally, the formulation and the statistical details of the proposed interestingness measure are presented in Section 2.3.

2.1 Analyzed Corpus

In our study, we used four novels, belonging to the same genre and the same literary time span, written by four famous classic French authors: Balzac's *"Eugenie Grandet"*, Flaubert's "*Madame Bovary*", Hugo's "*Notre Dame de* Paris" and Zola's "*Le ventre de Paris"*. This choice is motivated by our particular interest in studying the style of the classical French literature of the 19th century. At the time of the analysis of the syntactic patterns, each text written by one of the four authors is contrasted with texts written by the three other authors. That is to say that these three texts will be considered as norm corpus from which we will evaluate the hypothesis of the overrepresentation of syntactic patterns in the fourth remaining text, as explained later in this section.

2.2 Extraction of syntactic patterns

In our study we consider a syntagmatic approach. The text is first segmented into a set of sentences, each sentence is then represented by a sequence of syntactic labels (POS-tag)[1] corresponding to the words of the sentence using Treetagger (Schmid 1994). This produces at the end a set of syntactic sequences for each text. For exemple, the sentence "*Le silence profond régnait nuit et jour dans la maison.*" Will be represented by the sequence:

$$< DET, NOM, ADJ, VER, NOM, KON, NOM, PRP, DET, NOM, SENT >$$

Then, sequential patterns of a certain length with their supports (a number indicating how many sentences contain the pattern) are extracted from this syntactic sequential database using a sequential pattern extraction algorithm (Viger et al. 2014). Syntactic pattern consists of a sequential syntagmatic segment (with possible gaps) present in the syntactic sequences. It can be considered as a kind of generalization of the notion of n-gram widely used in the field of automatic language processing. Examples of syntactic patterns present in the sequence of the example above:

- $< DET >< NOM >< ADJ >$
- $< NOM >< ADJ >< VER >< NOM >$
- $< KON >< NOM > <*$[2] $> < DET >< NOM >$

To avoid the effect of statistical fluctuations on the analysis of patterns with low supports, we considered a support's threshold of 1%. That is to say that we focus only on patterns that are present in at least 1% of the sentences of the analyzed text. However, as sequential pattern mining is known to produce a large quantity of patterns even from relatively small samples of texts,

[1] Frech treetagger tagset:
http://www.cis.unimuenchen.de/~schmid/tools/TreeTagger/data/french-tagset.html

[2] <*> denotes a gap that can be filled with any POS tag

an interestingness measure should be applied on these patterns in order to identify the most important ones. This interestingness measure is explained in the next section.

2.3 Evaluation of the relevance of syntactic patterns

Our hypothesis to evaluate the relevance of a syntactic pattern is based on the fact that the most relevant ones should significantly reflect the stylistic choice of the author and should thus be characterized by a significant peculiar quantitative behavior, this peculiar behavior translate into a support's over-representation in his texts.
However, to capture this overrepresentation one cannot refer only to the absolute frequency of occurrence (support) Indeed, more frequent use of a syntactic pattern by an author (which translates into a relatively high support) does not necessarily indicate a stylistic choice since it can be very well a property imposed by the grammar of the language or by syntactic features that are characteristic of text's genre.
Thus, to assess the over-representation of a pattern, we use an empirical approach based on the comparison of the support of a syntactic pattern in a text to that found in a norm corpus. A ratio α between these two quantities is calculated as follow:

$$\alpha = \frac{\text{frequency of a pattern in the norm corpus}}{\text{frequency pattern in the text}}$$

In our experiments we found empirically that the distribution of the ratio α exhibits a Gaussian behavior. Indeed, the values of the α ratio are normally distributed around a central value (see Fig. 1). This is due to the fact that the frequency of occurrence of a syntactic pattern in a text is highly correlated with the frequency of occurrence in the norm corpus with a few exceptional special cases or outliers (see Fig. 2). These outliers represent the patterns of special interest for our study because they represent a certain linguistic deviation that is specific to the author's style compared to what one would expect to see in the norm corpus.

