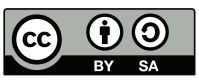
Text Analysis and Retrieval 2016

Course Project Reports

University of Zagreb

Faculty of Electrical Engineering and Computing



Publisher:

University of Zagreb, Faculty of Electrical Engineering and Computing

Organizers:

Domagoj Alagić
Mladen Karan
Jan Šnajder

Editors:

Domagoj Alagić
Mladen Karan
Jan Šnajder

ISBN 978-953-184-222-8

Course website:

http://www.fer.unizg.hr/predmet/apt/course_material

Powered by:

Text Analysis and Knowledge Engineering Lab
University of Zagreb
Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
http://takelab.fer.hr

TakeLab

Preface

It's hard not to notice how much information we are exposed to every day – we read friend's Facebook posts, perhaps a couple of daily gossips and news, we check reviews of new hardware we are planning to buy, and we send complaints to various service providers. Eventually, we both consume and produce a lot of information, which is, more often than not, expressed via text.

Making sense of such huge amounts of data (petabytes and counting!) has long been intractable using traditional data processing methods. However, in recent decades, machine learning (ML) made its way onto the scene, along with the text-specific fields of natural language processing (NLP), information extraction (IE), and information retrieval (IR). Considering how important and profitable data is today, it's hardly surprising that jobs related to these fields are now more sought after than ever.

In hope of making its Computer Science master students competitive in these fields, University of Zagreb, Faculty of Electrical Engineering and Computing (UNIZG FER) introduced Text Analysis and Retrieval (TAR) course three years ago in its AY 2013/2014 curriculum. TAR teaches the theoretical foundations of text analysis and retrieval methods, as well as their applications and best practices through hands-on practical assignments.

This booklet presents the results of student projects done as part of TAR coursework. This year, 68 students enrolled TAR – more than in the two previous editions of the course. Grouped in teams, they were offered 20 topics covering a wide range of NLP, IR, and IE tasks, targeting either English or Croatian data. Eventually, 12 topics piqued students' interest and got selected in the process. Almost all student teams (23/25) submitted their projects in time, along with their project reports, which you can find in this booklet.

We asked the students to write up their project reports in a form of a scientific paper, based on the guidelines we provided. In our opinion, by asking the students to put their project descriptions and results down on paper, as well as presenting it to the other students, we are teaching our students a valuable lesson – that presenting your own work is as important as the work itself. We are certain that the experience of writing a scientific paper will definitely help them in their future careers, be it in academia or industry. From a technical side, it is worth noting that the project results have not been reviewed or vetted, but we made sure to steer the students into the right direction, providing guidance and feedback on their work. Final reports and presentations were single-blind reviewed by the teaching staff, and students were asked to improve their reports based on our reviews. In the end, we think that no matter whether the project turned out to be decent or stunning, it will still be inspiring for the readers.

As our goal is to constantly improve this course, we are very grateful to all TAR 2016 students for enrolling this course and making it better through sheer enthusiasm and hard work.

Domagoj Alagić, Mladen Karan, and Jan Šnajder

Contents

Predicting Age, Gender, and Personality of Twitter Users

Antun Aleksa, Žad Deljkić, Ivan Sekulić

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
{antun.aleksa,zad.deljkic,ivan.sekulic}@fer.hr

Abstract

We propose a solution for a simplified version of the author profiling problem given at the PAN 2015 challenge. The solution is based on support vector machines for gender and age group generalization, as well as epsilon-support vector regression for estimating the personality of an individual. Our results are comperable to those of participants of PAN 2015.

1. Introduction

The goal of author profiling is to determine information about the unknown author of a given text. That information can range from age and gender estimation to estimating the personality of the individual. Author profiling is often applied in areas such as security, forensics and commercial domains (e.g., in a commercial domain, you can check what type of people like or dislike your product by examining their comments on public media). In this paper we propose a solution of a simplified version of the task that was given at the PAN 2015 event (Rangel et al., 2015).

PAN is a series of scientific events and shared tasks on digital text forensics. Shared tasks are computer science events that invite researchers and practitioners to work on a specific problem of interest. One of the tasks offered on the event is author profiling. The PAN 2015 author profiling task was to predict age, gender and personality traits (extroverted, stable, agreeable, conscientious, open) of an author bas on tweets writen in Spanish and English and also to predict only age and gender from tweets writen in Italian and Dutch. Our version of the task includes age, gender and personality traits prediction from English language based tweets. The models were trained on an annotated set of tweets gathered from 152 different people.

The remainder of this paper is organised as follows. Section 2 covers the state of the art methods, Section 3 describes our approach to solving the given task, and Section 4 presents the evaluation of our proposed method. We round up the paper with a conclusion in Section 5.

2. Related work

First contributions to author profiling task are made in predicting author's gender (Koppel et al., 2002). They employ simple lexical and syntactic features to infer the gender of the author of an unseen formal written document with approximately 80% accuracy. Cheng et al. (2011) try to tackle author gender identification problem of short length, multi-genre, content-free text, such as the ones found in many Internet applications. Argamon et al. (2003) explored differences between male and female writing in a subset of the British National Corpus. They managed to find significant differences between features of male and female authored documents. Pennebaker et al. (2003) show that there is a connection between words people use and their personality traits. These papers show that there is a basis for author profiling based on their written text. Schler et al. (2006) analyzed a corpus of tens of thousands of blogs with close to 300 million words and managed to achieve an accuracy of about 80% and about 75% for age identification.

Table 1: Example tweets

Happy Birthday! @username ????????????
WHAT THE ACTUAL FUCK. WHY IS THERE MORE TRAFFIC ????????????????
I kind of miss @username @username @username and @username ????

3. Approach

In this section we describe our approach to author profiling task. First, we preprocess the data, as described in next subsection. Then, we extract features from tweets of each author. Finaly, we employ machine learning algorithms on training data.

3.1. Preprocessing

First step in every natural language processing task is processing the text. We use a simple pipeline that consists of tokenizing the tweets, stemming the resulting tokens, and finaly removing the stop words and non-alpha-numeric characters. NLTK (Bird et al., 2009) is used for every part of the preprocessing pipline.

3.2. Features

We extract lexical features, representations of the text and task-specific features. Feature extraction consists of representing the tweet corpus as a bag of words, calculating the term frequency–inverse document frequency (tf-idf) for different words (both after doing preprocessing as described above) and a variety of additional features we considered to be relevant.

Those additional features consisted of calculating the ratio of:

- all caps words;
- internet slang words;

Text Analysis and Retrieval 2016: Course Project Reports (TAR 2016), pages 1–3, University of Zagreb, Faculy of Electrical Engineering and Computing, Zagreb, June 2016.

- hashtags;
- mentions;
- URL's;
- lenghtened words;
- exclamation points and question marks.

These features were extracted using simple regular expressions. Additionally, we compiled a list of common internet slang words ourselves that was used to extract them.

We decided on this specific set of features after several case studies of randomly selected users in our data set.

An example of such a user is user66, a young (18–24), extroverted (0.5) agreeable (0.3) and open (0.4) female. Few of her tweets are shown in Table 1.

Our theory was that many of these ratios may differ between genders and age groups in a statistically significant way which could then be used to differentiate between them.

Specifically, we theorized that a larger number of internet slang words (such as "lol" and "omg"), hashtags, all caps words and lengthened words (eg. "whaaaaaaat") may indicate a younger author. Similarly, out theory was that a larger number of mentions would indicate a more extroverted author.

3.3. Model

After extracting above described features, we proceed to the machine learning step. We use support support vector machine (SVM) for classification tasks (age and gender identification) and support vector regression (SVR) for computing personality traits. We did not focus on trying different models, since SVM is well known for tackling natural language processing tasks, and instead focused on better features.

We use grid search to optimize the hyperparametars for both SVM and SVR. There is no consistency in best parametars, so we did not present them here.

For classification tasks, we also employ feature selection. The goal is to find and use only the subset of features that will give the best results. In that way we simplify our model and reduce the learning time. We select k best features, as computed by χ^2 (Liu and Setiono, 1995) measure. The results with and without feature selection are given in the next section.

4. Evaluation

Table 2: Results for gender classification

Features	#feats	A	P	R	F1
BOW	all	0.55	0.57	0.57	0.57
Our model	all	0.57	0.57	0.58	0.57
BOW	100	**0.60**	**0.61**	**0.60**	**0.60**
Our model	100	0.54	0.55	0.54	0.54
BOW	500	0.57	0.57	0.57	0.57
Our model	500	0.55	0.57	0.57	0.57
BOW	2000	0.57	0.58	0.58	0.58
Our model	2000	0.56	0.59	0.58	0.58

Table 3: Results for age group classification

Features	#feats	A	P	R	F1
BOW	all	0.64	0.50	0.52	0.51
Our model	all	**0.68**	**0.57**	**0.56**	**0.56**
BOW	100	0.51	0.39	0.42	0.40
Our model	100	0.57	0.56	0.51	0.48
BOW	500	0.53	0.39	0.42	0.40
Our model	500	0.61	**0.57**	0.52	0.54
BOW	2000	0.63	0.49	0.51	0.49
Our model	2000	0.66	0.53	0.54	0.53

PAN 2015 corpora is composed of four datasets in different languages (English, Spanish, Italian and Dutch). The English dataset consists of 14166 tweets, composed by 152 people of different gender, age and personalities. We approach gender and age identification as classification tasks, while estimating personality traits is seen as a regression task.

Results for classification tasks are given in terms of accuracy, precision, recall and F_1 score, while regression is evaluated by computing mean absolute error. For evaluating the model, we use k-fold evaluation, with $k = 5$.

Results for age group identification are given in Table 3. Using SVM with all features generaly gives the best results for this task.

Table 2 shows our results for gender classification task. To our big surprise, the best results are achieved when using simple bag-of-words (BOW) text representation as the feature vector, combined with the feature selection that selects top 100 features.

Evaluation of our model on the big 5 personality traits is presented in Table 4. Our model mostly predicts values around 0.1 irregardless of everything else. Thus, we argue that results are not very applicable.

The results are in line with previous research from PAN 2015 author profiling task.

5. Conclusion

As the previous section shows, the goal of profiling authors from their tweets is achieved with reasonably good results.

Gender and age group classification had relatively good overall accuracy, while estimating the big 5 personality traits proved to be harder, which is in line with previous studies.

Overall, author profiling is an interesting problem with applications in forensics, security and marketing and as reasearch shows, today it's possible to achieve good results with well known machine learning algorithms. With future research, even further improvement should be possible.

References

Shlomo Argamon, Moshe Koppel, Jonathan Fine, and Anat Rachel Shimoni. 2003. Gender, genre, and writing style in formal written texts. *TEXT-THE HAGUE THEN AMSTERDAM THEN BERLIN-*, 23(3):321–346.

Table 4: Mean absolute error results for regression

Features	Agreeable	Conscientious	Extroverted	Open	Stable
BOW	0.12	0.14	0.13	**0.12**	**0.19**
Our model	**0.12**	**0.13**	**0.13**	0.13	0.20

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python*. " O'Reilly Media, Inc.".

Na Cheng, Rajarathnam Chandramouli, and KP Subbalakshmi. 2011. Author gender identification from text. *Digital Investigation*, 8(1):78–88.

Moshe Koppel, Shlomo Argamon, and Anat Rachel Shimoni. 2002. Automatically categorizing written texts by author gender. *Literary and Linguistic Computing*, 17(4):401–412.

Huan Liu and Rudy Setiono. 1995. Chi2: Feature selection and discretization of numeric attributes. In *tai*, page 388. IEEE.

James W Pennebaker, Matthias R Mehl, and Kate G Niederhoffer. 2003. Psychological aspects of natural language use: Our words, our selves. *Annual review of psychology*, 54(1):547–577.

Francisco Rangel, Paolo Rosso, Martin Potthast, Benno Stein, and Walter Daelemans. 2015. Overview of the 3rd author profiling task at pan 2015. In *CLEF*.

Jonathan Schler, Moshe Koppel, Shlomo Argamon, and James W Pennebaker. 2006. Effects of age and gender on blogging. In *AAAI Spring Symposium: Computational Approaches to Analyzing Weblogs*, volume 6, pages 199–205.

Plagiarism Detection Using Word2vec Model

Arijana Brlek, Petra Franjić, Nino Uzelac

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
{arijana.brlek, petra.franjic, nino.uzelac}@fer.hr

Abstract

Plagiarism denotes the act of presenting someone else's work as your own. Since the rise of the World Wide Web it has been identified as a serious problem for academia and the research community, among others, which is why there is a great interest for the development of automated plagiarism detectors. One of the core phases of an automated plagiarism detection system is text alignment. The task of the text alignment phase is to detect reused passages of text between two given documents. In this project report we present an approach to text alignment which builds on the winning algorithm from the PAN 2014 competition on plagiarism detection but adopts a semantic-based method for estimating sentence and passage similarity through the use of Word2Vec.

1. Introduction

Plagiarism can be denoted as the act of using another person's distinct ideas or words without explicit acknowledgment (Heneghan, 2012). In the recent decades an increase in text plagiarism has been reported both in academia and the research community. This phenomenon has largely been attributed to digitalization which made it possible to effortlessly reuse texts without appropriate citation. Since manual plagiarism detection is both time-consuming and severely limited by the human ability to memorize, recent years have shown a rising interest for the development of automated plagiarism detection systems.

Automatic plagiarism detection can be internal and external. Internal plagiarism detection aims to identify different writing styles within a suspicious document without specifying the documents which may have served as source for the plagiarized text. The goal of external plagiarism detection is to identify the plagiarized source documents, by comparing the suspicious document against a set of possible reference documents.

External plagiarism detectors typically carry out the process of detection in three steps. These are source retrieval, text alignment, and knowledge-based post-processing, as depicted in Figure 1.

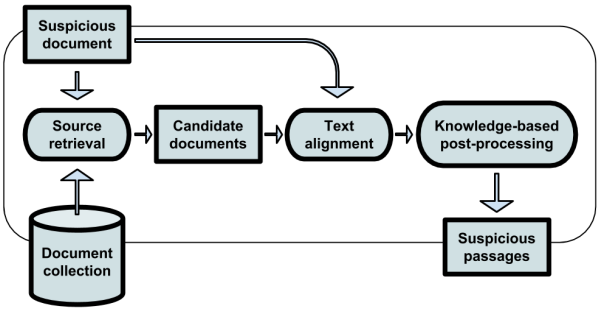


Figure 1: Three-stage process for plagiarism analysis (Stein et al., 2007)

The project presented in this report addressed the text-alignment phase. The text-alignment phase receives as its input a suspicious document and its possible source. Its task is to identify all maximal-length passages of reused text between the given pair of documents. This phase poses a difficult problem because plagiarized passages are rarely verbatim copied. Often the text is at least slightly modified, either by text insertion and deletion, sentence and word shuffling or synonym replacement.

The method described in this report builds upon the winning approach from the PAN 2014 competition (Sanchez-Perez et al., 2014), but relies on Word2Vec (Mikolov et al., 2013) and the approach suggested by Duhaime (2015) for measuring sentence similarity.

The algorithm employs a "seed and extend" paradigm borrowed from the field of gene sequence alignment in bioinformatics. The paradigm comprises four phases: (i) pre-processing, (ii) seeding, (iii) extension, and (iv) filtering. These phases have been identified as common building blocks for text alignment algorithms (Potthast et al., 2013).

We used Python and the Natural Language Toolkit (Bird et al., 2009) for the development of the algorithm, building on the code of Sanchez-Perez et al. (2014) which is publicly available in open source form.

The rest of the paper is structured as follows. In Section 2 we review related text alignment approaches in external plagiarism detection. Then, in Section 3 we explain the algorithm structure and how it works. In Section 4 we give the results of the algorithm run, alongside background information on the evaluation measures and the used corpora. Finally, Section 5 presents the conclusions drawn from the presented work.

2. Related work

Different text alignment algorithms have been described in detail in the notebooks for the PAN at CLEF competitions, so here our main focus is on plagiarism detection employing Word2Vec. Since Word2Vec is a relatively new addition to the set of NLP tools, no papers (that we are aware of) have been published dealing with its application on plagiarism detection. However, some web-sources like Duhaime (2015) deliver convincing evidence proving that Word2Vec is well-suited for cross-lingual plagiarism detection tasks.

Text Analysis and Retrieval 2016: Course Project Reports (TAR 2016), pages 4–7, University of Zagreb,
Faculy of Electrical Engineering and Computing, Zagreb, June 2016.

The mentioned method employs a modified version of the measure proposed by Alzahrani and Salim (2010) to determine the similarity between two sentences and we have based the similarity calculation of our seeding method on this approach. A detailed account of different fuzzy and semantic approaches to plagiarism detection can be found in (Alzahrani et al., 2012).

3. Methodology

3.1. Pre-processing

The goal of the pre-processing step is to filter out the irrelevant information contained in the suspicious and source documents. With this aim, we aplied the following substeps: (i) lowercasing, (ii) removal of apostrophes showing possesion (e.g. *woman's hat* $\rightarrow$ *woman hat*), (iii) sentence segmentation, (iv) joining of small sentences (less than 3 words) with the next sentence (v) removal of all characters except letters, (vi) tokenization, (vii) stop-words removal (using a list of 153 stop-words), (viii) WordNet lemmatization. In the last step quantifiers were excluded from lemmatization because of the observed anomaly of the WordNet lemmatizer which lemmatizes the quantifier 'less' to 'le'.

We also considered the possibility of expanding contractions (e.g. *how'd'y* $\rightarrow$ *how do you*), but this led to an overall deterioration of the results so this step was dismissed.

3.2. Seeding

The aim of the seeding phase is to find as many as possible small plagiarism cases connecting the suspicious text to its source. The seeding algorithm compares each of the sentences in the suspicious and the source text, evaluating how similar they are. In our particular approach, sentence similarity is computed through the cosine similarity between the Word2Vec representations of individual words in the sentences. If the similarity is greater than a certain threshold value $th1$, the pair is added to the set of possible plagiarism cases.

Word2Vec (Mikolov et al., 2013) denotes a group of algorithms that model semantic word representations. These algorithms are characterized by an unsupervised generation of word embeddings from textual documents. A word embedding $W : words \rightarrow \mathbb{R}^n$ is a parametrized function mapping words to a high-dimensional vector space. In this vector space, semantically similar words are closer.

The function used for the calculation of sentence similarity was adopted from Duhaime (2015) and is here presented as Algorithm 1.

Algorithm 1 Sentence similarity calculator

```
function SIMILARITY(sentenceA, sentenceB)
    stringSim ← empty_list
    for wordA in sentenceA do
        maxSim ← 0
        maxSimWord ← ""
        for wordB in sentenceB do
            sim ← cos(wordA, wordB)
            if sim > maxSim then
                maxSim ← sim
                maxSimWord ← wordB
        if maxSim > 0 then
            stringSim.add(maxSimWord)
    return sum(stringSim)/len(stringSim)
```

3.3. Extension and filtering

The task of the extension phase is to join the output seeds of the seeding phase into larger segments of plagiarized text. The filtering phase aims to improve the precision, at the expense of recall.The extension and filtering methods were adopted without significant alterations from the work of Sanchez-Perez et al. (2014) where they are described in detail.

In brief, in the extension step Sanchez-Perez et al. (2014) first group subsequent seeds into fragments of text with a certain gap allowed between the first and last sentence. These fragments (with dissimilar sentences possibly included) are then divided so that the their similarity exceeds a certain threshold value $th2$. If this condition cannot be met, the algorithm is applied with an adjusted, lower gap allowed between the outermost sentences. We calculate the similarity between fragments F_1 and F_2 through the Word2Vec representations (v) of words comprising them:

$$similarity(F_1, F_2) = \cos\left(\sum_{v \in F_1} v, \sum_{v \in F_2} v\right)$$

The filtering phase typically works by removing all aligned passages that fail to meet a certain criteria, either because they are too small or overlapping with other passages. Sanchez-Perez et al. (2014) attempted to merge overlapping fragments. The problem of too small fragments was dealt with by discarding those which had less than 150 characters.

4. Results

4.1. Corpus

The training and testing corpora were taken from the PAN 2014 competition.[1] The training corpus was left unchanged from the PAN 2013 competition and details of its construction are given by Potthast et al. (2013).

The corpus was constructed from the ClueWeb corpus 2009 [2] which consists of more than one billion web pages. These web pages formed the basis from which a set of documents dealing with specific topics was singled-out. The documents were then preprocessed and filtered and in the subsequent step a source set was generated that was used for the formation of each suspicious document. In the last step a suspicious document was constructed from the passages of the documents in the source set.

Four strategies were employed to mimic the result of typical plagiarist behavior while modifying passages: no obfuscation, random obfuscation, cyclic translation obfuscation, and summary obfuscation.

Random obfuscation applies operations such as shuffling, adding, deleting and replacing words or short phrases

[1] http://pan.webis.de/clef14/pan14-web/

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

at random, with the resulting passage ending-up semantically meaningless. Cyclic translation obfuscation submits the passage to a series of translations through web services, with the last step translating the passage into its original language. Lastly, summary obfuscation dealt with the uncredited presentation of another's ideas in one's own text in the form of a brief summary. The summary obfuscation corpora was generated by planting the summaries (human-generated) of documents from the Document Understanding Conference (DUC) 2001 in the DUC 2006 text corpus.

In its entirety, the training corpus amounted to 1827 suspicious and 3230 source documents which were grouped into 5185 pairs. Because of the computational cost we adjusted the parameters through grid search on the training corpus. This is also the reason why we tested only the smallest of the test corpora available (Test Corpus 1), which contained 1823 suspicious and 3164 source texts, combined into 518 pairs.

As for the Word2Vec model, we used the one pre-trained by Google on the Google News dataset. The model was trained on about 100 billion words and it contains 3 000 000 unique phrases represented as unit vectors in 300-dimensional space (Mikolov et al., 2013). Considering its size, the model significantly hampered the speed of the algorithm. We tried working with several other smaller corpora (Wordnet, Brown, Gutenberg) but found they lacked a significant portion of words contained in the plagiarism documents.

4.2. Evaluation measures

The performance of the plagiarism detector was evaluated with specific precision, recall, and granularity measures, all of which were combined into the so-called plagdet score to allow for unique detector ranking. The measures were designed by Potthast et al. (2010) and the evaluation script was downloaded from the PAN 2014 website[1].

The measures provide character-level evaluation of the detectors' performance. In the following expressions, S denotes the set of plagiarism cases in the corpus and R denotes the set of plagiarism detections. A plagiarism case $s \in S$ is a tuple $(s_{plg}, d_{plg}, s_{src}, d_{src})$ where the passage s_{plg} in d_{plg} is a plagiarized version of passage s_{src} in d_{src}. A plagiarism case is represented with a set $\mathbf{s}$ of references to characters of d_{plg} and d_{src} specifying passages s_{plg} and s_{src}. A plagiarism detection $r \in R$ is analogously represented with $\mathbf{r}$. Precision and recall are then defined as

$$prec(S,R) = \frac{1}{|R|} \sum_{r \in R} \frac{|\bigcup_{s \in S}(\mathbf{s} \sqcap \mathbf{r})|}{|\mathbf{r}|}$$

$$rec(S,R) = \frac{1}{|S|} \sum_{s \in S} \frac{|\bigcup_{r \in R}(\mathbf{s} \sqcap \mathbf{r})|}{|\mathbf{s}|}$$

$$\text{where } \mathbf{s} \sqcap \mathbf{r} = \begin{cases} \mathbf{s} \sqcap \mathbf{r}, & \text{if } r \text{ detects } s, \\ \emptyset, & \text{otherwise.} \end{cases}$$

The granularity measure specifically addresses the problem of multiple detections for a single plagiarism case. It is defined as

$$gran(S,R) = \frac{1}{|S_R|} \sum_{s \in S_R} |R_s|,$$

where $S_R = \{s | s \in S \wedge \exists r \in R : r \text{ detects } s\}$ and $R_s = \{r | r \in R \wedge r \text{ detects } s\}$.

The overall plagdet score is given with:

$$plagdet(S,R) = \frac{F_1}{\log_2(1 + gran(S,R))}$$

where F_1 is the equally weighted harmonic mean of precision and recall.

4.3. Parameter selection

To select the best parameters $th1$ and $th2$ for the algorithm, we performed grid search on the training corpus. The explored interval of the $th1$ parameter was from 0.3 to 0.7 with $step$ = 0.1, and the interval of $th2$ extended from 0.3 to 0.8, also with $step$ = 0.1. The results showed that the overall best Plagdet score was achieved for values $th1 = 0.6$ and $th2 = 0.6$.

4.4. Results

Table 1 shows the results of the algorithm on the PAN 2014 Test corpus 1.

Table 1: Results on the PAN 2014 Test corpus 1

Obfuscation	Plagdet	Recall	Precision	Gran.
None	0.7019	0.9167	0.5686	1.0
Random	0.6642	0.8585	0.5415	1.0
Translation	0.7096	0.9121	0.5808	1.0
Summary	0.8180	0.7004	0.9831	1.0
Entire	0.6986	0.8860	0.5766	1.0

The algorithm achieves a high Plagdet score (0.8180) on the summary obfuscation sub-corpus. It should be noted that the best result of the PAN 2012-2014 contestants with this particular type of obfuscation (although tested on PAN 2013 Evaluation Corpus 2 and 1) didn't exceed 0.6236 and the results were mostly within the range 0 - 0.5. This score gives reason to believe the algorithm could also perform well on the Test Corpus 2 so additional tests should be carried out to determine if it really is the best-performing algorithm with this type of task.

With its 0.6986 Plagdet score on the entire corpus, the algorithm could be placed in the 18th place in the overall algorithm line-up of the 2012-2014 PAN contestants. With its translation and random obfuscation sub-corpora scores it could be awarded 15th place. The worst performance of the algorithm, compared to the contestants is on the no-obfuscation corpus, where only one of the top-24 2012-2014 contestants doesn't outperform it.

It should be noted that in the first version of the algorithm we directly adopted the similarity from (Alzahrani and Salim, 2010). This initial calculation was unfairly punishing sentences which contained words which were not contained in the model. Upon appropriately modifying the

algorithm in the way proposed by Duhaime (2015), the score on nearly all sub-corpora improved by 0.1.

5. Conclusion

In this report we presented an algorithm for text alignment which is based on the winning approach from the PAN 2014 plagiarism detection competition but relies on semantic relations of word embeddings for calculating sentence and passage similarity.

According to our results, the combination of the approach of Sanchez-Perez et al. (2014) with Word2Vec seems particularly promising for the task of uncovering summary plagiarism cases. The results with other obfuscation techniques could be further improved, although they are still comparable to the ones of 2012-2014 PAN contestants. Since the algorithm for calculating sentence similarity was an adaptation of fuzzy similarity proposed by Alzahrani and Salim (2010), possible lines of future work could include exploring other fuzzy-based methods. It should be noted that more extensive performance evaluations should be carried out on larger corpora to adequately compare the proposed approach against the ones in the PAN competitions.

References

S. Alzahrani and N. Salim. 2010. Fuzzy semantic-based string similarity for extrinsic plagiarism detection: Lab report for pan at clef'10. presented at the 4th Int. Workshop PAN-10.

S. Alzahrani, N. Salim, and A. Abraham. 2012. Understanding plagiarism linguistic patterns, textual features, and detection methods. *IEEE Transactions on Systems, Man and Cybernetics*, 14(2):133–149.

S. Bird, E. Klein, and E. Loper. 2009. *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. O'Reilly.

D. Duhaime. 2015. Cross-lingual plagiarism detection with scikit-learn. `http://douglasduhaime.com/blog/cross-lingual-plagiarism-detection-with-scikit-learn`. Accessed: 09-06-2016.

Emily Heneghan. 2012. Academic dishonesty and the internet in higher education. `http://commons.trincoll.edu/edreform/2012/05/academic-dishonesty-and-the-internet-in-higher-education/`. Accessed: 04-06-2016.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. 2013. Efficient estimation of word representations in vector space. *Proceedings of Workshop at International Conference on Learning Representations*, pages 1576–1586.

M. Potthast, A. Barrón-Cedeño, Rosso, and B. Stein. 2010. An evaluation framework for plagiarism detection. *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 997–1005.

M. Potthast, M. Hagen, T. Gollub, M. Tippmann, J. Kiesel, E. Rosso, P. Stamatatos, and B. Stein. 2013. Overview of the 5th international competition on plagiarism detection. *CLEF 2013 Working Notes*.

B. Sanchez-Perez, G. Sidorov, and A. Gelbukh. 2014. A winning approach to text alignment for text reuse detection at pan 2014. *Notebook for PAN at CLEF 2014*.

B. Stein, S. Meyer zu Eißen, and M. Potthast. 2007. Strategies for retrieving plagiarized documents. *30th International ACM Conference on Research and Development in Information Retrieval*, pages 825–826.

Named Entity Recognition in Croatian Tweets using CRF

Filip Čulinović, Luka Kunić, Erik Perčić

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
{filip.culinovic,luka.kunic,erik.percic}@fer.hr

Abstract

Automatically recognizing named entities in a document is an important part of many natural language processing tasks. This paper presents a system for recognizing named entities in Croatian tweets. Due to the brevity and informal nature of tweets, this task can be significantly more difficult than finding named entities in news articles and other similar documents. Nevertheless, tweets contain an abundance of useful information and can be a very good source of data for information extraction systems.

1. Introduction

Named entity recognition (NER) is the task of extracting named entities from a text document. In the context of natural language processing, named entities can be the names of people, locations, and organizations, as well as dates, times, and other measurable quantities. Named entities are often the focus of many information retrieval tasks such as question answering or text summarization.

The most basic approach for recognizing named entities in a document would be to define a large set of rules which are used to detect various patterns in text. The main problem with this approach is the unstructured nature of text, which produces many exceptions to the predefined rules. An alternative and more successful approach is to use manually annotated data to train a NER system using supervised machine learning. This can be done using basic classifiers like Naive Bayes or Support Vector Machines (SVM), but better results are achieved using either Hidden Markov Models /HMM) or Conditional Random Fields (CRF). These models predict the output label for each token by estimating either the joint probability or the conditional probability of a sequence of labels in a given sentence.

The performance of NER systems is affected by various factors: language, writing style, document length, etc. A good example of a language that is not suited for NER is German in which all nouns are capitalized, making the task much more difficult because capitalization is usually a very good feature for determining whether a token is part of a named entity. Writing style is also very important because formally written documents like news articles are much more consistent than informal text found in social media posts. Also, when working with very short text snippets like tweets, it can be more difficult to properly identify the named entities due to the lack of context.

In this paper we present a system which recognizes named entities in a given set of Croatian tweets. This is done by training a CRF model using a set of manually annotated tweets. The output of the model is a set of labels that indicate whether a token is a part of a named entity or not. The preparation of the training dataset is described in Section 3. The CRF model and the selected features are described in Sections 4 and 5. The results and model evaluation are presented in Section 6.

2. Related work

Named entity recognition is a very popular task in NLP and many papers tackle the challenges of building reliable NER systems. The task becomes even more difficult when dealing with short texts such as tweets, and it has been shown by Liu et al. (2011) that NER systems designed for working with regular documents mostly will not suffice when finding named entities in tweets. They proposed a solution which combines a KNN classifier with CRF for a semi-supervised learning approach. They also used gazetteers to mitigate the lack of training data, and they have shown significant improvement over previously presented baselines.

The performance of many NLP systems which use supervised or semi-supervised machine learning largely depends on the choice of features for the models. Tkachenko and Simanovsky (2012) present comprehensive research on the affect of different feature combinations when training NER models. They used the same data set to perform feature comparison using their CRF system with Viterbi inference. They have shown that most features improved the performance of the recognizer, but they managed to find an optimal configuration by gradually removing some of the features in order to achieve similar results with a significantly reduced feature count.

3. Input data preparation

The tweets used for training the system were obtained from a dataset containing several million entries from Croatian and Serbian users. From this dataset we manually extracted 1200 tweets which were mostly published by news portals. Such tweets were chosen because they usually contain news titles or short descriptions of events containing named entities. For each tweet, part-of-speech tags were generated, and the named entities were manually annotated.

3.1. Part-of-speech tagging

Part-of-speech (POS) tagging is a method of determining the role of words in a sentence. Parts of speech can be verbs, nouns, adjectives, pronouns, etc. POS tagging is very useful for named entity recognition because named entities usually consist of nouns or noun phrases, which makes the POS tag a good feature for the model.

POS tagging was done using TakeLab's NLPToolkit (*http://www.takelab.fer.hr*) which is a simple framework for

Text Analysis and Retrieval 2016: Course Project Reports (TAR 2016), pages 8–10, University of Zagreb, Faculy of Electrical Engineering and Computing, Zagreb, June 2016.

preprocessing Croatian text. The tool takes unprocessed text as input, removes all stop-words, and outputs a CoNLL style formatted table containing POS tags for the remaining words.

3.2. Annotating the training dataset

The tweets were annotated by adding labels which denote whether a token is a part of a named entity. The labels are used to highlight three types of named entities: persons (PER), locations (LOC), and organizations (ORG). Also, if a named entity is a sequence of tokens, we differentiate the first token (by appending B) from the remaining tokens in the sequence (which are appended by I). Each token is assigned a label in form of short tags: e.g., [ORG-B] for the beginning of an organization name or [LOC-I] for the second word of a location name. All tokens which are not part of any named entity are labeled as *other* using [OTH].

Apart from regular words, tweets can contain several specific token types. Most common are *hashtags* which are prefixed by the # symbol, and *mentions* prefixed by @. Before preprocessing the test data, hashtags are completely removed because they often just indicate the topic of the tweet and do not have any syntactic meaning. Mentions are kept, but we do not treat them as named entities despite the fact that they are often used to name a specific user.

Annotating the tweets entirely manually would be a long and tedious process. Therefore, we designed an annotation app which facilitates the process by providing an easy to use labeling UI, as shown in Figure 1. The app takes raw text as input and splits it into tokens. The user can highlight the tokens which are part of the named entity by chosing the named entity type and clicking the token. After all named entities are properly highlighted, the app generates the annotation labels.

The tweet annotation was performed by three annotators, where each annotator had labeled 333 tweets, and an additional 200 tweets were labeled by all three annotators. The combined IAA score for the 200 tweets was measured per token and only strict matches were considered as positive results. Correlation of around 91% was achieved. Most disagreements were caused by the misinterpretation of whether a named entity is a location or organization, which largely depends on the context. The most correlation was found in person tags as those are fairly common and easy to identify in text.

4. Feature selection

Arguably the most important step in the model design process is feature selection. In order to identify good features, we compiled a list of possible candidates which we later evaluated using cross-validation. The features which offered the most significant improvements to the results are:

- the POS tags we extracted during data preparation,
- the content of the token being labeled,
- whether the first character of the token is capitalized,
- whether the entire word is capitalized,
- whether the token ends in 'ic' or 'ić' (which is very common for Croatian last names and can improve the recognition of person named entities),

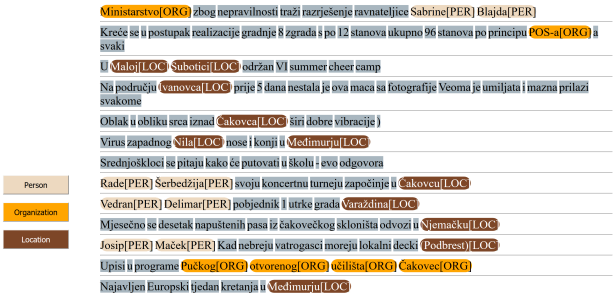


Figure 1: Highlighting the named entities in given tweets using the annotation app. The annotation mode can be chosen using the buttons on the left. The tokens with blue background will be labeled as [OTH].

