Exploring Multidimensional Continuous Feature Space to Extract Relevant Words

Márius Šajgalík, Michal Barla, Mária Bieliková

Slovak University of Technology in Bratislava
Faculty of Informatics and Information Technologies
Ilkovičova 2, 842 16 Bratislava, Slovakia
{marius.sajgalik,michal.barla,maria.bielikova}@stuba.sk

Abstract. With growing amounts of text data the descriptive metadata become more crucial in efficient processing of it. One kind of such metadata are keywords, which we can encounter e.g. in everyday browsing of webpages. Such metadata can be of benefit in various scenarios, such as web search or content-based recommendation. We research keyword extraction problem from the perspective of vector space and present a novel method to extract relevant words from an article, where we represent each word and phrase of the article as a vector of its latent features. We evaluate our method within text categorisation problem using a well-known 20-newsgroups dataset and achieve state-of-the-art results.

1 Introduction

The distributional hypothesis that features of a word can be learned based on its context was formulated sixty years ago by Harris [5]. In the area of natural language processing (NLP), the context of a word is commonly represented by neighbouring words that surround it (i.e. both the preceding and succeeding words). Since Harris, many NLP researchers built on his hypothesis and developed methods for learning both syntactic and semantic features of words. Hinton [6] was one of the pioneers who represented words as vectors in continuous feature space. Collobert and Weston [2] showed that such feature vectors can be used to significantly improve and simplify many NLP applications.

With advent of deep learning, more efficient methods for training feature vectors of words have been discovered. This allows us to use larger sets of training data, leading to higher quality of learned vectors, which in turn open possibilities for more practical applications of those vectors.

The advantage of such distributed representation of words, which maps words onto points in hyperspace is that we can employ standard vector operations to achieve interesting tasks e.g., to measure similarity between pairs of words. We can also calculate what words are the most similar by finding the closest vectors, or a vector that encodes a relationship between a pair of words, e.g., a vector transforming singular form into plural one, etc. Mikolov et al. showed that with such word vectors, we can encode many semantic and also syntactic relations [13].

One of the open problems in the current NLP is the computation of meaning of complex structures. The solvability of this problem is unanimously presupposed by notion of language compositionality, which is regarded as one of the defining properties of human language. One of the definitions of language compositionality, as almost an uncontroversial notion among linguists, can be found in [3]:

"(...) the meaning of (that) complex expression is fully determined by the meanings of its component expressions plus the syntactic mode of organization of those component expressions."

There are various approaches to solve the problem of compositionality in distributional semantics. The standard approach to model the composed meaning of a phrase is the vector addition [12]. There are also other approaches to model composition in semantic spaces that include vector point-wise multiplication [14], or even more complex operations like tensor product [4]. For purpose of this paper, we utilise vector addition to compose meaning of a phrase as it is reported to give very good results and to hold multiple relations between vectors [13] on the corpus we use in our experiments. We do not experiment with other methods of composition, but focus our experiments on applying various weighting schemes to analyse and compare their performance.

In this paper, we leverage feature vectors of words to extract relevant words from documents and evaluate such representation in text categorization problem. Such representation has a big potential in NLP and we can only anticipate that in a few years it will gradually supersede all those manually crafted taxonomies, ontologies, thesauri and various dictionaries, which are often rather incomplete (like most of artificial data). Moreover, there is no means of measuring similarity directly between pairs of words in such hand-crafted data. Most relations are just qualitative and described by their type (e.g. approach in [1] reveals only a relation type, but it cannot determine the relation quantitatively) and thus, all existing methods for measuring (semantic) similarity are limited to achieving only rather imprecise results.

2 Related work

In the past, we were studying a problem of extracting key-concepts from documents. Our approach described in [16], uses key-concepts (instead of features) to classify documents. We evaluated the proposed method using 20-newsgroups dataset, giving classification accuracy 41.48% using naïve Bayes classifier. However, 20-newsgroups dataset is one of the most commonly used in text categorisation problem and most of the researchers use micro-average F1 score to evaluate their classification performance. If we calculate micro-averaged F1 score for results in [16], it yields 58.63% using naïve Bayes classifier and 55.85% using kNN classifier.