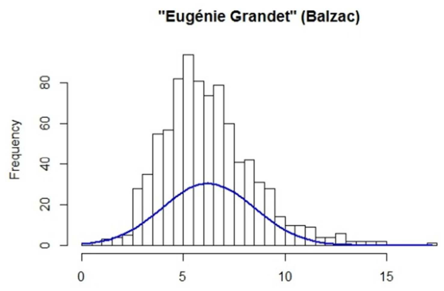

Fig. 1. Gaussian behaviour of the ratio α in Balzac's "*Eugénie Grandet*" novel

The configuration described above allows us to use an outlier detection method based on Gaussian distribution and Z-score to identify such special patterns (Chandola et al. 2009). The over-representation of a pattern in this case will result in a greater negative aberrant behavior compared to other patterns. The most over-represented patterns will be those associated with lowest values of standard *z*-score Z. The *z*-score values are calculated as follows:

$$Z_i = \frac{\alpha_i - \hat{\alpha}}{S}$$

Where α_i and Z_i are respectively the ratio α and the *z*-score corresponding to the i-th syntactic pattern. $\hat{\alpha}$ and S are respectively the mean and standard deviation of the ratio α .

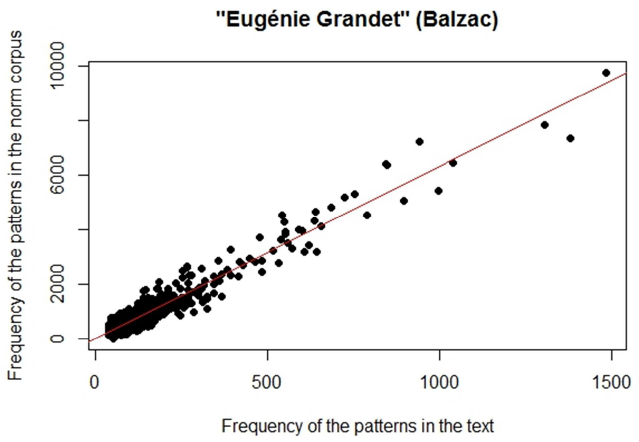

Fig. 2. Frequencies of syntactic patterns in a text with respect to their frequencies in the norm corpus for the studied novel. Each point in the graph represents a syntactic pattern. The plotted lines represent the linear regression lines capturing the expected behaviour of the α ratio

3 Results and Discussion

In this section, we present some examples of relevant syntactic patterns extracted from our corpus. Using the proposed method, the extracted patterns seem to have a strong relevance to characterize the style of the authors of our corpus but also to the novels' content and the literary genre in which it operates. In the Flaubert's *Madame Bovary*, several extracted patterns well represent the rhythmic rather than functional role of punctuation that is peculiar to the style of Flaubert (Mangiapane 2012). For example pattern (1) captures instances of a comma preceding the conjunction, followed by a parenthetical clause.

Pattern (1) <PUN> < KON>< PUN> <PRP>, with support= 113, sample instances of the pattern in the text:

- , et , à
- , mais , avant
- ; et , à

In *le Ventre de Paris* of Zola, and in the same direction, the syntactic patterns extracted as relevant clearly represent the use of nested clauses to describe situations or attitudes in the novel such as in the pattern (2), or to describe public places and objects in displays in long lists as in the pattern (3):

Pattern (2) : <PUN> <PRP> <PRP> <NOM>, support= 104, sample instances of the pattern in the text (bold text):
« Florent se heurtait à mille obstacles **, à des porteurs** qui se chargeaient **, à des marchandes** qui discutaient de leurs voix rudes ; il glissait sur le lit épais d' épluchures et de trognons qui couvrait la chaussée , il étouffait dans l' odeur puissante des feuilles écrasées .»

Pattern (3): <NOM> <PUN> <PRP> <NOM> <ADJ>, support= 68, sample instances of the pattern in the text (bold text):

- angles , à fenêtres étroites
- très-jolies , des légendes miraculeuses
- écrevisses , des nappes mouvantes

In *Eugénie Grandet* of Balzac, other different communicative functions are performed by the syntactic patterns and their textual instances, for example:

Pattern (4): <PUN> <VER> <NAM> <PRP>, support= 49, which is used as post-introducer of direct speech. This rather formulaic way of specifying (in a parenthetical form) the utterer of a reported speech is common to all, but seems to be strongly preferred by Balzac, while the other authors have shown a more varied style in introducing dialogues. Sample instances of the pattern in the novel:

- , dit Grandet en
- , reprit Charles en
- , dit Cruchot en

Pattern (5): <NUM> <NUM> <NOM>, support= 54, is a pattern used to refer to money, which is typical for the novel scenario where money plays a very important role. Sample instances of the pattern in the novel:

- vingt mille francs
- deux mille louis
- sept mille livres

Pattern (6) : <ADV> <VER> <PRO> <ADV>, support= 59, is used to express negative questions :

- n' avait -il pas
- ne disait -on pas
- ne serait -il pas

Pattern (7) : <PUN> <NOM> <PUN> <VER>, support= 44, represent the punctuation extensively used to mimic spoken intonation and even to reproduce performance phenomena such as stutter. :

- , messieurs , cria
- , madame , répondit
- , mademoiselle , disait

The few analyzed examples indicate that the presented technique is effective in extracting interesting syntactic patterns from a single text, and this seems particularly promising for the analyses of such classic literary texts.
On the other hand, this technique, as well as other similar ones, prompts the question of what is really captured by significant patterns. Some structures may be significant because they are typical of an author's style, its fingerprint - as we may say borrowing a metaphor often used in attribution studies, or they may be dictated by functional needs, due to the particular topic of the novel, or to the conventions of the chosen genre. This is particularly true for syntactic analysis, where the functional constraints on the authorial freedom are more evident. Much further works have to be carried out concerning this issue.

4 conclusion

In this paper, we have presented an objective interestingness measure to extract meaningful stylistic syntactic patterns from a given author's work. Our hypothesis is based on the fact that the most relevant syntactic patterns should significantly reflect the author's stylistic choice and thus they should

exhibit some kind of peculiar overrepresentation behavior controlled by the author's purpose. To evaluate the effectiveness of the proposed method, we conducted an experiment on a classic French Corpus. The analyzed results show the effectiveness in extracting interesting syntactic patterns from this type of text.
Based on the current study, we have identified several future research directions such as exploring other statistical measures to assess the interestingness of a given syntactic pattern, and expanding the analysis to include morpho-syntactic patterns (form and lemma words). Finally, we intend to experiment with other languages and text sizes using standard corpora employed in the field of computational stylistics at large.

References

Biber, D., 2006. *University language: A corpus-based study of spoken and written registers*, John Benjamins Publishing.

Biber, D. & Conrad, S., 2009. *Register, genre, and style*, Cambridge University Press.

Chandola, V., Banerjee, A. & Kumar, V., 2009. Anomaly detection: A survey. *ACM Computing Surveys (CSUR)*, 41(3), p.15.

Craig, H., 2004. Stylistic analysis and authorship studies. *A companion to digital humanities*, 3, pp.233–334.

Frontini, F., Boukhaled, M.A. & Ganascia, J., Linguistic Pattern Extraction and Analysis for Classic French Plays.

Mahlberg, M., 2012. *Corpus stylistics and Dickens's fiction*, Routledge.

Mangiapane, S., 2012. Ponctuation et mise en page dans Madame Bovary: les interventions de Flaubert sur le manuscrit du copiste. *Flaubert. Revue critique et génétique*, (8).

Quiniou, S. et al., 2012. What about sequential data mining techniques to identify linguistic patterns for stylistics? In *Computational Linguistics and Intelligent Text Processing*. Springer, pp. 166–177.

Ramsay, S., 2011. *Reading machines: Toward an algorithmic criticism*, University of Illinois Press.

Schmid, H., 1994. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of the international conference on new methods in language processing*. pp. 44–49.

Siemens, R. & Schreibman, S., 2013. *A companion to digital literary studies*, John Wiley & Sons.

Viger, P.F. et al., 2014. SPMF: A Java Open-Source Pattern Mining Library. *Journal of Machine Learning Research*, 15, pp.3389–3393.

Mining Comparative Sentences from Social Media Text

Fabíola S. F. Pereira

Faculty of Computer Science
Federal University of Uberlândia (UFU)
Uberlândia, Minas Gerais, Brazil
fabfernandes@comp.ufu.br

Abstract. Comparative opinions represent a way of users express their preferences about two or more entities. In this paper we address the problem of comparative sentences mining focused on social medias. We propose a genetic algorithm able to mine comparative sentences from short sentences based on sequential patterns classification. A comparison among classifiers regarding comparative sentences analysis is also presented. Our results indicate better accuracy for the proposed technique against literature baseline approaches, reaching accuracy levels of 73%.