- whether the token is a number,
- the last 2 characters of the token,
- the last 3 characters of the token (similar to 2 characters, but it was shown that combining the two features provides better results than having either of the two features individually),
- all of the above features for the previous and next tokens in the sentence.

We chose not to use any external dictionaries for finding common person names, locations, and organizations, although such data is often used in NER systems. Adding boolean features which indicate whether a word is in one of the dictionaries would most certainly increase the recognition accuracy, but we wanted to investigate what results can be achieved while only using features extracted from the input text.

5. The NER model

The input for the model consists of two text files containing the annotated tweets, and the preprocessed data with the POS tags. Since the preprocessing tool removes all stopwords, the POS tags need to be matched to the corresponding tokens in the annotated text. For each word we build a feature vector by extracting the previously described features.

The feature vectors are used as input for the recognition model based on Conditional Random Fields (CRF). As proposed by Lafferty et al. (2001), CRF are a framework for building probabilistic models to segment and label sequential data. The model accounts for the fact that the probability of an occurrence of a label depends not only on the current observation, but also on past and future observations. CRF is used to directly model the probability distribution of the entire output vector based on the input feature vector, giving the most probable label sequence $\mathbf{y}^*$ as

$$\mathbf{y}^* = \text{argmax}_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}) \quad (1)$$

where $P(\mathbf{y}|\mathbf{x})$ is the conditional probability of a label sequence $\mathbf{y}$ for the given input feature vector $\mathbf{x}$. While calculating the conditional probability, each feature in the feature

vector is assigned a weight λ_i indicating the relative importance of the feature. The conditional probability is given by

$$P(\mathbf{y}|\mathbf{x},\lambda) = \frac{1}{Z(\mathbf{x})} \exp\left\{\sum_{i=1}^{n}\sum_{j}\lambda_i f_j(y_{i-1}, y_i, \mathbf{x}, i)\right\} \quad (2)$$

where $Z(\mathbf{x})$ is the sum of the above exponents over all output vectors $\mathbf{y}$, and is used as a normalization factor in order to achieve a proper probability distribution. The weights $\lambda_\mathbf{i}$ are calculated using supervised machine learning, by training the model on a set of input data.

5.1. Implementation

To build our NER system, we used the python-crfsuite library (*https://github.com/tpeng/python-crfsuite*) which implements CRF using a form of stochastic gradient descent for parameter optimization. The library provides a trainer and a predictor. The trainer uses the input data to optimize the model parameters, and it outputs a configuration file containing the parameter values. The predictor uses the configuration file produced by the trainer to predict the label sequences based on given input.

The model does not look at every token individually, but bases its predictions on the entire input sentence. That is why the test cases consist of a set of feature vectors, where each vector contains features for one token in a sentence. The output of the model is a vector of labels which have been predicted by the model for the given input.

6. Results and evaluation

In order to evaluate the results of the CRF model, we built a simple baseline model which will be used for comparison. The baseline model does not make predictions based on extracted features, but uses label frequencies to randomly assign labels to the tokens. The frequencies are calculated from the manually annotated training data. As expected, the baseline model did not perform very well and only managed to correctly label several tokens by chance.

On the other hand, our model performed considerably better than the baseline with a combined F1 score of 37%, which is still quite low, but it provides a good starting point for further investigation. Breaking down the scores with regards to each label model provides good insight into the areas where the model had difficulties recognizing correct named entities. The breakdown of the results is shown in Table 1.

It turns out that the model is quite conservative when assigning labels, as precision is generally much higher than recall. This is probably caused by the lack of training data, as well as the fact that tweets often contain named entities written in lower case, which has a great impact on named entity recognition. The extent of this issue would most likely be reduced if we significantly increased the training data size.

The model was quite good at recognizing last names of people, which is clear from the fact that the [PER-I] label had the highest F1 score. The main reason for this is the 'ic/ić' feature which significantly increased the number of matches for the person labels when it was added. However, this has caused some false positives in cases where the tweet only contained the last name of a person, but the model wrongly labeled the last name's preceding word as the person's first name.

Table 1: The breakdown of the results a confusion matrix calculated by comparing the actual output labels with the model predictions.

Label	Precision	Recall	F1 score	True count
LOC	0.66	0.21	0.32	81
ORG	0.58	0.14	0.23	52
PER	0.59	0.35	0.44	128
Average	0.61	0.27	0.37	261

The model had most trouble with organizations, primarily because they were not very frequent in the training data. Also, organization names are often comprised of multiple words, some of which are commonly found in other parts of the text, so it can be difficult to distinguish whether a word is part of the organization named entity.

7. Conclusion

We implemented a CRF model for finding named entities in tweets. This task has proven to be quite challenging when compared to named entity recognition in larger text documents due to scarcity of data, unstructured and informal nature of the tweets, as well as a general lack of named entities in tweets. However, the results provide a good indication about the areas of the solution which might be improved. The primary focus should be expanding the training data set by manually annotating a larger number of tweets, as well as incorporating dictionaries of common named entities in order to generate additional relevant features.

References

John Lafferty, Andrew McCallum, Fernando Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of the 18th International Conference of Machine Learning (ICML 2001)*, pages 282-289.

Xiaohua Liu, Shaodian Zhang, Furu Wei, Ming Zhou. 2011. Recognizing Named Entities in Tweets. In *Proceedings of the 49th Annual Meeting of the Association for Computation Linguistics: Human Language Technologies - Volume 1*, pages 359-367. Association for Computational Linguistics.

Diego Mollá, Menno van Zaanen, Daniel Smith. 2006. Named Entity Recognition for Question Answering. In *Proceedings of the 2006 Australasian Language Technology Workshop (ALTW2006)*, pages 51-58.

Maksim Tkachenko and Andrey Simanovsky 2012. Named Entity Recognition: Exploring Features. In *Proceedings of KONVENS 2012 (oral presentations), Vienna, Austria, 2012*.

Named Entity Extraction on Croatian and Serbian Tweets Using CRFs

Matija Folnović, Antea Hadviger, Ivan Paljak

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
{mfolnovic,antea.hadviger,ipaljak}@gmail.com

Abstract

In this paper we present a system for named entity recognition (NER) in tweets written in Croatian and Serbian. Current work regarding NER is mostly focused on formal texts and state-of-the-art NER systems show a significant drop in performance when applied to informal and short texts like tweets. However, tweets are a valuable source for information extraction which makes the field worth exploring. We manually annotated a set of tweets and used conditional random fields (CRFs), a discriminative probabilistic model often used as an alternative to hidden Markov models (HMMs). We present the results in terms of precision, recall, and F1 measure.

1. Introduction

Social media generates endless streams of data, flowing in from Twitter, Facebook, Instagram, and other social sites and discussion boards like never before. Information found in tweets is often more up-to-date and inclusive than news articles, due to simplicity of tweeting and widespread availability of mobile devices. Given the fact that the volume of tweets is already extremely large and rapidly growing, it is natural to consider applying natural language processing methods over tweet corpora for data mining.

Named entity recognition (NER) is a subtask of information extraction which is used to locate and classify parts of text into predefined categories, such as names of persons, organizations, locations, events, temporal expressions, etc. (Wikipedia, 2016). It is highly valuable in practice, for example, for detecting trending people and events on Twitter or automatically tagging news articles. It could also be used as a subtask of more complex tasks, like relation extraction between named entities.

2. Related work

Current work regarding named entity recognition is mostly focused on formal texts, such as news articles. Those systems show a significant drop in performance when applied to a different genre of text, like tweets, as they are short (up to 140 characters), informal, prone to noise, unstructured, lacking of context, often grammatically and syntactically poor, or simply meaningless gibberish. Furthermore, named entities are not always capitalized and are often written in a colloquial manner, which directly affects selecting features for the NER model. For example, the average F1 of the Stanford NER (Finkel et al., 2005), which achieves state-of-the-art performance on documents, drops from 90.8% (Ratinov and Roth, 2009) to 45.8% on tweets (Liu et al., 2011).

Li et al. (2012) present an unsupervised method that does not depend on the unreliable local linguistics features, but it builds *local context* and *global context* for tweets. Experimental results show comparable performance with the state-of-the-art NER systems in real-life targeted tweet streams.

3. Methods

3.1. Data

Our data set is a collection of tweets written in Croatian and Serbian (mostly Serbian) which are not domain-specific. Annotation was done completely manually using an internally developed tool by the following rules:

- each token is annotated as a person name, organization, location or nothing
- all tokens are labelled according to the context
- PER (person) labels tokens which represent personal or fictional names, surnames, and nicknames
- ORG (organization) labels tokens which represent organizations, institutions, sports teams, political parties, music bands, etc.
- LOC (location) labels tokens which represent places, countries, cities, streets, rivers, mountains, etc.
- tokens starting with '@' (mentions) are not labelled
- tokens starting with '#' (hashtags) are labelled normally

We have annotated a total of around 5000 randomly selected tweets, where around 1250 tweets contained a labelled named entity. For tagging, we used BIO notation (B - beginning, I - inside, O - outside). In order to measure inter-annotator agreement (IAA), we annotated 200 tweets and computed Cohen's kappa coefficient, which showed a 91% agreement.

3.2. Model

Conditional random fields (CRFs) are a type of discriminative partially directed probabilistic model introduced by Lafferty et al. (2001), often used as an alternative to hidden Markov models (HMMs) as they offer several important advantages over them. CRFs directly model a conditional probability distribution over label sequences given a certain observation sequence, rather than a joint distribution over both label and observation sequences, resulting in the relaxation of the independence assumptions required by

Text Analysis and Retrieval 2016: Course Project Reports (TAR 2016), pages 11–13, University of Zagreb, Faculy of Electrical Engineering and Computing, Zagreb, June 2016.

HMMs. Furthermore, CRFs overcome the label bias problem of maximum entropy Markov models (MEMMs) and other conditional Markov models.

We used a linear chain CRF which defines the conditional probability as follows:

$$P(\mathbf{y}|\mathbf{x}, \boldsymbol{\lambda}) = \frac{1}{Z(\mathbf{x})} \exp\left\{ \sum_{i=1}^{n} \sum_{j=1}^{m} \lambda_j f_j(y_{i-1}, y_i, \mathbf{x}, i) \right\} \tag{1}$$

where $\mathbf{x}$ are the observations (tokens), $\mathbf{y}$ are the hidden labels (tags), f are the feature functions, n is the length of the sequence and m is the number of used features. $\boldsymbol{\lambda}$ is a vector of weights for each feature that is determined by supervised learning, by maximizing the likelihood of manually annotated data. Typically, parameter optimization is done using iterative scaling or gradient descent-based techniques.

To ensure that $P(\mathbf{y}|\mathbf{x})$ are indeed probabilities, a normalizing constant Z in equation 1 is calculated as follows:

$$Z(\mathbf{x}) = \sum_{\mathbf{y}} \sum_{i=1}^{n} \sum_{j=1}^{m} \lambda_j f_j(y_{i-1}, y_i, \mathbf{x}, i) \tag{2}$$

In linear chain CRFs, the feature functions f compute real values by taking into account our current and previous labels, y_{i-1} and y_i, the whole observed sequence $\mathbf{x}$, and the current position in the sequence i. For example, one of the binary feature functions we use is defined as:

$$f_j(\cdot) = \begin{cases} 1 & \text{if } x_i \text{ is all caps} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

We used an implementation of conditional random fields called CRFsuite, written in C++ by Okazaki (2007).

3.3. Features

Feature selection is an extremely important part of information extraction with machine learning. One of the key advantages of CRFs is their great flexibility to include a wide variety of arbitrary, non-independent features of the input. When using NER models on tweets rather than formal documents, different features should be selected in order get satisfying results.

In our work, we used the following features:

- original word and lowercased word
- stemmed word
- word shape (e.g., for word "Zagreb": ULLLLL)
- word type (e.g., *AllUpper*, *AllDigit*, *AllLetter*, etc.)
- POS tag (whole tag and only word category)
- prefixes (first k letters, $k \in [1, 4]$)
- suffixes (last k letters, $k \in [1, 4]$)
- binary features (contains only upper case letters, contains only lower case letters, contains an upper case letter, etc.)

For every token, all features are calculated in a window which is 5 tokens long, so 2 tokens before and 2 tokens after the current token are taken into consideration.

For stemming and POS tagging we used tools developed for Croatian language by Agić et al. (2013) and Pandžić (2012).

4. Evaluation

We have developed a simple baseline model B in order evaluate the performance of CRFs. The model B randomly chooses a tag for all tokens in a tweet, taking into account only probabilities for each tag, calculated using the training set. It does not consider any features of the observed token.

For evaluating our models, we filtered the annotated data by discarding those tweets that did not contain any tokens annotated as named entities. We split that set of tweets into two disjoint sets: training set containing 80% of the data, and test set containing the remaining 20%. We performed 5-fold cross-validations on the training set, trying to remove the features proved to be useless or negatively affecting the average F1-score over the 5 folds. However, our final model uses all the considered features as that gives the best results on the test set.

Table 1: Evaluation results (%)

Model	P	R	F1
B	14.77	14.97	14.86
CRFs	65.44	51.85	56.56

Table 1 shows evaluation results for both of our models evaluated on the test set, expressed in terms of macro-averaged precision, recall, and F1 score. It is evident that CRFs gives a significantly better performance than the baseline model. That was highly expected as the baseline model is very simple, whereas CRF uses various features of a token and its context.

Table 2: Evaluation results by tag for CRFs (%)

Tag	P	R	F1
O	95.07	98.36	96.68
I-PER	68.71	64.74	66.67
I-ORG	50.77	27.73	35.87
I-LOC	73.74	52.90	61.60
B-PER	73.33	72.13	72.73
B-ORG	25.00	17.65	20.69
B-LOC	71.43	29.41	41.67

Evaluation results for every type of tag on the CRFs model are shown in Table 2. We see that the model performs best when detecting persons and worst when labeling organizations. Personal names are mostly consisted of a single or a pair of tokens and often appear at the beginning of the sentence. Organization tokens probably pose a

problem because they can appear in various contexts and shapes, sometimes being consisted of multiple words, and sometimes as an abbreviation. Locations are somewhat easier to detect as there are a few prepositions that often appear directly before them.

5. Conclusion

In this paper we presented a system for named entity recognition on tweets based on a linear-chain conditional random fields (CRFs) model. Our model is built to recognize persons, organizations and locations in Croatian and Serbian tweets. It is trained and tested on a manually labeled set of randomly selected tweets. As state-of-the-art NER systems still perform rather poorly on informal texts like tweets, we did not expect spectacular results. Nevertheless, evaluation showed that CRFs model with an F1 score of 56.56% performs significantly better than our simple baseline model.

In the future, we propose training and testing the model on more annotated tweets and exploring more different features to try and match the models applied to formal texts. We could also consider adding more types of named entities to label.

References

Željko Agić, Nikola Ljubešić, and Danijela Merkler. 2013. Lemmatization and morphosyntactic tagging of croatian and serbian. In *4th Biennial International Workshop on Balto-Slavic Natural Language Processing (BSNLP 2013)*.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370. Association for Computational Linguistics.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Chenliang Li, Jianshu Weng, Qi He, Yuxia Yao, Anwitaman Datta, Aixin Sun, and Bu-Sung Lee. 2012. Twiner: named entity recognition in targeted twitter stream. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pages 721–730. ACM.

Xiaohua Liu, Shaodian Zhang, Furu Wei, and Ming Zhou. 2011. Recognizing named entities in tweets. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 359–367. Association for Computational Linguistics.

Naoaki Okazaki. 2007. Crfsuite: a fast implementation of conditional random fields (crfs), 2007. *URL http://www. chokkan. org/software/crfsuite*.

Ivan Pandžić. 2012. *Development of the Croatian stemmer for usage in information retrieval*. Ph.D. thesis, Filozofski fakultet, Sveučilište u Zagrebu.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, pages 147–155. Association for Computational Linguistics.

Wikipedia. 2016. Named-entity recognition — wikipedia, the free encyclopedia. [Online; accessed 30-June-2016].

Named Entity Extraction on Tweets in Croatian Language Using CRF

David Geček, Iva Marušić, Goran Rumin

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
{david.gecek,iva.marusic,goran.rumin}@fer.hr

Abstract

This paper describes the task of named entity recognition (NER) on tweets, mostly in Croatian language. For that task Conditional Random Fields (CRF) model was used for labeling names of persons, locations, and organizations. Prior to training the model we had to obtain a set of tweets, manually label them and calculate Inter Annotator Agreement (IAA). As the result of our work, we provided an evaluation of our model and stated problems we encountered while conducting the task.

1. Introduction

Named entity recognition (NER) is a method of recognizing named entities, atomic elements in text (Molla, 2008), in case of this paper: persons, organizations, and locations. For annotating purposes, we used "CONLL-2003: List of tags with associated categories of names" (Meulder, 2013). Persons, meaning any alias, first, middle, last names of people, animals or fictional characters. Organizations, meaning any word or words representing company, brand, political movement, government and its bodies, public organizations etc. Locations, such as roads, regions, structures, natural locations, public places, commercial places, countries, abstract places, etc. Some of the entities that do not fit in these categories are: any adjectives derived from entities important for this paper, religions, nationalities, events, wars, slogans etc.

Named entities are analyzed on tweets. NER could be done with carefully handcrafted rules, however, most of such systems tend to work well just in very specific situations. Secondly, those systems need properly structured text. Tweets have many misspellings, colloquialisms, and foreign words, so supervised machine learning is used. This gives us a lot of flexibility.

There are two supervised approaches (Baluja et al., 2000) to NER: classification and sequence labeling. Classification can not use labels from both sides of a token as features, so sequence labeling is used, concretely, Conditional Random Fields (CRF) (Lafferty et al., 2001). Supervised machine learning needs annotated data for learning. The authors of this paper served as annotators. Inter annotator agreement (IAA) is used for evaluating agreement between annotators and F-score is used for evaluating how well model annotates test tweets after training.

2. Methods

2.1. Dataset

To successfully accomplish our task, we were given a large textual file containing short messages called tweets. Given tweets were supposed to be written in Croatian language, but because it is sometimes difficult to distinguish Croatian from Serbian, file contained both Croatian and Serbian tweets.

Large text files are difficult to read, mostly because not many text editors can open large files. In order to train model used for NER task, we needed to extract appropriate number of tweets to annotate them. For this task we used files with predefined Croatian names of people, locations, and keywords that are usually found in the names of organizations. Based on those files we wrote a script that extracted tweets which contained at least one word defined by those files. We set a limit of 20 thousand tweets that were extracted from the original file. Unfortunately, not all of those tweets contained named entities, but most of them did. Another problem we encountered was that ratio of languages was 70:30 in favor of Serbian.

2.2. Data annotation

Three people worked on this project and annotated 5010 tweets, selected from extracted tweets. For annotations we used different labels based on type of named entity, and we made a distinction between first word of named entity and the rest of the words. For that task we used labels:

- PB - denoting first word of name of person, e.g., if named entity is Goran Ivanišević, Goran should be labeled as PB
- PI - denoting other words that are part of name of person besides first word, e.g., using same example as above, Ivanišević should be labeled as PI
- LB - denoting first word of name of location
- LI - denoting other words of name of location besides first word
- OB - denoting first word of name of location
- OI - denoting other words of name of organization besides first word
- O - denoting word which is not part of named entity

Data annotation is a crucial part of every NLP task that is based on the use of a supervised machine learning model. Data should be annotated consistently among all annotators and to decide if annotated data are of good quality, Inter Annotator Agreement needs to be calculated. If IAA is not high enough, the model will perform poorly. For determination of IAA score, each pair of annotators was given a

Text Analysis and Retrieval 2016: Course Project Reports (TAR 2016), pages 14–17, University of Zagreb, Faculy of Electrical Engineering and Computing, Zagreb, June 2016.

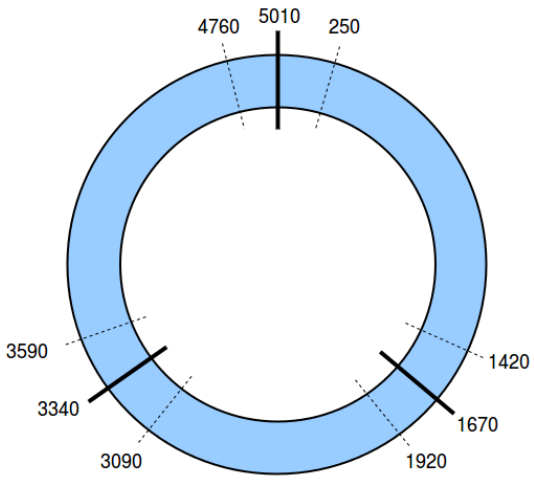


Figure 1: Way of splitting tweets between annotators

Table 1: IAA scores

Annotators	IAA score(%)
A1, A2	77.54
A1, A3	87.86
A2, A3	82.02
Average	82.47

collection of 500 tweets to annotate. Both annotators of same pair were annotating same set of tweets, while tweets between pairs were different. The way in which collection of tweets was split can be seen in the figure 1.

Based on annotation of those tweets, Cohen's kappa was calculated for each pair of annotators. During annotation we faced problems of deciding which label should be given for some entities, because some words can be interpreted in different ways which is the main reason why this task can be hard to accomplish. Besides, data annotation can be a very tedious task to do and people often lose concentration while doing it (Finin et al., 2010). Our results of data annotation can be seen in Table 1. Each pair of annotators compared their annotation and for each difference they agreed which label is better. After that we obtained our Gold Standard which was used for training model.

2.3. CRF

Conditional Random Fields (CRF) is a model that accounts for some of the shortcomings of the Hidden Markov Model (HMM):

- HMM models direct dependence between each state and only its corresponding observation - NER may depend not just on single word, but also on surrounding words from both sides,
- Large number of parameters,
- Inability to adjust well to the concrete problem,
- HMM models P(X,Y) while P(Y|X) is enough for prediction,
- CRFs are more expressive.

CRF directly models the distribution of the entire output vector over the entire input feature vector. The most probable output vector as the sequence of labels for the given input vector (Nallapati, 2013):

$$y* = \arg\min_{x} P(y|x)$$

In this project, we used CRFSuite (Okazaki, 2007), implementation of CRF for labeling sequential data. CRFSuite tends to be fast in training and tagging. It uses a data format similar to those used in other machine learning tools. Each line consists of a label and features of an item (empty line denotes an end of item sequence).

After annotating tweets, preprocessing tool NLPToolkit was used, provided to us by TakeLab members, our TA's. On the given output Python script was used that generated features needed for training the CRFSuite model. The following features were generated in the output file:

- word - the original word from tweet itself, not changed in any way,
- lemma - the word in standard form, for instance: vladu - vlada, osnuje - osnovati,
- POS - part of speech - N (noun), V(verb), etc.,
- lowercased token - the lowercased version of the token,
- token type - token type (AllUper, AllDigit...),
- prefixes, suffixes - prefixes and suffixes of tokens,
- contains an uppercase letter, contains a lowercase letter, contains an alphabet letter, contains a digit, contains a symbol - boolean value about containing certain characters.

All these features are not generated only for the word itself, but also for surrounding words, two from each side. At the beginning of each line there is also a label from annotator. The output file line looks something like this (very short version of it):

```
 O w[-2]=za w[-1]=Vladu w[0]=da
w[1]=osnuje w[2]=Agenciju wl[-2]=za
wl[-1]=vladu (...) lemma[-1]=vlada
pos[-1]=N (...) type[-2]=AllLetter
type[-1]=InitUpper (...) p1[-2]=z
(...) s4[-1]=ladu (...) au[2]=no
al[-2]=yes (...) cs[-1]=yes
cs[0]=yes (...) w[-2]|w[-1]=za|Vladu
w[-1]|w[0]=Vladu|da (...)
lemma[-2]|lemma[-1]=za|vlada (...)
type[0]|type[1]=AllLetter|AllLetter
(...) w[-4..-1]=Apel
w[-4..-1]=za (...).
```

This file can now be input for CRFSuite training. In our case, training is started with:

$$./crfsuite\ learn\ -m\ ./7030.model\ ./treningNer.txt$$

Training took approximately 130 seconds and we got the model '7030.model'.

CRFSuite can now tag our test set using:

$$./crfsuite\,tag\ -t\ -m\ ./7030.model\ ./testNer.txt$$

We used 70% of our annotated data for training and 30% for testing.

3. Evaluation

After we have finished training the model, the next step is the evaluation of its performance. For that purpose we used Precision (P), Recall (R), and F1 score (Goutte and Gaussier, 2005). These measures are calculated on class level and on instance level. Calculation of each measure begins with a confusion matrix where t_p represents the number of retrieved instances that were expected to be retrieved, f_p represents the number of retrieved instances that should not have been retrieved, f_n represents the number of non retrieved instances that were expected to be retrieved and t_n represents the number of non retrieved instances that should not have been retrieved. The formulas used for calculation are given bellow.

$$\begin{bmatrix} t_p & f_p \\ f_n & t_n \end{bmatrix}$$

$$P = \frac{t_p}{t_p + f_p} \qquad R = \frac{t_p}{t_p + f_n} \qquad F1 = \frac{2 \times P \times R}{P + R}$$

Evaluation at class level means that we look each class as independent and calculate how good the model was at assigning each class individually. Since we have seven classes, we will end up with 7×7 confusion matrix. For each class we create standard 2×2 confusion matrix as seen above. After that we use macro and micro measures to calculate the scores. Macro measures are average of individual scores for each class, while micro measures are obtained by summing all confusion matrices and then calculating scores. The results calculated for individual classes are given in Table 2.

Table 2: Class level evaluation results

Score	Macro	Micro
P	0.81707	0.95572
R	0.59811	0.95572
F1	0.67540	0.95572

Significantly larger micro measures mean that the model is more efficient at assigning common classes and less efficient at assigning rare classes. The fact that our training data, which we extracted heuristically from a collection of tweets, among named entities had around 60% persons, 20% organizations, and 20% locations explains this situation.

Evaluation at instance level is done by measuring how precise the model annotates named entity instances by comparing his output with gold standard at an instance level. This can be done by using strict or lenient approach. Strict approach says that an instance is correctly tagged if it is tagged with the same type and within the same extent of words. Lenient approach allows an instance to be marked correct even when there is only partial overlap between the extents of words. In Table 3 are given the results.

Table 3: Instance level evaluation results

Score	Strict	Lenient
P	0.74382	0.75813
R	0.52526	0.56706
F1	0.61572	0.64882

It is expected for lenient approach to give better results. In the Table 3, it can be seen that our model has a relatively small difference between strict and lenient approach, which implies that our model in most cases tags correct extent of words. The F1 score is consistent with class level F1 score for macro evaluation, which means that a bigger amount of organization and location entities would also improve instance tagging accuracy.

4. Conclusion

Extracting named entities from tweets is difficult task because they are short sequences of text, often written in informal language. For our task we have manually labeled set of tweets and trained CRF model on training set. Obtained F1 score of roughly 65% is reflecting stated problems. In spite of the results, we are content because we tackled task like this for the first time.

We realised crucial obstacles while conducting this task, so in future we will know what to expect. More time could be invested in obtaining good dataset and for our next project we could try a different model for NER.

References

S. Baluja, V. O. Mittal, and R. Sukthankar. 2000. Applying machine learning for high performance named-entity extraction.

Tim Finin, Will Murnane, Anand Karandikar, Nicholas Keller, Justin Martineau, and Mark Dredze. 2010. Annotating named entities in twitter data with crowdsourcing. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 80–88. Association for Computational Linguistics.

Cyril Goutte and Eric Gaussier. 2005. A probabilistic interpretation of precision, recall and f-score, with implication for evaluation. In *Advances in information retrieval*, pages 345–359. Springer.

J. Lafferty, A. McCallum, and F.C.N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Fien De Meulder. 2013. Conll-2003: List of tags with associated categories of names. http://www.cnts.ua.ac.be/conll2003/ner/annotation.txt.

Diego Molla. 2008. What is named entity recognition? http://afner.sourceforge.net/what.html/. Online; accessed 8th june 2016.

Ramesh Nallapati. 2013. Conditional random fields: Probabilistic graphical models. http://www.cs.stanford.edu/ nmramesh/crf.ppt. accessed 8th june 2016.

Naoaki Okazaki, 2007. *CRFsuite: a fast implementation of conditional random fileds*.

Sentence-Level Semantic Similarity

Goran Golub, Tena Perak, Ivan Relić

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
{goran.golub, tena.perak, ivan.relic}@fer.hr

Abstract

Semantic Textual Similarity (STS) measures the degree of semantic equivalence between texts. This paper describes a regression model for estimating semantic similarity between two sentences developed as a part of Text Analysis and Retrieval course at FER, Zagreb. Our work focuses on exploring the ability to capture main semantic components of two sentences with simple knowledge-based and corpus-based similarity measures. The model is trained and evaluated on the train and test datasets of SemEval 2012 Task 6 using Pearson and Spearman correlation metrics. We compare model results with the two best submitted systems of 2012.

1. Introduction

Semantic similarity of texts has applications in many NLP tasks such as information retrieval (IR), text classification, word-sense disambiguation, and automatic evaluation of machine translation. STS is closely related to paraphrase detection and textual entailment task but instead of producing a binary yes/no decision, it generates graded output which determines how semantically similar two text are. Task described in (Agirre et al., 2012) provides 3 datasets of sentence pairs: MSRpar (paraphrases), MSRvid (short descriptions of videos), and SMTeuroparl. Each sentence pair is associated with a real number in range 0-5 measuring its semantic similarity - 0 meaning no similarity and 5 meaning semantic equivalence.

Our model consists of multiple features combining knowledge-based and corpus-based similarity measures, as well as simple lexical overlaps of sentences. We employ Support Vector Regression (SVR) implemented in LIBSVM library (Chang and Lin, 2011) for obtaining a regression model in a supervised fashion. Outputs of our model are compared to gold standard using Pearson and Spearman correlation coefficients.

As we did not want to simply replicate earlier work on feature extraction (Šarić et al., 2012; Bär et al., 2012), we set our goal to create a simplified model without overly complex features which could achieve comparable results. We found that previous work on this task contains preprocessing steps and features too specific for given datasets, so we aim to build a more general and thus more robust model. Related to this line of thought, we tried to incorporate neural network approach to modeling sentences as research in this field shows promising results (He et al., 2015).

This paper is structured as follows: in section 2 we give a detailed description of used features in SVR model, section 3 contains information about preprocessing steps and training procedure, and in section 4 we present results of our model on previously mentioned datasets accompanied by comparison with winning systems.

2. Similarity Measures and Features

Capturing semantic similarity between short texts asks for similarity measures more flexible than simple word overlap. Since the number of words in each text is quite limited, intuition tells us the model should have a notion of semantic similarity between word pairs instead of just comparing their lexical forms. Mihalcea et al. (2006) present evidence of this assumption using WordNet-based similarity measures such as Wu and Palmer, Lin, and Resnik similarity. Most of the features we use are taken from Šarić et al. (2012) while others are developed as a simplification of their more elaborate counterparts.

2.1. Knowledge-based Features

All of our knowledge-based word similarity measures are based on WordNet. We use two different similarity measures of words: PathLen (Leacock and Chodorow, 1998) and Lin (Lin, 1998) similarity which are defined on synsets (a set of synonyms sharing a common meaning). Both of them rely on the lowest common subsumer (LCS) of two synsets c_1 and c_2, which represents the lowest node in the WordNet hierarchy that is a hypernym of both c_1 and c_2. PathLen similarity is defined as:

$$pathlen(c_1, c_2) = \frac{1}{1 + path(c_1, LCS) + path(c_2, LCS)}$$

while Lin similarity takes into account information content of words:

$$lin(c_1, c_2) = \frac{2ic(LCS(c_1, c_2))}{ic(c_1) + ic(c_2)}$$

Information content of words was retrieved from the Brown corpus. We used NLTK library (Loper and Bird, 2002) for calculation of this similarities.

WordNet-Augmented Word Overlap

This similarity measure is defined as in (Šarić et al., 2012): similarity between two sentences is the harmonic mean of similarity coverage of sentence S_1 over S_2 and vice versa:

$$f_1 = \frac{2P_{WN}(S_1, S_2)P_{WN}(S_2, S_1)}{P_{WN}(S_1, S_2) + P_{WN}(S_2, S_1)}$$

We use both PathLen and Lin similarity for measuring resemblance of two synsets. Instead of taking the maximum

Text Analysis and Retrieval 2016: Course Project Reports (TAR 2016), pages 18–21, University of Zagreb, Faculy of Electrical Engineering and Computing, Zagreb, June 2016.

similarity value of all synsets attached to a single word, we use only the most common meaning of the word given its part-of-speech tag.

WordNet Root Similarity

The central part of every sentence is its predicate upon which other words and phrases depend on. If sentences differ greatly in their predicates, one can assume high probability of them not being similar. This is why we obtain parse tree roots of both sentences using Stanford Neural Dependency Parser (Chen and Manning, 2014). Lemmatized roots are then compared using previously described similarity measures.

2.2. Corpus-based Features

Information about a word meaning can be extracted from the contexts that word appears in. Popular approaches include latent semantic models and neural word embeddings. We experimented with both Latent Dirichlet Allocation (LDA) and word2vec neural model (Mikolov et al., 2013). The latter was utilized in construction of corpus-based features as it achieved better results in capturing semantic relations between words.[1] We used implementation of word2vec model from *gensim* python module (Řehůřek and Sojka, 2010) trained on Google News Corpus.

Sentence and Root Embedding

To construct sentence embedding from vector representations of its constituent words we use *tf-idf* weighted sum, giving more importance to words bearing more information:

$$sent2vec(S) = \sum_{w \in S} tfidf(w) \cdot word2vec(w)$$

where the *idf* weighting factor of each word is calculated on Wikipedia corpus. Similarity of representations is calculated as cosine similarity.

Along with WordNet root comparison, we use word2vec model to estimate similarity of sentence root lemmas. These features proved to be the most useful ones, alone having high correlation with output labels.

2.3. N-gram Overlap Features

Word Overlap

If two sentences share substantial number of words and phrases, it is reasonable to believe they are somewhat semantically similar. Word n-gram is a list of *n* consecutive words and each sentence is represented as a set of n-grams. Similarity score between two sets of n-grams is defined as:

$$score(set_1, set_2) = \frac{2|set_1 \cap set_2|}{|set_1| + |set_2|}$$

Our model uses three such features: unigram, bigram, and trigram similarity scores of input sentences.

[1] Systems were evaluated on WordSimilarity-353 test collection.

Part-of-Speech Tag Overlap

Part-of-Speech tag provides information about how is particular word used in a sentence. This means that representing a sentence as a list of POS tags of its constituents can carry some syntactic information. We experimented with three n-gram POS tag overlaps: unigram, bigram, and trigram, although only unigram feature improved overall performance of our model.