There are many other researchers, who used the same dataset to evaluate and compare their work. The most important and relevant are those that study novel weighting schemes. Authors of [8] propose TF-RF for relevant term extraction, which can be viewed as an improvement of well-known TF-IDF weighting. The problem of TF-IDF is that it treats each word equally across different categories. The novelty of TF-RF weighting is that it discriminates the relevance of words based on the frequency differences in different categories. Evaluation using 20-newsgroups dataset gives micro-averaged F1 score 69.13% using kNN and 80.81% using SVM classifier.

Approach in [18] is based on two concepts – category frequency and inverse category frequency. Authors propose ICF-based weighting scheme that is reported to give better results than TF-RF using Reuters-21578 dataset, which has many categories (52). However, using 20-newsgroups dataset, they report not as good performance as TF-RF. Our explanation is that 20-newsgroups dataset has fewer categories and thus, the discriminating power of their "icf assumption" is weaker.

There are also approaches that do not focus on weighting of words, but try to employ various more sophisticated methods. In [9] we can find an approach using Discriminative Restricted Boltzmann Machines that achieves 76.2% micro-averaged F1 score using 20-newsgroups dataset and in [10] authors propose an error-correcting output coding method, which using naïve Bayes classifier achieves 81.84%.

3 Data pre-processing

We processed each article with Stanford CoreNLP [15] to transform raw text into sequence of words labelled with part-of-speech tags. Approach in [17] inspired us to build a finite automaton (Fig. 1) that accepts candidate phrases for further processing. We assume that extracted candidate phrases describe the content of given article and have a potential to contribute with good features to topical representation, which would help us to extract better words. Using these patterns of labelled words, we got rid of stopwords and retained only valid candidate phrases that could possibly influence the process of choosing the most relevant words that would be most distinctive for classifying the article into its correct category.

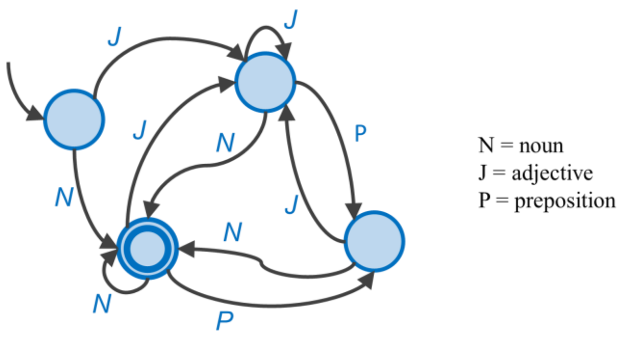

Fig. 1. Finite automaton for generating candidate phrases.

Also, we tried choosing only noun phrases obtained by chunking, however, it skipped noun phrases prefixed by adjectives, which sometimes significantly influence the meaning of a phrase. We also tried choosing only the words that are nouns, adjectives, verbs or adverbs, but it gave us worse results than considering candidate phrases accepted by this automaton.

We simulate the understanding of word semantics by using distributed word representation [12], which represents each word as vector of latent features. We use pre-trained feature vectors[1] trained on part of Google News dataset (about 100 billion words) using neural network and negative sampling algorithm. The model contains 300-dimensional vectors for 3 million words and phrases. The vectors of phrases were obtained using a simple data-driven approach described in [12].