Keywords: opinion mining, comparative sentences, genetic algorithm, social media mining

1 Introduction

Comparative opinions represent a way of users express their preferences about two or more entities. Mining comparative sentences from texts can be useful in several applications. For instance, a company might be interested in social media rumors of a new product release among consumers. Or, what are the best and worst features of the new product from consumers viewpoint? Nowadays, social medias are great source of this kind of information and mining comparative opinions from them seems to be a very promising direction to unveil valuable knowledge.

Many researches have been done in the field of regular opinion and sentiment classification [3,2]. However, *comparative opinions* represent a different viewpoint of users and an interesting research area. According to [8], a regular opinion about a certain car X is a statement like *"car X is ugly"*. On the other hand, a comparison is like *"car X is much better than car Y"*, or *"car X is larger than car Y"*. Clearly, these sentences have rich information from which we can extract knowledge with specific mining techniques.

In [5] the authors proposed a classification technique for mining comparative sentences based on grammatical sequential patterns. In this paper, based on [5]'s background, our goal is to stress techniques for mining comparative sentences focused on Twitter social data analysis. We argue that social medias *corpora*, as a great source of users opinions, must be explored and specific mining algorithms are needed.

The main contributions of this paper are: (1) a publicly available dataset crawled from Twitter. We manually labeled 1,500 tweets as comparative or non-comparative; (2) the genetic algorithm GA-CSR to aggregate to the problem of mining comparative sentences; and (3) a set of experiments comparing our approach with state-of-the-art techniques.

This paper is organized as follows: in Section 2 we introduce the problem of mining comparative sentences, highlighting social medias texts. In Section 3 we discuss techniques proposed in related work and present our proposal. In Section 4 the experimental results are showed. Finally, Section 5 concludes the paper.

2 The Problem of Mining Comparative Sentences

In the context of our study, comparative opinions are opinions that express a relation based on similarities or differences between two or more entities. According to [6], there are four types of comparisons: *non-equal gradable comparison* ("XBox is better than Wii-U"), *equative* ("XBox and Wii-U are equally funny"), *superlative* ("XBox is the best among all video games") and *non-gradable comparison* ("XBox and Wii-U have different features"). The first three types are called *gradable comparative* and are our focus because the sentences allow to establish a preference order among entities being compared.

Definition 1 (Comparative Opinion [6]). *Comparative opinion is a sextuple (E1, E2, A, PE, h, t), where $E1$ and $E2$ are the entity sets being compared based on their shared aspects A, $PE \in \{E1, E2\}$ is the preferred entity set of the opinion holder h, and t is the time when the comparative opinion is expressed. For a superlative comparison, if one entity set is implicit (not given in the text), we can use a special set U to denote it. For an equative comparison, we can use the special symbol $EQUAL$ as the value for PE.*

Example 1. Let us consider the following comparative sentence: "@stephthelamekid tbh wii u games do have better graphics than ps4 and xbox 1 games", posted by user Dinotia_4 in 12/06/2014. The comparative opinion extracted is: ({Wii U games}, {PS4 games, XBox One games}, {graphics}, {Wii U games}, Dinotia_4, 12/06/2014)

One challenge on the problem of comparative sentences is that not all sentences with POS tags JJR, RBR, JJS and RBS (comparative and superlative POS tags) are comparative. For example, *"faster is better."* Moreover, some expressions are comparative, but just can be identified through context, e.g *"PS4 is expensive, but Wii-U is cheap."*

3 Comparative Sentences Mining Techniques

To the best of our knowledge, the most representative technique in literature that addresses the problem of comparative sentences is [5], which is based on sequential pattern mining. In the following we present two new approaches: a naive approach based on *n-grams* classification and a genetic algorithm approach. The technique from [5] is also summarized in this Section.

3.1 *N-grams* Classification

The technique of document representation through term vector is the most common in the sentiment analysis field and can be used as our baseline. In this approach, each sentence in the corpus is a *document*, *terms* are the most relevant words and we use TF-IDF matrix [6] to represent them. Such matrix is, therefore, submitted to a classifier that builds a model able to identify whether a given sentence is comparative or not.

In this work, just *unigrams* have been considered. We did three pre-processing steps: (1) stop words removal, (2) stemming and (3) 1000 features extraction based on information gain index [10]. As we will present in Section 4, the results obtained with this approach were not expressive, even varying the classification algorithms.