Character Overlap

Since word overlap features capture similarities only between words in the same grammatical form (we are not using lemmas) and since misspelled words are a common phenomenon, we decided to add features that could find similarities between words based on their character overlap. Equation for computing this score is the same as before, only we use n-gram measures for $n \in 3, 4, 5, 6$.

In addition to described features, we defined our own NuFER similarity feature which measures resemblance of sentences based on the difference in their lengths. Feature value is defined as follows:

$$NuFER(S_1, S_2) = 1 - \frac{|len(S_1) - len(S_2)|}{len(S_1) + len(S_2)}$$

3. Preprocessing and Training

3.1. Preprocessing

Before feature extraction, every sentence pair is preprocessed as follows:

- tokenized using TreebankWordTokenizer from nltk library;
- words containing hyphens are split into multiple parts (e.g., '10-year-old' becomes '10 year old');
- two consecutive words are merged if they appear in conjunction in the other sentence;
- punctuation symbols are removed;
- words are lowercased;
- consecutive tokens *ca + n't* are replaced with *can + not* (similar for *wo + n't*);
- part-of-speech tags are obtained using nltk library;
- dependency parse trees are generated with Stanford Neural Dependency parser;
- stopwords are removed;
- words are lemmatized with WordNetLemmatizer.

3.2. Feature Selection

We experimented with features described in Section 2. to study their influence on the performance of the model. To determine the best of the 17 described features, we performed a full forward-search on the space of similarity measures. In forward-search, we perform 5-fold cross-validation on the training set for each measure, and pick the best one; in the next round, that best metric is retained, and the remaining metrics are considered for addition. Search is stopped when performance no longer increases.

Table 1: Average CV score on training datasets

	MSRpar	MSRvid	SMTeuropar
Pearson	0.6330	0.8687	0.6982
Spearman	0.5518	0.8678	0.5304

We found that the weighted word2vec sentence embedding could independently achieve very high correlations with the gold standard. Other features that proved themselves useful are root word2vec embedding, PathSim root similarity, NuFer, POS unigram, and WordNet-augmented word overlap for both PathSim and Lin similarity measures.

3.3. Model Training

We train SVR using 5-fold cross-validation and grid search for obtaining optimal hyperparameters C, γ and ϵ. Since training datasets are quite heterogeneous, we train the model on each set separately. Average cross-validation Pearson and Spearman correlation coefficients for the training datasets are displayed in Table 1.

This results indicate that our model might be overly simple and unable to capture resemblance every time. We tried to compensate this with a convolutional neural network model described in He et al. (2015). In the paper, authors train and test the network only on MSRvid datasets. We trained the model on all training examples through 30 epochs which took 24 hours to complete. Resulting network turned out not to be of great use as its output was weakly correlated with the gold standard. Time limitations prevented us from conducting more experiments with the neural model. We also tried out a different regression model - Random Forests, but it did not improve overall performance on the training sets.

4. Results

In Table 2 we present Pearson correlation scores on five test sets in the first column (SVR) and compare them with the two best submitted systems of 2012 (TakeLab, UKP).[2] The last column displays Spearman correlation coefficients of our model on all datasets.[3] As mentioned above, for MSRpar, MSRvid, and SMTeuroparl test sets we train the model on corresponding train sets separately. The model evaluated on the two surprise sets (OnWN and SMTnews) was trained on all training datasets combined.

Our model outperformed the two top submitted systems on three out of five test sets, two of which are surprise datasets. This confirms our hypothesis about better generalization due to model simplicity.

However, our model seems to lack potential to capture semantic similarity of longer sentences (MSRpar set). This is probably due to the fact that we use very little information about the syntax of sentences and the ordering of words. Poor performance on this dataset causes weak Pearson correlation coefficient with gold standard for all five datasets.

[2]Best systems according to normalized rank and mean rank.

[3]Systems on SemEval task were ranked using Pearson correlation.

Table 2: Correlation coefficients on test sets

	Pearson			Spearman
	SVR	TL\simple	UKP	SVR
MSRpar	0.5965	**0.7343**	0.6830	0.5730
MSRvid	**0.8847**	0.8803	0.8739	0.8804
SMTeurpar	0.4962	0.4771	**0.5280**	0.5410
OnWN	**0.6911**	0.6797	0.6641	0.6614
SMTnews	**0.5122**	0.3989	0.4937	0.4528
All	0.6972	0.8133	**0.8239**	0.5891
Mean	0.6632	0.6753	**0.6773**	0.6484

Still, obtained weighted mean correlation score is comparable to the other two models.

5. Conclusion

Measuring semantic similarity of sentences is a difficult task. In this paper we show that obtaining results close to the *state-of-the-art* is possible using very simple features and minimal knowledge of syntactic structure of sentences. To achieve this, we train a SVR model on features based on similarity measures between words and n-gram overlaps between two texts.

Our model uses word2vec embeddings to capture context-based similarity of words and WordNet knowledge base to approximate distance between them. We show that model with features constructed for a general dataset does in fact generalize better than the models whose features are dataset specific. However, our model seems to have a problem with longer sentences because of scarce syntactic information it utilizes.

For future work we would like to experiment more with neural models for sentence representations as recent work shows great potential in capturing semantics and syntax of text.

References

Eneko Agirre, Mona Diab, Daniel Cer, and Aitor Gonzalez-Agirre. 2012. Semeval-2012 task 6: A pilot on semantic textual similarity. In *Proceedings of the 6th International Workshop on Semantic Evaluation (SemEval 2012), in conjunction with the First Joint Conference on Lexical and Computational Semantics (*SEM 2012)*. ACL.

Daniel Bär, Chris Biemann, Iryna Gurevych, and Torsten Zesch. 2012. Ukp: Computing semantic textual similarity by combining multiple content similarity measures. In *Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 435–440. ACL.

Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27.

Danqi Chen and Christopher D Manning. 2014. A fast and accurate dependency parser using neural networks.

In *Empirical Methods in Natural Language Processing (EMNLP)*.

Hua He, Kevin Gimpel, and Jimmy Lin. 2015. Multi-perspective sentence similarity modeling with convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1576–1586. ACL.

Claudia Leacock and Martin Chodorow. 1998. Combining Local Context and WordNet Similarity for Word Sense Identification. In Christiane Fellbaum, editor, *WordNet: An electronic lexical database.*, chapter 13, pages 265–283. MIT Press.

Dekang Lin. 1998. An information-theoretic definition of similarity. In *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML '98, pages 296–304. Morgan Kaufmann Publishers Inc.

Edward Loper and Steven Bird. 2002. Nltk: The natural language toolkit. In *In Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics. Philadelphia: Association for Computational Linguistics*.

Rada Mihalcea, Courtney Corley, and Carlo Strapparava. 2006. Corpus-based and knowledge-based measures of text semantic similarity. In *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 1*, pages 775–780. AAAI Press.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May. ELRA.

Frane Šarić, Goran Glavaš, Mladen Karan, Jan Šnajder, and Bojana Dalbelo Bašić. 2012. Takelab: Systems for measuring semantic text similarity. In *Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 441–448, Montréal, Canada, 7-8 June. ACL.

Experiments on English Lexical Text Simplification

Paula Gombar, Ivan Katanić

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
{paula.gombar, ivan.katanic}@fer.hr

Abstract

In this paper, we consider the task of lexical text simplification by building a system to replace complex and hard-to-comprehend words with less complex semantically-matching words. We use several different approaches, combining both supervised and unsupervised methods relying on word vector representations, regular and simplified corpora, context-dependent and context-independent features. One supervised approach makes use of ranking SVM and different kernel functions, while the other obtains the best linear combination of features. Unsupervised approaches make use of linguistic features and external lexico-semantic resources such as WordNet. The system was tuned, trained and evaluated on the SemEval-2012 Task 1 dataset and outperforms two of three given baselines.

1. Introduction

Lexical simplification is the task of finding less complex semantically-appropriate words or phrases, and replacing those that are difficult to comprehend, especially for certain groups of people such as non-native speakers, people with low literacy or intellectual disabilities, and language-impaired people.

Common steps in a pipeline for a lexical simplification system include:

1. complexity analysis: finding out which words or phrases are considered complex,
2. substitute lookup: retrieving adequate replacements, simpler than the original word or phrase, from a thesaurus or lexico-semantic resources,
3. context-based ranking: ranking of substitutes to produce the final replacement.

To give an example, let us consider the following sentence: "The incisions will feel constricted for the first 24–48 hours." The system would first identify complex words, e.g., "constricted", then search for substitutes that might adequately replace it. Suppose the retrieved candidates were "uncomfortable", "tight", "stretched", "compressed" and "constricted". Finally, the system would score each of the candidates on simplicity and context-adequacy, rank them and determine the simplest one, e.g., "tight", and use it to replace the complex word, yielding the sentence "The incisions will feel tight for the first 24–48 hours."

This paper mainly focuses on the third step, context-based ranking, as complexity analysis and substitute lookup have already been given in the SemEval-2012 Task 1 resources this paper is based on. The definition of SemEval-2012 Task 1 is, given a short context, a target word in English, and several substitutes for the target word that are deemed adequate for that context, rank these substitutes according to how "simple" they are, allowing ties.

We decided to use several different approaches to solving this task, mixing both supervised and unsupervised methods, using external resources such as word vector representations, Simple Wikipedia, WordNet, linguistic features, as well as context-dependent and context-independent features.

2. Related work

There have been multiple different approaches to lexical text simplification. Before simplified corpora, the approach was based on substituting longer words with shorter and more frequent alternatives, such as in (Carroll et al., 1998; De Belder and Moens, 2010), and referred to lexico-semantic resources like WordNet (Fellbaum, 1998).

After the emergence of simplified corpora, most notably Simple Wikipedia[1], the focus was shifted to using this type of resource, either using context-aware unsupervised methods (Biran et al., 2011), or supervised methods to learn substitutions from the sentence-aligned corpora (Horn et al., 2014).

Recent advantages in word vector representations (Pennington et al., 2014) have paved the path for unsupervised approaches that do not use simplified corpora, but regular corpora and vector representations (Glavaš and Štajner, 2015).

3. Dataset description

The dataset used in SemEval-2012 Task 1 originates from SemEval-2007 and the Lexical Substitution task. It is extracted from the English Internet Corpus of English (Sharoff, 2006), and is a selection of sentences, or contexts. In total, there are 2010 contexts which are divided into trial and test sets, consisting of 300 and 1710 contexts, respectively. The datasets cover a total of 201 (mostly polysemous) target words, including nouns, verbs, adjectives and adverbs, and each of the target words is shown in 10 different contexts.

Given the list of contexts and each respective list of substitutes, the annotators ranked substitutes for each individual context in ascending order of complexity. The trial dataset was annotated by four people, while the test dataset was annotated by five people. In both cases, each annotator tagged the complete dataset, and the inter-annotator agreement was computed using the kappa index with pairwise

[1] https://simple.wikipedia.org.

Text Analysis and Retrieval 2016: Course Project Reports (TAR 2016), pages 22–25, University of Zagreb, Faculy of Electrical Engineering and Computing, Zagreb, June 2016.

rank comparisons, the same metric being used for system evaluation. On the trial dataset, a kappa index of 0.386 was reported, while for the test dataset, a kappa index of 0.398 was reported. Regarding the low kappa scores, the highly subjective nature of the annotation task must be taken into account. It is also worth noticing that this agreement metric is highly sensitive to small differences in annotation, thus leading to rather pessimistic scores.

Finally, here is an example of a sentence, or context, from the trial dataset, as well as the gold-standard ranking of substitutions.

Sentence: "The Governor *took* the big sheets of imitation parchment, glanced over them, signed his name to each laid down the pen, and handed the papers across the table to Dreier."

Gold rankings: {get} {take} {pick up} {collect, grasp} {gather}.

4. Features

We rank the simplification candidates according to several features, both context-dependent and context-independent.

Inverse word length. It is a valid assumption that a word is more complex if it is longer than another candidate. We use the inverse of the candidate's length, because in our system a higher score means the word or phrase is simpler.

Number of synsets in WordNet. WordNet[2] is a lexico-semantic resource that groups together words based on their meanings, or synsets. The number of candidate's synsets correlates with interchangeability in several contexts, so a candidate with a higher number of synsets is considered to be simpler.

Frequency in Simple Wikipedia. We obtained a list of word frequencies from an up-to-date Simple Wikipedia dump.[3] Simple Wikipedia is primarily written using basic English, making it a useful resource for this task.

Frequency in Wikipedia. We obtained a list of word frequencies from an English Wikipedia dump[4] from 2014. The assumption is that an often-used word must be simpler than words that are not used as often.

Corpus complexity. As seen in (Biran et al., 2011), corpus complexity is defined as the ratio of a candidate's frequency in Wikipedia and Simple Wikipedia:

$$C_w = \frac{f_{w,English}}{f_{w,Simple}}$$

where $f_{w,c}$ is the frequency of candidate w in corpus c. We use the inverse score, because we rank higher the simpler candidates.

Context similarity. As seen in (Glavaš and Štajner, 2015), the idea is that the simplification candidates that are synonyms of the correct sense of the original word should be more semantically similar to the context of the original word. This feature is obtained by computing the semantic similarity of the simplification candidate and each content word from the original context. The semantic similarity of two words is computed as the cosine of the angle between their corresponding GloVe[5] vectors. In all experiments, we used 200-dimensional GloVe vectors pretrained on the English Wikipedia corpus. Formally defined:

$$csim(w, c) = \sum_{w' \in C(w)} cos(\mathbf{v_w}, \mathbf{v_{w'}})$$

where $C(w)$ is the set of context words of the original word w, and $\mathbf{v_w}$ is the GloVe vector of the word w.

Semantic similarity. Similar to context similarity, this feature is obtained by computing the semantic similarity of the original word marked for replacement and the simplification candidate:

$$ssim(w, c) = cos(\mathbf{v_w}, \mathbf{v_{w'}})$$

where $\mathbf{v_w}$ is the GloVe vector of the original word w, and $\mathbf{v'_w}$ is the GloVe vector of the replacement candidate.

5. Methods used

We used a total of four different methods, three supervised and one unsupervised. When using a supervised approach, we optimized the algorithm's parameters using grid search and cross-validation. Cross-validation is performed by splitting the trial dataset into a training set and validation set, the ratio being 70:30. The systems are then evaluated on the test dataset.

We initially used *SVM^{rank}* described in (Joachims, 2006), but found the implementation too rigid to perform hyperparameter optimization. We ended up implementing our own version of a ranking SVM algorithm, with the help of SVM implementation in scikit-learn (Pedregosa et al., 2011). The ranking SVM algorithm comes down to mapping the similarities between pairs of candidates onto a feature space, calculating the distances between any two of them, then, using pairwise comparisons, converting the problem into a classification one, and, finally, solving the optimization problem with the regular SVM solver.

Ranking SVM with RBF kernel. Hyperparameter ranges that we tested are shown in Table 1. The scaler parameter defines the method used to scale the feature array. Standard scaler translates the data so the mean value is zero, then scales is so that the standard deviation is one, whereas the MinMax scaler scales and translates each feature between zero and one. PolynomialFeatures is used to add complexity to the model by considering nonlinear features. If the degree is 2, and we have features x_1 and x_2, the resulting feature array is $(1, x_1, x_2, x_1^2, x_1x_2, x_2^2)$.

Ranking SVM with linear kernel. Similar to the previous method, the only difference is the kernel used in the SVM algorithm. Hyperparameter ranges that we tested are shown in Table 2. What is interesting is that the linear kernel outputs the coefficients belonging to each of the features. The following optimal linear coefficients were obtained: $weights = (-0.355, -0.109, -0.296, 0.273, -0.041, -0.835, 0.004)$, whereas the order of features is as described beforehand.

[2] https://wordnet.princeton.edu

[3] https://dumps.wikimedia.org/simplewiki/

[4] https://dumps.wikimedia.org/enwiki/

[5] http://http://nlp.stanford.edu/data/glove.6B.zip.

Table 1: Hyperparameter ranges for Ranking SVM with RBF kernel.

Hyperparameter	Optimal value	Possible values
Scaler	Standard	StandardScaler, MinMaxScaler, None
PolynomialFeatures degree	1	1, 2
C	2^5	$\{2^{-15}, 2^{-14}, \ldots, 2^8\}$
γ	0.00098	$\{2^{-15}, 2^{-14}, \ldots, 2^8\}$

Table 2: Hyperparameter ranges for Ranking SVM with linear kernel.

Hyperparameter	Optimal value	Possible values
Scaler	Standard	StandardScaler, MinMaxScaler, None
PolynomialFeatures degree	1	1, 2
C	2^4	$\{2^{-15}, 2^{-14}, \ldots, 2^8\}$

Linear combination of features. We used grid search to find the optimal combination of linear coefficients, similar to the output of ranking SVM with linear kernel. The optimal hyperparameters can be seen in Table 3. Additionally, the optimal scaler used is MinMax scaler.

Unsupervised approach. The unsupervised approach was more popular in SemEval-2012, eight out of twelve teams using it instead of a supervised one, as well as the winning team. The reason might be due to the limited number of training examples given. In this approach, we first scale the data using the MinMax scaler and then declare all coefficients as 1, so the resulting function is equivalent to obtaining the average of all features. We do not use cross validation in this approach, so after tuning the model on the trial dataset, we evaluate it on the test dataset.

6. Evaluation

Evaluation for all methods was done on the test dataset, consisting of 1710 contexts. In the SemEval-2012 task description, three baselines were provided.

L-Sub Gold: This baseline uses the gold-standard annotations from the Lexical Substitution corpus of SemEval-2007 as is. In other words, the ranking is based on the goodness of fit of substitutes for a context, as judged by human annotators. This method also serves to show that the Lexical Substitution and Lexical Simplification tasks are indeed different.

Random: This baseline provides a randomized order of the substitutes for every context. The process of randomization is such that is allows the occurrence of ties.

Simple Freq.: This simple frequency baseline uses the frequency of the substitutes as extracted from the Google Web 1T Corpus (Brants and Franz, 2006) to rank candidate substitutes within each context.

The evaluation metric used is the same measure used for inter-annotator agreement, the metric based on the kappa index (Cohen, 1960). It is used for both contrasting two human annotators, and contrasting a system output to the average of human annotations that together forms the gold-standard, and is defined as the following:

$$\kappa = \frac{P(A) - P(E)}{1 - P(E)}$$

where $P(A)$ denotes the proportion of times the system output corresponds to the gold-standard, and $P(E)$ denotes the probability of agreement by chance between the two. The overall kappa score is calculated for every pair of ranked items for a given context, and then averaged.

Table 4: Baseline kappa scores on trial and test datasets.

	Trial	Test
L-Sub Gold	0.050	0.106
Random	0.016	0.012
Simple Freq.	**0.397**	**0.471**

Table 5: Implemented methods kappa scores on the test dataset.

Method name	Test score
Ranking SVM with RBF kernel	**0.461**
Ranking SVM with linear kernel	0.443
Linear combination of features	0.459
Unsupervised approach	0.313

Results in table 4 show that the "Simple Freq." baseline performs very strongly, despite being simple. In fact, it surpasses the average inter-annotator agreement on both trial and test datasets.

Results in table 5 show that the ranking SVM with RBF kernel performs best, although all supervised methods perform similarly on the test set, the unsupervised method being somewhat weaker. All four methods outperform two of the three given baselines. It is possible that the supervised

Table 3: Optimal hyperparameters for linear combination of features.

Feature	Weight
Inverse word length	1
WordNet synsets	0
Simple Wikipedia frequency	10
English Wikipedia frequency	0
Corpus complexity	0
Context similarity	9
Semantic similarity	0

approaches would outperform the "Simple Freq." baseline if there were more training data available.

7. Conclusion

We presented four different methods in solving the lexical simplification task, using both context-dependent and context-independent features, as well as external resources such as state-of-the-art word vector representations and simplified corpora.

The top-performing method in our approach was the ranking SVM with RBF kernel, although it is followed closely by linear combination of features and the ranking SVM with linear kernel. The unsupervised approach does not perform as well, but it outperforms two out of three given baselines.

We believe that the performance of supervised approaches is likely to improve with larger training sets, and that the scarcity of training data is the main reason why the context-independent "Simple Freq." baseline outperforms these methods. Additionally, the evaluation metric is very susceptible to penalize slight changes, making it rather pessimistic about the system's performance.

In the linear combination of features, the largest weight was given to Simple Wikipedia frequency, followed by context similarity. This shows that there is a very strong relation between distributional frequency of words and their perceived simplicity, as well as the importance context-dependent features and word vector representations. On the other hand, it appears that relying on the entire English Wikipedia corpus and lexico-semantic resources like WordNet do not have any impact on the system's performance.

References

Or Biran, Samuel Brody, and Noémie Elhadad. 2011. Putting it simply: a context-aware approach to lexical simplification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 496–501. Association for Computational Linguistics.

Thorsten Brants and Alex Franz. 2006. {Web 1T 5-gram Version 1}.

John Carroll, Guido Minnen, Yvonne Canning, Siobhan Devlin, and John Tait. 1998. Practical simplification of english newspaper text to assist aphasic readers. In *Proceedings of the AAAI-98 Workshop on Integrating Artificial Intelligence and Assistive Technology*, pages 7–10. Citeseer.

Jacob Cohen. 1960. A coefficient of agreement for nominal scale. *Educ Psychol Meas*, 20:37–46.

Jan De Belder and Marie-Francine Moens. 2010. Text simplification for children. In *Prroceedings of the SIGIR workshop on accessible search systems*, pages 19–26. ACM.

Christiane Fellbaum. 1998. *WordNet*. Wiley Online Library.

Goran Glavaš and Sanja Štajner. 2015. Simplifying lexical simplification: Do we need simplified corpora? *Volume 2: Short Papers*, page 63.

Colby Horn, Cathryn Manduca, and David Kauchak. 2014. Learning a lexical simplifier using wikipedia. In *ACL (2)*, pages 458–463.

Thorsten Joachims. 2006. Training linear svms in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 217–226. ACM.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *The Journal of Machine Learning Research*, 12:2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Serge Sharoff. 2006. Creating general-purpose corpora using automated search engine queries. *WaCky*, pages 63–98.

A System for Lexical Text Simplification

Jakov Ivančan, Ivan Kovačević, Antun Maldini

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
{jakov.ivancan,ivan.kovacevic,antun-maldini}@fer.hr

Abstract

We describe a system for lexical simplification, as defined in the SemEval 2012 task 1, combining a simple statistical-based approach with a context-aware vector-based one. We discuss the relative contribution of various parts, and compare the results with those in the SemEval task.

1. Introduction

Text simplification consists of converting a text into a less complex or more readable form (Biran et al., 2011). The reason for doing so is to render the text more understandable and easier for children, language-impaired people, and non-native speakers to comprehend.

We took on the task of lexical text simplification: to replace complex words with semantically-matching simple words. This is similar to the lexical substitution task in (Mccarthy et al., 2007), the main difference being that the substitutes are chosen here according to how simple they are. Specifically, the task as given in (Specia et al., 2012), where it is defined as follows: given a short context, the word to be replaced, and a set of candidate substitutes which are deemed to be semantically similar enough, rank these candidates by how "simple" they are. Simple words are loosely defined as those which can be understood by a wide range of people, including those with low literacy levels or some cognitive disability, children, or non-native speakers.

The dataset consists of 2010 context sentences, split into a trial set (300 sentences) and a test set (the remaining 1710 sentences). The words to be replaced are marked and PoS-tagged, and up to 10 candidate substitutes are given for each one. Gold-standard rankings are also given; these were generated by averaging the human annotators' rankings. The system is evaluated against them.

We attempted to solve the problem by combining two different approaches: one context-independent, and one context dependent.

2. Related work

In (Sinha, 2012), several systems are described, for solving the same task as we. They all use a subset of the following resources:

- Simple English Wikipedia — the Simple Wikipedia consists of *simple* words; words that appear frequently in it are thus more likely to be simple
- Transcribed Spoken English Corpus — a corpus of spoken English language; conversational language is more likely to contain simple words
- WordNet — a lexical database, described in (Fellbaum, 1998), that combines a dictionary and a thesaurus; it is used under the hypothesis that simple words are used very frequently, and are therefore polysemous
- Web1T Google N-gram Corpus — a collection of n-gram frequencies, for n in [1,5], as observed on the Web; the hypothesis is that simple words will have higher frequencies on the Web as a whole

One of those systems is SIMPRANK, which calculates its simplicity score as:

$$simplicity(word) = \frac{1}{len(word)} + c_{word}(SimpleWiki) + c_{word}(Discourse) + c_{word}(WordNet) + c_{word}(Unigrams) \quad (1)$$

where c_{word} is the (non-normalized) frequency of a word in a corpus. The second one is SIMPRANKLIGHT, which leaves out the Web1T corpus (the last term). The last one is SALSA (Stand-alone Lexical Substitution Analyzer), which substitutes the candidate, creates all possible 3-grams from the resulting context, sums their frequency counts in the Web1T corpus, and uses that as a simplicity score for the candidate used. It is the only one of the three which tries to take advantage of the context.

In (Biran et al., 2011), a more general solution is presented, which takes as its input a text, generates the substitution rules, and applies them where it determines it would be OK to do so, while taking care to maintain grammaticality of the original sentence.

They generate potential substitution rules thus: from all non-stopwords in the corpus, generate all word-pairs; filter out those which are not synonyms using WordNet; filter out the rules where the first word is less complex than the second, with word complexity χ_w being defined as the product of *lexical complexity* L_w and *corpus complexity* C_w. They are in turn defined as

$$L_{word} = \frac{f_{word,English}}{f_{word,Simple}} \quad (2)$$

$$C_{word} = len(word) \quad (3)$$

where $f_{word,c}$ is the frequency of $word$ in the corpus c – here English Wikipedia or Simple English Wikipedia.

Since context vectors are used for triggering the rules, a definition would be fitting here: a context vector CV_w

Text Analysis and Retrieval 2016: Course Project Reports (TAR 2016), pages 26–28, University of Zagreb, Faculy of Electrical Engineering and Computing, Zagreb, June 2016.

of a word w with a k-token window radius is a frequency count of all the words that appear at a distance of at most k tokens from w in a sentence, summed over all sentences in a given corpus. The sentence vector for word *fox* and corpus *"The quick brown fox jumps over the lazy dog."* with a window radius of 2 would thus be $\{(quick, 1), (brown, 1), (jumps, 1), (over, 1)\}$.

For triggering a substitution rule $\{w \rightarrow x\}$, the procedure is:

- Build the context vector $SCV_{s,w}$ from the context of the target sentence s
- Calculate the cosine similarity of CV_w and $SCV_{s,w}$; if this is larger than a predefined value, the rule is not used
- Create a common context vector $CCV_{w,x} = min(CV_w, CV_x)$, where min is the elementwise minimum. Calculate context similarity as $ContextSim = cosine(CCV_{w,x}, SCV_{s,w})$.

The rule with the highest context similarity is then used for replacement.

This second approach is not directly applicable to our task, since it goes beyond the simple ranking requirement, and does not have explicit candidate ranking. However, we adapt the last part – vector similarity between global and local context vectors – as a component of our system.

3. The procedure

We decided to use a combination of SIMPRANKLIGHT and context vectors to produce the simplicity score function, defined as follows:

$$\begin{aligned} simplicity(word) = w_1 \cdot \frac{1}{len(word)} \\ + w_2 \cdot f_{word}(WordNet) \\ + w_3 \cdot f_{word}(SimpleWiki) \\ + w_4 \cdot ContextSimilarity \end{aligned} \quad (4)$$

where w_1 to w_4 are weights, $f_{word}(c)$ is the normalized frequency of $word$ in corpus c, and $ContextSimilarity$ is the context similarity of $word$ in the given context.

The assumption provided with the data is that the candidate substitutes already fit the target sentence *well-enough* semantically. Since the $ContextSimilarity$ used here is a measure of *semantic similarity*, we were not sure how much of a difference would it make, if any. Still, we thought there might still be enough variability for this component to have a positive effect on total perfomance.

The $ContextSimilarity$ measure is defined as the cosine between CV_w, the local context vector of the word to replace, extracted from the context sentence, and CV_x, the global context vector of the replacement candidate, extracted from the entire Simple Wikipedia corpus.

3.1. Data and preprocessing

The contexts, provided in XML form, were originally unparsable – a single regexp search-and-replace operation solved that. The Simple Wikipedia corpus is a collection of all articles (without edit history or talks) from Simple Wikipedia, extracted from the 2016-06-01 database dump using the Wikipedia Extractor Toolkit[1] and standard command-line tools. The resulting corpus was about 100MiB of text, with close to 17 million words. As corpora go, this is a rather small one. We then counted the word frequencies, and generated context vectors with a 4-token radius, for all the words in the corpus.

Some of the words given as substitution candidates, however, are not single words, but sequences of several words (*n-grams*). We counted their frequencies separately from the single-word frequencies, and generated their context vectors separately from the single-word context vectors. Since there were only about 700 distinct n-grams in the set of all candidates, this was much more efficient than if we were doing so for all appropriately-sized n-grams in the corpus.

All words were considered, i.e., stopwords were not thrown out. Some of the substitution candidates had the same stems; stemming was thus not used. We initially did not want to use lemmatization either for the same fear, though later on we concluded that the fear was unfounded. Due to time constraints, however, we did not redo everything with lemmatization.

4. Results

Due to time and resource constraints, a grid search (or a more advanced heuristic) of the weight-space for weights w_1 through w_4 in equation (4) was not done. In its stead, we did a hill-climbing search. We first normalized the first three subscores (inverse length, number of WordNet meanings, and Simple Wikipedia frequency) by dividing them with their largest encountered values. The weight-space used was $\{\, 10^i \mid i \in [0, 7] \,\}^4$. The resulting best weights are shown in table 1, by indices into this weight-space. A heavy emphasis on the Simple Wikipedia frequency component is apparent. The table 2 shows the performance of single components only (with other weights set to zero). Note the slight discrepancies in tables 1 and 2. We think this is due to floating point errors – Simple Wikipedia frequencies are normalized by a factor of roughly 10^6. Taking into consideration the very small differences between the trial set scores, we chose the set of weights with indices (1,1,6,0) for our final system.

Table 3 shows the performance of the original task participants, grouped by statistical significance. There were three baselines in the task. One was a random permutation ranking, with a score expectedly very close to zero. Another was a simple format change, with the ranking remaining the same as in the provided input file (for the substitute candidates), also with a low score. The third one was using frequencies from the Web1T corpus as a simplicity score. The baseline scores on trial and test datasets are shown in 4. SIMPRANK achieved the same score as the third baseline, while SIMPRANKLIGHT scored somewhat lower, though still close. SALSA, which focused on context, failed miserably – it scored significantly lower than even the random

[1] Taken from `http://medialab.di.unipi.it/wiki/Wikipedia\_Extractor`

Table 1: Optimization best results

Weights	Score
(0,0,7,0)	0.3963
(0,0,6,0)	0.3963
(1,0,6,0)	0.3963
(0,1,6,0)	0.3951
(0,0,5,1)	0.3945
(0,0,5,0)	0.3945
(0,0,6,1)	0.3938
(1,0,5,0)	0.3963
(0,0,4,0)	0.3926
(0,1,5,0)	0.3914

Table 2: Single component scores

Component	Trial set score	Test set score
Inverse length	0.176	0.237
WordNet score	0.192	0.271
SimpleWiki freq.	0.393	0.465
Context similarity	0.191	0.197

baseline. With the final set of weights, our system achieves a score of 0.467 on the test set.

We see that the context similarity component, while better than the basic baselines, does not significantly contribute to the overall best score. We suspect that while the small corpus size might have an effect, the blame lies in the original assumption that it might be of additional help.

Table 3: SemEval 2012 task results

Rank	Team ID	System ID	Score
1	WLV-SHEF	SimpLex	0.496
2	baseline	Sim Freq	0.471
2	UNT	SimpRank	0.471
2	annlor	simple	0.465
3	UNT	SimpRankL	0.449
4	EMNLPCPH	ORD1	0.405
5	EMNLPCPH	ORD2	0.393
6	SB	mmSystem	0.289
7	annlor	lmbing	0.199
8	baseline	No Change	0.106
9	baseline	Rand	0.013
10	UNT	SaLSA	-0.082

Table 4: SemEval 2012 baseline scores

Baseline	Trial dataset	Test dataset
Random	0.016	0.012
Unchanged	0.050	0.106
Simple frequency	0.397	0.471

5. Conclusion

Our lexical simplification system, when compared to those in the SemEval 2012 task, holds its ground – not a surprise, given that it was based on a system that did well in the task. As it turns out, the context-vector component did not really improve the performance; given that only one participant had succeeded in beating the simple frequency baseline, and not by much, one might based on this reasonably conclude that there exists a strong correlation between frequency and simplicity. In other words, either simplicity is a mostly statistical thing, or humans are very efficient at communication with respect to the cognitive load it imposes.

We achieved better results than SIMPRANKLIGHT, the system on which we built upon, by putting a relatively large emphasis on the Simple Wikipedia frequency component. The system is relatively fast (when ignoring context vectors), and uses only Simple Wikipedia (a small corpus, widely available, and free) and WordNet (freely available). WordNet can also be thrown out for almost identical results and even faster processing. To conclude: in terms of time, money, and computing resources, mere SimpleWiki frequency as a simplicity score is very effective, with very little spent.

References

Or Biran, Samuel Brody, and Noémie Elhadad. 2011. Putting it simply: a context-aware approach to lexical simplification.

Christiane Fellbaum, editor. 1998. *WordNet An Electronic Lexical Database*. The MIT Press, Cambridge, MA ; London, May.

Diana Mccarthy, Falmer East Sussex, and Roberto Navigli. 2007. Semeval-2007 task 10: English lexical substitution task. In *In Proceedings of the 4th workshop on Semantic Evaluations (SemEval-2007*, pages 48–53.

Ravi Sinha. 2012. Unt-simprank: Systems for lexical simplification ranking.

Lucia Specia, Sujay Kumar Jauhar, and Rada Mihalcea. 2012. Semeval-2012 task 1: English lexical simplification.

2016. Wikipedia extractor. http://medialab.di.unipi.it/wiki/Wikipedia_Extractor. Accessed: 2016-06-20.

Application of Latent Semantic Indexing and BM25 Methods in an Information Retrieval System

Kristijan Kecerin, Simon Knežević, Juraj Oršulić

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
{kristijan.kecerin, simon.knezevic, juraj.orsulic}@fer.hr

Abstract

In this paper we examine the application of Latent Semantic Indexing (LSI) in Information Retrieval (IR). The model has been built using distributed computing from a corpus which consists of all articles of the English Wikipedia, while the queries and indexed documents used for testing the model are from the European Media Monitor collection. The performance of this model is compared with the BM25 probabilistic retrieval model, along with a model which attempts to combine the two approaches.

1. Introduction

Information retrieval is the activity of obtaining information resources (documents) relevant to a user's information need from a collection of information resources (Manning, 2008). The user's information need is expressed in the form of a textual query, which can be considered a short document itself.