4 Transforming phrases into vectors

After we have obtained the list of noun phrases for each article, we transform each phrase into the corresponding feature vector. The model contains not just words, but also many multi-word phrases that have a specific meaning different from what we would get if we just summed up the vectors of words in those phrases. Let us call both the words and the phrases present in the model simply as unit phrases. Since there is no direct mapping in the model for every possible phrase, we try to assemble the longer phrases by concatenating the most probable unit phrases that are present in the model. We use a simple dynamic programming technique to minimise number of concatenated unit phrases and thus, prefer the longer unit phrases present in the model. In case of ambiguity we choose the sequence of unit phrases that are more similar to each other. Here follows the description of our algorithm:

```
Input: string PHRASE of length LEN
Output: feature vector VECTOR

Initialise array of integers NO_UNITS of size (LEN + 1),
set all elements to 0
Initialise array of floats SCORE of size (LEN + 1), set
all elements to 0
Initialise array of vectors V of size LEN+1, set V[0] to
zero vector, the rest is undefined
For 0 <= a < LEN do
  If V[a] is not defined then continue
  For a < b <= LEN do
    If there is not end of word at PHRASE[b] then con-
    tinue
    If substring PHRASE[a:b] is in the model
      Let V1 be the word vector for PHRASE[a:b]
      If V[a] is undefined
        Let SCORE1 = 0
      Else
```

[1] Available online at - https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUT-TlSS21pQmM/edit?usp=sharing

```
        Let SCORE1 be similarity of V1 and V[a]
      If V[b] is undefined or
      NO_UNITS[a] + 1 < NO_UNITS[b] or
      (NO_UNITS[a] + 1 = NO_UNITS[b] and
      SCORE[a] + SCORE1 < SCORE[b])
        NO_UNITS[b] = NO_UNITS[a] + 1
        V[b] = V[a] + V1
        SCORE[b] = SCORE[a] + SCORE1
  If no substring PHRASE[a:b] was found in the model
    Let b be the ending position of nearest word in
    PHRASE[a:]
    NO_UNITS[b] = NO_UNITS[a] + 1
    V[b] = V[a]
    SCORE[b] = SCORE[a]
Let VECTOR = V[LEN]
```

We use NO_UNITS array to track the minimal number of unit phrases that comprise longer phrases. Thus, we prefer unit phrases with multiple words, since their learned feature vector has higher quality than if we just summed up feature vectors of their individual words. In case of ambiguity, we use SCORE array to track the similarity of the composing unit phrases. Thus, we prefer more common expressions. The array V serves as an intermediate storage of computed vectors, which is used to retrieve the final result. The step 4.3 is a handler for unknown words (tokens) present in the phrase, which simply skips to the next available word. This mostly handles the common cases of various punctuation characters present in the phrase, which are not included in the model we used.

5 Searching for relevant words with t-SNE visualisation

Since in the corpus we use each word is a real-valued vector, we can map each word to a point in 300 dimensional feature space. We used t-SNE [11] (t-Distributed Stochastic Neighbour Embedding), which is a technique for dimensionality reduction particularly well suited for visualisation of high-dimensional datasets, and visualised word vectors to analyse and better understand what exactly is going on with these vectors. A priori, we state the following hypothesis:

"If a word in the article is more relevant (i.e. if it is a keyword), it means that it is more relevant to the discussed topic. On the other hand, if a word is not relevant to the discussed topic, it will diverge randomly in the vector space. If divergence of non-relevant words was not random, those words would form another topic in the article."

There are multiple possibilities of what exactly to visualise. First, we can visualise all the words and phrases extracted from the article by Stanford CoreNLP parser. However, that results in a big incomprehensible mess of words. Alternatively, we can reduce the set to only the candidate phrases as described in section 3. Although it is much better, we cannot really see any natural word clusters as could be expected a priori. Although similar words are grouped together, there are not just a topical word clusters relevant to the article and thus, we cannot infer easily which of those words should be chosen as keywords. An explanation for why the sole candidate phrases are not sufficient, is that since we do not use any frequency statistics, we cannot infer which words are more or less common than others and thus approximate the relevance.

To cope with that we could visualise the whole corpus along the extracted words and phrases to simulate general language understanding. However, this proves to be not usable and mostly impractical. Although there exists the fastest t-SNE implementation so far called Barnes-Hut-SNE that is specifically designed for big dataset and utmost performance, we did not succeed to process the whole corpus of word vectors due to its huge size. We analysed the source code of t-SNE implementation and applied multiple optimisations, which improved the overall performance even more, however, it did not suffice. We identified the bottleneck of the computation, which was the search of k nearest neighbours. We found out that in case of our corpus of word vectors, the simple linear search is faster than the vantage-point tree used in Barnes-Hut-SNE. This is probably due to bigger ratio between number of dimensions (300) and number of words (3000000) and division by vantage-point does not help to speed up the algorithm in practice. However, computation of k nearest neighbours for each word in vocabulary would require roughly at least $(3*10^6)^2*300 = 2.7*10^{15}$ operations, which is impractically too much even with parallelisation over multiple processor cores.