3.2 Sequential Patterns Classification

Sequential patterns classification for comparative sentences mining had been proposed in [5]. Sequential pattern mining (SPM) is an important data mining task [1]. A sub-sequence is called sequential pattern or frequent sequence if it frequently appears in a sequence database, and its frequency is no less than a user-specified minimum support threshold *minsup* [4].

According to [5], a class sequential rule (CSR) is a rule with a sequential pattern on the left and a class label on the right of the rule. Unlike classic sequential pattern mining, which is unsupervised, in this approach sequential rules are mined with fixed classes. This method is thus supervised. For a formal definition, please refer to [5].

After defined the task of mining class sequential rules, then we deploy the algorithm to our problem. However a sentence cannot be handle simply from raw words, as we did on *n-grams* classification approach (Subsec. 3.1). To find sequential POS tags patterns in sentences and, then, build an input dataset of sentences to be classified (supervised learning) as comparative or non-comparative the following steps are needed:

1. **Sentences with pivot keywords.** Many words in English language indicate comparisons, for example *beat*, *exceed*, *outperform* etc. Moreover, those ending with *-est* and *-er* are naturally comparative or superlative adverbs and adjectives. Thus, a set of comparative keywords is considered. The idea is to identity sentences with at least one keyword and use the words that are within the radius of 3 of each keyword in the sentence as a sequence in our data.
2. **Replacing with POS tags.** For each sequence of max length of 7 obtained in previous phase, replace all words with their corresponding POS tags.

3. **Labeling sequences.** For each sequence, we have to label it as *comparative* or *non-comparative*. This is the same label that originated the sequence.
4. **Generating CSR.** In this phase we have to mine sequential patterns. The algorithm PrefixSpan [9] have been used with minimum support 0.1 and minimum confidence 0.6.
5. **Building dataset for classification task.** To translate class sequential rules into input for classification algorithms, the following steps are considered: each CSR is a feature. The classes are *comparative* and *non-comparative*. Each sentence from original corpus is a tuple in dataset. If the sentence matches a given CSR, the value is 1. Otherwise, 0. Each sentence keeps with its class. In this way, we have a well-formed input to a classifier algorithm.
6. **Running the classifier.** In paper [5] the authors use just the Naive Bayes classifier. In our experiments, we also considered the algorithms SVM, Multi-Layer Perceptron (MLP) and Radial Basis Function (RBF).

Example 2. In order to illustrate steps 1 to 4, let us consider the sentence: *"this/DT game/NN is/VBZ significantly/RB more/JJR fun/NN with/IN Kinect/NN than/IN without/IN it/PRP."* It has the keyword *more* and the generated CSR is:

$$<\{NN\}\{VBZ\}\{RB\}\{moreJJR\}\{NN\}\{IN\}\{NN\}> \rightarrow \text{comparative}$$

3.3 Genetic Algorithm

In this paper we propose a genetic algorithm for mining comparative sentences. The idea is to mine class sequential rules (CSR) from [5] (Subsec. 3.2). However, we do not use a classifier, but a genetic algorithm (GA-CSR) to get rules.

Each chromosome represents a CSR. Chromosomes have fixed length of 8 genes, where the first 7 are the sequential patterns with itemsets of length 1 (default gene) and the last one is the class with value *comparative* or *non-comparative* (class gene). Each default gene is an itemset and can assume POS tags domain values. Moreover, for each default gene we have the additional bit '1' or '0' representing whether or not it is part of sequential pattern. The example chromosome coding and its meaning is described in Figure 1. In the following we detail GA-CSR features.

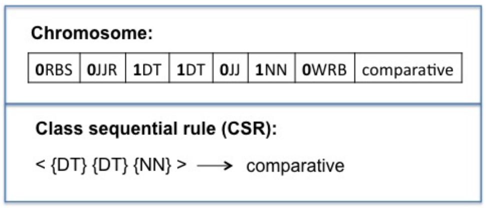
Chromosome:
0RBS
0JJR
1DT
1DT
0JJ
1NN
0WRB
comparative
Class sequential rule (CSR):
< {DT} {DT} {NN} > ⟶ comparative