A well known family of IR algorithms are the vector space algorithms, which transform both the indexed documents and the query into a common vector-space representation and use a similarity measure between these representations (such as the cosine similarity) to compute a numerical similarity of the respective document with the query. The documents are then ranked according to their calculated similarity score and returned in this order to the user with the hope that most highly ranked documents are truly the most relevant ones. The features which span the vector space are typically the document-term frequencies that can be supplemented with a weighting scheme such as the Inverse Document Frequency (IDF), which gives more preference to terms with more information (which appear in fewer documents).

A common pitfall of a model like the one described above is that it will not recognize documents which are relevant to the user's information need, but use synonyms or troponyms while expressing semantically similar content. E.g., a user may not be served a document which uses the words "freight" or "shipment" when they query the system with the word "cargo".

A remedy for this problem may be found in the Distributional hypothesis, which states that words that occur in similar contexts tend to have similar meanings (Harris, 1954). In the models we work with, the context of a word is considered to be the whole document it appears in. Using linear algebra it can be shown that finding a singular value decomposition (SVD) of the term-document matrix and keeping n strongest singular values gives a new vector space basis in which semantically similar documents have similar representations, which solves the above problem.

2. Related work

The algebraic foundation for LSI was first described in (S. C. Deerwester, 1990). This paper describes the SVD process and interprets the resulting matrices in a geometric context. It is shown that the SVD, truncated to k dimensions, gives the optimal low-rank k-approximation to the original matrix. The base vectors with weak singular values are thought to be responsible for differentiating between semantically similar terms which belong to the same *topic*; thus, when they are rejected, we get similar vector-space representations for semantically similar texts.

Other researchers have proposed theoretical approaches to understanding LSI. In (Zha and Simon, 1998) LSI is described in terms of a *subspace model*, and a statistical test is proposed for choosing the optimal number of dimensions for a given collection. They also discuss LSI's relationship to statistical regression and Bayesian methods.

(Ding, 1999) constructs a statistical model for LSI using the cosine similarity measure, showing that the term similarity and document similarity matrices are formed when using the maximum likelihood estimation, and that LSI is the optimal solution to this problem. They also conclude that the unsupervised dimension reduction is closely related to unsupervised learning, and that the basis vectors belonging to the strongest singular values can be used to identify good starting centroids for a K-means clustering algorithm.

3. Background

3.1. Vector space model

The vector space model is an algebraic model for representation of text documents as vectors of identifiers, such as vocabulary terms. Its first use was in SMART (*System for the Mechanical Analysis and Retrieval of Text*) Information Retrieval System developed at Cornell University in 1960s (G. Salton, 1975). Documents are represented by vectors from $\mathbb{R}^n$, where each document can be viewed as a linear combination of words from a dictionary of size n.

3.2. Latent Semantic Indexing (LSI)

Latent Semantic Indexing (LSI) has been applied to a wide variety of learning tasks involving textual data. It is often applied to tasks such as search and retrieval, classification

Text Analysis and Retrieval 2016: Course Project Reports (TAR 2016), pages 29–31, University of Zagreb, Faculy of Electrical Engineering and Computing, Zagreb, June 2016.

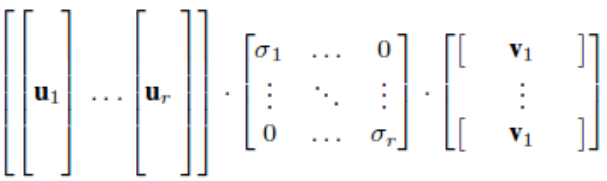


Figure 1: SVD decomposition of a rectangular matrix is analogous to diagonalization of square matrices. The main difference is that there are two new bases, spanned by vectors in $\mathbf{u}$ and $\mathbf{v}$, respectively. Singular values are analogous to eigenvalues.

and filtering. The idea that LSI could be used to bring out the 'latent semantics' within a corpus of documents was originally thought of by (Kontostathis, 2007).

LSI is based on a mathematical technique called Singular Value Decomposition (SVD). The SVD process decomposes a term-by-document matrix $\mathbf{A}$ of rank r into three matrices: a term-by-rank matrix $\mathbf{U}$, a singular-value matrix $\mathbf{\Sigma}$, and a document-by-rank matrix $\mathbf{V}$. Although the original matrix may be obtained through matrix multiplication using all singular values as shown in (1), as stated before, we do not compute the full decomposition, as rejecting weak singular values is a core principle of LSI.

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \tag{1}$$

Thus, in an LSI system, matrices $\mathbf{U}$, $\mathbf{\Sigma}$, and $\mathbf{V}$ are truncated to k dimensions. This truncation is accomplished by removing columns $k+1$ through r of $\mathbf{U}$, columns and rows $k+1$ through r of $\mathbf{\Sigma}$, and rows $k+1$ through r of $\mathbf{V}^T$. This process is shown graphically in Figure 1. (Berry et al., 1995.). Queries are represented in the reduced space as $\mathbf{q}\mathbf{U}_k$ where $\mathbf{U}_k$ is the term-by-dimension matrix, after truncation to k dimensions. Queries are then compared to the reduced document vector using a similarity measure such as the cosine similarity.

3.3. BM25 information retrieval function

In information retrieval, BM25 (BM stands for *Best Matching*) is a modification of the classic probabilistic retrieval. It is based on the probabilistic retrieval framework developed in the 1970s and 1980s by Stephen E. Robertson, Karen Spärck Jones, and others (Stephen E. Robertson, 1998). BM25 is not a single function, but actually a whole family of scoring functions, with slightly different components and parameters. Probably the best known and most used one is:

$$score(D,Q) = \sum_{i=1}^{n} IDF(q_i) \cdot \frac{f(q_i,D)\cdot(k_1+1)}{f(q_i,D)+k_1(1-b)+k_1\cdot\frac{|D|}{avgdl}\cdot b} \tag{2}$$

where n is the number of keywords that the query contains, $f(q_i, D)$ is the term frequency of q_i in the document D, $|D|$ is the length of the document D in words, and $avgdl$ is the average document length in the text collection from which documents are drawn. k_1 and b are free parameters, usually chosen, in absence of an advanced optimization, as $k_1 \in [1.2, 2]$ and $b = 0.75$.

4. Implementation

To implement an LSI-based semantically-aware information retrieval system, a large term-document matrix is required. For this purpose a WaCky (*Web-As-Corpus Kool Yinitiative*) dump of the English Wikipedia was used as a training corpus. We have decided to use the *gensim* software library which has an LSI model training implementation that can stream documents one by one (in chunks of 10,000) from the training corpus without loading the whole corpus into memory (Řehůřek and Sojka, 2010). Furthermore, *gensim* supports offloading this training work to worker nodes in a distributed computing approach, which has come in as a big help since the task of computing the SVD on the term-document matrix of all Wikipedia articles demanded great computational power.

For this, we have prepared a custom Ubuntu live image with the necessary tools installed and booted it using PXE network boot on 15 machines in one of the computer laboratories for students on the Faculty of Electric Engineering and Computing, with one of the machines having a role of the job dispatcher.

We have used three variants of the LSI model with various numbers of topics: 1,000, 700 and 400 due to a number of around 500 topics being recommended in various literature, along with the dictionary size being 100,000. All words that appear in more than 10% of the documents have been removed. Before calculating the SVD, we had applied tf-idf weighting and document vector length normalization. These steps were inspired by examining the existing *gensim* scripts for preparing a Wiki corpus for topic modeling.

Additionally, we have implemented the classical probabilistic retrieval model with the term weights being scaled by the Okapi BM25 function, in multiple variations: first, using the dictionary from Wikipedia, alongside with its IDF weights being used as term weights; second, also using the dictionary from Wikipedia, but building the IDF weights from the indexed test corpus (the EMM collection); third, both the dictionary and the IDF weights being built again exclusively from the test corpus.

Finally, we have tried out combining the outputs from the top performing LSI and BM25 models. The initial attempt was a guess with no optimization performed which is a simple linear combination of the retrieval status values of the models: $\mathrm{RSV}_{BM25}(query)+6\mathrm{RSV}_{LSI}(query)$. Then, the queries along with their relevance judgments were split into training and test sets for cross-validation, where the weights for the linear combination were the hyperparameters being optimized. The optimization objective was to maximize the mean average precision on the training set.

5. Evaluation results

The evaluation results of the various developed models are shown in table 1 and figure 2. The best BM25 model is the one built with a fresh dictionary from the test corpus. In general, the BM25 models outperform the LSI models by a considerable margin. Out of the LSI models, the best performing was one with 1000 topics, which suggests that we could have perhaps done better by building one with even more topics. However, a naïve combination of the best BM25 model with the best LSI model gives an even

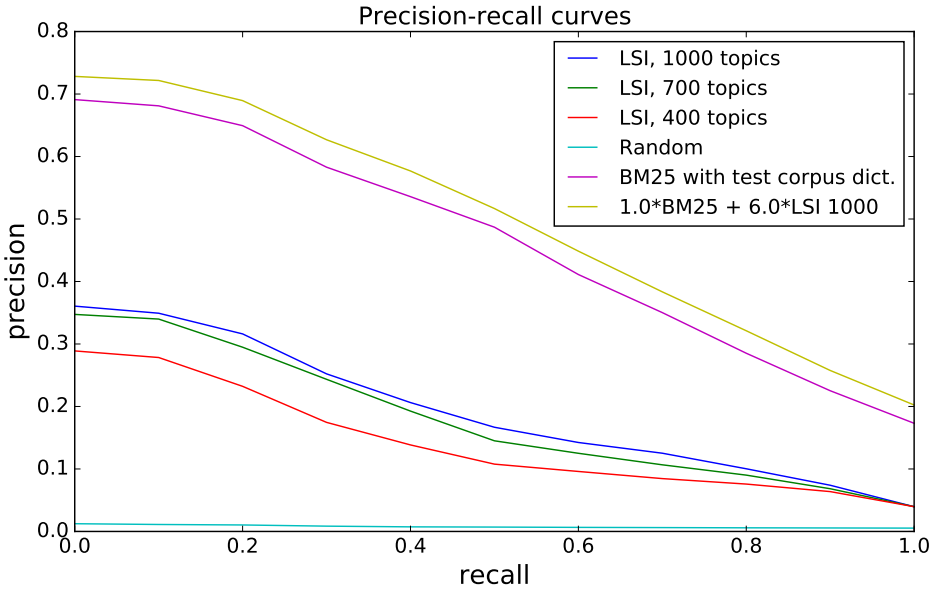


Figure 2: Precision-recall curves for various models, evaluated on the full test set (without hyperparameter optimization). As a baseline, there is a model which returns a random permutation of the indexed corpus.

Model	Average R-precision	Mean average precision (MAP)
LSI, 1000 topics	0.1446	0.1514
LSI, 700 topics	0.13561	0.1411
LSI, 400 topics	0.1091	0.1118
Random	0.00618	0.0090
BM25 with test corpus dict.	0.3259	0.3714
1.0*BM25 + 6.0*LSI 1000	0.3594	0.4080

Table 1: Performance metrics of various models

slightly better model. In addition to that, running a simple optimization algorithm to find the optimal linear combination did not yield any significantly better results than our initial guess.

6. Conclusion

We have built an information retrieval system using two different approaches: the classical probabilistic approach combined with the state-of-the-art BM25 formula, and using LSI – a vector space model approach combined with a transformation into the latent topic space. Out of the two, the BM25 performed significantly better, however, we were able to augment its performance slightly by doing a linear combination of its output with the output of the LSI model.

References

C. H. Q. Ding. 1999. A similarity-based probability model for latent semantic indexing. In Proceedings of the Twenty-second Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval, pages 59–65.

C. S. Yang G. Salton, A. Wong. 1975. A vector space model for automatic indexing. Communications of the ACM, v.18 n.11, p.613-620.

Z. S. Harris. 1954. Distributional structure. Word, 10, 146–162. Reprinted in J. Fodor and J. Katz, The Structure of Language, Prentice Hall, 1964.

April Kontostathis. 2007. Essential dimensions of latent semantic indexing (lsi). *Proceedings of the 40th Xiaoyong*, pages 1–8.

Prabhakar; Schütze Hinrich Manning, Christopher D.; Raghavan. 2008. Introduction to information retrieval. Cambridge University Press.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May. ELRA. http://is.muni.cz/publication/884893/en.

T. K. Landauer G. W. Furnas R. A. Harshman S. C. Deerwester, S. T. Dumais. 1990. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, pages 391–407.

Micheline Hancock-Beaulieu Stephen E. Robertson, Steve Walker. 1998. Okapi at trec-7. Proceedings of the Seventh Text REtrieval Conference (TREC 1998). Gaithersburg, USA.

H. Zha and H. Simon. 1998. A subspace-based model for latent semantic indexing in information retrieval. In Proceedings of the Thirteenth Symposium on the Interface, pp. 315–320.

Sexual Predator Identification Using Ensemble Learning Classifiers

Vinko Kodžoman, Petra Marče, Ana Škaro

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
{vinko.kodzoman,petra.marce,ana.skaro}@fer.hr

Abstract

The aim of this paper is to present a possible solution to the Sexual Predator Identification problem as a part of Text Analysis and Retrieval course at the Faculty of Electrical Engineering and Computing in Zagreb. The given problem was originally presented at the PAN 2012 competition, where the task was divided into two parts: 1) identifying the predators among all the users, and 2) identifying the most distinctive features of the predators' bad behavior by singling out the incriminating lines. We approached these tasks using machine learning, specifically ensemble learning. Features used in our ensemble model are lexical and behavioral features extracted from a dataset consisting of online chat conversations.

1. Introduction

Social network revolution has changed the way people interact and communicate. Everyone who has access to a computer or a mobile phone also has access to some form of chat messaging. Connections between people, acquaintances or strangers, have become much easier to establish. This allows for misbehavior and cybercriminal because it provides the users with much desirable privacy and anonymity for such acts. People, especially children, are easily lured into some kind of unwanted relationship where a person, first presented to them as a friend or peer, wants to take sexual advantage of them.[1]

The task of Sexual Predator Identification presented in PAN 2012 competition consisted of two stages. Given the chat logs involving two (or more) people, the first stage was to determine who was the one trying to convince the other(s) participant(s) to provide some sexual favour. After obtaining the list of author IDs which are considered predators, the second stage was to identify the most suggestive lines of their conversation. We approached these tasks using different machine learning models, with the emphasis on ensemble learning. This decision was based on the fact that this is a modern approach, which combines multiple traditional learning algorithms. The first part of our task is what is considered to be supervised learning; more specifically, classification. Our experiments showed that it is possible for machine learning models to recognize sexual predators using behavioral and lexical analysis of their chat logs.

The second part does not contain any labeled data, so clustering had to be done to determine which lines are considered suggestive. Other works on this topic are presented in Section 2. Feature ablation study is described in Section 3. Our solution with detailed results for both the first and the second stage are given in Section 4. Finally, the conclusion is given in Section 5.

[1] According to the Oxford dictionary, a predator is "A person who ruthlessly exploits others."

2. Related work

In their paper, Villatoro-Tello et al. (2012) proposed the method for detection of misbehaving users in chats based on two hypotheses. The first one claims that the terms used by predators (i.e. pedophiles) differ from terms used in normal chatting, and the second states that predators apply a detectable sort of pattern when approaching the victim. The authors did not include any kind of word normalization into their working pipeline, suggesting that the text in chat conversations had unique characteristics, and often does not follow any grammar rules. Furthermore, they claim that intentional misspelling can be significant in the grooming phase. As a part of prefiltering, they removed conversations with only one participant and those containing images, as well as those that has less than six interventions per user. As their classifiers, authors suggested Neural Networks (NN) with a single hidden layer of ten units and Support Vector Machines (SVM). They used two-fold cross validation to estimate the performance of their models. The second stage, line detection, was approached with language models based approach. The contestants won the first place at PAN 2012 competition, as their model for classification of predators had 0.935 $F_{0.5}$ score. Parapar et al. (2012) defined 11 chat based features in addition to tf-idf model, with the goal to capture behavioral patterns of individuals in chats. As a part of prefiltering, they removed long words and long character sequences present in images. The proposed classification model was SVM, and they got the final F_1 score of 0.78.

3. Feature ablation study

Due to the success of (Parapar et al., 2012) in the first task of predator classification we decided to follow their approach. Two groups of features used in our models are td-idf features and chat-based features. Chat-based features are those that are supposed to capture a sexual predator's specific behavior in chat rooms. The starting premise is that sexual predators use chat more frequently, ask more questions and use specific words, which was confirmed by the feature importance study shown in Figure 1.

The original set of chat-level features from (Parapar et

Text Analysis and Retrieval 2016: Course Project Reports (TAR 2016), pages 32–35, University of Zagreb, Faculy of Electrical Engineering and Computing, Zagreb, June 2016.

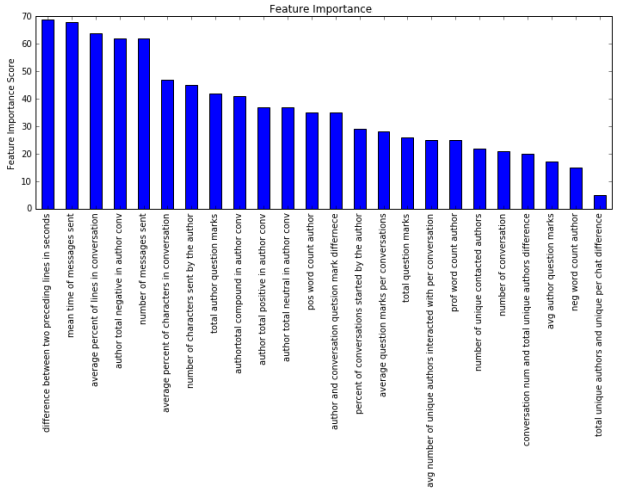


Figure 1: Chat-based feature importance for XGBoost model

al., 2012) was expanded with additional features, some of which can be found in Table 1.

One of the key additional features was the sentiment analysis of the text found in the author's conversation. To get the sentiment of the text, we used the VADER sentiment analysis tool described in (Hutto and Gilbert, 2014). Alongside sentiment analysis, text emotional mapping was done using a positive and a negative corpus[2] of words as well as our own manually created corpus of profane words, part of which is shown in Table 2.

To capture the lexical features specific to sexual predators a tf-idf matrix was used. We experimented with different parameters for the tf-idf matrix, using five-fold cross validation (grid search) and a single XGBoost. XGBoost system is described in (Chen and Guestrin, 2016).[3] Parameter tuning has presented the unigram model as the best model. The tf-idf matrix was formed using the standard weighting scheme:

$$tf/idf_{t,d} = (1 + log(tf_{t,d})) \times log(\frac{N}{df_t}) \qquad (1)$$

All conversations were in English, thus all English stop words were removed from the tf-idf matrix. To prune the tf-idf matrix, only the top 3500 features ordered by term frequency across all conversations were used. As most of the text found in chat conversations was informal, no normalization was used for the terms in the tf-idf matrix. As noted in previous PAN 2012 publications, normalization does not increase the performance of models.

Table 3 shows the performance on test set. We used scaled and not scaled features. Scaling was done with the standard scaler.

4. Implementation

In this section we describe our approach and implementation for both stages of the given task. For the first stage, we needed to classify predators from chat logs. For this, we tried several machine learning models: Naive Bayes, SVM, random forest and XGBoost. For the second stage of identifying suggestive lines, we used KMeans clustering with the target set to two clusters, as our presumption was that suggestive lines would group together, given the right criterium.

4.1. Sexual predator classification

As mentioned in (Villatoro-Tello et al., 2012), some filtering can help focus only on the most important cases and to reduce the computational cost. However, the authors of the aforementioned paper stated that their filtering was successful despite having eliminated a few of the predators in the process. Considering this, we decided to filter all conversations with less than 5 interactions. Nevertheless, it was shown that filtering worked fine with SVM model, but lowered the score when using XGBoost.

Our first attempt was to use random forest classifier, but even though its precision was high, its recall was extremely low. We did not succeed in improving this, so we decided to choose XGBoost using scaled tf-idf features as our baseline model. Our experiments continued with changing chosen features first to chat based only, and finally to the combination of both chat based and tf-idf features. Last experiment has shown the best results. All aforementioned experiments can be seen in Table 3.

Besides ensemble learning, we tried out a more traditional approach. The first classifier we used was the naive Bayes classifier, which performed poorly. The second one was SVM with linear and radial basis functions. Optimal parameters for both RBF and linear SVM were found using the grid search. Optimal linear SVM performed slightly better in terms of $F_{0.5}$ score and F_1 score so we decided to use it in our final solution.

Finally, we decided on creating a bootstrap aggregated (bagged) model, consisting of two linear SVMs, one random forest classifier and three XGBoost classifiers. The final prediction result was constructed using a voting system of all six classifiers. In the final results, an author is labeled as a sexual predator if at least three of six classifiers label him as such. All bagged models used different features and parameters tuned with five-fold cross validation as shown in Table 4. Using a bagged model, we managed to increase the performance of our final model. Results are shown in Table 5. Bootstraping allows for more control over precision and recall, as the number of voters needed for a final decision or their weights can be tuned. Our final $F_{0.5}$ score was 0.823.

4.2. Identifying suggestive lines

Our second task was to identify the most distinctive features of the predators' bad behavior by singling out the incriminating lines. Unfortunately, the results of this task depend largely on the success of the classification from the first stage, because in identifying lines, we only consider chats of authors classified as sexual predators. Also, for this task there was no labeled data so we performed clustering in two groups. In this case, one learning example was one line of chat. Feature vector for each line was extracted using tf-idf

[2] http://www.unc.edu/

[3] Described system is available as an open source package at https://github.com/dmlc/xgboost.

Table 1: Subset of chat-based features with the appropriate descriptions.

Feature name	Feature description
total question marks	Number of question marks found in all author's conversations.
total author question marks	Number of author's question marks found in conversations.
neg. words count author	Number of negative words used by author.
prof. words count author	Number of profane words used by author.
author total negative in author conv.	Percent of negative emotion found in text (sentiment analysis).
author total positive in author conv.	Percent of positive emotion found in text (sentiment analysis).
author total neutral in author conv.	Percent of neutral emotion found in text (sentiment analysis).

Table 2: Subsets of manually created sets of words, used as features, both in classification and clustering phases.

Negative	Profane	Positive
abandoned	darn	able
anger	dear	amazed
awful	legs	brave
scared	hot	kind
worse	skirt	sure
zealous	spank	trust

Table 3: XGBoost model scores for different features.

Measure	tf-idf scaled	chat based	tf-idf + chat based
accuracy	0.999	0.999	0.999
precision	0.791	0.769	**0.879**
recall	0.209	0.461	**0.571**
F_1 score	0.330	0.576	**0.692**
$F_{0.5}$ score	0.508	0.679	**0.793**

with fixed vocabulary. Here, we experimented with different vocabulary.

The first vocabulary consisted of k-most important words from the classification stage as derived from SVM with tf-idf feature matrix from our bagged model. Higher absolute value of the feature associated weight implied its higher importance in final classification decision. In addition to this vocabulary, we also used manually created sets of negative and profane words, mentioned in Section 3. For clustering algorithm we used simple K-means algorithm with K-means++ method for initialization of centroids. Our final F_3 score was 0.384. The results are shown in Table 6.

Table 4: Bootstrap aggregated models and their features.

No. of models	Classifier	Features
1	SVM (linear)	chat based
1	SVM (linear)	tf-idf
2	XGBoost	chat based + tf-idf
1	XGBoost	chat based
1	random forest	chat based + tf-idf

Table 5: Final score for the bagged model for sexual predator classification.

Measure	Bootstrap aggregated model
accuracy	0.999
precision	0.901
recall	0.610
F_1 score	0.728
$F_{0.5}$ score	0.823

Table 6: Clustering results in terms of recall, precision and F_3 measure on test set for different subsets of words in vocabulary.

Measure	negative	profane	SVM	combination
recall	**0.779**	0.777	0.759	0.774
precision	0.068	0.038	**0.069**	**0.069**
F_3 score	0.382	0.381	0.381	**0.384**

5. Conclusion

Sometimes it is hard to distinguish between sexual predator behavior and harmless flirting based on chat messages only, even for a human being. Chat based features give us valuable information about individual chatting habits and behavior. However, it is shown that the combination of these two groups of features can give satisfying results in sexual predator identification problem.

Despite successful feature extraction, it is crucial to choose the right method for classification. In our work we tried some traditional methods, naive Bayes and SVM. We also decided to experiment with a new and modern approach – ensemble learning. It is shown that ensemble learning models, with the features we extracted, gave better results. However, it is important to note that our score is still not better than the official best score achieved at the PAN 2012 competition.

The second problem was more complicated, but we managed to obtain satisfying result when comparing it to the official PAN 2012 results.

References

J. Parapar, D. E. Losada, and A. Barreiro. 2012. A learning-based approach for the identification of sexual predators in chat logs. *PAN 2012*.

E. Villatoro-Tello, A. Juarez-Gonzales, H. J. Escalante, M. Manuel-y-Gomez, and L. Villasenor-Pineda. 2012. A Two-step Approach for Effective Detection of Misbehaving Users in Chats. *PAN 2012*.

Hutto, C. J. and Gilbert, E. E. 2014. VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. *Eighth International Conference on Weblogs and Social Media (ICWSM-14). Ann Arbor, MI, June 2014*.

T. Chen and C. Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. *22nd SIGKDD Conference on Knowledge Discovery and Data Mining, 2016*.

Domain Adaptation for Classification of Newswire Data

Janko Kovačević, Vilim Stubičan, Josip Vujnović

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
janko.kovacevic@fer.hr, vilim.stubican@fer.hr, josip.vujnovic@fer.hr

Abstract

This document provides information about classification of two different learning algorithms with simple adjustment for domain adaptation.

$$\Phi^s(x) = \langle x, x, 0 \rangle, \Phi^t(x) = \langle x, 0, x \rangle$$

Figure 1: *Feature selection for domain adaptation*

comp.graphics	rec.autos	sci.crypt
comp.os.ms-windows.misc	rec.motorcycles	sci.electronics
comp.sys.ibm.pc.hardware	rec.sport.baseball	sci.med
comp.windows.x	rec.sport.hockey	sci.space
misc.forsale	talk.politics.misc	talk.religion.misc
	talk.politics.guns	alt.atheism
	talk.politics.mideast	soc.religion.christian

Table 1: *Topic groups*

1. Introduction

Text categorization is the classification of documents into a fixed number of predefined categories. Each document can be in multiple, exactly one or no category at all. Main objective of text categorization is to learn machine models known as classifiers from examples. There are two types of automated text classification: handcrafted rule-based systems and supervised machine learning models. Handcrafted rule-based systems are based on occurrence or non-occurrence of certain word or words in the document. They are expensive to design and maintain and require a domain expert for both actions. On the other hand, supervised machine learning models use a small labeled data set. Labeling is quite easier in comparison with designing rules. Most often used are Naïve Bayes and Support Vector Machine (SVM).

2. Domain adaptation

Domain adaptation is classification that is appropriate exactly in the case when one has enough "target" data to do slightly better than just using only "source" data. System used in this project is quite easy to implement and straight forward to understand. Adaptation is done through generating three versions of features: one general one, one source specific and one target specific. Meaning, augmented source data is going to contain general and source specific features, while augmented target will contain general and target specific features.

By expanding features vector like this, there will be more specific word per domain. Let us consider an example where source domain is Wall Street Journal and target domain are reviews of computer hardware. Words like "the" would not make a difference due to their frequency. However, word "monitor" should have greater effect for the computer hardware, even if it has meaning in Wall Street Journal domain.

3. Dataset

Our dataset is a collection of twenty Newsgroup datasets, it contains twenty thousand newsgroup documents, partitioned (nearly) evenly across twenty different newsgroups. This dataset was originally collected by Ken Lang. The twenty newsgroups collection is a popular data set for experiments in text applications of machine learning techniques, such as text classification and text clustering. The data is organized into 20 different newsgroups, each corresponding to a different topic. Some of the newsgroups are very closely related to each other (e.g. *comp.sys.ibm.pc.hardware* / *comp.sys.mac.hardware*), while others are highly unrelated (e.g. *misc.forsale* / *soc.religion.christian*). Each newsgroup is divided to a training set and a test set in ration 3 : 1. Topics grouping in the dataset is shown on Figure 2.

4. Inverted index

In computer science, an inverted index (also referred to as postings file or inverted file) is an index data structure storing a mapping from content, such as words or numbers, to its locations in a document. The purpose of an inverted index is to allow fast full text searches, at a cost of increased processing when a document is added to the database. It is the most popular data structure used in document retrieval systems, used on a large scale for example in search engines. There are two main variants of inverted indexes: a record-level inverted index (or inverted file index or just inverted file) contains a list of references to documents for each word. A word-level inverted index (or full inverted index or inverted list) additionally contains the positions of each word within a document. We used the following, full inverted index. It was accomplished using a HashMap and all the words were mapped in it accordingly to their containing file and their positions in it. During the mapping process we ignored the stop words so they weren't mapped. This use of inverted index allowed us to have quick access to information about which documents contained the specific words that we were looking.

Text Analysis and Retrieval 2016: Course Project Reports (TAR 2016), pages 36–38, University of Zagreb,
Faculy of Electrical Engineering and Computing, Zagreb, June 2016.

$$P(c \mid d) \propto P(c) \prod_{1 \leq k \leq n_d} P(t_k \mid c)$$

Figure 2: *Naïve Bayes probability*

$$P(w_i \mid c_j) = \frac{N(W{=}w_i, C{=}c_j)+1}{N(W, C{=}c_j)+|V|}$$

Figure 3: *Multinomial probability*

5. Naïve Bayes classifier

The first supervised learning method we introduce is the multinomial Naïve Bayes or multinomial NB model, a probabilistic learning method. The probability of a document d being in class c is computed as in Figure 2.

where $P(t_k|c)$ is the conditional probability of term t_k occurring in a document of class c. We interpret it as a measure of how much evidence t_k contributes that c is the correct class. Furthermore, there are several different types of NB classification models. One used in this project is multinomial document model which takes into account frequency of unique words. Also, to avoid 0-values for classification results, this model (as most of them do) uses Laplace smoothing. Using multinomial model, we can derive probability for classification of single word into class c as in Figure 3 where $N(W = w_i, C = c_j)$ is number of time word w_i in class c and $N(W, C = c_j)$ is number of words in class c. $|V|$ is number of unique words from dataset. Also, to avoid decimal error underflow in computer systems, probability logarithms are summed for the final result.

6. SVM

A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. Given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples. Support vectors are the input samples that lie closest optimal hyperplane. One remarkable property of SVMs is that their ability to learn can be independent of the dimensionality of the input space which makes it great tool for text classification. On average, our input space contained 40 000 features (120 000 features for domain adaptation). In order to use SVM for text classification, samples needed to be preprocessed. Preprocessing of samples is described in next subchapter.

6.1. Data preprocessing

We assume access to a labeled set of text newswire samples D which is used to train SVM model. Let X represents input space and $Y \in \langle -1, 1 \rangle$ represents output space. Goal of data preprocessing is to represent text sample $d \in D$ as vector of input space. Input space model which is used is bag-of-words model. Bag-of-words model is a vector of size n, where n is number of unique words amongst all samples D. Each unique word has its own index in bag-of-words model. Stop words should not be included in said model since they are common and they are not domain unique.

	A	$A \cup B$ no domain adaptation	$A \cup B$ domain adaptation
NB	-	0.9105	0.9105
SVM	0.6963	0.708	0.7477

Table 2: *Results for comp.* and sci.* datasets*

	A	$A \cup B$ no domain adaptation	$A \cup B$ domain adaptation
NB	-	0.9616	0.956
SVM	0.7248	0.7561	0.7822

Table 3: *Results for talk.* and rec.* datasets*

For example, let us say we classify text documents in two classes: computer versus science newsire. Stop word 'all' brings no information whether text should be classified as computer or science, whereas word 'monitor' is more likely to be a noun in computer newswire. Input space vector Xd is created fom every sample $d \in D$ after the bags-of-word model had been created. Firstly, text sample d is indexed in order to retrieve word count for each word. Afterwards, new empty input space vector is created. Finally, number of repetitions of each word in sample is assigned to input space vector at index i, where i is index of same word in bag-of-words model. If the index i does not exists (e.g. stop word had been found in sample d) then don't include that word in input space vector. For longer texts it is possible to add additional condition to exclude all words whose frequency of appearing in sample is less than 3. Since texts in datasets we are using in this project are short in length, this condition was not included. Input space for domain adaptation is slightly altered version of said approach and was described in domain adaptation chapter.

7. Results

Measuring was done on two different data sets from following domains: comp.graphics and sci.space with tests in *comb.windows.x* and *sci.med*, and *rec.sport.baseball* and *talk.politics.guns* with tests in *rec.sport.hockey* and *talk.politics.mideast*. Both NB model and SVM model were trained and tested on sets.

For NB model results could be measured for unison of domains with and without domain adaptation. SVM was trained and tested over the same datasets. Results are displayed as F1-score in Table 2. and Table 3.

8. Conclusion

Considering differences between model that has no additional parameters besides input dataset (NB model) and model with adjustable parameters for training (SVM), both models performed well.

NB model provided better result due to type of the dataset: training datasets were mostly short text containing important information about specific domains, which provided more accurate probability distribution than anything than SVM could train itself to classify. Nevertheless, SVM model provided pretty good results for machine learning model with large bag of words. Final conclusion would be that classification is hard, but it can be optimized with

proper selection of model considering input dataset and, furthermore, additional improvement can be derived from simple domain adaptation as described in this project.

9. Literature

Joachims 2016., Text Categorization with Support Vector Machines: Learning with Many Relevant, https://www.cs.cornell.edu/people/tj/publications/joachims_98a.pdf

Daume 2016., Frustratingly Easy Domain Adaptation, http://www.umiacs.umd.edu/ hal/docs/daume07easyadapt.pdf

2016. Text Classification & Clustering, http://www.fer.unizg.hr/_download/repository/TAR-06-TCC.pdf

A Machine Learning Approach for Sexual Predator Identification in Chat

Luka Križan, Marko Lukić, Josip Užarević

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
{luka.krizan, marko.lukic, josip.uzarevic}@fer.hr

Abstract

This paper describes the system for the identification of sexual predators in chat, developed as a project for Text Analysis and Retrieval course at Faculty of Electrical Engineering and Computing. The tasks for the project were to identify sexual predators within all the users and to identify the lines of conversations which are the most distinctive of the predator bad behaviour. We developed a simple system using machine learning with TF-IDF weighted vector space model and compared our results with the results from PAN 2012 competition.

1. Introduction

A sexual predator is a person seen as obtaining or trying to obtain sexual contact with another person in a metaphorically "predatory" or abusive manner (Filler, 2001). During the past few decades, Internet has become the most dominant social media giving the opportunity to users from all around the world to communicate, e.g., through instant messaging services, social networks, forums and blogs. Such services give users option to hide their personal information, which gives an opportunity to users to establish new connections, but it can also represent a threat from sexual predators, especially to children and underage users. According to NBC news program *Dateline*, USA law enforcement officials in 2006 estimated that around 50000 sexual predators are online at any given moment. Although the origins of that figure have been questioned by many sources, the figure still shows that the threat from sexual predators to underage users in online activities is present and real.