Finally, we can use a golden mean of above approaches and enrich the feature vectors of candidate phrases in the article with feature vectors of k nearest neighbours present in the corpus. For each such unit phrase that we get, we increase its relevance relative to its cosine similarity to the query noun phrase. Thus, we sum up the total relevance for each unit phrase and are able to visualise the top 100 keywords.
Initially, we optimised our method to extract relevant words from articles on Washington Post website. The reason was that we intended to create a dataset of articles annotated with keywords. Although it turned out to be not suitable for our needs, it helped us to tune our method and not to overfit on the evaluation dataset used in this paper. We can see the t-SNE visualisation of top 100 most relevant words for a random article[2] produced by our method in Fig. 2.

We can see that visualisation of top 100 most relevant words forms several clusters. Most of these clusters hide at least one of the keywords. The red words are those keywords that are present in the keyword list on the webpage of the article. The blue ones are keywords that we would also manually and subjectively choose as the most relevant for this article and we can clearly see their relevance just from looking at the headline of the article. The absence of these words in the original list of keywords is probably due to SEO (search engine optimisation), since SEO probably focuses just on a few words and their variants.

[2] Available online at - http://www.washingtonpost.com/world/national-security/cybersecurity-poll-americans-divided-over-government-requirements-on-companies/2012/06/06/gJQAmWqnJV_story.html (last accessed on May 14, 2014)

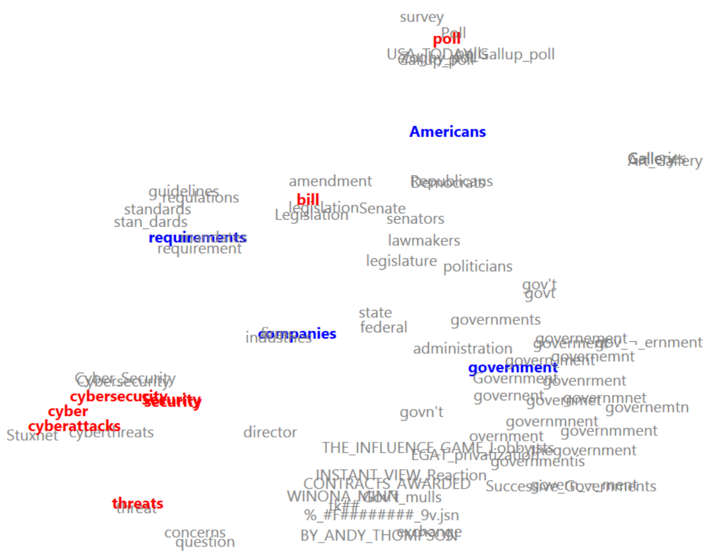

Fig. 2. Top 100 most relevant words computed by our method and visualised by t-SNE. Red words are keywords from website and blue words are keywords that we would also manually and subjectively choose as the most relevant for this article.

However, what is more significant for our method is that there are many words in used corpus, which represent only a noise (mostly typos). Thus, we reduced the original corpus to only 200 000 most frequent words. We calculated frequency of each word in corpus by processing whole Google N-gram dataset[3]. This cleaned up the corpus and also sped up further processing, which would be intractable in some cases. We can see that many words representing words with typos disappeared. We can see the result of this second alteration in Fig. 3.

As we can see, most of the keywords remained in the top, which is a positive result. However, there are still some stopwords, which represent determiners, prepositions, names and some letters that probably emerged from name initials present in the training data. Although the finite automaton described in section 3 ignores such stopwords present in text, after the selection of k nearest neighbours, there still emerge such stopwords. Therefore, we employed TF-IDF statistics to filter out these stopwords. We also removed all words shorter than 3 letters. As we can see in Fig. 4, it didn't clean out all stopwords (names) as discussed in previous paragraph, but still it removed a good portion of them (there are only 4 stopwords, which are all in one small cluster). However, it might be speculative if names are really stopwords, since there are some persons mentioned in the article as well. We can see that most words are more relevant than in the previous case, since they are more coherent and focused on given topic.