Fig. 1: Chromossome coding

- **Fitness.** The rules fitness ($Fitness$) in the population is calculated based on a function containing two terms, namely Specificity (Sp) and Sensitivity (Se), where $Sp = TN/(TN + FP)$, $Se = TP/(TP + FN)$ and $Fitness = Se * Sp$. The variables TP, TN, FP and FN correspond to true positives, true negatives, false positives and false negatives, respectively.
- **Population creator.** The population is randomly generated.
- **Selection and crossover.** Two best chromosomes are selected by applying roulette wheel selection method and two point crossover method is applied over them to generate new children chromosomes. The class gene is not considered.
- **Mutation.** The mutation process changes the value of an attribute to another random value selected from the same domain. It can occurs in any gene type and does not consider the flag bit of each gene.
- **Insertion and removal operator.** Insertion and removal operators control the size of a rule. Insertion operator activates the gene by setting its flag bit and removal operator deactivates a gene by resetting the flag bit with a varying probability P_i and P_r, respectively.
- **Survivor selection.** GA-CSR uses fitness-based selection where individuals are selected from the set composed by parents and offspring. The top T_p fitness individuals are selected, where T_p is the population size.

In the end, we have a set of class sequential rules and just those greater than minimum support and confidence are considered for test and model validation.

4 Experimental Settings

In this section we report our experiments. We aim to compare best classification accuracies. Section 4.1 describes the datasets used to train and test the models. It also presents our experiments set-up and parameter setting. Finally, we expose our results in terms of success rate of the classifiers in Section 4.2.

4.1 Datasets and Parameterization

We tested our algorithms over two datasets: Amazon product reviews and Twitter texts. Our goal is to show how text mining social medias is different because of specific features of text length and language.

The Amazon product review is about mp3 players and was obtained from [7]. Twitter dataset contains tweets about PlayStation 4 and XBox video games and we collect them from Twitter API[1]. Both datasets were manually labeled. In Table 1 we detail the datasets features.

Our test set is composed by 9 runs for each dataset. For the *n-grams* approach, we used 4 classifiers: SVM (SVM-Unigram), NB (NB-Unigram), MLP (MLP-Unigram) and RBF (RBF-Unigram). In the CSR approach we also use 4 classifiers: SVM-CSR, NB-CSR, MLP-CSR and RBF-CSR. Finally, we run our proposed genetic algorithm GA-CSR. In Figure 2 we present detailed parameters used for each approach.

[1] https://dev.twitter.com/rest/public

	DB-Amazon	DB-Twitter
# sentences	1000	1500
# comparative sentences	97 (9.7%)	199 (13.26%)
Texts dates	2003-2007	Dec 2014
Topic	mp3 players	XBox and PS4

Table 1: *Datasets* used for tests

	Unigrams	
Train/Test	10-fold cross-validation	
Pre-process	stop words, stemm, infogain	
# Features	1000	
	MLP-Unigram	RBF-Unigram
Momentum	0.8	-
Learning rate	0.6	-
# neurons in hidden layer	15	2
# hidden layers	1	1

(a) Parameters for *n-grams* approach

	CSR	
Train/Test	10-fold cross-validation	
Radius	3	
minsup	0.1	
minconf	0.6	
Sequential pat. algorithm	PrefixSpan	
	MLP-RCPS	RBF-RCPS
Momentum	0.8	-
Learning rate	1.0	-
# neurons in hidden layer	7	7
# hidden layers	1	1

(b) Parameters for CSR approach

	GA-CSR
Train/Test	10-fold cross-validation
Learning rate (T_m)	0.8
Crossover rate (T_c)	0.8
Insertion rate (P_i)	0.3
Removal rate (P_r)	0.3
minsup	0.1
minconf	0.6
# generations	100
Population size (T_p)	50
Fitness	$Se * Sp$

(c) Parameters GA-CSR approach

Fig. 2: Parameterization

4.2 Experimental Results

The first test set was performed over DB-Amazon dataset (Figure 3). We can observe a poor performance for *n-grams* approach. As expected, it is a simple baseline that does not take into account elaborated features of our mining problem. Varying classifiers algorithms does not impact on results that reach a maximum accuracy of 68.6% for RBF neural network.