In 2012, *PAN*[1] ran a competition in sexual predator identification with a task to identify sexual predators from given chat logs which involved two or more people. The task was divided into two parts:

- identifying the predators within all the users;
- identifying the lines of the predator conversations which are the most distinctive of the predator bad behaviour.

The training corpus for the competition included 66928 conversations between 97690 different users from which 148 were tagged as sexual predators. For the second part of the task, no direct training data was given.

2. Related work

As this task was the part of PAN 2012 competition, we focused on the participants' approaches (Inches and Crestani, 2012).

2.1. Identifying predators

The collection given as a training set was intentionally unbalanced (having less than 1% of positive class examples), so most of the participants used two stage classifier to filter out the negative examples. In the first stage, classifier distinguished between conversations involving a predator and conversations without a predator. The second stage was used for the predator-vs-victim classification. This approach was used by participants who got the highest score in the competition (Villatoro-Tello et al., 2012). The other most commonly used approach was to remove the conversations that included only one participant, those that had less than six messages per user, etc. Most of the participants used unigram or bigram model with TF-IDF weighting to extract features. In general, features have been used without stemming and stopword removal to preserve author's style, including misspelling and grammatical errors. Some of the participants included "behavioural" features, e.g., the number of times a user starts a dialogue, the number of questions asked, etc. For classification, the participants used various methods, focused mostly on the machine learning algorithms. The most used method was support vector machine (SVM).

2.2. Identifying predators' lines

The simplest solutions used for this problem was to return as relevant all the conversation lines of all identified predators from the first problem. The most used method was filtering of all predator conversations through a dictionary of "perverted" terms or with a particular score (e.g., TF-IDF weighting).

3. Proposed method

To implement a system for sexual predator identification, we used a simpler approach than most of the participants and included some of the methods mentioned above, but with a focus on the lexical features of messages sent between the users. The system was implemented in Python 3 using text analysis tools from NLTK[2] module and machine learning tools from scikit-learn[3] module.

[1] Series of scientific events and shared tasks on digital text forensics. http://pan.webis.de/

[2] http://www.nltk.org/

[3] http://scikit-learn.org/

Text Analysis and Retrieval 2016: Course Project Reports (TAR 2016), pages 39–42, University of Zagreb, Faculy of Electrical Engineering and Computing, Zagreb, June 2016.

Table 1: The number of users removed from the training data related to the value of threshold

Threshold	Users removed	Positive users removed
1	20804	1
2	57962	4
3	72720	4
4	75644	4
5	**76578**	**4**
6	77236	4
7	77956	4
10	81013	5

3.1. Identifying predators

3.1.1. Preprocessing

After taking into account the nature of chat messages, we decided to perform tokenization and lemmatization in preprocessing. Chat messages are usually poorly structured with many grammatical errors and misspellings. They also include emoticons, internet slang and intentional deviations from rules of grammar. Performing stopword removal, stemming or lemmatization in this case doesn't have significant effect in reducing the total number of words and classification.

On account of inefficient preprocessing, the total number of words appearing in messages was too large to efficiently do any kind of further computation on the input data. To reduce the total number of words, we decided to remove the words which in total occurred less than five times. After word filtering, the total number of words was reduced to approximately a half of its original value.

3.1.2. User filtering

As was mentioned in the introduction, the training data included conversations between 97690 users, from which only 148 were tagged as sexual predators. To reduce the number of users tagged as negative, we decided to put a threshold to the minimal number of messages sent per user. If number of messages sent by a particular user was less or equal than the value of threshold, they were removed from the training data. The total number of users (tagged both positive or negative) removed from the data related to the value of threshold is shown in Table 1. After analysis, we decided to set the value of threshold to five because no significant improvement was made with a higher threshold value. The total number of users was reduced almost to a quarter of its original value, while only four users tagged as positive were removed.

Also, to reduce the computation time for the user representation, we removed one user who was tagged as negative, because their TF-IDF computation time was longer than the combined computation time of all the other users.

3.1.3. User representation

For the user representation, we used a very simple approach, similar to the approach used by the participants who were ranked as third in the competition (Parpar et al., 2012). Every chat user is represented with a document – a collection of all messages that they sent in any conversation that they participated. Every document was transformed into a unigram representation, weighted with a standard TF-IDF weighting scheme. As a result, every user was simply represented as a floating point vector. Additionally, we also included a non-text feature – the average time of the day when the user was chatting. We divided the time of the day into three categories:

- morning – between 7:00 and 14:59;
- afternoon – between 15:00 and 22:59;
- night – between 23:00 and 6:59.

Because of an equal distance between these categories, we added them to the feature vectors using one-hot encoding. In the end, total representation of a user was a vector with a size of 59378.

The simplicity of this approach also has its negative aspects. It is not possible to extract most of the "behavioural" features that could also be important for classification, e.g., the number of conversations started by the particular user, the number of users that were included in conversations with this particular user, the frequency of turn-taking, etc. However, we expect that this representations still contains enough clues to identify the bad behaviour distinctive for the sexual predators.

3.1.4. Training

Due to a highly unbalanced dataset, the straight-forward approach in the classifier training (training performed with the exact data which was left after filtering) would result in a trivial classification with all users classified as negative. To tackle that problem, we considered two approaches – oversampling the positive class and undersampling the negative, majority class. As we already encountered problems with high computational costs, we decided to undersample the negative class.

In the first step of training, we split the training collection into training and test parts (with 70-30 ratio, while keeping the number of the positive class examples balanced). In the second step, we undersampled the training part, using the different amount of the negative class examples ($\{\frac{1}{16}, \frac{1}{8}, \frac{1}{4}, \frac{1}{2}, 1\} \cdot n$, where n is the number of negative class examples in the training part). After undersampling, we performed 5-fold cross-validation on the resized training part, which was used to find the classifier model and its hyper-parameters that maximize $F_{0.5}$ score (the measure which was used in PAN 2012 for the evaluation of the predator identification). We used SVM (with linear and RBF kernel), logistic regression and k-nearest neighbours classifier. The classifier with the best score in the cross-validation was tested on the test part and also evaluated with $F_{0.5}$ measure. To find the optimal amount of undersampling, the whole cross-validation process was then performed again, with a lower undersampling rate. Due to a non-deterministic way in which negative users were chosen for undersampling, we repeated each test several times and took the best result.

Both logistic regression and SVM with linear kernel showed promising results, while KNN classifier showed

Table 2: The results for the sexual predator identification task with L2-regularized logistic regression on the test collection

Precision	Recall	F_1 score	$F_{0.5}$ score
0.9175	0.7008	0.7946	0.8641

very poor results. The highest score on the test part was scored by L2-regularized logistic regression, while using only a quarter of negative class examples from the training part. Also, SVM with linear kernel using the same number of negative class examples scored almost the same score as logistic regression. Due to very high computation costs of training and classification with SVM, we decided to focus only on the logistic regression.

Unfortunately, we did not keep track of which negative examples were used in the cross-validation process, so it was not possible to retrain the classifier using all of the positive examples from the whole training collection (as it would not be retrained with the same negative examples). As a consequence, the classifier was trained only in the cross-validation process, using only 70% of the positive class examples.

3.2. Identifying predators' lines

To identify the predators' lines, we considered two approaches. The first approach was to return all the messages from all the users who were classified as predators in the previous step. Even though it is an oversimplified approach, the participants who used it in PAN 2012 competition received good results. For the second approach, we decided to modify the previous approach. Instead of returning every message sent by the users classified as predators, we filtered messages by classifying them individually using the classifier from the previous step. Our assumption was that the second approach would outperform the first approach because it should have a better precision score while keeping a similar recall score.

4. Results

The test collection contained conversations between 218702 users, from which 254 were marked as sexual predators. Both training and the test collection were equally unbalanced, containing around 0.1% of positive class examples. For the second part of the problem, 6478 messages from the test collection were tagged as distinctive of the predator bad behaviour. $F_{0.5}$ score was used for evaluating the system in the predator identification and F_3 for identifying the predators' lines.

4.1. Identifying predators

The results for the predator identification task are shown in Table 2. The classifier had a high score in precision and a lower score in recall, but it was trained to maximize $F_{0.5}$ score (which emphasizes precision) so the total result was not unexpected. $F_{0.5}$ score of 0.8641 would place our system at the fifth place (out of 16 participants) for the predator identification task in PAN 2012 competition, with less than 0.01 lower score than the systems that were ranked as third and fourth.

Table 3: The results for the task of identifying predators' lines for both approaches used (returning all messages sent by the users marked as predators in the first step and filtering the messages by their individual classification using the classifier from the first step)

Method	Precision	Recall	F_3 score
Returning all messages	**0.0967**	**0.8902**	**0.4889**
Message filtering	0.0721	0.0012	0.0013

4.2. Identifying predators' lines

Our results for the second part of the problem are shown in Table 3. The results were completely opposite from our initial expectation, since the second approach, which used message filtering, scored a really low score, both in precision and recall. The first approach had a low score in precision because it returned all messages sent by predators, but it showed a good result in total because F_3 score was used (which emphasizes recall). Our system, while using very simple approach of returning all the messages sent by the users classified as positive in the previous step, outperformed all systems at the task of identifying predators' lines and with a F_3 score of 0.4889 it would be ranked as first in PAN 2012 competition.

5. Conclusion

Our task was to develop a system for identification of sexual predators among all the users in chat and for identification of the lines of conversations which are the most distinctive of their bad behaviour. We used a fairly simple machine learning approach with TF-IDF weighted vector user representation.

Our system, even though it was much simpler than most systems used by the participants in PAN 2012 competition, ended with good results in both parts of the sexual predator identification task. As always, there is a lot room left for improvement and with inclusion of more behavioural features in user representation, system would probably end up with better results for the task of predator identification (and consequentially with even better results for the task of identifying predators' lines).

References

Daniel M. Filler, Making the Case for Megan's Law: A Study in Legislative Rhetoric. Indiana Law Journal, Vol. 76, No. 2, 2001.

Giacomo Inches and Fabio Crestani, Overview of the International Sexual Predator Identification Competition at PAN-2012

Esaú Villatoro-Tello, Antonio Juárez-González, Hugo Jair Escalante, Manuel Montes-y-Gómez, and Luis Villaseñor-Pineda, A Two-step Approach for Effective Detection of Misbehaving Users in Chats - notebook for PAN at CLEF 2012

Javier Parpar, David E. Losada and Alvaro Barriero, A learning-based approach for the identification of sexual predators in chat logs - notebook for PAN at CLEF 2012

Supervised Tweet Classification

Josip Milić, Tomislav Marinković, Domagoj Pereglin

University of Zagreb, Faculty of Electrical Engineering Computing
Unska 3, 10000 Zagreb, Croatia
{josip.milic,tomislav.marinkovic, domagoj.pereglin}@fer.hr

Abstract

In this paper we describe our implementation of a method for topic classification of Twitter messages written in Croatian language into one of six predefined classes. We present the results of three machine learning algorithms (SVM, logistic regression, and k-NN) used for classification. Our training data consists of Croatian twitter messages acquired through Twitter API. These messages are manually labeled and passed to classifiers which are trained and then evaluated.

1. Introduction

Twitter[1] is a popular microblogging service where users post messages called *tweets*. These messages can contain different media types, but usually are consisted of different forms of text. Twitter's most valued differentiating feature from other social media services is the mandatory brevity because the size of each message is limited to 140 characters. This makes it a great platform for sharing information in a short and straightforward form such as user comments and news (or at least an essential part of it). Each Twitter user can subscribe and read tweets from other users which are usually their friends or people who they admire and value their opinions. Users can also be official representatives of news agencies and post news tweets. Many of them are specialized for different kinds of news (e.g. information technology, politics), but by subscribing to them users can also expect possibly unwanted information such as advertisements or tabloid news. Tweet classification could be used for filtering tweets related to topics specified by the user. For the purpose of demonstration, topics were news (segmented as IT, politics, sports and general news) and deals (e.g. job offers, user ads).

Several Croatian Twitter users were chosen because of their language and expected types of tweets (e.g. @bugonline for IT news, @hrtsport for sports news) and their tweets were acquired via Twitter API.

In order to train a classifier, supervised learning usually requires hand-labeled training data. Six descriptive labels were used each for every topic, detailed in subchapter 3.2. Labeling.

In grammar, inflection is the modification of a word to express different grammatical categories such as tense, case, voice, gender, and other. Croatian language is highly inflective so special care was taken regarding this matter in preprocessing step. For feature extraction we used the bag-of-words model. This is a simplifying representation where text (such as a sentence or a document) is represented as the bag (multiset) of its words, disregarding grammar and word order. It is commonly used in methods of document classification where the frequency of occurrence of each word is used as a feature for training a classifier.

[1] http://www.twitter.com

2. Related works

There are several research papers regarding tweet classification.

One approach to short text classification is to use a small set of domain-specific features. Siriam et al. extracted these features from the author's profile and text (Sriram et al., 2010). Their set of features consisted of one nominal and seven binary features. Nominal feature is author of a tweet, and binary features are shortened words and slangs, time-event phrases, emphasized words, currency and percentage signs, opinioned words, "@username" at the beginning of the tweet, and "@username" within the tweet. They classified tweets to a predefined set of generic classes such as News, Events, Opinions, Deals, and Private Messages. Naïve Bayes classifier is trained using 5-fold cross validation.

Batool et al. analyzed extracted information from tweets using keyword based knowledge extraction (Batool, 2013). They extracted keywords, entities, synonyms, and parts of speech from tweets which are then used for classification and sentiment analysis. The extracted knowledge was enhanced using domain-specific seed based enrichment technique in a way that decreases information loss. They reported improvement from 0.1% to 55% with their proposed system.

3. Data and methods

Implemented method consists of five stages: gathering data, labeling, preprocessing, feature extraction, and machine learning.

3.1. Gathering data

Croatian tweets were acquired by using GrepTweet [2]. It retrieved tweets from chosen users as text files where each file row had form *tweet ID* | *posting date* | *text content*. Tweet IDs were used for differentiation and labeled tweet texts were used for model training and testing.

3.2. Labeling

Table 1 shows chosen descriptive category (class) names and a number of labeled tweets per each category.

[2] http://greptweet.com/

Text Analysis and Retrieval 2016: Course Project Reports (TAR 2016), pages 43–46, University of Zagreb, Faculy of Electrical Engineering and Computing, Zagreb, June 2016.

category	number of labels
NEWS_TECHNOLOGY	1043
NEWS_POLITICS	1657
NEWS_SPORT	1052
NEWS_REST	2738
DEALS	976
REST	3257
Σ	10723

Table 1: Category names and number of labels per category

Tweets containing news were split into four subcategories and tweets containing deals refer to all kinds of ads and offers, ranging from job offers to item selling ads. Tweets that do not belong to any of these categories were labeled as REST.

Tweets were manually labeled by the authors. Each tweet received a single label. Tweets with multiple topics were labeled as their primary topics or were discarded as REST if the author couldn't decide. Cohen's kappa was used to measure inter-rater agreement. It measures the agreement between two annotators who each classify N items into C mutually exclusive categories. The equation for kappa is: $\kappa = \frac{p_0 - p_e}{1 - p_e}$, where p_o stands for relative observed agreement and p_e for hypothetical probability of chance agreement.

Each annotator provided 50 labeled tweets from each category. Calculated values are shown in Table 2.

	$Annotators(\kappa)$		
	$A1$	$A2$	$A3$
$A1$	1.000	0.742	0.745
$A2$	0.742	1.000	0.805
$A3$	0.745	0.805	1.000

Table 2: Cohen's kappas between annotators

Kappas for all pair combinations of annotators were summed and averaged to get mean Cohen's kappa. Calculated mean Cohen's kappa is $\bar{\kappa} = 0.764$.

3.3. Preprocessing

Before classification, retrieved tweet texts were preprocessed. Firstly, tweet texts were purified by removing punctuation marks (e.g. commas, brackets, quotation marks, stroke) which were considered bad for classifier training. Secondly, TakeLab preprocessor was used for NLP pipeline preprocessing of tweet text. The NLP pipeline consists of sentence segmentation $\triangleright$ tokenization $\triangleright$ POS tagging $\triangleright$ morphological processing (stemming and lemmatization).

Sentence segmentation and tokenization splits sentences into smaller meaningful parts. Lemmatisation is a process of transforming a word to its basic form.

3.4. Feature extraction

TF-IDF is the weight computed as the product of the term frequency component and the inverse document frequency component.

$$tf(k_i, d_j) = 0.5 + \frac{0.5 * freq(k_i, d_j)}{max(freq(k, d_j)|k \in d_j)} \quad (1)$$

$$idf(k_i, D) = log \frac{|D|}{|d \in D|k_i \in d_j|} \quad (2)$$

TF-IDF value (product of TF and IDF values) reflects how important a word is to a document in a collection or corpus. Training and test datasets were converted to corresponding TF-IDF vectors by using created vocabulary of words retrieved from the training dataset. The created vocabulary consisted of 14822 words.

For the purpose of "helping" the classifiers, map of special words for each category was created. Each TF-IDF vector is enhanced by adding six (one for each category) values at the end of the vector. Each value represents the size of intersection of a set of special words from category and current set of words. Evaluation results were slightly improved with using only dozens of special words per category.

4. Machine learning and evaluation

Dataset was split into train and test parts (train dataset = 70%). Three classifiers were used for tweet classification. Two baseline classifiers were used for comparing evaluated results. Main classifiers were: SVM, Logistic Regression and k-NN. Statistical values used for classifier evaluation ($True, False, Positive, Negative$ denoted as T, F, P, N) are:

- $precision = \frac{T_P}{T_P + F_P}$, the fraction of retrieved instances that are relevant
- $recall = \frac{T_P}{T_P + F_N}$, the fraction of relevant instances that are retrieved
- $F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} = \frac{2 \cdot T_P}{2 \cdot T_P + F_P + F_N}$, the harmonic mean of precision and recall
- $accuracy = \frac{T_P + T_N}{T_P + T_N + F_P + F_N}$, the proportion of true results (both true positives and true negatives) among the total number of cases examined

Cross validation was performed on the whole dataset (training and test labeled data combined). The dataset was split randomly to train and test data twenty times for classifier fitting/testing purpose.

4.1. Dummy classifier

Dummy classifier is a classifier that makes predictions using simple rules. Two dummy classifiers were used for baseline classification. The second gave single prediction for all test data - most common category in dataset.

category	$precision$	$recall$	$F1 - score$
NEWS_TECH	0.000	0.000	0.000
NEWS_POLITICS	0.000	0.000	0.000
NEWS_SPORT	0.000	0.000	0.000
NEWS_REST	0.000	0.000	0.000
DEALS	0.000	0.000	0.000
REST	0.293	1.000	0.453
Σ	0.049	0.167	0.075
Accuracy	29.282 %		

Table 3: Evaluated dummy classifier values

4.2. SVM classifier

A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (supervised learning) the algorithm outputs an optimal hyperplane which categorizes new examples.

Implemented SVM is using linear kernel because of the number of features. SVM classifier was fitted with labeled and preprocessed training data represented as TF-IDF vectors. For the purpose of optimal classification, penalty parameter C was chosen from predetermined values with grid search method. Optimal C for training data is $C = 0.7$.

category	$precision$	$recall$	$F1 - score$
NEWS_TECH	0.736	0.650	0.691
NEWS_POLITICS	0.871	0.844	0.857
NEWS_SPORT	0.902	0.856	0.879
NEWS_REST	0.577	0.597	0.586
DEALS	0.851	0.795	0.822
REST	0.687	0.732	0.709
Σ	0.771	0.746	0.757
Accuracy	72.61 %		

Table 4: Evaluated SVM classifier values

Calculated cross validated accuracy with standard deviation is $accuracy_{CV} = 68.24(\pm 9)\%$.

4.3. Logistic regression classifier

Logistic regression measures the relationship between the categorical dependent variable and one or more independent variables by estimating probabilities using a logistic function, which is the cumulative logistic distribution.(Freedman, 2009)

Logistic regression classifier was fitted with labeled and preprocessed training data represented as TF-IDF vectors. For the purpose of optimal classification, inverse of regularization strength C was chosen from predetermined values with grid search method. Optimal C for training data is $C = 12.75$.

category	$precision$	$recall$	$F1 - score$
NEWS_TECH	0.773	0.626	0.692
NEWS_POLITICS	0.869	0.835	0.852
NEWS_SPORT	0.9126	0.834	0.871
NEWS_REST	0.564	0.604	0.583
DEALS	0.865	0.788	0.825
REST	0.680	0.738	0.708
Σ	0.777	0.737	0.755
Accuracy	72.27 %		

Table 5: Evaluated logistic regression classifier values

Calculated cross validated accuracy with standard deviation is $accuracy_{CV} = 68.02(\pm 9)\%$.

4.4. k-NN classifier

k-Nearest Neighbors algorithm (k-NN) is a non-parametric method used for classification. k-NN is a type of instance-based learning, or lazy learning, where the function is only approximated locally and all computation is deferred until classification. The k-NN algorithm is among the simplest of all machine learning algorithms.

k-NN classifier was fitted with labeled and preprocessed training data represented as TF-IDF vectors. For the purpose of optimal classification, number of neighbors k was chosen from predetermined values with grid search method. Optimal k for training data is $k = 28$.

category	$precision$	$recall$	$F1 - score$
NEWS_TECH	0.797	0.374	0.509
NEWS_POLITICS	0.859	0.681	0.759
NEWS_SPORT	0.910	0.744	0.819
NEWS_REST	0.682	0.376	0.485
DEALS	0.565	0.752	0.645
REST	0.515	0.843	0.639
Σ	0.721	0.628	0.643
Accuracy	63.35 %		

Table 6: Evaluated k-NN classifier values

Calculated cross validated accuracy with standard deviation is $accuracy_{CV} = 63.07(\pm 7)\%$.

5. Conclusion

Tweet classification evaluation values of used classifiers were satisfiable considering the language of text and used preprocessing methods and tools. Equivalent version of WordNet (lexical database) designed for Croatian language would be of a great help for classification of tweets. There is a number of things which can be done to improve results such as enlargement of labeled dataset (use of supervised annotating of large scale would certainly help), including more special features (with more intelligent approach than just counting the length of the intersection), and further tweaking of used classifiers.

Web application for demonstration of a practical use of tweet classification was created. Its basic idea is that users can filter tweets by choosing desired categories. Only those tweets which were classified into desired classes are shown on screen. For the purpose of experimenting, classifier selection and manual input of text is also included.5.

By improving the tweet classifier, users could be less exposed to tweets considered as noise and irrelevant, but also wouldn't miss tweets from desired categories misclassified as undesired categories.

References

Maqbool Batool, Khattak. 2013. Precise tweet classification and sentiment analysis. IEEE.

David A. Freedman. 2009. Statistical models: Theory and practice. page 128. Cambridge University Press.

B. Sriram, D. Fuhry, E. Demir, H. Ferhatosmanoglu, and M. Demirbas. 2010. Short text classification in twitter to improve information filtering. In *33rd international ACM SIGIR conference*, pages 841–842. ACM.

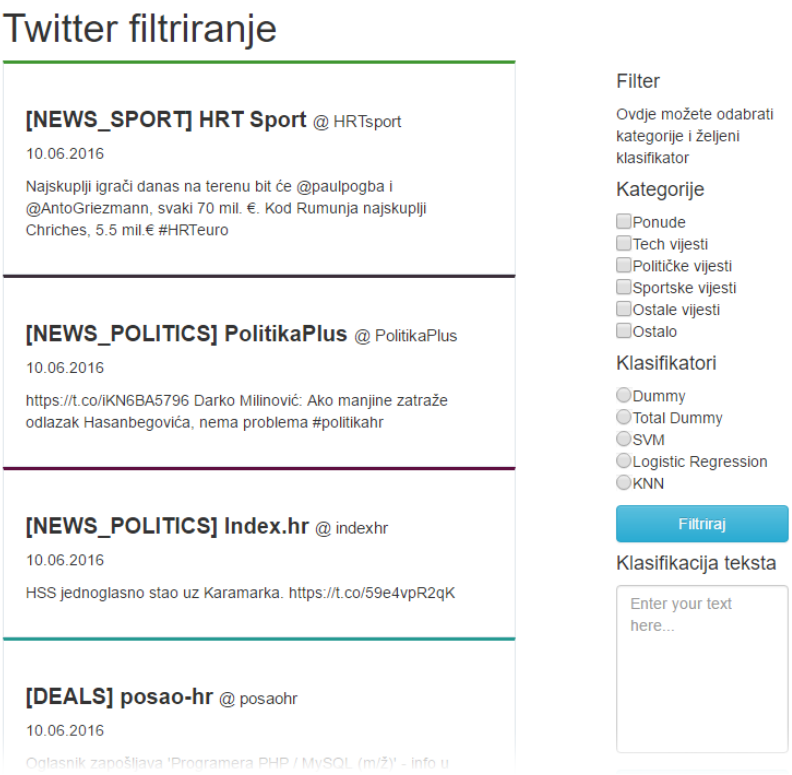


Figure 1: Web page of tweet filtering web application

Age, Gender, and Personality Profiling Based on Tweet Analysis

Lovre Mrčela, Marko Ratković, Ante Žužul

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
{lovre.mrcela, marko.ratkovic, ante.zuzul}@fer.hr

Abstract

The goal of this project was to profile an author by analyzing a set of texts written by them, and then determining the degree of each the Big Five personality traits. In addition, gender and age-group for each author are derived as well. The dataset was collected from twitter profiles, in English, Italian, Spanish, and Dutch languages. Approach was based on *tf-idf*, considering occurrences of trigrams. The implementation is done in Python programming language, using *nltk* and *sklearn* libraries.

1. Introduction

Author profiling deals with the problem of describing someone's personality, by means of extracting information from their writing style. Personality can be described using five traits (the so-called *"Big Five personality traits"*[1]), which are: extraversion, stability, agreeableness, conscientiousness, and openness to experience. Degrees of each trait range from -0.5 (indicating the total opposite), to 0.5 (indicating the exact match).

Provided with degrees of the five traits, it is possible to determine author's gender and age-group, via classification based on a model trained on previously labeled data. In this project, we used the linear SVC and Gaussian naive Bayes models for the classification into gender and age-group, and the linear regression with squared error measure for determining the degrees of personality traits. The training set we used was a collection of twitter posts in English, Spanish, Italian, and Dutch authors, ranging from around 35 authors in Dutch to 150 in English, each author's file containing about 100 posts. Of these four sets, English and Spanish are labeled with age-group, while the Italian and Dutch sets are not.

2. Approach

In this section, the methods of our approach are thoroughly explained. First, the preprocessing of input text is carried out, and weighted vector of trigrams (three consecutive letters) is obtained. Then, from preprocessed text some additional feature vectors, which were reasonably expected to be discriminative, are extracted. Finally, gender and age-group classification, and personality traits regression models are trained on extracted features, and final results are compared for various parameters.

2.1. Text preprocessing

For the rest of the process to be optimal, some sort of text preprocessing needs to be done on the raw input data. The input data we use is given in *xml* format, so the first step in preprocessing was to parse the actual sentences from the *xml* structure. When that is done, following steps are also applied:

- *urls* to other sites are substituted with an URL tag, and usernames (when referenced in replies) are substituted with a REPLY tag,
- all the text is converted to lower case because we don't deal with capitalization of words, only with words themselves;
- more than 3 repetitions of the same character are reduced to 3 letters, so that the words like *"coooool"* (5 repetitions) and *"coooooool"* (7 repetitions) are both treated as the same word, but distinctly from *"cool"*, because while we want to take repetitions into account, we would like to ignore the quantity of repeated characters (see the Section 2.2.);
- stop words (Bird, Klein, and Loper, 2015) for that particular language are deleted from the text, because they are considered insignificant for author profiling.

Each three consecutive letters are grouped into trigrams, and weighted vector of trigrams is obtained, using *tf-idf* weighting scheme. The extracted trigram weighted vector is used as one feature. More features are then extracted from preprocessed text, as described in the next subsection.

2.2. Additional feature extraction

In addition to weighted vector of trigrams, we decided to investigate some further characteristics of the written corpora, which were expected to be discriminative for the gender and/or age-group. Here is the list of considered additional features, and explanation for each of them:

- **number of emoticons:** the average number of emoticons used in a post (e.g. :) , <3; not considering each emoticon distinctly but all of them in total),
- **number of consecutive long repetitions of characters:** as mentioned before, we count only occurrences of repetitions longer than 3 characters, not the length of repetitions themselves - these repetitions most of the time do not have constant number of characters, even for the same author, or the same post, so it is a

[1] https://en.wikipedia.org/wiki/Big_Five_personality_traits

Text Analysis and Retrieval 2016: Course Project Reports (TAR 2016), pages 47–50, University of Zagreb, Faculy of Electrical Engineering and Computing, Zagreb, June 2016.

better approach to take into account only instances of repetitions;

- **number of replies:** the average number of replies to another user per each post,
- **number of hashtags:** the average number of hashtags per post,
- **number of exclamation marks:** the average number of exclamation marks (!) per post - each exclamation marks is counted, as we considered that, opposed to the consecutive repetition of letters, repeated exclamation marks do indicate author's stronger emotion to a some degree.
- **average length and standard deviation of posts:** we were inspecting average post length, as we presume it may also be correlated with age-groups;
- **average length and standard deviation of words:** as above, but considering just words.

It was expected for some of the features to be present in a greater degree in some subpopulations compared to the other (i.e. younger vs. older, male vs. female). The obtained results with respect to each feature are shown in the section 4..

The final feature set was obtained by selecting *n* best features, where *n* is also hyperparameter that needs to be optimized as well as the model. We select *n* best features using the ANOVA F-value[2].

2.3. Gender and age-group classification

For the gender and age-group classification subproblem, following approaches were considered:

- Logistic Regression
- Naive Bayes Classifier
- Decision Tree Classifier
- Random Forest Classifier
- SVC (using *rbf*, linear, poly- and sigmoid kernels)

The best results for age-groups were obtained using SVC with linear kernel, and for binary classification of gender, the Gaussian Naive Bayes.

2.4. Personality traits regression

For the personality traits regression, following approaches were considered:

- Linear Regression
- Decision Tree Regressor
- Random Forest Regressor
- SVR (using various kernels)

[2] `http://scikit-learn.org/stable/modules/generated/sklearn.feature\_selection.SelectKBest.html`

After testing each method, the best results turn out to be obtained by using SVR and linear regression.

All of these models are implemented in *sklearn* library, which we are using in our project solution.

3. Testing

Due to the lack of access to the official testing dataset (because of an ongoing competition), the official training dataset was divided across the authors into a subset for training (70%) and a subset for testing (30%).

From the above mentioned models, optimal model and hyperparameters were selected by using *10-fold* cross-validation on training set. Criterion for classifier (which was also used in PAN contest (Rangel et al., 2015)) was accuracy score. Aside from accuracy, we also used precision, recall, and F1 measures (micro and macro, for multi-class classification) (Rangel et al., 2014). For the regressor, we used the *root-mean-square error* for each of five traits individually:

$$\mathrm{RMSE}_i = \sqrt{\frac{\sum_{k=1}^{N}\left(\hat{\theta}_k^{(i)} - \theta_k^{(i)}\right)^2}{N}},$$

where $\hat{\theta}_k^{(i)}$ is a degree of i-th trait for k-th user estimated by our model, similarly $\theta_k^{(i)}$ is the given, real degree, and N is the number of users (of course, only users from the test set are taken into account when measuring the error).

In the absence of official testing dataset, it was obligatory to set baseline score as a referent measure. The baseline was set by using dummy models. Baseline classifier always gives the most frequent class, and baseline regressor always outputs the mean value. Thus, achieved results can be put into a more real perspective.

4. Results

Further observation of additional features values shows there is a correlation between some features and age-groups and/or gender of the author. This can be seen in tables 1 (age-groups) and 2 (gender). For example, average post length tends to be longer for older users than for younger users, for both the English and Spanish corpora. Number of user replies in average is also greater for older users than for younger users, for both languages. Average number of hashtags per posts seems to slightly increase towards older users in English corpus; however it is not in linear correspondence with age in Spanish corpus.

Also, there is correlation between the average number of emoticons per posts considering the gender of user: in the English, Italian, and Dutch corpora, female users in average tend to use up to three times as many emoticons than male users. However, in Spanish corpus the situation is reversed: male users in average tend to use in more emoticons than female users. Same goes for number of exclamation marks: female users in all languages use in average more exclamation marks than male users.

Precision, recall, F1 micro score, and macro score measures for each language are shown in tables 3 (age-groups), 4 (gender). The *root-mean-square error* measure for the personality traits can be seen in table 5.

Table 1: Overview of additional features values for each age-group, per language.

Language	English				Spanish			
Age-group	18-24	25-34	35-49	50-XX	18-24	25-34	35-49	50-XX
Post length	60.714	85.853	86.680	93.753	75.728	85.246	92.804	101.991
Post length deviation	29.656	29.401	29.532	32.192	31.377	31.234	30.719	29.200
Word length	4.908	5.984	6.312	6.013	4.935	5.295	5.647	5.409
Word deviation	3.493	4.726	5.088	4.452	3.356	3.882	4.230	3.963
Emoticon count	0.057	0.064	0.046	0.038	0.135	0.104	0.053	0.030
Hashtags	0.127	0.658	0.267	0.514	0.168	0.340	0.259	0.231
Character repetitions	0.040	0.012	0.017	0.003	0.065	0.022	0.030	0.022
Exclamation marks	0.137	0.207	0.195	0.527	0.183	0.244	0.257	0.276
User replies	0.492	0.540	0.632	1.293	0.579	0.715	0.818	0.854

Table 2: Overview of additional features values for gender, per language.

Language	English		Spanish		Italian		Dutch	
Gender	Female	Male	Female	Male	Female	Male	Female	Male
Post length	76.786	77.222	86.030	86.949	91.555	87.513	77.442	77.239
Post length deviation	29.711	29.764	30.767	31.131	32.908	30.594	29.574	30.829
Word length	5.529	5.718	5.328	5.281	5.898	6.153	5.255	5.229
Word length deviation	4.187	4.385	3.916	3.786	4.202	4.705	3.489	3.476
Emoticons count	0.075	0.039	0.082	0.102	0.214	0.072	0.119	0.072
Hashtags	0.380	0.394	0.357	0.190	0.540	0.700	0.424	0.120
Character repetitions	0.026	0.020	0.041	0.025	0.008	0.006	0.026	0.016
Exclamation marks	0.275	0.133	0.252	0.221	0.228	0.168	0.271	0.121
User replies	0.641	0.548	0.745	0.698	0.725	0.556	0.647	0.909

Table 3: Overview of results of age-group classification per language. Baseline scores are given for comparison. Best results are in bold (the higher, the better).

Language	Accuracy	Precision	Recall	$F1^{(micro)}$	$F1^{(macro)}$
English	**0.782**	**0.717**	**0.640**	**0.652**	**0.782**
English (baseline)	0.326	0.081	0.250	0.122	0.326
Spanish	**0.933**	**0.975**	**0.900**	**0.924**	**0.933**
Spanish (baseline)	0.600	0.150	0.250	0.187	0.600

Table 4: Overview of results of gender classification per language. Baseline scores are given for comparison. Best results are in bold (the higher, the better).