Based on analysis in this section, we finally used TF-RF weighting instead of TF-IDF, since it has been reported to give better results in text categorisation task [8].

[3] Available online at - http://storage.googleapis.com/books/ngrams/books/datasetsv2.html (last accessed on May 14, 2014)

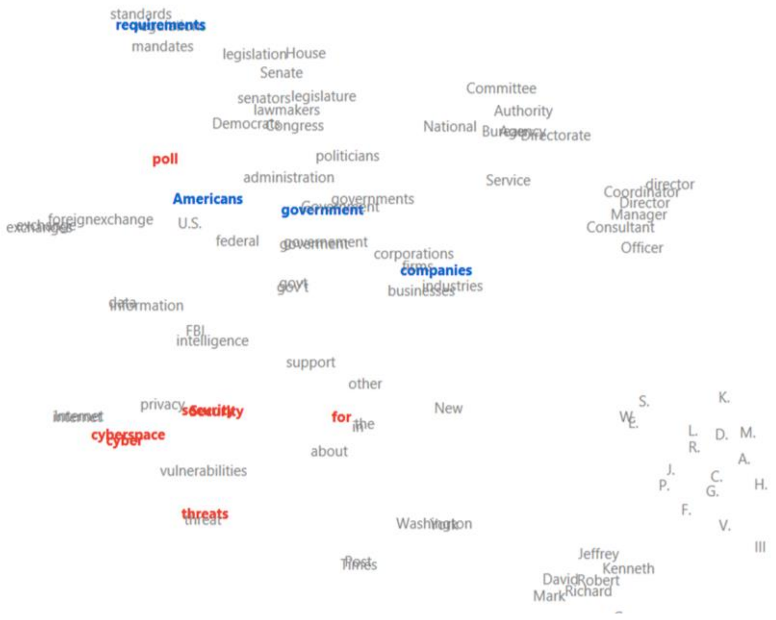

Fig. 3. Top 100 most relevant words computed by our method using top 200k word vector corpus and visualised by t-SNE.

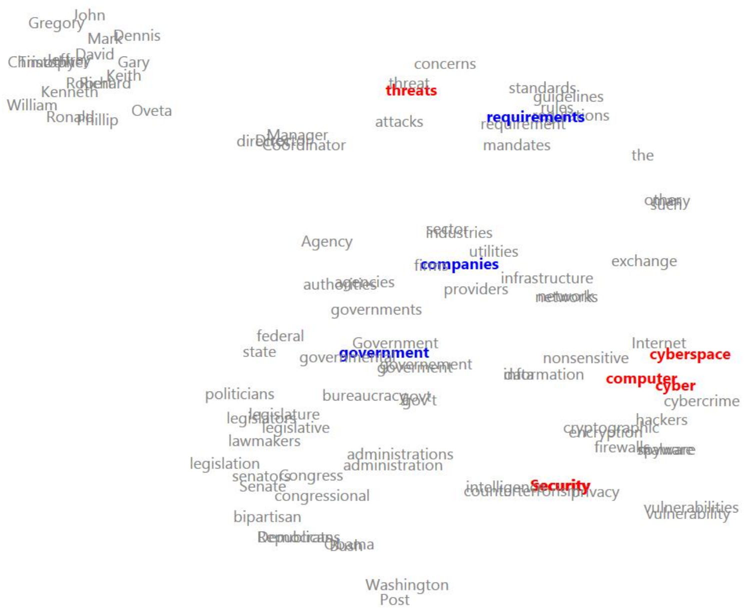

Fig. 4. Top 100 most relevant words computed by our method with TF-IDF weighting using top 200k word vector corpus and visualised by t-SNE.