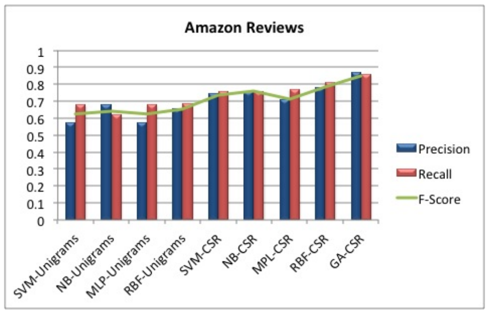

Fig. 3: Experimental results over DB-Amazon

Regarding CSR approach from [5], the results were similar to original paper. The difference is that in [5] just Naive Bayes classifier had been used. In our experiments we also considered other classification algorithms. The neural network RBF-CSR reached the best accuracy of 81.13%. Finally, our proposed genetic algorithm reached 85.23% of accuracy indicating the best approach for DB-Amazon dataset.

The second test set ran over DB-Twitter (Figure 4). Graphics curves maintained the trend, however the average accuracy decreased around 10%. This can be explained due to the large amount of noise in Twitter texts. Moreover, sentences grammatical errors potentially harm the grammatical pattern approaches.

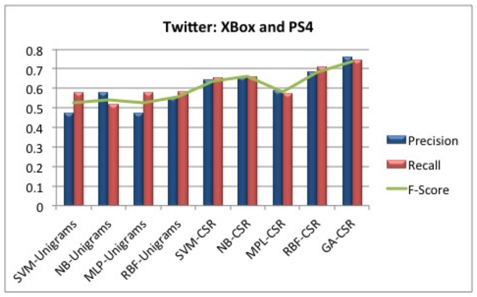

Fig. 4: Experimental results over DB-Twitter

5 Conclusion

In this paper we addressed the problem of mining comparative sentences. We carried out an experiment using 1,500 short sentences from Twitter.com, equally divided into two domain categories: *comparative* and *non-comparative* sentences. The results showed that the higher success rate was obtained with our genetic algorithm approach (73%). As our sample is relatively small, we used cross-validation (10-fold) to avoid overfitting and increase the accuracy of the success rate of the classifiers.

To ensure reproducibility of our results, in conjunction with the publication of this paper, we have released the full genetic algorithm GA-CSR code and Twitter data in the format used by our algorithm[2].

As future work, once mined comparative sentences, our focus will be on mining user preferences. We consider that comparative sentences are good source of users opinions, enabling the development of reasoning user preferences models from social data.

References

1. Agrawal, R., Srikant, R.: Mining sequential patterns. In: Proceedings of the Eleventh International Conference on Data Engineering. pp. 3–14. ICDE '95 (1995)
2. Arias, M., Arratia, A., Xuriguera, R.: Forecasting with twitter data. ACM Trans. Intell. Syst. Technol. 5(1), 8:1–8:24 (2014)
3. Ceron, A., Curini, L., Iacus, S.M.: Using sentiment analysis to monitor electoral campaigns: Method matters-evidence from the united states and italy. Soc. Sci. Comput. Rev. 33(1), 3–20 (2015)
4. Fournier-Viger, P., Wu, C.W., Tseng, V.: Mining maximal sequential patterns without candidate maintenance. In: Advanced Data Mining and Applications, vol. 8346, pp. 169–180 (2013)
5. Jindal, N., Liu, B.: Identifying comparative sentences in text documents. In: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 244–251. SIGIR '06 (2006)
6. Liu, B.: Sentiment Analysis and Opinion Mining. Morgan Claypool Pub. (2012)
7. McAuley, J., Leskovec, J.: Hidden factors and hidden topics: Understanding rating dimensions with review text. In: Proceedings of the 7th ACM Conference on Recommender Systems. pp. 165–172. RecSys '13 (2013)
8. Pang, B., Lee, L.: Opinion mining and sentiment analysis. Foundations and Trends in Information Retrieval 2, 1–135 (2008)
9. Pei, J., Han, J., Mortazavi-Asl, B., Wang, J., Pinto, H., Chen, Q., Dayal, U., Hsu, M.C.: Mining sequential patterns by pattern-growth: The prefixspan approach. IEEE Trans. on Knowl. and Data Eng. 16(11), 1424–1440 (Nov 2004)
10. Sharma, A., Dey, S.: An artificial neural network based approach for sentiment analysis of opinionated text. In: Proc. of the 2012 ACM Research in Applied Computation Symposium. pp. 37–42 (2012)

[2] `http://lsi.facom.ufu.br/~fabiola/comparative-mining`