Language	Accuracy	Precision	Recall	F1
English	**0.956**	**0.888**	**1.000**	**0.941**
English (baseline)	0.347	0.347	1.000	0.516
Spanish	**1.000**	**1.000**	**1.000**	**1.000**
Spanish (baseline)	0.400	0.400	**1.000**	0.571
Italian	**1.000**	**1.000**	**1.000**	**1.000**
Italian (baseline)	0.416	0.000	0.000	0.000
Dutch	**1.000**	**1.000**	**1.000**	**1.000**
Dutch (baseline)	0.454	0.454	**1.000**	0.625

Table 5: Overview of RMSE of personality traits regression per language. Baseline scores are given for comparison. Best results are in bold (the lower, the better).

Language	Extraversion	Stability	Agreeableness	Conscientiousness	Openness
English	**0.122**	**0.179**	**0.153**	**0.140**	**0.130**
English (baseline)	0.164	0.235	0.182	0.167	0.155
Spanish	**0.080**	**0.143**	**0.103**	**0.155**	**0.131**
Spanish (baseline)	0.123	0.220	0.149	0.211	0.183
Italian	**0.048**	**0.123**	**0.067**	**0.086**	**0.099**
Italian (baseline)	0.136	0.166	0.116	0.162	0.162
Dutch	**0.087**	**0.112**	**0.129**	**0.064**	**0.041**
Dutch (baseline)	0.137	0.197	0.155	0.115	0.116

5. Conclusion

Experimenting with various models and features, we obtained results similar to other published works (Rangel et al., 2015) (in our case, tested on reduced training set). Unfortunately, we were not able to test our solution on the official data due to the data not yet having been released.

The possible upgrade of this work would be researching approach of Latent Semantic Analysis, as it may further improve detection of author personal traits.

References

Steven Bird, Ewan Klein, and Edward Loper 2015. Natural Language Processing with Python

Francisco Rangel, Martin Potthast, Fabio Celli, Benno Stein, Paolo Rosso and Walter Daelemans 2015. Overview of the 3rd Author Profiling Task at PAN 2015

Francisco Rangel, Paolo Rosso, Irina Chugur, Martin Potthast, Martin Trenkmann, Benno Stein, Ben Verhoeven and Walter Daelemans 2014. Overview of the 2rd Author Profiling Task at PAN 2014

Unsupervised Text Segmentation Based on Latent Dirichlet Allocation and Topic Tiling

Mirela Oštrek, Luka Dulčić

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
{mirela.ostrek,luka.dulcic}@fer.hr

Abstract

In this paper we describe an unsupervised method for topical segmentation of text. This method represents text as sequence of semantically coherent segments using the Bayesian topic modeling approach and one of the recently developed text segmentation algorithms. We developed and evaluated this method on synthetic Choi dataset. After the initial dataset cleanup, Latent Dirichlet Allocation model is applied to it in order to predict a certain probability distribution over topics which are then used as topic vectors for Topic Tiling algorithm. The latter divides text in semantically coherent segments by calculating cosine similarities and depth scores between the neighboring topic vectors. Performance of this method is evaluated with both Pk and *WD* measure, and results are shown.

1. Introduction

Most people today are searching through digital repositories which contain a great number of documents such as web pages, articles, emails, forum and blog posts, and so on. Despite this information need, finding a relevant topic for a query is extremely difficult task, unless documents are manually annotated with topics (Alpaydin, 2014). Our aim is to do this annotation automatically and use it to divide a number of documents into semantically coherent units – paragraphs. This will prove to be helpful later on, when search engine tries to retrieve all relevant documents and to pinpoint a specific location in document which best suits user's query.

In order to tackle this problem, we decided to use Bayesian Latent Dirichlet Allocation (LDA) model to predict a probability distribution over topics and Topic Tiling algorithm to segment given text into paragraphs, in line with work of Riedl and Biemann (2012a; 2012b; 2012c; 2012d). We give a short overview of LDA in Section 3 and continue by explaining Topic Tiling in Section 4. Dataset preprocessing is illustrated in Section 2. In Section 5 we present evaluation metrics and our results. Section 6 concludes the paper.

2. Dataset

2.1. Choi Dataset description

The Choi dataset (Choi, 2000) is commonly used in text segmentation field. It is artificially generated from Brown corpus and consists of 920 documents. Each document consist of 10 segments. Document generation was performed by extracting snippets of 3–15 sentences from different documents from Brown corpus. Documents in Choi dataset are divided into 6 categories based on number of sentences in segments. Categories are 3–5, 3–11, 3–15, 6–8, 9–11, 12–15 where 3–5 indicates there are 3 to 5 sentences in each segment.

2.2. Preprocessing

Preprocessing of Choi dataset consists of standard tasks such as sentence segmentation, tokenization and stemming. Sentence segmentation was easy task because every sentence in Choi dataset is divided by new line character, additionally we needed to remove irrelevant sentences which were empty or consisted of only stop words and other irrelevant tokens. Task of sentence segmentation is essential for this project because Topic Tiling algorithm predicts segment boundaries based on similarities between sentences. Tokenization of sentences was done using standard nltk[1] tokenizer. Obtained tokens were then filtered using list of irrelevant tokens which includes standard english stop words list. In further analysis of Choi dataset and experimenting with different lists of irrelevant tokens we decided to also remove all digits, tokens which appeared in more than 95% of documents or appeared in only one document and some punctuations tokens which were present in dataset but were not covered with stop words list. This significantly improved performance. After filtering tokens are stemmed using nltk Porter Stemmer.

3. LDA

LDA in text processing is an application of the Bayesian approach, namely topic modeling (Blei et al., 2003; Alpaydin, 2014). It serves its purpose for our method as it outputs probability distribution over topics for each sentence of given text. Topic distribution is assumed to have Dirichlet prior as in:

$$\text{Dirichlet}(\theta|\alpha) = \frac{\Gamma(\alpha_0)}{\prod_i^K \Gamma(\alpha_i)} \prod_i^K \theta_i^{\alpha_i - 1}, \qquad (1)$$

where $\theta = [\theta_1, \ldots, \theta_K]^T$ and $\alpha_0 = \sum_i \alpha_i$. In equation (1) K is the number of topics and θ denotes probabilities that correspond to the proportions of different topics. Prior allows us to calculate our prior beliefs in these proportions.

LDA is parametric model and its size is fixed, but we can make this model nonparametric by making the number of topics increase as necessary and adapt to data using a Dirichlet process (Alpaydin, 2014). However, in our

[1] http://nltk.org/

Text Analysis and Retrieval 2016: Course Project Reports (TAR 2016), pages 51–54, University of Zagreb, Faculy of Electrical Engineering and Computing, Zagreb, June 2016.

project we did not automatically adjust the number of topics to data. We chose several fixed numbers of topics and evaluated our method against them to see which one gives the most satisfying results.

LDA model in general works in the following way:

1. Generate topics in advance.
2. Assign topic to each word.
3. Check up and update topic assignments iteratively.

LDA iterates over the third step defined number of times. This is one of the parameters for fine–tuning LDA model. Other than that, LDA model has three more parameters and those are: α , β and number of topics K. Parameter α regulates the sparseness of topic–document distribution. Lower values result in documents being represented by fewer topics. Reducing β increases the sparsity of topics, by assigning fewer terms to each topic, which is correlated to how related words need to be, to be assigned to a topic (Riedl and Biemann, 2012c).

We conducted a research on three different ways of training our LDA model regarding semantically coherent units which are given to LDA as inputs. It is possible to send units such as sentences, paragraphs or even whole documents to LDA, all with goal to fit our model so that it can perform well on set of unseen documents. Inputs for predictions are always sentences because we need to obtain word–topic vectors, which are represented as probability distributions over topics for each sentence, and pass them on to algorithm for text segmentation.

4. Topic Tiling

With the aim of being able to segment the given textual document into semantically coherent units, we applied the Topic Tiling algorithm to it. In the previous section we discussed LDA model and how it outputs probability distribution over topics for each sentence of given text. Those outputs are actually called topic vectors and they serve their purpose as inputs for Topic Tiling algorithm. In contrast to some older algorithms for text segmentation such as Text Tiling, Topic Tiling algorithm does not use real words, but it uses topic distribution over words in sentence.

So, in Topic Tiling algorithm, each sentence of a text is represented by a topic vector. Neighboring topic vectors are then compared in terms of similarity. We decided to use cosine similarity because it is efficient for our task and it is not computationally too expensive. The next step is calculating depth scores d_p with the following expression:

$$\mathrm{d_p} = \frac{1}{2}(hl(p) - c_p + hr(p) - c_p), \tag{2}$$

where $hl(p)$ denotes highest peak on the left side of the current depth score point p, $hr(p)$ denotes highest peak on the right side of the p, and c_p is cosine similarity score calculated for p (Riedl and Biemann, 2012c). Figure 2 shows highest left and right peak for certain local minimum. Finally, depth scores are searched for M local minimums, and boundaries are set between topically different segments of text – paragraphs. Number of paragraphs M can be chosen in two different ways:

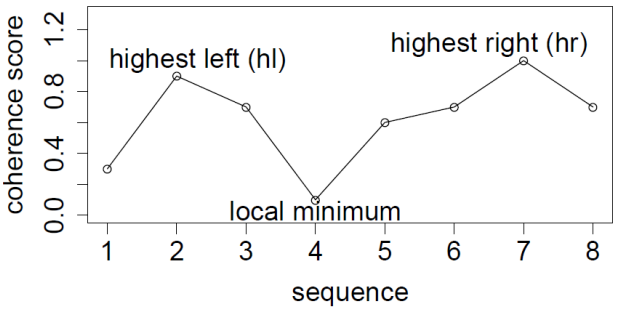


Figure 1: Illustration of the highest left and the highest right peak according to a local minimum (Riedl and Biemann, 2012c).

1. Number is fixed according to developer's "intuition" or already known fact.
2. Number is "calculated" according to certain condition performed on depth scores.

First case is commonly used if we already know how many thematic paragraphs textual document should consist. Second case counts all depth scores which satisfy the following condition as boundaries:

$$\mathrm{d_p} > \mu - x \cdot \sigma. \tag{3}$$

Condition (3) must be met for depth score to be marked as boundary between two paragraphs. For that specific purpose, standard mean μ and standard deviation σ are calculated using depth scores as data. In that way, a threshold for depth score being marked as boundary is defined and number of paragraphs is selected dynamically. In condition (3) there is also x variable present. In our research, we chose a few values for x ranging roughly from 0 to 1 to see which is the optimal value of x for dynamical segmentation. Results are shown in the next section.

5. Performance evaluation

5.1. Pk and *WD* measure

Early articles on text segmentation (Hearst, 1994) used precision and recall for evaluating segmentation models. These methods are nowadays considered inappropriate for this task because the distance between false positive segment boundary and correct one is not considered at all. For this reason Pk (Beeferman et al., 1999) and *WD* (Pevzner and Hearst, 2002) measures were developed. Descriptions of Pk and *WD* measures below are presented here following the description in (Riedl and Biemann, 2012c).

Pk measure uses a sliding window of k tokens which is moved over the text to calculate segmentation penalties. First we generate pairs: $(1, k), (2, k+1), (3, k+2), ..., (n-k, n)$ where n is the length of the document. Then for each pair (i, j) it is checked whether positions i and j belong to same segment or not. This is done separately for estimated and gold standard boundaries. If the gold standard and estimated segments do not match, a penalty of 1 is added. Finally error score is computed by normalizing penalty score by the number of pairs $(n - k)$, which produces number

between 0 and 1. A score of 0 indicates perfect match between gold standard and estimated boundaries. The value of parameter k is usually set to half of the average segment length, given by the gold standard.

WD measure is according to its authors Pevzner and Hearst (2002) enhancement over Pk measure, where drawback of Pk is that it is unaware of number of segments between pairs (i, j). Pk and *WD* measures are very similar, both use sliding window of k tokens. First step in *WD* is same as in Pk, dividing document into $(n-k)$ pairs. Then, for each pair (i, j), number of segments between position i and j is counted, separately for gold standard and estimated segments. If count for gold standard and estimated segments does not match, a penalty of 1 is added. Finally error score is obtained by normalizing penalty score by the number of pairs $(n - k)$ which produces number between 0 and 1. A score of 0 indicates perfect match between gold standard and estimated boundaries.

In practice, Pk and *WD* measures are highly correlated. For this reason we used only Pk measure for validation of model, to reduce the overhead of parameter grid search, and used both measures for testing model.

5.2. Cross validation

First of all, it is important to note that we were unable to conduct a proper cross validation which is usually mandatory for methods with lot of parameters like this one. Main reason for this is because training of LDA model is very computationally demanding, with certain parameters set, it can take up to 30 minutes. Since we had no high computation hardware available, this was considerable limitation. Because of the latter reasons we were forced to reduce parameters grid search space and to even neglect some parameters for which we used recommended values (Riedl and Biemann, 2012c).

Following parameters are subject to optimization in this segmentation model:

- K: Number of topics used in the LDA model. Commonly used values vary from 50 to 500.
- α : Document topic prior in LDA model. Recommended value is 0.1.
- β : Word topic prior in LDA model. Recommended values are 0.1 or 0.01.
- i : Inference iterations in LDA model. Recommended value is 70 to 120.
- x : Multiplier of standard deviation in (3). Commonly used value is 0.5.

Parameters β and i were not part of grid search. Recommended values $\beta = 0.01$ and $i = 70$ were used. Grid search included parameters K, α and x with following values:

- K: $\{60, 80, 100, 120, 130, 140, 150, 160\}$.
- α : $[0.1, 1]$ with step of 0.1.
- x : $[0.1, 0.7]$ with step of 0.1.

Table 1: Segmentation results on validation set, for simplicity only scores regarding x parameter are shown with respect of optimal parameters K and α.

K=150, α=0.1	Pk score
x = 0.1	0.093
x = 0.2	**0.087**
x = 0.3	0.095
x = 0.4	0.104
x = 0.5	0.108
x = 0.6	0.122
x = 0.7	0.143

Beside these parameters, we can also consider LDA input as another parameter. We can input documents, segments or sentences into LDA as mentioned in article before. Again, this parameter was not included in grid search because it would extend grid parameter space considerably. Instead we estimated performance on smaller subset of documents (100) with fixed parameters (K=100, α=0.1, β=0.01, i=70, x=0.5). Input consisting of segments as basic units significantly outperformed other two input types, therefore we used segments as basic input units for model evaluation.

Cross validation was performed with simple train–validation–test split of dataset with ratios of 60:20:20 which indicate percentage of each set. We were unable to use common method of k–fold cross validation because of its high computational demand. Table 1 shows segmentation results on validation set.

5.3. Results

Cross validation showed that optimal parameters are: K=150, α=0.1 and x=0.2. Pk measure score for optimal model on test set is 0.09 and *WD* measure score is 0.1. These results may be suboptimal because of limitations we have faced when conducting model selection. Also these results may vary +-0.02 because of the stochastic nature of LDA model and train–validation–test dataset split. Figure 2 shows segmentation of one randomly chosen document from test set.

6. Conclusion

Throughout this article we deal with one of the unsupervised methods for text segmentation which is based on the LDA topic model and Topic Tiling algorithm. In the previous section, we carefully examined obtained results of our method on Choi dataset. The performance that ensued from cross validation process was fairly satisfying, so we deem this state of the art method to be worthy of its label, but we also imply the need to test it out on sets of real world textual documents. By doing that it would be possible to make proper parameter adjustments and gain new clues on possible correlations between them.

References

E. Alpaydin, 2014. *Introduction to Machine Learning*, pages 449–450, 480–482. MIT Press.

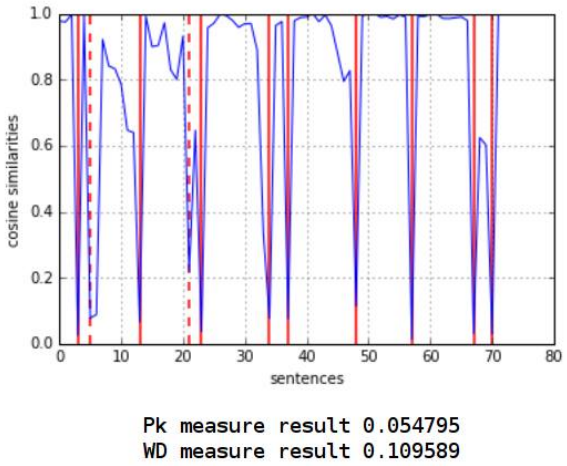


Figure 2: Plot of document segmentation. Blue graph shows cosine similarities between sentences, solid vertical lines indicate correct boundaries and dashed lines indicate false boundaries.

D. Beeferman, A. Berger, and J. Lafferty. 1999. Statistical models for text segmentation. pages 177–210.

D. M. Blei, A. Y. Ng, and M. I. Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, pages 993–1022.

F. Y. Y. Choi. 2000. Advances in domain independent linear text segmentation. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 26–33, Seattle, WA, USA.

M. A. Hearst. 1994. Multi-paragraph segmentation of expository text. *In Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, pages 9–16.

L. Pevzner and M. A. Hearst. 2002. A critique and improvement of an evaluation metric for text segmentation. *Computational Linguistics*, pages 19–36.

M. Riedl and C. Biemann. 2012a. How text segmentation algorithms gain from topic models. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2012)*, Montreal, Canada.

M. Riedl and C. Biemann. 2012b. Sweeping through the topic space: Bad luck? roll again! In *Proceedings of the Joint Workshop on Unsupervised and Semi-Supervised Learning in NLP held in conjunction with EACL 2012*, Avignon, France.

M. Riedl and C. Biemann. 2012c. Text segmentation with topic models. In *Journal for Language Technology and Computational Linguistics (JLCL)*, volume 27, pages 47–70.

M. Riedl and C. Biemann. 2012d. Topictiling: A text segmentation algorithm based on lda. In *Proceedings of the Student Research Workshop of the 50th Meeting of the Association for Computational Linguistics*, Jeju, Republic of Korea.

Offensive Text Detection With Naive Bayes Classifier

Jure Pajić, Mateo Šimonović, Rafael Slekovac

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
jure.pajic@fer.hr, mateo.simonovic@fer.hr,rafael.slekovac@fer.hr

Abstract

The aim of this study is to build a basic system that automatically classifies a piece of text as either offensive or not offensive, as well as impolite and polite. We evaluate two classification methods, one of which is Multinomial Naive Bayes and the other is Bernoulli Naive Bayes. Dataset used to create this project was derived from database of online comments written in Croatian language.

1. Introduction

In modern days with the spread of internet and posibility of creating virtual communities, people from all around the world can share their opinions, feelings, religious beliefs and other views. Even though one of the main human rights is freedom to speek and to express their opinion on certain topics, many people abuse that power and often insult and offend other people. Therefore, to protect people from being abused by other, these platforms need to have ways of preventing users from writing offensive and impolite texts. Furthermore, some other fields of study that would benefit from automatic detection of offensive text are sentiment analysis, autorship attribution, automatic censorship and others.

Our goal was to solve the task in the best way possible implement two classification methods. These methods are Multinomial and Bernoulli Naive Bayes. Goal of our study is to develop system which will in real time show user if the text he is writing is offensive or not and if it is polite or not. In order to make our program be as precise as possible we used an annotated dataset in Croatian by the TAs.

A brief overview of a theory behind Naive Bayes and Multinomial and Bernoulli document model are given in Section 2. In Section 3. we provide information about dataset we used to develop solution. Section 4. describes preprocessing of dataset, while Section 5. provides information about created classifiers. Futher on, in Section 6. we will show classification of previously mentioned classifiers. Interactive application that was created for this project is described in Section 7. Finally, Section 8. shows our conclusion.

2. Theory

2.1. Naive Bayes

When it comes to machine learning and text analysis, Naive Bayes classifiers are a family of simple probabilistic classifiers based on applying Baye's theorem with naive independence assumptions between the features. This model is a probabilistic generative model formally defined as follows:

$$h(x) = \frac{P(x|y)P(y)}{\sum_y^{'} P(x|y)P(y')}$$

where $P(y)$ are prior class probabilities, computed as MLE and $P(x|y)$ are class likelihoods, factorized into:

$$P(x|y) = P(x_1|y)P(x_2|y)...P(x_n|y).$$

If x_i is a discrete feature, the factor $P(x_i|y)$ is computed as the fraction of instances with value x_i among those labeled as y. In order to avoid zero-probabilities, we use Laplace smoothing described as follows:

$$P(w_i|C_j) = \frac{N(W = w_i, C = c_j) + 1}{N(C = c_j + K_i)}$$

where K_i represents number of values of feature w_i.

As any other method, Naive Bayes has both positive and negative characteristics. Some of the positive characteristics are that it is simple to train, it can work with various types of features, it gives probabilistic outputs and there is no need for model selection because it has no hyperparameters. Since this is a linear model, sometimes it's not powerful enough to model complex feature interaction and this is one of it's weaknesses.

2.2. Multinomial document model

At each position k in a document, any of the words w_i from the vocabulary can occur. We take W_k and say that it is categorical variable. If and only if w_i occurs at position k, then we can say that $W_k = w_i$. Next to model the probability of word w_i occuring at position k in a document from class c_j we use the following formula:

$$P(W_{k_1} = w_i|C = c_j) = P(W_{k_2} = w_i|C = c_j)$$

Using this, as a result we get one multinomial variable W for all word positions which tells us how many times w_i has occured in the document. Using previously shown formulas we can get estimation:

$$P(w_i, c_j) = \frac{N(W = w_i, C = c_j) + 1}{N(W, C = c_j) + |V|}$$

which gives us fraction of times in which word w_i appears in documents c_j.

2.3. Bernoulli document model

Presumably, each word w_i taken from vocabulary can either occur or not occur in the document at all. If we take W_i as

Text Analysis and Retrieval 2016: Course Project Reports (TAR 2016), pages 55–57, University of Zagreb, Faculy of Electrical Engineering and Computing, Zagreb, June 2016.

binary variable, er can then say that $W_i = 1$ if and only if word w_i occured in the document. Next on, to model the probability of word w_i occuring in a document from class c_j we use:

$$P(W_i = 1, c_j)$$

Using that, we can move on and make estimation:

$$P(w_i, c_j) = \frac{N(W_i = 1, C = c_j) + 1}{N(C = c_j) + 2}$$

that represents fraction of documents from class c_j in which the word w_i occurs. As we can see from previous definitions, number of occurences is completely ignored.

3. Dataset

For this project we've been provided with a database with online comments. The text in the dataset is written in Croatian. It consists of 1000 examples annotated with classes for politeness and offensiveness.

3.1. Dataset data relation

In table 3.1. we will show the relationship between classes. In this part we are using preprocessed dataset because if we would use original dataset than we would get false results. We will talk more about preprocessing in the next section (4.).

Table 1: Table showing intersection between classes. Column total shows union of two sets.

	Total	Intersection	Ratio
Polite-impolite	5351	986	18.4%
Offensive-not offensive	5351	1027	19.2%
Polite-offensive	5006	1428	28.53%
Polite-not offensive	4732	3916	82.76%
Impolite-offensive	2600	1467	56.42%
Impolite-not offensive	4956	1323	26.68%

4. Preprocessing

Since the dataset we've been provided with has a lot of grammatical errors, localizations, non character words like emojis, word cases, plurals etc. we needed to preprocess our dataset in order for it to be usable. In preprocessing we first tokenize our dataset, set all characters to lowercase and remove non character words and stop words. We process the remaining words with lemmatization and after that with stemming. Some words are exceptions to this rule and we will cover that later in Section 4.3..

4.1. Lemmatization

We are using package hunspell (Crawler lib 2016) in c# for lemmatization. For lemmatization. Hunspell provides functions like: spell checking, spell suggestion, word lemmatization etc. With this tool we process all words and check spell correction of the entire dataset.

4.2. Stemming

Since the hunspell tool is not perfect because it doesn't reduce the same words to their unique form, we continue with word stemming. For stemming we are using the Croatian stemmer (Pandžić 2007). With stemming process we reduce the error of multiple versions of the same word.

4.3. Exceptions

By analysing the dataset we found out that around 2700 words were grammatically incorrect. We applied automatic spell correction on those words. By manually inspecting all those 2700 automatic corrections we found out that around 200 words couldn't be corrected and only 600 corrections were actually correct. This means that around 1900 words (70%) were incorrectly corrected, for example "Batman" corrected into "bankomat". With those manually separated words we created a list of exceptions. Those words are only going through stemming process.

5. Classifiers

For text classification we have created two classes MultinomialNB and BernoulliNB from the ground up. For this project we needed maximal performance during text prediction and that is the main reason why we used our own code and not pre-existing tools.

5.1. Input

For fitting, the classifier is provided with two lists: texts and classifications. Every element of the list texts contains words separated with only one white space. In order for classifier to work well, input texts must be preprocessed.

5.2. Data structure

The classifier contains a set of words contained in list texts. Individual words are separated in two dictionaries: bag of words of class one and bag of words of class two. Those dictionaries contain the number of word occurrences. The dictionary is used to minimize data retrieval time. Classifiers also store information about the total number of words contained within each class and the number of documents belonging to each class.

5.3. Classifier learning

During fitting stage, the classifier iterates through the list of texts and separates words according to text classification. We could have stored dataset in memory as it is, but then we would suffer performance issues during prediction stage. It is important to mention that words are stored in their original form and not in form of N $\times$ M matrix (binary code) where N is number of documents and W is number of words set. The reason for doing this is still performance.

6. Evaluation

In the following section we will show evaluation of our classifiers.

6.1. Evaluation method

The classifier is evaluated with cross validation method. The ratio of training set and test set is 9:1. Cross validation is done 100 times. In each training-testing session F1

score is calculated. Average of all F1 scores is used as a final evaluation result.

6.2. Results

In the following table we listed our evaluation results. It is important to mention that these results came from fitting on database which is heavily processed, which is the results can not be 100% reliable.

Table 2: Evaluation F1 scores for Bernoulli and multinomial variants of naive Bayes

Algorithm	Offensiveness F1	Politeness F1
Multinomial NB	0.83	**0.84**
Bernoulli NB	**0.80**	0.79

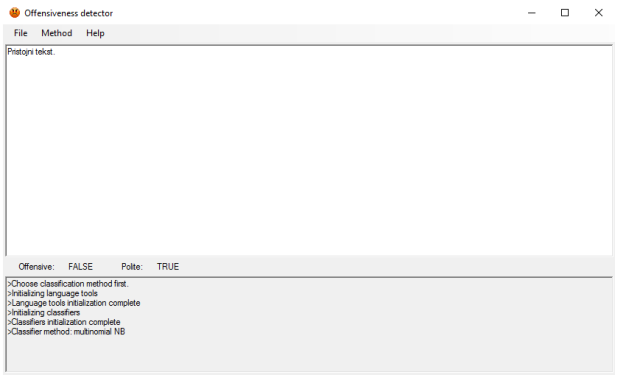


Figure 1: Classification example.

7. Conclusion

At the end of this project we conclude that our classifier for offensiveness and politeness works really well. Even with very bad database in terms of grammatical correction and words variety we can get good results by applying some manual work. Standard probabilistic methods have proven themselves to be quite good in this particular task.

Acknowledgements

We would like to thank our professor and assistants for giving us support and good pieces of advice throughout this project.

References

Crawler lib. 2016. Hunspell for c sharp. http://www.crawler-lib.net/nhunspell Spell Check Hyphenation Thesaurus.

Ivan Pandžić. 2007. Stemmer for croatian. http://nlp.ffzg.hr/resources/tools/stemmer-for-croatian/ A simple rule-based stemmer for Croatian.

Cristian Danescu-Niculescu-Mizi , Moritz Sudhof , Dan Jurafsky , Jure Leskovec , Christopher Potts. 2013. Acomputational approach to politeness with application to social factors. http://aclweb.org/anthology//P/P13/P13-1025.pdf.

Information Retrieval for Croatian Economics Library

Petra Rebernjak, Marin Oršić, Željko Findak

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
{petra.rebernjak,marin.orsic,zeljko.findak}@fer.hr

Abstract

In this project we've built an information retrieval application for a collection of economics books. We covered different aspects of satisfying information needs, providing users with an application capable of searching through dozens of books fast and accurate, along with providing summarized chapters. Building with modularity on mind, our application can be extended to the level of a production system.

1. Introduction

Information explosion is well known phenomenon of 21st century. The amount of online information and knowledge grows rapidly. Still, the source of the knowledge lies in the books. It would be useful to search the full text of books as easy as to surf the World Wide Web. Croatian students, entrepreneurs and scientists would be able to get first-hand information and a peek into the book based on their information need, all of which would possibly encourage them to purchase the whole book.

Firstly, our task was to implement and evaluate information retrieval system that does full text indexing and search of economics collection of books in Croatian language. Information retrieval (IR) is an activity of obtaining information resources relevant to an user's information need from a collection of information resources (Manning and Schütze, 1999). Information need is an individual or group's desire to locate and obtain information to satisfy a conscious or unconscious need (Manning, 2008). The field of information retrieval has moved from being a primarily academic discipline to being the basis underlying most people's preferred means of information access. Nevertheless, in recent years, a principal driver of innovation has been the World Wide Web, unleashing publication at the scale of tens of millions of content creators.

Secondly, the system needs to provide an automatic summary of every book or chapter in the database. Summary is a concise representation of a document's content to enable the reader to determine its relevance to a specific information. Document summarization is useful in cases where documents, such as books, are very long.

We evaluated retrieval effectiveness of search engine using average precision and recall evaluation metrics.

Section 2. gives an overview of related work, Section 3. covers the implementational aspects of the project and Section 4. describes how evaluation was made and provides results of the system evaluation.

2. Related work and inspiration

Probably the most popular information retrieval system of similar kind is Google Books.[1] Google has scanned, converted to text using OCR and stored in database more than 25 million books provided through partnership with libraries, publishers and authors. When provided with query, Google Books allows users to view full pages from books in which the search terms appear, if the book is out of copyright or if the copyright owner has given permission. If book is still under copyright, user sees "snippets" of text around the queried search terms. This inspired us to expand and apply knowledge about IR on particular real world problem.

Our task was to build web application which will provide IR for economics books in Croatian language which were provided by MATE d.o.o.

3. Implementation

In Section 3.1. an IR system is introduced with description of it's features and configuration. Section 3.2. introduces a text summarizer for generating document summarizations. Lastly, Section 3.3. describes a web application for generating information retrieval test collections.

3.1. Information retrieval

As underlying IR system for our application we chose Elasticsearch.[2] It is a search server based on Apache Lucene.[3] Its usage is similar to NoSQL databases; communication is based on exchanging JSON document through REST API. We chose it because of its modern API and, among other notable features, is fairly extensible for Croatian language.

In order to obtain good search results, we had to preprocess our books. Our books were provided in EPUB format. Its advantage was easier parsing of text. However, resulting strings contain a lot of formatting (such as HTML tags), so we had to address this problem accordingly.

Elasticsearch does not support Croatian language natively, so we decided to preprocess it using three different ways. Initially, we decided to use Hunspell and lemmatization. [4] Hunspell is a spell checker and morphological analyzer designed for languages with rich morphology and

[1] https://books.google.com

[2] https://www.elastic.co/products/elasticsearch

[3] https://lucene.apache.org

[4] https://hunspell.github.io

Text Analysis and Retrieval 2016: Course Project Reports (TAR 2016), pages 58–60, University of Zagreb, Faculy of Electrical Engineering and Computing, Zagreb, June 2016.

complex word compounding and character encoding, originally designed for the Hungarian language. We used public domain Croatian dictionary, which is used in variety of open-source tools, like Firefox and LibreOffice. [5] [6] Advantages of using dictionary based tool is that sometimes algorithmic stemmers can stem words in a strange way because they don't know any of the underlying language. In Hunspell, dictionary is used to map words to its origin. Because of this, quality of the preprocessing is directly related to the quality of the dictionary that we use. Disadvantage is that Hunspell will only be able to stem words it has in the dictionary.

On second try, we used a simple rule-based stemmer for Croatian by Natural Language Processing group at Faculty of Humanities and Social Sciences. Building algorithmic stemmer for languages with rich and complex dictionary is quite challenging, however, our results are still comparable with Hunspell based preprocessing. Elasticsearch doesn't have interface for custom stemmers, so we had to manually apply stemmer on our library and build index. Text is first broken into words, punctuation is removed and then each word is stemmed. After that we join it and supply to Elasticsearch as JSON document. Similar preprocessing is applied to user queries.

As another solution for comparison, we used out of the box configuration provided by Elasticsearch, which included only basic settings (Gheorghe and Lee Hinman, 2015). Text is first lowercased, then transfered to a tokenizer as a string, which then processes the string to break it into individual words, or tokens (perhaps discarding some characters like punctuation), and emits a token stream as output.

3.2. Summarization

The target of the automatic text summarization is to reduce a textual document in order to create a summary that retains the pivotal points of the original document. The research about text summarization is very active and during the last years many summarization algorithms have been proposed.

Many types of summaries exists. In this paper we focus on extractive and generic summaries that retain only an unchanged subset of overall content of the particular document. We follow the basic idea of text summarization by extracting sentences that contain important terms. The text is represented as an unordered set of terms — so-called *bag-of-word* representation. Term frequency and its variants is a commonly used term-weight for summarization and information retrieval tasks. The term frequency f_i of term i, is defined as the number of times term i appears in all sentences.

The input is tokenized into sentences and words, and the term frequency map of the words is computed as described above. The frequency map is then filtered in order to remove stop words. [7] Finally, the sentences are ranked according to the frequency of the words they contain and the top sentences are selected for the final summary. Despite oversimplifying, we obtained satisfiable performance.

The evaluation of a text summarizer is not an easy task. Most of the methods require manually generated summaries which can cause number of problems: a need for the experts that are well acquainted with books, problems with task subjectivity and expensive time cost.

3.3. Relevance labeling

For the purpose of evaluating the IR system, we developed a web application for labeling relevances and later evaluation of retrieval effectiveness. Where standard test collections are widely available, we find that creating a test collection for domain-specific information needs and Croatian language may be more useful. An information retrieval test collection consists of:

- document collection;
- set of information needs: descriptions and queries;
- set of relevance judgements for each query-document pair: we used binary relevance judgement.

The developed system allows each person marking relevances to view the documents from the collection to have better insight in it's content. Furthermore, each marking can be corrected and resubmitted thus improving the quality of relevance judgements.

The marking pipeline is as follows:

1. For each query in the set of information needs, the user is asked which of the books from the document collections is relevant in their opinion;
2. The user submits his binary relevance judgements.

The application is easy to use, as the users marking relevances are able to give their relevance judgements using any device which has a Web browser. Moreover, relevance judgements are stored for each user separately, so the system keeps track of the progress the user has made, hence all of the work doesn't have to be submitted in a single session.

The collected data is stored in a relational database for further processing. Using collected relevance judgements, evaluation of an IR system is improved:

- different settings used can be evaluated against a unique testing set;
- comparison against other system is made possible.

Using this data, evaluation metrics may be measured for the IR system, as described in Section 4. The developed Web application also features an efficient workflow for importing a different document collection and creating another set of information needs.

4. Evaluation

This section describes how the evaluation was made and gives an overview of the results.