6 Evaluation by text categorisation

We evaluated proposed method in text categorisation problem. We used 20-newsgroups dataset, which consists of 18 846 news documents divided into 20 categories. We used the version "by date" divided into training and testing sets, consisting of 11 314 and 7 532 documents respectively.

For each document, we computed one feature vector as a normalised sum of vectors of top relevant words output from our method. Thus, each document was expressed by 300 features. We tried different number of top words to create the feature vector of a document and also multiple common classifiers, which seemed most reasonable to us.

We tried using k-NN classifier, however, it didn't yield very good results (only below 80%). We used linear discriminant analysis presuming that each category can be expressed as a linear combination of features in a vector and thus not treating all features as equal could yield better results. Truly, we achieved better F1 score 82.85% for top 1200 words. We also applied linear SVM classifier, which we found having better performance in other researchers' work [8]. Our expectation was fulfilled by SVM yielding state-of-the-art performance of 84.5% micro-averaged F1 score (compared to approaches from section 2). We can see summarisation of achieved results in Figure 5.

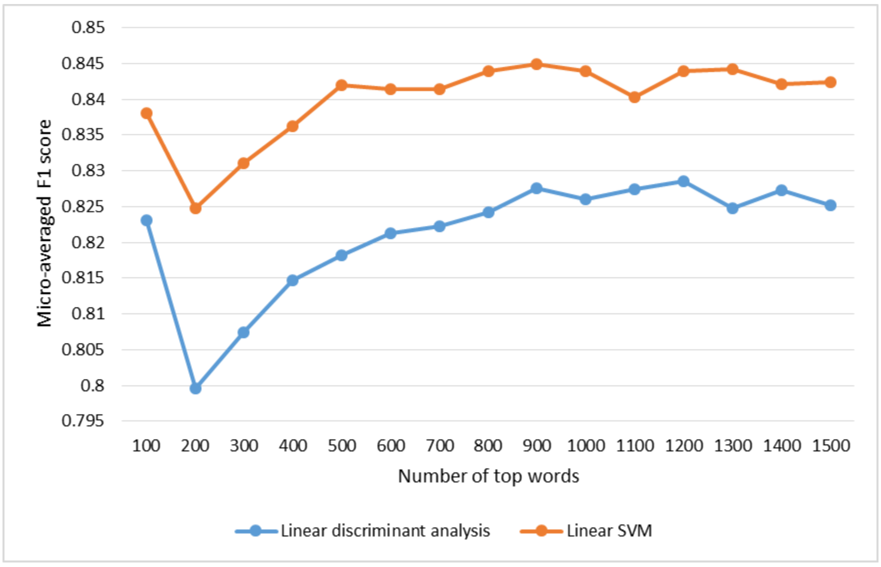

Fig. 5. Micro-averaged F1 score using different number of top words and different classifiers.

In results of SVM classification for sum of top 900 words (Table 1), we can see that the vast majority of the erroneous predictions fall into categories that are very similar to the predicted ones. We can see that categories like rec.sport.* and talk.politics.mideast that are rather different from the rest, were classified with very high precision. On the other hand, we can see that wrong predictions from categories comp.* mostly retained within these, since they are very similar. Also, misclassifications in categories about politics remain roughly within the respective categories. We can see that classification performance for articles in talk.religion.misc category is very poor, although majority of misclassification are classified into very similar categories – alt.atheism and soc.religion.christian, which are also about religion. Our explanation is that although we form better thematic representation by using higher number of top words to compute feature vector of a document, we are limited in differentiation of articles about miscellaneous religion topics, since they probably include also topics discussed in other two more specific categories about religion.