[5] https://www.mozilla.org/en-US/

[6] https://www.libreoffice.org

[7] https://bitbucket.org/trebor74hr/text-hr/src/tip/text_hr/std_words.txt?fileviewer=file-view-default

Table 1: Evaluation results for used indexes

Measure	Default search index	Hunspell analyzer	Croatian stemmer
Precision micro	**48.11%**	45.95%	46.53%
Recall micro	90.90%	**93.07%**	92.86%
F1 micro	**62.92%**	61.53%	61.99%
Precision macro	**63.73%**	53.43%	55.24%
Recall macro	88.37%	93.86%	**94.37%**
F1 macro	**74.05%**	68.09%	69.69%

4.1. Evaluation metrics

Using the gathered test collection, evaluation metrics may easily be produced. Each document may be relevant or irrelevant for a query, and it also may be retrieved or not (Büttcher et al., 2010). This produces a 2-by-2 confusion matrix.

The information retrieval system used 3 indexes, which represent three different approaches we made using Elasticsearch:

1. An out of the box system which uses default settings;
2. A system which uses the Hunspell analyzer;
3. A system which uses a stemmer and stop word removal for Croatian.

Evaluation was made using the aforementioned document collection and a set of 30 queries for which the relevance judgements were given using the Web application. People giving their relevance judgements have knowledge of the specific domain and were also the ones creating the query set.

4.2. Results

Evaluation results are shown in Table 1. The results show an increase in recall when using Hunspell or Croatian stemmer. Compared to default settings, a trade off is made by sacrificing precision. However, an increase in recall indicates that documents which are releveant for a specific information need are not left unretrieved.

Search queries which consist of words that individually have different meaning than the query itself produce the worst results, where queries which are domain specific and consist of one single word yield best results. A larger document collection to test on would give more precise evaluation results.

5. Conclusion

We obtained satisfiable performance of the text summarizer, despite its oversimplified implementation. Using more sophisticated models for assessing term and sentence importance, applying redundancy reduction and content ordering would very much improve current performance. Hence, proper multi-document summarization would be helpful in compiling more efficient browsing and searching and, potentially, clustering similar documents.

Elasticsearch turned out to be a powerful tool for IR, especially for it's speed. Using Elasticsearch's ability to have different configuration for each index, we were able to create different indexes and fill them using different methods and finally test them simultaneously.

Support for Croatian language in natural language processing is extremely meager. We are certain that with appropriate support, larger book database and richer query set, the overall system performance would be more satisfactory. This shouldn't be a problem, considering Elasticsearch's scalability potential.

Future work may include more advanced IR methods, such as query expansion. Also, relevance feedback (or pseudo-relevance feedback) may be implemented in the system to improve results. Having in mind that users who don't satisfy their information needs quickly (i.e., on the first page of results), evaluation metrics which take ranking into account, like P@k and R-precision, may be implemented to maximize.

Acknowledgements

We would like to thank MATE d.o.o. for providing us with the document collection, creating the query set and marking relevances.

References

Stefan Büttcher, Charles L. A. Clarke, and Gordon V. Cormack. 2010. *Information Retrieval: Implementing and Evaluating Search Engines*. MIT Press.

Radu Gheorghe and Matthew Lee Hinman. 2015. *Elasticsearch in Action*, volume 1. Manning Publications.

Christopher D. Manning and Hinrich Schütze. 1999. *Foundations of statistical natural language processing*, volume 2. MIT Press.

Christopher D. Manning. 2008. *Introduction to Information Retrieval*, volume 2. Cambridge University Press.

Offensive and Impolite Text Detection

Bruno Rosan, Jure Perović

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
{jure.perovic,bruno.rosan}@fer.hr

Abstract

This document provides insight into text sentiment analysis and classification system. Text is classified in two ways, as OFFENSIVE or NOT OFFENSIVE and as POLITE or IMPOLITE. Paper discusses methods and steps taken into developing system, from preparing data set, text preprocessing to feature extraction and classifier optimization. Dataset used consists of manually annotated forum user replies in Croatian language. At the end system is evaluated using the standard classification evaluation metrics.

1. Introduction

Politeness is a battery of social skills whose goal is to ensure everyone feels affirmed in a social interaction (Foley1997). While communicating through internet forums, one may find or write offensive or impolite messages. These messages can make online experience unpleasant for some users. To address these concerns, many user forums, blogs, and social networking sites apply several mechanisms to screen and filter offensive contents. However, most of the filtering happens using lexicon based approach. It implies that if the user's comments involve any profanity or offensive content, the system will search for the typed word or synonyms of the words in the repository built, and it will remove or redact the word on the forum.

Given the manually annotated set of various pieces of text, challenge is to create system that could learn from data set and determine politeness of any new piece of text. Since our data set consists of internet forum replies, there was no need for two-step classification approach. More precisely, subjectivity classification is not necessary because all pieces of text in data set are subjective opinions. We approach the problem as text classification problem with assumption that pieces of text are considered impolite if they contain impolite words. Solution would be to extract impolite words from data set and if new piece of text contains such words it is classified as impolite. Piece of text is considered polite if there is no such words. Problem with this approach is when new piece of text contains impolite words that were not present in training data set, it would be wrongly marked as polite. Our idea to solve problem of impolite words not present in train set is to learn model in word2vec vector space where vectors represent words. By using large number of words we can extract more impolite words based on the extracted set of impolite words from data and hopefully get positive results for problematic sentences. Same approach is used in offensive text classification.

Structure of the paper is as follows. Section 2. describes dataset used for evaluation and preparation of data for machine learning. Classification strategy and keywords extraction are discussed in section 3. Section 4 presents results of text classification and in Section 5 we summarize and bring the conclusion of this work.

2. Dataset

Dataset we used are manually annotated pieces of text with two sets of labels, one for politeness and other for offensive texts. Dataset contains 1000 different pieces of text in Croatian language. There are 813 pieces of text marked as polite and 187 marked as impolite. For offensive text classification we have 182 pieces of text marked as offensive and 818 as not offensive. We split the data in train and test sets in ratio 2:1 taking care that we have nearly same ratio of positive and negative examples in both sets.

2.1. Preprocessing

To represent every piece of text as list of words and to extract important words every example is put through some steps of natural language processing. We used sentence segmentation and tokenization, POS (part-of-speech) tagging and lemmatization options from preprocessing suite. After POS tagging, nouns, verbs, adjectives and pronouns are extracted as mainly those parts of speech make sentences impolite or offensive. As forum users often do not use diacritic characters we replaced them with corresponding characters.

2.2. Dataset difficulties

Our dataset is collection of internet forum replies and in some, words with mistakes are found. Words like that and few more from dataset do not match with words given in word2vec so we have to discard them. Pieces of text contain few sentences so single words do not appear many times in one piece of text. Because of that term frequency score for words is usually same and plays little part in keyword extraction.

3. Classification strategy

After all the preprocessing and data preparation steps we have dataset consisting of lists of lemmatized words. To learn model based on training subset of data set we approach as follow:

1. Compute TF-IDF score of every word in corpus.
2. Extract keywords, words with highest TF-IDF score.
3. Mark extracted words as positive or negative based on label given to piece of text where words occur.
4. Use machine learning model with word feature vectors as input.

Text Analysis and Retrieval 2016: Course Project Reports (TAR 2016), pages 61–63, University of Zagreb, Faculy of Electrical Engineering and Computing, Zagreb, June 2016.

Pieces of text in test set are classified as impolite if they contain word that machine learning model classifies as impolite. If all the words are classified as polite by model piece of text is classified as polite. Same approach is used for offensive texts.

3.1. Keywords

To find keywords in corpus TF-IDF score is computed for every word. Term frequency is the number of times term t occurred in a piece of text d. Inverse document frequency diminishes the weight of terms that occur very frequently in the document set D and increases the weight of terms that occur rarely. Term frequency combined with inverse document frequency is called TF-IDF score and calculated as $tfidf(t, d, D) = tf(t, d)idf(t, D)$. Term frequency $tf(t, d)$ is normalized as follows:

$$tf(t, d) = 0.5 + 0.5 \times \frac{f_{t,d}}{max\{f_{t',d} : t' \in d\}}$$

Inverse document frequency is usually calculated as:

$$idf(t, D) = \log \frac{N}{1 + |\{d \in D : t \in d\}|}$$

Where N is total number of documents. With TF-IDF score we can find keywords in given data set but information alone tells nothing about politeness of word. Computing the score for labeled pieces of text gives keywords for polite and impolite sets. Furthermore, we modify inverse document frequency factor with numerator as total number N of documents labeled as polite and denominator as $\{1 + |\{d \in D : t \in d\}|$ where D stands for all pieces of text marked as polite. We do so in order to distinguish between keywords in impolite set and words that make piece of text impolite. Since examples in data set contain few sentences and terms usually do not repeat in same example, term frequency is not that important, so emphasis on inverse document frequency is done by setting denominator as $\{0.001 + |\{d \in D : t \in d\}|$.

3.2. Classification

Classification for every piece of text is done by classifying every word in text by machine learning model. If the piece of text contains impolite or offensive word it is considered impolite. For classification we used k-NN algorithm with Croatian word2vec vectors.

Word2vec takes as its input a large corpus of text and produces a vector space, typically of several hundred dimensions, with each unique word in the corpus being assigned a corresponding vector in the space. Word vectors are positioned in the vector space such that words that share common contexts in the corpus are located close to one another in the space (Mikolov2013). We used already produced vector space.It contains 161410 word vectors. Each word is presented as 300-dimensions vector.

Classification with k-NN algorithm is done by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors. Choosing parameter k is done by cross validation as shown in table 1. We also used random forest classifier consisting of 10 decision trees that gives just slightly better results.

Table 1: Statistical scores for impolite text classification with different numbers of nearest neighbors in k-nn

k	$Precision$	$Recall$	$F_2 - score$
1	0.2222	**0.5573**	**0.4282**
3	0.2149	0.3770	0.3276
5	**0.3191**	0.2459	0.2577
7	0.2777	0.0819	0.0954

4. Results

Evaluation of the system is done through standard evaluation metrics. Precision, that gives percent of examples classified with correct class, recall and F-score. Recall is computed as percentage among examples marked as impolite or offensive by classifier and all impolite or offensive examples. F-score combines precision and recall and is calculated as:

$$F_\beta = (1 + \beta^2) \times \frac{precision \times recall}{\beta^2 precision + recall}$$

Since our goal is to determine offensive and impolite texts recall is of more value than precision. Parameter β in F-score is used to put weight on certain score, so we set it to 2 to emphasize recall.

Percentages calculated from classifier results are shown in tables 1 and 2 for impolite and offensive text classification respectively. Highest $F_2 score$ for impolite text classification amounts to 0.43 with Nearest neighbor classifier. We get similar results for offensive text classification with highest score of 0.49.

Table 2: Offensive text classification results.

k	$Precision$	$Recall$	$F_2 - score$
1	0.2146	**0.7096**	**0.4856**
3	0.2138	0.5967	0.4394
5	**0.3289**	0.4032	0.3858
7	0.2941	0.1612	0.1773

5. Conclusion

Main task was to create system that could detect offensive or impolite texts. To achieve that, system should efficiently solve multiple tasks, from natural language processing, calculating polarity of text to text classification using machine learning models. We were given manually annotated dataset to evaluate our solution.

We tried to solve the problem by calculating polarity of keywords in texts from the annotated dataset. Words marked as offensive or impolite would serve as seeds in vector space which we use for classification of new texts. First and main assumption made is that one impolite word makes whole piece of text impolite. For offensive pieces of text that may not be completely true but strangely score, using the same strategy, is slightly higher. That may point to that there is too many different offensive or impolite words and larger and more balanced data set would be needed for model to find them all. Regardless of relatively small data

set, nearest neighbor classifier gets too low scores in both classifications.

Further work should focus on deeper sentence analysis considering offensive text classification, as well as obtaining more examples for dataset and using other machine learning methods that could be more suitable for the problem.

References

Muhammad Abdul-Mageed and Mona Diab. 2012. A multi-genre corpus for modern standard arabic subjectivity and sentiment analysis. In Proceedings of LREC, pages 3907–3914.

Cristian Danescu-Niculescu-Mizil. 2015. A computational approach to politeness with application to social factors. Computer Science Department, †Linguistics Department †Stanford University, ‡Max Planck Institute SWS.

William Foley. 1997. Anthropological linguistics: An introduction.

Tomas Mikolov. 2013. Efficient estimation of word representations in vector space.

Shashank H. Yadav. 2015. An approach for offensive text detection and prevention in social networks. Department of Computer Technology, Yeshwantrao Chavan College of Engineering, Nagpur, India.

(Yadav, 2015) (Abdul-Mageed and Diab, 2012) (Foley, 1997) (Mikolov, 2013) (Danescu-Niculescu-Mizil, 2015)

NER in Croatian and Serbian Tweets Using Machine Learning

Arian Sibila

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
arian.sibila@fer.hr

Abstract

This paper presents our work on NER for tweets in Croatian and Serbian. NER for tweets is related to NER for literary language in that the same tools are used but additional processing steps are needed to achieve an acceptable F1 score. The reasons are that (1) tweets are often grammatically incorrect, and (2) working with Croatian and Serbian makes it more difficult because they are morphologically rich languages. Machine learning frameworks such as CRF allow the data to "speak for itself" and thus potentially obsolete rule based methods.

Table 1: CRF data set example

Label	Features
B-PER	LEMMA[0]=josip POS[0]=N
I-PER	LEMMA[0]=radošević POS[0]=N
NULL	LEMMA[0]=na POS[0]=S
NULL	LEMMA[0]=posudba POS[0]=N

1. Introduction

Named-entity recognition is a sub task of information extraction whose goal is to locate so called Named entities withing a corpus of a text, such as (1) person names, (2) organization names and (3) location names. The reason one might need NER on tweets specifically is to, for example, keep a track of the number of mentions as to come up with a popularity measure of a particular NE (e.g. a politician). Indeed, many private and public organizations have been reported to monitor targeted Tweeter streams.

The difficulties specific to Tweeter NER include:

(1) most tweets are shorter than 140 characters,

(2) tweets are not grammatically and/or syntactically correct,

(3) named entities are not always properly spelled and/or capitalized.

As a result of that, state-of the art tools for NER perform poorly when applied to Tweeter.

As previously reported by many researchers, CRF (Conditional Random Fields) is the technology of choice for the task at hand because it allows us to compactly model multivariate data with the ability to leverage a large number of input features for prediction. It is also true that an accurate conditional model can have much simpler structure and be quicker to train than a joint model (e.g. Hidden Markov Models). Therefore it makes sense to ask right away which features should we use to train our model. Table 1 shows an example of data set of a particular CRF tool called CRFsuite [3]. For our purpose we consider each line to correspond to a word in a tweet and proceed to generate such data set from manually annotated tweets. Different tweets are separated with a blank line.

2. Techniques

In machine learning, feature is an individual measurable property of a phenomenon being observed. Sequence labeling is a task of assignment of categorical labels to each member of the observed sequence. This task can be treated as a set of independent classification tasks, one per member of the sequence. However, accuracy is generally improved by making the classification decision dependent on the choices of nearby elements.

Typical NLP stages in a NER pipeline involve (1) parsing, (2) POS tagging, (3) lemmatization, (4) feature extraction, and (5) training the model/tagging the data.

We were given manually annotated set of tweets split into training set and testing set in ratio 8:2. Each word either has an explicit annotation tag (such as B-PER for beginning of a person entity) or is otherwise given a default annotation tag of NULL.

2.1. Parsing

Parsing is a task of recognizing an input and extracting information from usually with the intention of transforming the original input into a different language or a format. The input data was given as a rootless XML with one tweet per tag. First we create a parser in C++ (extract1.cc) to convert the given data set into one word per line with an empty line between tweets, a format suitable for the use with the scripting tools. Words starting with # or @ are considered tags and are ignored.

2.2. POS tagging

POS (part-of-speech) tagging is a task of assigning an each word a tag to describe its word category (noun, verb, etc.) as well as its inflection (number, case, gender, etc.) Examples of POS tags for Croatian language are given in Table 3.

The tool used for POS tagging is called hunpos tagger [2], originally developed for Hungarian language but is able to work with Croatian language as well with the addition of *model* file It is important that POS tagging be performed before lemmatization or, to put it another way, that lemmatization be performed on a POS-tagged input for accuracy.

Text Analysis and Retrieval 2016: Course Project Reports (TAR 2016), pages 64–66, University of Zagreb, Faculy of Electrical Engineering and Computing, Zagreb, June 2016.

Table 2: POS tags

POS tag	Description
N-msn	Noun, masculine, singular, nominative
Agpmsn	Adjective, masculine, singular, nominative
Vmr3s	Verb, masculine, 3rd person, singular
Sl	Adposition, locative

2.3. Lemmatization

Lemmatization is the task of converting a word to what is called a lemma or a base form of a word. Its main purpose is to reduce the language complexity that CRF has to deal with which is especially important for morphologically rich languages like Croatian. The tool used for lemmatization is called CST'lemmatizer [4], originally developed for Danish language but with a proper model can be used for Croatian.

2.4. Feature extraction

After lemmatization we have the following features to work with (table 3). Those features are combined into new features (i.e. composite features) in the following ways: (1) including individual features of previous and/or next word(s), (2) including joint features of previous and/or next words. (3) feature scaling. This is all done in a program called extract2.sh

2.4.1. Individual features

CRFsuite allows us to easily capture the context surrounding the word in question by using array-subscript notation, e.g. POS[-1]=N-msn would tell the CRFsuite to condition the classification decision on the event of previous (-1) POS tag being N-msn.

2.4.2. Joint features

Just like the individual features, we can incorporate joint features such as POS[-1]—POS[0]=Agpmsn—N-msn. This would tell CRFsuite to condition the classification decision on the event of the both previous and next POS tags being Agpmsn and N-msn respectively.

2.4.3. Feature scaling

CRFsuite allows us to assign weights to individual or joint features. For example, POS[0]=N-msn:2 would tell CRFsuite to condition the classification decision on the event of current POS tag being N-msn 2 times more heavily than it otherwise would.

2.5. Training the model/tagging the data

CRFsuite is designed to run on the output of extract2.sh. The training procedure can be tweaked by selecting (1) a training algorithm, (2) a number of iterations (3) a regularization factor. After training, a model file is created to be used by the same program but in a tagging mode, which means that the same data format is used and the same program extract2.sh is run twice, once on the training set and again on the testing set.

Linear-chain CRF is given by the two equations below:

$$p(y_{1:N}|x_{1:N}) = \frac{1}{Z} exp(\sum_{n=1}^{N}\sum_{i=1}^{F}\lambda_i f_i(y_{n-1}, y_n, x_{1:N}, n)) \quad (1)$$

$$Z = \sum_{y_{1:N}} exp(\sum_{n=1}^{N}\sum_{i=1}^{F}\lambda_i f_i(y_{n-1}, y_n, x_{1:N}, n)) \quad (2)$$

Z in the equation (2) is the scaling factor to make the result in (1) be a probability. N is the size of the sequence (i.e. number of members), F is the number of features, lambda is the weight of the i-th feature. Features are represented by the function f which in this example takes (hidden) labels from the pair of adjacent states

2.5.1. Training algorithms

Out of the available algorithms, lbfgs (Limited-memory Broyden-Fletcher-Goldfarb-Shanno) performs the worst and pa (passive aggressive) performs the best both in terms of speed of training and the quality of training.

2.5.2. Number of iterations

By default, the training terminates after finishing 100 iterations. The results can be improved slightly by increasing the maximum number of iterations to 200.

2.5.3. Regularization factor

By default, the regularization factor in various algorithms is set to 1.0 and is presented as complexity parameter (C, the inverse of regularization factor). This means that selecting lower value for C should make model simpler (i.e. prevents over-training.)

2.5.4. Precision, Recall, F1

Upon evaluating (tagging) the test set, CRFsuite [3] can be told via an option to calculate and output scores known as macro-average precision, recall and F1. The most important score of the above 3 is F1 because it aggregates precision and recall into a single value.

2.6. Additional features

Because the quality of the trained model and the score of the test data ultimately depends on the quality of the features and because CRFsuite can learn which features are not useful it is in our best interest to include as many features as we have.

2.6.1. Beginning-of-sentence

"BOS" feature is appended to the first word of the tweet to indicate that this is where the tweet begins.

2.6.2. End-of-sentence

"EOS" feature is appended to the last word of the tweet to indicate that this is where the tweet ends.

2.6.3. Gazetteers

Gazetteers are geographical dictionaries used in conjunction with an atlas or a map. For our purpose, those would be text files containing named entities one at a line. Next, we might query the gazetteer for a word frequency and use it as a feature.

Table 3: Features

#	Features
1	POS tag (e.g. N-msn)
2	Word lemma (e.g. radošević)
3	Capitalization state (1 if capitalized, 0 otherwise)
4	Alphanumeric state (1 if all letters, 0 otherwise)

Table 4: Results

Algorithm	Training time	F1
* CRF - lbfgs	65.960	0.774774774775
* CRF - l2sgd	44.590	0.775782155272
* CRF - ap	36.210	0.783868219256
CRF - arow	32.410	0.719372021306
HMM	0.637	0.682548476454

2.6.4. Shortened POS tag

Shortened POS tag is a feature that takes only the word category from the POS tag (the first letter of the POS tag). For example, N-msn would be converted to N. The idea being that the first letter might contain the most accurate information.

3. Hidden Markov Model

For the purpose of comparison and being complete, we repeat the process employing a different model - Hidden Markov model (HMM). Compared to CRF, HMM is simpler and, for this use, less capable. The main reason is that in HMM we are not able to use arbitrary features such as surrounding words, capitalization, grammatical case, etc., but instead we are limited to using only words as features.

4. Results

At the end we present the score on the test set for various training algorithms.

F1 score was calculated using exact matching of named entities.

As can be seen from table 4, the test scores are lower than expected for all the algorithms. That is to be expected because (1) we are dealing with tweets which are noisy and are not the regular (literary) language, and (2) the training data set is relatively small which makes the model prone to over-training.

Of the 5 algorithms, 3 have been identified as having sufficient capacity to learn the model (as witnessed by their train-scores.)

5. Conclusion

It is a well known fact that the errors in NLP pipeline compound from one stage to the next. This means that part of the error could be due to the lacking POS model and/or the lemmatizer. It might also be worthwile looking into rule based rewriting to bring tweets into more lexically correct form before CRF training.

6. Local references

[1] Agić, Ž.; Ljubešić, N.; Merkler, D. (2013.) Lemmatization and Morphosyntactic Tagging of Croatian and Serbian. In Proceedings of the 4th Biennial International Workshop on Balto-Slavic Natural Language Processing (BSNLP 2013). Sofia, Bulgaria, Association for Computational Linguistics, 2013, pp. 48–57.

[2] P. Halácsy, A. Kornai, and C. Oravecz. 2007. HunPos – an open source trigram tagger In: Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions, Association for Computational Linguistics, Prague, Czech Republic, 209–212.

[3] N. Okazaki (2007). CRFsuite: A fast implementation of conditional random fields (CRFs).[Online] Available: `http://www.chokkan.org/software/crfsuite/`.

[4] Jongejan, B., Haltrup, D. (2005). The CST Lemmatiser. Center for Sprogteknologi, University of Copenhagen (version 2.9)

[5] D. Nadeau, S. Sekine. A survey of named entity recognition and classification.

Link Analysis Algorithms (HITS and PageRank) in the Information Retrieval Context

Dario Smolčić, Deni Munjas

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
{dario.smolcic, deni.munjas}@fer.hr

Abstract

We implemented an information retrieval system that re-ranks the documents based on the analysis of between-document links with the link analysis algorithms PageRank and HITS. The information retrieval system was built using a binary vector-space retrieval model. Scores produced by vector space model were refined with the HITS and PageRank algorithms. The performance of the IR system both with and without PageRank and HITS was evaluated on the European Media Monitor test collection using standard IR metrics (R-Precision, Mean Average Precision, mean reciprocal rank).

1. Introduction

Information retrieval (IR) is the activity of obtaining information resources relevant to an user's information need from a collection of information resources. The focus of information retrieval in this paper are unstructured text documents. The classical problem in IR is the ad-hoc retrieval problem. In ad-hoc retrieval, the user enters a query describing the desired information. The system then returns a list of documents.

The biggest information retrieval systems are search engines like Google and Yahoo. Search engines are nowadays becoming necessity tools for information retrieval, learning and many other aspects of life. While different information retrieval systems may have different search models and principles, vector space model, PageRank and HITS are commonly used by most popular ones, and they can be combined together to improve the search performance.

2. Vector space model

In vector space model documents and queries are represented as vectors of index terms:

$$\mathbf{d_j} = [w_{1j}, w_{2j}, ..., w_{tj}]$$

$$\mathbf{q} = [w_{1q}, w_{2q}, ..., w_{tq}]$$

Where t is the number of index terms in the vocabulary. Since we are using binary vector space model, weights in vectors are binary numbers. Weights are computed according to the following formula:

$$w_{ij} = \begin{cases} 1, & \text{if document } \mathbf{d_j} \text{ contains term } i \\ 0, & \text{otherwise} \end{cases}$$

The relevance of the document for the query is estimated by computing some distance or similarity metric between the two vectors.

Some possible similarity metrics are cosine similarity and dice similarity. Some possible distance metrics are euclidean distance and Manhattan distance. In this project, we use cosine similarity and dice similarity.

$$Cosine(\mathbf{d_j}, \mathbf{q}) = \frac{\mathbf{d_j q}}{\|\mathbf{d_j}\|\|\mathbf{q}\|}$$

$$Dice(\mathbf{d_j}, \mathbf{q}) = \frac{2\|\mathbf{d_j q}\|}{\|\mathbf{d_j}\|^2 + \|\mathbf{q}\|^2}$$

The most relevant documents for the given query are the ones with the highest similarity score (Bašić and Šnajder, 2014).

3. Link analysis algorithms

Information retrieval content scores are not efficient for web usage due to web's massive size, therefore, popularity/importance scores are introduced. Popularity scores harness information from the immense graph defined by web's hyperlink structure. In our case we are not operating online on a website database, but we are performing information retrieval on a great number of documents and we did create links among them.

Hyperlinks among documents are interpreted as recommendations. Documents with more recommendations are more important than documents with less recommendations. When assessing importance of a document, it is necessary to take into account the importance of the recommender. Recommendations from more important recommenders are worth more while the overall number of recommendations issued by the recommender also matter (Bašić and Šnajder, 2015).

3.1. PageRank

Let $\mathbf{H}$ be the row normalized Web graph adjacency matrix.

$$\mathbf{H_{ij}} = \begin{cases} 1/|P_i| & \text{if there is a link from page } P_i \text{ to page } P_j \\ 0, & \text{otherwise} \end{cases}$$

Assume there is a random surfer who surfs the web by following the hyperlink structure of the Web represented by $\mathbf{H}$. When on a page containing no hyperlinks, surfer may hyperlink to any other page therefore a stochasticity adjustment needs to be applied to $\mathbf{H}$:

$$\mathbf{S} = \mathbf{H} + \mathbf{a}(\frac{1}{n}\mathbf{e}^T)$$

Where $a_i = 1$ if page i has no hyperlinks and $\mathbf{e}$ is a vector of ones.

Text Analysis and Retrieval 2016: Course Project Reports (TAR 2016), pages 67–69, University of Zagreb, Faculy of Electrical Engineering and Computing, Zagreb, June 2016.

Sometimes, a random surfer abandons the hyperlink method of surfing, randomly choosing the next page. With that in regard a primitivity adjustment needs to be applied according to the following formula:

$$\mathbf{G} = \alpha\mathbf{s} + (1-\alpha)\mathbf{E}$$

Where α is a scalar between 0 and 1 determining the frequency of abandoning the hyperlink structure and $\mathbf{E}$ is the normalized matrix of ones.

Let $\boldsymbol{\pi}$ be the probability distribution of the surfer across the web pages (the vector of PageRank scores) then the following formula must be valid:

$$\boldsymbol{\pi}^T = \boldsymbol{\pi}^T\mathbf{G}$$

Thus if we were to compute the principal left eigenvector of the matrix $\mathbf{G}$ — the one with eigenvalue 1 — we would have computed the PageRank values (Manning et al., 2008).

3.2. HITS

HITS defines hubs and authorities. A page is considered a hub if it contains many hyperlinks. On the other hand, it is considered an authority if many hyperlinks point to it. A page is a good hub if it points to good authorities, and a good authority if it is pointed to by good hubs. We need to assign two scores to every document: an authority score and a hub score. We can update these scores iteratively according to the following formulae:

$$\begin{aligned}\mathbf{h} &\leftarrow \mathbf{A}\mathbf{a} \\ \mathbf{a} &\leftarrow \mathbf{A}^T\mathbf{h}\end{aligned} \tag{1}$$

Where $\mathbf{h}$ and $\mathbf{a}$ are the vectors of hub and authority scores and $\mathbf{A}$ is adjacency matrix. A_ij is 1 if there is a hyperlink from page i to page j, and 0 otherwise. Now the right-hand side of each line of equation 1 is a vector that is the left-hand side of the other line of the equation. Substituting these into one another, we may rewrite equation 1 as:

$$\begin{aligned}\mathbf{h} &\leftarrow \mathbf{A}\mathbf{A}^T\mathbf{h} \\ \mathbf{a} &\leftarrow \mathbf{A}^T\mathbf{A}\mathbf{a}\end{aligned} \tag{2}$$

Now we can compute vector of hub scores $\mathbf{h}$ as principal eigenvector of $\mathbf{A}\mathbf{A}^T$ and vector of authority scores $\mathbf{a}$ as principal eigenvector of $\mathbf{A}^T\mathbf{A}$.

In this paper the links were created in that way that adjacency matrix $\mathbf{A}$ is always symmetric. That means that hub and authority scores turn out to be identical, so in future references we will denote this score as HITS score.

4. Pipeline

In this section we present the pipeline of the entire information retrieval system.

1. Each document is preprocessed with a Porter stemmer and stopword removal.
2. Vocabulary is constructed from the entire collection.
3. Every document is converted to a binary vector.

Table 1: Mean average precision.

VSM	VSM + PageRank	VSM + HITS
0.264	0.267	0.264

4. For each document and a query, vector space model score (similarity score) $S_{vsm}(d_j, q)$ is calculated.
5. PageRank and HITS scores are calculated ($S_{la}(d_j)$).
6. In the last step vector space model scores are refined with PageRank or HITS scores according to the following formula:

$$S(d_j, q) = wS_{vsm}(d_j, q) + \frac{(1-w)S_{la}(d_j)}{log(r(d_j, q)) + log5} \tag{3}$$

Where $r(d_j, q)$ is the rank of document d_j according to the vector space model and w is the number between 0 and 1 that determines the importance of vector space model score in relation to the link analysis score. This formula was taken from (Cheng, 2012).

5. Test collection

We worked with European Media Monitor test collection which contains 1742 documents and a set of 48 queries with relevance judgments. As mentioned in Section 3. we created the hyperlinks among documents ourselves. Two different methods were used in hyperlink creation:

1. Each document came with a number. We created links between documents whose number distance was smaller than 200. This method produced more or less random links among documents.
2. We calculated cosine similarity for each pair of documents. Documents with a similarity greater than 0.7 were linked. In reality, it is expected that similar documents would probably link to each other.

6. Results

Information retrieval system was evaluated using mean average precision, mean reciprocal rank and mean R-precision. Three different setups were evaluated:

- Vector space model score.
- Vector space model score refined with PageRank.
- Vector space model score refined with HITS.

Our results are presented in Tables 1, 2 and 3. Each table represents a different evaluation method. We find that mean reciprocal rank is the most important evaluation method for interpretability.

Random link generation only caused a decrease in all precisions. With that in regard we decided to generate links using cosine similarity as described in Section 5. Among

Table 2: Mean reciprocal rank.

VSM	VSM + PageRank	VSM + HITS
0.502	0.525	0.502

Table 3: Mean R-precision.

VSM	VSM + PageRank	VSM + HITS
0.246	0.243	0.246

earlier mentioned similarity metrics we decided to use cosine similarity due to dice similarity poor results.

Mean reciprocal rank in each setup is around 0.5 which means that on average the first relevant document will be found on the second rank in the retrieved document list.

Refining vector space model scores with PageRank shows an improvement in mean average precision and mean reciprocal rank according to Tables 1 and 2. However refining vector space model with HITS doesn't show any improvement.

7. Conclusion

For this project we implemented an information retrieval system based on binary vector space model in combination with link analysis algorithms PageRank and HITS. It was shown that refining vector space model scores with PageRank improves system precision while HITS made no noticeable difference.

In the future the retrieval system could be improved. An obvious idea would be to use tf-idf weighting scheme instead of binary. We could also try further parameter adjustment to achieve better results. Also the system could be tested on a dataset that already contains links among documents.

References

Bojana Dalbelo Bašić and Jan Šnajder. 2014. 3. basics of information retrieval. University course material.

Bojana Dalbelo Bašić and Jan Šnajder. 2015. 4. improved search, evaluation, and web search. University course material.

Guangchun Cheng. 2012. Design and implementation of a search engine using vector space and pagerank. Technical Report 5200, University of North Texas Discovery Park.

Christopher D Manning, Prabhakar Raghavan, Hinrich Schütze, et al. 2008. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge.

Sexual Predator Identification Using *word2vec* Features

Jan Tomljanović, Luka Zuanović, Tomislav Šebrek

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
{jan.tomljanovic,luka.zuanovic,tomislav.sebrek}@fer.hr

Abstract

This paper describes our approach to two problems regarding sexual predators in internet chats. First problem is identification of sexual predators among chat users, and the second problem consists of finding messages in chats which are predatory in nature. We tackle these problems using text analysis, more specifically, word to vector algorithm along with some extra features extracted from text. For both problems we used machine learning techniques as the final step.

1. Introduction

Chat based communication or similar features can nowadays be found on many web pages, including not only obvious ones, like giant social networks such as Facebook, Twitter, Instagram etc., but also many others, more obscure, which often provide anonymity for the users.

In such conditions, many people exploit possibilities for missuse of the chatroom. Among such people sexual predators[1] are arguably most dangerous, especially when their victims are under-aged children. Children are often inexperienced internet users and are easily tricked into trusting a person they are communicating with. Therefore, predators can then persuade children to engage in sexually explicit text or video chats and children can potentially even agree to meet them in person.

This serious threat necessitates an adequate remedy. One possible pre-emptive approach is for trained professionals to impersonate young children and lure potential paedophiles to meet them in person and apprehend them when they show up. This approach works, but it is impossible for a group of people to monitor all chats that exist in digital space and that is why a need for automated sexual predator classification arises.

In this paper we tackle two problems: 1. sexual predator identification 2. identification of predatory parts of conversations These problems were part of International Sexual Predator Identification Competition at PAN-2012[2] (**PAN12**). For the first problem participants were provided with a labelled training set which enabled supervised machine learning approaches. For the second problem there were no labelled messages, thus making it an unsupervised learning problem.

The rest of the paper is organized as follows. Section 2 gives a brief overview of related work. In Section 3 our approach to the first problem is explained followed by description of our approach to the second problem in Section 4. Section 5 presents experimental results for both problems. Finally, Section 6 concludes our work and proposes future improvements.

[1]"A sexual predator is a person seen as obtaining or trying to obtain sexual contact with another person in a metaphorically 'predatory' or abusive manner" - Wikipedia.org

[2]http://pan.webis.de/clef12/pan12-web/author-identification.html

2. Related work

In (Parapar et al., 2012) for the first problem authors represented each user with feature vector made up of a combination of three types of features. First type were tf/idf features constructed over standard unigram, bigram and trigram representation of the chats. Authors pruned the vocabulary by removing infrequent and too long terms (unigram representation) or expressions (bi/trigram representation).

Second type of features used were linguistic inquiry and word count (LIWC) psycholinguistic features (Pennebaker et al., 2015). And for the third type, authors used chat-based features that tried to represent overall user activity. Examples of such features are average time of day user had sent messages to other users, number of different users approached by that user or average time between two consecutive messages sent by that user.

For the final step authors most often used SVM classifier. As for the second problem, the best approach submitted (Popescu and Grozea, 2012) was returning all lines that user classified as predator wrote, which suggest the difficulty of the problem and the lack of well thought out solution.

For an extensive overview of methods used to solve both problems see (Inches and Crestani, 2012).

3. Predator identification

We approach sexual predator identification problem as a supervised machine learning problem. From the training set we extract two types of features for each chat user. Then we train several classifiers and test them on the test set.

3.1. Dataset pre-filtering

Dataset provided in **PAN12** contains chat data, i.e., conversations between one, two, or more users.

We decided to remove all conversations with three or more users because it is highly unlikely for a sexual predator to display bad behaviour in non-private chats. Training set data proved us right as there were no sexual predators that participated in such conversations.

Text Analysis and Retrieval 2016: Course Project Reports (TAR 2016), pages 70–72, University of Zagreb, Faculy of Electrical Engineering and Computing, Zagreb, June 2016.

Unlike (Villatoro-Tello et al., 2012) we decided against the removal of conversations which had only one user because we reckoned they were either failed attempts to start a conversation or a continuation of a conversation that had started in the past but failed to be continued. Both of which we thought could be indicative of predatory behaviour.

All users that wrote less than fifty words were removed from the dataset since we thought it would be unlikely to recognize a predator with a such small amount of data. By doing this, we removed four sexual predators from the training set, however, we consider them to be errors because after manual inspection we found it to be impossible to infer that they were indeed sexual predators.

In this pre-filtering steps training set has been reduced to approximately ten percent of its original size which made significant impact on the running time of the classifiers.

All of the steps above were applied equally for the training set and the testing set. All the predators we lost with pre-filtering were marked as false negatives and were included in the final results.

3.2. Chat-based features

As mentioned in Section 2. in (Parapar et al., 2012) several chat-based features were extracted from the dataset. Similarly, we extracted five chat-based features but with just one feature overlap between ours and theirs (i.e., the first one). We thought these features could describe some predatory trends.

Features are as follows:

1. average size (in characters) of user's messages
2. percentage of commenced conversations

$$\frac{number\ of\ conversations\ user\ commenced}{number\ of\ conversation\ user\ is\ involved\ in}$$

3. percentage of user's words in conversation

$$\frac{number\ of\ user's\ words\ in\ conversations}{total\ number\ of\ words\ in\ those\ conversations}$$

4. average number of user's consecutive messages

$$\frac{total\ number\ of\ user's\ messages}{total\ number\ of\ user's\ message\ blocks}$$

5. percentage of questions asked

$$\frac{number\ of\ questions\ user\ asked}{number\ of\ user's\ messages}$$

3.3. Word2vec features

After we generate chat-based features for each user we proceed to additionally represent each user with a set of *word2vec* features.

First, we preprocess the dataset (what remained after initial pre-filtering) by removing stop words and then applying lemmatization on each remaining word. Other authors (Parapar et al., 2012; Villatoro-Tello et al., 2012) skipped this step in order to preserve users' own unique styles. We wanted to identify the predator by the core meaning of his/her messages, instead of relying on the fact that maybe many predators use the same form of a word or make the same grammatical errors.

After that, all of user's words are transformed into 300 dimensional vector using Google word2vec dictionary.[3] For each user we added all vectors of his/her words together and then normalised them to compensate for different users having produced different amounts of words.

We also tried using GloVe[4] (Global vectors for word representation) for transforming words into vectors. Since the classifiers which used GloVe had slightly worse results we decided to use Google *word2vec* dictionary instead.

After all of this, each user is represented with 305 dimensional feature vector consisting of five chat-based features and 300 *word2vec* features. All that remains is to train the classifiers.

3.4. Classifier training

We decided to train three classifiers: support vector machine (SVM), random forest and logistic regression.

For each of them hyperparameter selection was conducted on stratified 3-fold nested cross-validation. Data was standardized properly every time the classifiers were trained or tested (both training and testing data were standardized but with mean and standard deviation calculated only on the training data).

Once hyperparameters were decided all three models were fitted on the training set and evaluated on the test set. Evaluation results are described in Section 5.

4. Predatory message identification

Since no labelled data was provided for the second problem of **PAN12** we approached it somewhat differently. First we classify users in the test set with the classifier that was performing best on our training set and that was SVM classifier, in the same way as for the first problem. By doing that, we got the list of predators (predicted by our classifier). Since our classifier is not working perfectly, we missed some of the predators, and therefore some of the predatory lines they wrote. We counted those, marked them as false negatives, and included them in the results.

Now we process the list of predators, i.e., each of the lines they wrote in the same way we processed words in the first problem (stop word removal, lemmatization and word2vec). As the result we got a 300 dimensional vector for each line detected predators wrote, and we will refer to them as line vectors.

For the classification of line vectors as either predatory or not we decided to use one-class SVM. As the training data for one-class SVM we used previously calculated word2vec vectors for all non-predator users from the training set for the first problem.

As the final step, for each of the line vectors trained one-class SVM model predicted whether it belongs to the class of decent (non-predatory) messages it previously was trained on, or not. If it decided that the line vector doesn't

[3] https://code.google.com/archive/p/word2vec/

[4] http://nlp.stanford.edu/projects/glove/

belong to the class of decent messages that line vector was marked as predatory. Evaluation results are described in Section 5.

5. Results

For the first problem we trained three classifiers: SVM, random forest (RF) and logistic regression (LR). Evaluation was conducted on the test set provided for **PAN12** and the results are reported in table 1. We used $F_{0.5}$ measure to evaluate our classifiers because it was the metric used for ranking contest participants. If we were to compare our results for the first problem with that of the contest participants, we would place sixth out of seventeen with a result fairly close to the top ranks.

Table 1: Classifier evaluation results for the first problem

	precision	recall	$F_{0.5}$
SVM	0.973	0.571	**0.853**
RF	**1**	0.205	0.563
LR	0.798	**0.622**	0.755

Results for second problem are reported in table 2. Three versions of one-class SVM were tried out, with respect to the key parameter ν. Here, F_3 measure was used for the same reason $F_{0.5}$ was used in the first problem. Our results are especially notable when compared with results of other **PAN12** participants. Our result is second best and just slightly worse then the winner's.

Table 2: Evaluation result for the second problem

precision	recall	F_3	ν
0.1	0.757	0.457	0.1
0.101	0.77	0.463	0.3
0.101	**0.77**	**0.463**	0.5

6. Conclusion

We have proposed a new set of features which can be used to represent a chat user in such a way that automated sexual predator identification is possible with a very high precision. Future improvements could be made by combining our set of features with features developed by participants of **PAN12** such as tf/idf features or LIWC psycholinguistic features.

Results indicate that our approach to the identification of predatory lines is still highly unsuccessful, even though it is better than most of the approaches used by other authors.

References

Giacomo Inches and Fabio Crestani. 2012. Overview of the international sexual predator identification competition at pan-2012. In *CLEF (Online Working Notes/Labs/Workshop)*, volume 30.

Javier Parapar, David E Losada, and Alvaro Barreiro. 2012. A learning-based approach for the identification of sexual predators in chat logs. In *CLEF (Online Working Notes/Labs/Workshop)*.

James W Pennebaker, Ryan L Boyd, Kayla Jordan, and Kate Blackburn. 2015. The development and psychometric properties of liwc2015. *UT Faculty/Researcher Works*.

Marius Popescu and Cristian Grozea. 2012. Kernel methods and string kernels for authorship analysis. In *CLEF (Online Working Notes/Labs/Workshop)*.

Esaú Villatoro-Tello, Antonio Juárez-González, Hugo Jair Escalante, Manuel Montes-y Gómez, and Luis Villasenor Pineda. 2012. A two-step approach for effective detection of misbehaving users in chats. In *CLEF (Online Working Notes/Labs/Workshop)*.

Classification Approaches for Short Texts as Tweets

Luka Vranješ, Tomislav Gracin, Mihael Presečan

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
{vranjes.luka,tgracin,mihael.presecan}@gmail.com

Abstract

As the Twitter social network is constantly loaded with new content, new interests were aroused for automatically classifying the tweets into categories of interests. In contrary to the large texts, classifying short texts as tweets, gives us new challenges in text classification, because traditional approaches like *Bag-Of-Words* do not provide sufficient word occurrences and thus having notable limitations. In this paper, *Word Embeddings for Semantic Similarity* has been introduced as a new approach to the traditional *Bag-of-Words* approach of feature selection, and over the classification approaches, *multilabel* and *multiclass* has been tested in comparison to the proposed feature selection. This study was conducted on the Croatian-Serbian corpus of tweets, categorizing it in the five broad and general categories (News, Events, Opinions, Deals, Private Messages) using supervised machine learning on *SVM model*.

1. Introduction

Twitter is known as a micro-blogging social network used to discuss and promote public opinion, or simply to share news and promote events or products. One tweet is limited to 140 characters and thus satisfies the definition of micro-blogging (mic). Classifying short text requires new techniques, and approaches other then classifying the large texts. Traditional approaches like Bag-of-Words *(BoW)* create large word vector length of whole dictionary, so vector of one short tweet becomes sparse, and it does not adapt well to its semantic representation.

For short text, new techniques are necessary for adapting to proper semantic representation. *Word Embeddings (or W2V - Word to Vector)* is technique which maps the words into the vectors of real numbers, and grouping the words/vectors that occur in similar context in low-dimensional space, thus adapting to the semantic representation more accurately. This paper gives the comparison between *BoW* and *W2V* in performance on classifying Croatian-Serbian tweets, and combination of both.

2. Previous work and new approaches

Tweet classification has been a subject of interest in recent years, and several studies had been conducted by students of *Faculty of Electrical Engineering and Computing, University of Zagreb*. On previous studies on *Tweet Classification* (J. Bzik), SVM, KNN and Naive Bayes had been tested, and SVM had shown to be the best choice. In this study focus had been on SVM and comparing *BoW* and *W2V* feature extraction combined with *multiclass* and *multilabel* algorithm approach.

3. Approaches to feature extraction

3.1. Bag of Words

In previous work, the chosen feature extraction approach for tweet classification was *Bag of Words* (*BoW*) (J. Bzik). All words have been taken from annotated tweet corpus, and were put in a dictionary, each word having it's index. A vector representing a tweet is then created in a simple manner of putting the number of occurrences of certain word in a tweet in the vector at the corresponding index.

3.2. Word Embeddings for Semantic Similarity (W2V - Word to Vector)

Word embedding is a feature learning techniques, where words from the vocabulary are mapped to vectors of real numbers in a low-dimensional space relative to the vocabulary size ("continuous space"), thus they insinuate semantic similarity. Creating the function of mapping the words into the vectors is the task for itself, and it includes the methods like neural networks, dimensionality reduction on the word co-occurrence matrix, probabilistic models, and explicit representation in terms of the context in which words appear. Today's most famous toolbox for Word Embeddings is *Word2Vec* toolbox created by Google team led by Tomas Mikolov.(T. Mikolov2013)

For this project, mapping function has been provided by *TakeLab* which maps predefined set of Croatian words into vectors (TakeLab2016). This predefined set of Croatian words are created from different corpus of words other then the corpus of Croatian-Serbian tweets used in this project.

Feature vector of the tweet has been made by aggregated word embeddings by following formulas:

$$vec(c) = \sum_{w \in s} ic(w) \cdot vec(w) \tag{1}$$

$$ic(w) = log \frac{\sum_{w' \in C} freq(w')}{freq(w)} \tag{2}$$

Each word $vec(w)$ in tweet is wighted by formula (2) and summed up to one vector, thus one tweet is represented with one vector (feature vector) suitable for training the classifier. Weight $ic(w)$ in (2) is calculated as the logarithmic ratio of all occurrences of all words in training set over the frequency of that particular word in the training set. If new word occurs in the classification and is not present in the dictionary of training set, then $ic(w)$ is assigned to 0 and that word is not included in the feature vector.

3.3. Combined BoW and Word Embeddings

The new feature has been extracted by concatenating *BoW* and *W2V* into single feature vector. Feature vector becomes

Text Analysis and Retrieval 2016: Course Project Reports (TAR 2016), pages 73–76, University of Zagreb, Faculy of Electrical Engineering and Computing, Zagreb, June 2016.

longer and *SVM* classifier has been shown to be endurable and robust to feature changes. Over the training phase, *SVM* classifier will weigh higher features that are more relevant than the others.

4. Approaches to classification

In this work, five broad general categories has been chosen: *News*, *Events*, *Opinions*, *Deals* and *Private Messages*. Classification task is to classify tweet into one or more, or none of the categories. Uncategorized case is allowed as well.

4.1. Multiclass classification

Multiclass or *multinomial* classification is a task to classify an instance between two or more classes. Most of the classification models, including SVM, are *binary* classification algorithms, and to perform *multiclass* classification the problem needs to be solved by variety of strategies. Two most famous strategies are *one-vs.-one* and *one-vs.-rest*. In this work, *SVM* binary classifier has been turned into *multinomial* classifier over the *one-vs.-one* strategy. *Multiclass* classification makes the assumption that each sample is assigned to one and only one class. Tweet classification problem indicates that one tweet can be in one or more categories at same time. To meet the requirements of *multiclass* classification, these five categories had been expanded to thirty two classes in all inner combinations, thus providing distinct multiple classes. Therefore only one classifier is need to classify one tweet.

4.2. Multilabel classification

Multilabel classification is a variant of the classification problem where multiple target labels must be assigned to each instance. Strongly related problem to *multilabel* classification is *multi-output* classification, where model maps vector **x** into binary vector **y**, rather than scalar outputs as in the ordinary classification problem. To achieve multilabel properties, problem has been transformed into a set of binary classification problems, which can then be handled using single-class classifiers. To implement this approach, each category needs to have it's own binary classifier and data has to be prepared for each one of them - as of particular tweet is or is not in that particularly category.

5. Research

The research of this paper is to explore combination of mentioned feature extraction with the multiclass and multilabel approach. Features of *BoW*, *W2V* and concatenated were combined with *multiclass* and *multilabel* approach, thus providing 6 cases. For each case, a validation had been conducted, and has been presented in this paper.

6. Method

6.1. Tweets corpus and annotating

In this work, three annotators annotated 2000 tweets, which were randomly taken out of big initial tweet corpus. There was a set of 500 tweets which were annotated by all annotators. Agreement between each annotators has been presented by table (1). *Fleiss' kappa* measure (Fleiss1971) has been calculated to correct the agreement of annotators and chance agreement. According to Landis and Koch this is slight agreement, while 0.21 - 0.4 represent fair agreement (Landis and Koch1977). Agreement could be higher, if annotators had prescribed guidelines initially.

Table 1: Annotators agreement and Fleis' Kappa measure

	News	Events	Opinions	Deals	Priv. msg.
Agr. AB	0.87	0.96	0.56	0.97	0.67
Agr. BC	0.89	0.96	0.69	0.97	0.66
Agr. CA	0.95	0.98	0.37	0.99	0.91
Avg. Agr.	0.90	0.97	0.54	0.98	0.75
Kappa	0.17	0.35	0.02	-0.01	0.45
Total avg. agreement			**0.8310**		
Total Fleis' Kappa (κ)			**0.1988**		

Between three annotators, one was chosen to be the representative for 500 tweets that were interested. Decision criteria was an annotator who had highest agreement with other two annotators.

6.2. Data reprocessing and modeling

Tweets usually contain a lot of noise produced by using unofficial language or bad grammar. These tweets had been filtered and normalized. This step is called preprocessing text data. First step was to make every letter a lowercase and punctuation marks were removed. The text had been tokenized string of words, and special characters connected with semantics of Twitter (like @ and #) had been preserved, creating a new "word" in BoW vector.

7. Results

The set of 2000 annotated tweets was split to on training and testing test over 7:3 ratio. Testing has been conducted on three different measures with following equation numbers: *Precision* (3), *Recall* (4) and *F1-Score* (5). For the multiclass classification, additional *accuracy* (6) measure has been added for clearer understanding of data. The measures has been calculated on following formulas:

$$Precision = \frac{True_{positive}}{True_{positive} + False_{positive}} \tag{3}$$

$$Recall = \frac{True_{positive}}{True_{positive} + False_{negative}} \tag{4}$$

$$F1 = \frac{2 \cdot True_{positive}}{2 \cdot True_{positive} + False_{positive} + False_{negative}} \tag{5}$$

$$Accuracy = \frac{True_{positive} + True_{negative}}{total} \tag{6}$$

7.1. Multiclass approach

In multiclass approach only one classifier was developed. That classifier has 32 classes that represent all the combination of the assignment of the 5 categories.

Table 2: Results for multiclass approach with BoW feature set.

Class	Precision	Recall	Acc.	F1-Score
NEWS	0.00	0.00	0.84	0.00
EVENTS	0.00	0.00	0.95	0.00
OPINIONS	0.00	0.00	0.60	0.00
DEALS	0.00	0.00	0.93	0.00
PRIV. MSG.	0.47	1.00	0.47	0.63
TOTAL	0.09	**0.20**	**0.77**	0.13

Table 3: Results for multiclass approach with W2V feature set

Class	Precision	Recall	Acc.	F1-Score
NEWS	0.71	0.75	0.85	0.13
EVENTS	0.00	0.00	0.95	0.00
OPINIONS	0.55	0.03	0.60	0.06
DEALS	0.00	0.00	0.98	0.00
PRIV. MSG.	0.47	0.97	0.48	0.64
TOTAL	0.34	**0.35**	**0.77**	0.17

Table 4: Results for multiclass approach with combined feature set.

Class	Precision	Recall	Acc.	F1-Score
NEWS	0.56	0.17	0.85	0.26
EVENTS	0.00	0.00	0.95	0.00
OPINIONS	0.58	0.46	0.65	0.59
DEALS	0.00	0.00	0.98	0.00
PRIV. MSG.	0.60	0.82	0.66	0.69
TOTAL	0.35	**0.29**	**0.82**	0.31

In BoW feature extraction, all categories except *Private Messages* have been classified as *false negative*, due to following reasons: training data has been taken randomly among big corpus of Croatian-Serbian tweets, and they do not represent all the categories equally. *Private Messages* has been labeled in 47,25% of training data, while *Deals* has only 1,5% presence, and *Events* are present in 4,95% of training dataset. The other reason is due to *Private Messages* have character @ as characteristic, which significantly takes weight training model that outweigh other features. The *recall* has shown that classifier has no *false negatives* and all positives classifier classifies correct, but other measures shows that in area of negatives examples classifier works poorly.

In W2V feature extraction, progress has been shown that other categories has been included as true positive examples. Events and Deals do not have high presence rate in training data and still other features that are characteristic for other categories outweigh, as well in combined case.

In combined case, results have shown higher improvement in *accuracy* but lower value in *recall*, which still shows a progress.

Over all, multiclass approach shows bad performance, due to unrepresentative training data, but most of all sparse data. From five categories, to gain all inner combinations, classes has been expanded into 32 classes. To rightly train the this case, few conditions needs to be met. All classes has to have adequate number of training examples, and all 32 different classes should be contextually independent from each other, or if not independent the feature vectors should be near each other in space, but that is not the case in this example. For instance, some tweets can be in the *Opinions* and *Private Message* categories in same time, which are contextually dependent, and multiclass approach has not cover this case.

7.2. Multilabel approach

Multilabel approach has shown better performance when each category has dedicated classifier to determent is particular tweet in that particular category or not. Therefore five binary SVM classifiers were trained for each category. The results are following:

Table 5: Results for multilabel approach with BoW feature set.

Class	Precision	Recall	F1-Score
NEWS	0.71	0.84	0.77
EVENTS	0.90	0.95	0.93
OPINIONS	0.36	0.60	0.45
DEALS	0.97	0.98	0.98
PRIV. MSG.	0.28	0.53	0.37
TOTAL	0.64	**0.78**	0.70

Table 6: Results for multilabel approach with W2V feature set.

Class	Precision	Recall	F1-Score
NEWS	0.82	0.85	0.78
EVENTS	0.90	0.95	0.93
OPINIONS	0.36	0.60	0.45
DEALS	0.97	0.98	0.98
PRIV. MSG.	0.72	0.58	0.47
TOTAL	0.75	**0.79**	0.72

Table 7: Results for multilabel approach with combined feature set.

Class	Precision	Recall	F1-Score
NEWS	0.81	0.85	0.81
EVENTS	0.90	0.95	0.93
OPINIONS	0.62	0.63	0.61
DEALS	0.97	0.98	0.98
PRIV. MSG.	0.68	0.68	0.68
TOTAL	0.80	**0.82**	0.80

Multilabel approach has shown better characteristics for each individual category, because each classifier has adapted to its particular category. Compared to previous work (J. Bzik) this approach with combination of BoW and W2V has shown better performance. *Recall* and *accuracy* measure are the same values, and therefore *accuracy* have not been presented in this paper separately.

8. Conclusion

Results have shown that tweets can be classified fairly well with correct classification approach. Results also showed that combined feature extraction of BoW and W2V give better results than in separate cases. While BoW is the good approach in large documents, in small texts like tweets is not representative enough. Benefits of BoW are that dictionary is created from the annotated set of tweets, and all words in testing will be used. Special features provide special characters like @ or # that are really useful in tweeter case.

Word Embeddings with Semantic Similarity has shown better results than BoW. The downside for this project is that not all words that were in tweets dictionary are not present in W2V dictionary. Words used to create W2V were taken from more formal Croatian language, and in this project tweets were written in informal way, in Croatian and Serbian language as well. To improve results, we propose to create W2V from the twitter data, with informal and Serbian words included. Another improvement would be to add steaming into NLP pipeline to reduce number of words that have same semantic value but different form.

This work has shown that multilabel approach is better approach to classification, and that Word Embeddings for Semantic Similarity is a good approach to small texts as are tweets. The testing has shown good results, but they are still more room for improvement.

Acknowledgment

In this work, a great help was done by *TakeLab* team Mladen Karan mag. ing. comp., Domagoj Alagić mag. ing. and dr. sc. Goran Glavaš. There has been a great portion of patience invested in preparing this work and working with us in lectures, consultations and correspondence. Special thanks to professor doc. dr. sc. Jan Šnajder for leading the whole team and conducting this lecture.

Appendex

The project has been open sourced and published online.[1] As well, in more technical details this project has been discussed online at *minimizef.com* [2] portal.

References

J. L. Fleiss. 1971. *Measuring nominal scale agreement among many raters*. Psychological Bulletin, vol. 76, no. 5 pp. 378–382 edition.

M. Stepanić J. Bzik. Tweet classification.

J. R. Landis and Koch. 1977. G. g. pages 159–174.

Microblogging. https://en.wikipedia.org/wiki/Microblogging. Accessed: 2016-08-02.

G. Corrado J. Dean T. Mikolov, K. Chen. 2013. Efficient estimation of word representations in vector space. *http://arxiv.org/pdf/1301.3781.pdf*.

TakeLab. 2016. Words to vectors. http://takelab.fer.hr/studentdownload/fhrwac-vectors/.

[1] https://github.com/mpresecan/HrSr-TweetClassification

[2] http://minimizef.com/classification-approaches-for-short-texts-as-tweets/

Author Profiling in English Tweets

Filip Zelić, Borna Sirovica, Ivan-Dominik Ljubičić

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
{filip.zelic, borna.sirovica, ivan-dominik.ljubicic}@fer.hr

Abstract

In this paper, we present our approach to the author profiling task, a student project for Text Analysis and Retrieval course. Given a set of tweets by the same person, the task aims at identifying age, gender and personality traits of that person. We address age and gender prediction as a classification task and a personality prediction as a regression problem. We experimented with Support Vector Machine for classification and regression and other machine learning algorithms using a variety of custom designed features as well as features extracted from publicly available resources.

1. Introduction

Author profiling distinguishes between classes of authors by studying their sociolect aspect, i.e., how language is shared or how an author can be characterized from a psychological viewpoint. This information helps in identifying profiling aspects such as gender, age, native language, or personality type.

Author profiling is a problem of growing importance, among others for applications in forensics, security, and marketing. However, social profiling still remains a less-explored topic, even though the exponential growth of social networks increased its importance even further. While there were several publications that were trying to predict some demographical information such as gender, age, and native language (Argamon and Shimoni, 2003), (Peersman and Vaerenbergh, 2011), as well as the personality type, and to perform author profiling in general (Argamon and Schler, 2009), the real push forward was enabled by specialized competitions such as the PAN shared tasks on author profiling (Pardo and Inches, 2013), (Rangel and Daelemans, 2014), (Rangel and Daelemans, 2015), which ran in 2013–2015.

This paper presents our approach to the author profiling task. The task focused on predicting an author's demographics (age and gender) and the big five personality traits (McCrae and Costa, 2008) (agreeable, conscientious, extroverted, open, stable) from a set of tweets by the same target author.

This paper is organized as follows: Section 2 gives a detailed description of dataset used for training and testing of our model. Section 3 presents general approach and methodology used in the experiments, including a description of the preprocessing, the features, and the learning algorithms we used. Section 4 presents and discusses our results and provides some deeper analysis. Finally, Section 5 concludes and points to possible directions for future work.

2. Dataset

The dataset we used consisted of English tweets from 152 users in XML format along with one *truth.txt* file with age, gender and the Big Five personality traits labels for each user (PAN, 2015). For labeling age, the following classes were considered: 1) 18—24, 2) 25—34, 3) 35—49, 4) 50+ and gender was labeled as male (M) or female (F). The distribution of the gender and age labels in the corpus is reported in Table 1. For the case of gender classes the corpus was balanced with 50% of the tweets labeled as male and other half as female, but regarding age the distribution was skewed due to the lower number (around 22%) of the older users (labels *35—49* and *50+*) and higher number (around 78%) of the younger users (labels *18—24* and *25—34*).

Table 1: Distribution of Twitter users with respect to the age and gender labels in the corpus.

Trait	Label	Number of users
Age	18—24	58
	25—34	60
	35—49	22
	50+	12
Gender	Male	76
	Female	76

Regarding personality traits normalized numeric rating in [-0.5,0.5] range was given for each of the following properties: extroverted, stable, agreeable, conscientious, and open. The mean for each trait is reported in Table 2.

Table 2: Mean values of the Big Five personality traits in the corpus.

Personality trait	Mean value
Extroverted	0.16
Stable	0.14
Agreeable	0.12
Conscientious	0.17
Open	0.24

Text Analysis and Retrieval 2016: Course Project Reports (TAR 2016), pages 77–81, University of Zagreb, Faculy of Electrical Engineering and Computing, Zagreb, June 2016.

3. Approach

Along this section, we describe the steps we took to prepare features for the training of our supervised machine learning models. In subsection 3.1 we explain the preprocessing step in which we cleaned the data in order to get better performances for stylometric features. Subsection 3.2 explains extraction of various features organized by their category and the last subsection 3.3 describes classification and regression models we used tackling the task of author profiling.

3.1. Preprocessing

Preprocessing is done by creating a document for each user joining all his/hers tweets from the dataset. After the creation, the document is initially stripped of XML tags and cleared of all twitter specific characteristics such as hashtags, @replies as well as URLs from the text. While doing this step we save the count of mentions, hashtags and URLs for later use in feature modeling. When the preprocessing step is done, features need to be extracted.

3.2. Feature extraction

As mentioned in the introduction, we used many different self-designed features and features extracted from publicly available resources for our machine learning algorithms.

Lexicon Features. We used several lexicons, both manually crafted and automatically generated:

- **NRC Word-Emotion Association Lexicon** (Mohammad, 2013): one dictionary for each of the eight primary human emotions: anger, anticipation, disgust, fear, joy, sadness, surprise, trust as well as dictionaries for positive and negative emotion word classification.
- **Internet slang word lexicon** : personally made dictionary with most common words younger people like to use on the Internet.
- **Frequent male and female words lexicon** (Schwartz and Ungar, 2013): dictionaries containing words most likely to be used by male or female person. Based on word usage frequency per million words.
- **Frequent male and female function words lexicon** (Kiprov, 2015) : dictionaries containing function words, such as 'this','the','she', or 'not', most likely to be used by male or female person.

For each lexicon, we found the number of occurrences of lexicon terms in all tweets for each user, normalized by the total number of words in tweets. We instantiated separate features for the different lexicons.

Twitter-specific Features. We used the following Twitter-specific features:

- **Hashtags:** the number of hashtags;
- **URLs:** the number of URLs posted;
- **User mentions:** the number of mentions of users using the pattern @username;
- **Emoticons:** the number of emoticons used in tweets.

All of the above counts are normalized by the total number of available tweets for the target user; so they could be viewed as "average number per tweet".

Orthographic Features. We used the following orthographic features:

- **Letter case:** the number of upper-case words and upper-case characters;
- **Character flooding:** the number of redundant character reduplication;
- **Word length:** average word length;
- **Tweet length:** average tweet length;
- **Specific characters:** usage of specific characters, such as the number of occurrences of the exclamation points, question marks or apostrophes.

Term-level Features. We used the following term-level features:

- **n-grams:** TF-IDF matrices of unigrams and trigrams;
- **F-score:** measure of contextuality and formality based on the frequency of the part of speech (POS) usage in a text (f.x below means the frequency of the part-of-speech x):

$$F = 0.5 * [(f.noun + f.adj + f.prep + f.art) – (f.pron + f.verb + f.adv + f.int) + 100]$$

- **POS tags:** frequency of certain POS tags such as nouns, verbs, prepositions, determiners, interjections, adverbs, adjectives etc.

3.3. Supervised learning models

Supervised machine learning models are used with hyperparameter optimization using grid search and k-fold cross validation on 70% of training data. Regarding classification task for age and gender, we tested the following classification models:

- Support Vector Machine Classification (SVC) with linear kernel and RBF kernel
- Logistic Regression
- Random Forest

On the other hand regarding regression task for personality traits, following regression models were tested:

- Support Vector Machine Regression (SVR) with linear kernel and RBF kernel
- Linear Regression

Values of features were scaled using StandardScaler from sklearn python library in order to avoid complications that can arise in the classification stage when features with numeric values that differ a lot.

4. Evaluation

After extraction of features and normalization of feature values, we trained models from Section 3.3 on 70% of data and evaluated them on other 30% of data trough multiple iterations in which train and test data were shuffled. In this process we tried combinations of many features from Section 3.2 for each subtask. Best feature combinations are stated in the next few paragraphs.

Regarding age subtask best scores were achieved using combinations of twitter-specific and orthographic features along with TF-IDF scores of frequent trigrams. SVM model with RBF kernel proved to be the best choice model wise.

In Gender classification subtask best results were obtained with the usage of male and female based lexicons together with orthographic features and F-score measure. SVM model with RBF kernel was proven to be the best choice for model selection also. The results of these two classification subtasks are given in the Table 3.

Table 3: Age and Gender prediction accuracy

Subtask	Classifier	Accuracy score
Gender	SVM (RBF kernel)	77.2 %
Age	SVM (RBF kernel)	76.1%

Personality traits subtask was modeled as a regression problem. For every trait we chose the most appropriate feature combination which in general included word-emotion based lexicons as well as the F-score measure, POS tags frequencies and orthographic features such as upper-case word count, average tweet length and exclamation overload count. To measure accuracy of personality traits prediction root mean squared error (RMSE) measure was used. Lowest errors between predicted and actual scores were obtained using SVM Regression model. Results of this task are given in following Table 4.

Table 4: Personality traits prediction accuracy

Personality trait	Regression model	RMSE
Extroverted	SVR (RBF kernel)	0.142
Stable	SVR (RBF kernel)	0.170
Agreeable	SVR (RBF kernel)	0.145
Conscientious	SVR (RBF kernel)	0.153
Open	SVR (RBF kernel)	0.156

5. Conclusion and future work

This paper proposes a supervised machine learning technique for solving the author profiling problem of determining author's age, gender and Big Five personality traits trough feature extraction from tweet corpus.

Tackling this challenging task we combined a large number of various features in order to train and evaluate our machine learning models. The main problem was that we could not evaluate our models on the most real case scenario because we did not find a publicly available evaluation dataset so the models were evaluated on 30% of the training dataset.

Even though evaluation showed good results for age gender and personality trait prediction (Rangel and Daelemans, 2015), we feel that there is a lot more to gain. For example, in the future work we could develop a more sophisticated approach for personality trait identification, considering more specific features and preprocessing for each personality trait separately. We could also experiment with descriptive LSA (Latent Semantic Analysis) and discriminative SOA (Second Order Attributes) features analyzed in paper (Álvarez Carmona and Escalante, 2015).

References

Pennebaker Argamon, Koppel and Schler. 2009. Automatically profiling the author of an anonymous text. *Commun. ACM*, 2:119–123.

Argamon and Shimoni. 2003. Automatically categorizing written texts by author gender. *Literary and Linguistic Computing*, 17:401—-412.

Nakov-Koychev Kiprov, Hardalov. 2015. Experiments in author profiling—notebook for pan. *CLEF PAN 2015*.

McCrae and Costa. 2008. A contemplated revision of the neo five-factor inventory. *The SAGE handbook of personality theory and assessment*.

Mohammad. 2013. Research in computational linguistic. *National Research Council Canada*.

PAN. 2015. Pan15-author-profiling-training-dataset-english. *https://github.com/dmincu/Author-Profiling/tree/master/pan15-author-profiling-training-dataset-2015-03-02/pan15-author-profiling-training-dataset-english-2015-03-02*.

Koppel-Stamatatos Pardo, Rosso and Inches. 2013. Overview of the author profiling task at pan 2013. *Working Notes for CLEF 2013 Conference*.

Daelemans Peersman and Vaerenbergh. 2011. Predicting age and gender in online social networks. *Workshop on Search and Mining User-generated Contents, SMUC '11*, pages 37–44.

Chugur-Potthast Trenkmann-Stein-Verhoeven Rangel, Rosso and Daelemans. 2014. Overview of the 2nd author profiling task at pan 2014. *CLEF 2014 Evaluation Labs and Workshop*.

Potthast-Stein Rangel, Rosso and Daelemans. 2015. Overview of the 3rd author profiling task at pan 2015. *CLEF 2015 Labs and Workshops*.

Kern-Dziurzynski Ramones-Agrawal-Shah Stillwell-Seligman Schwartz, Eichstaedt and Ungar. 2013. Personality, gender, and age in the language of social media: The open-vocabulary approach. *In PLOS ONE*, 8.

Montes-y-Gómez Villaseñor-Pineda Álvarez Carmona, López-Monroy and Escalante. 2015. Author profiling task notebook for pan at clef 2015. *INAOE's participation at PAN'15*.

Author Index