Table 1. Classification results using linear SVM and sum of top 900 word vectors used as a feature vector for document.

	alt.atheism	comp.graphics	comp.os.ms-windows.misc	comp.sys.ibm.pc.hardware	comp.sys.mac.hardware	comp.windows.x	misc.forsale	rec.autos	rec.motorcycles	rec.sport.baseball	rec.sport.hockey	sci.crypt	sci.electronics	sci.med	sci.space	soc.religion.christian	talk.politics.guns	talk.politics.mideast	talk.politics.misc	talk.religion.misc
alt.atheism	183	1	3	1	2	1	0	1	0	2	0	2	5	4	3	20	8	10	9	57
comp.graphics	4	273	34	18	14	58	9	1	3	1	0	1	18	16	4	1	2	0	2	1
comp.os.ms-windows.misc	4	19	211	33	30	49	3	0	0	0	0	2	16	2	0	4	0	0	1	2
comp.sys.ibm.pc.hardware	1	19	35	194	42	7	12	0	0	0	0	1	22	0	0	0	0	0	0	0
comp.sys.mac.hardware	1	10	30	56	202	6	8	0	0	0	0	3	14	0	2	1	0	1	1	1
comp.windows.x	1	19	41	10	7	230	1	0	1	1	0	2	7	1	1	0	1	0	1	1
misc.forsale	3	11	9	17	27	7	314	1	0	2	0	3	9	4	2	1	3	0	3	2
rec.autos	0	3	3	1	2	0	15	357	36	0	0	1	10	3	1	0	2	0	0	3
rec.motorcycles	10	1	3	2	2	2	5	4	321	1	0	2	5	2	1	1	3	0	3	4
rec.sport.baseball	3	3	0	2	3	2	5	0	5	376	6	1	1	2	1	0	1	4	2	0
rec.sport.hockey	0	2	7	1	1	1	1	0	2	10	390	2	3	1	0	1	2	0	0	2
sci.crypt	3	7	6	4	6	4	0	0	2	0	0	321	23	0	1	0	8	3	5	0
sci.electronics	1	12	2	49	32	8	9	15	7	1	1	18	245	5	3	1	2	1	1	1
sci.med	7	1	2	0	1	2	2	1	0	1	0	3	6	338	6	2	4	0	5	8
sci.space	11	1	4	4	5	3	0	2	1	0	0	1	2	2	360	0	2	0	7	9
soc.religion.christian	48	1	0	0	1	1	1	0	0	1	0	0	0	2	0	356	2	1	2	93
talk.politics.guns	8	0	0	0	3	0	3	10	7	1	0	7	2	3	1	2	298	2	90	34
talk.politics.mideast	8	4	1	0	0	1	1	0	0	0	0	3	0	1	0	0	2	344	3	1
talk.politics.misc	14	1	2	0	4	1	0	3	7	0	1	15	2	6	6	0	20	8	170	5
talk.religion.misc	9	1	1	0	1	12	1	1	6	0	1	8	3	4	2	8	4	2	5	27

7 Conclusions

We can see in this paper that word vectors can significantly help in the task of relevant words extraction. We have presented a working method of extracting relevant words based on the feature vectors of words. We evaluated our method in text categorisation and succeeded to achieve state-of-the-art results, which we consider the main contribution of our work.

The method we have proposed in this paper has wide application. Basically, relevant words can be used as a metadata for compact description of a document. Further, such metadata can be used for indexing purposes and can be utilise to ease search in document collections. Perhaps the most active area is web, since it contains ever growing huge amounts of web documents that need to be organised. We have shown that extracted relevant words are good for categorisation, but that also implies that the underlying representation captures the semantics of documents quite accurately.

On web, the application is twofold. In the first case, it can be used on the "Wild Web" to categorise web pages by their topic, e.g. in personalised search [7], or more broadly to assist in regular keyword-based web search. Alternatively, it can be utilised also in some specific domains, not just to help categorise new documents or to facilitate the search using keywords where we are aware of the topical structure of the domain, but even to create the topical structure of the domain from scratch, based solely on the documents in a collection and thus let the data speak itself.

In our future research, we plan to focus on more thorough analysis of vector representation of words for the purpose of text classification, where we have found a great open space for further research. We would also like to research utilisation of vector representation in other tasks, which till now, used manually crafted semantic representations like taxonomies.

Acknowledgement: This work was partially supported by grants No. VG1/0675/11, APVV-0208-10 and it is the partial result of the Research and Development Operational Programme project "University Science Park of STU Bratislava", ITMS 26240220084, co-funded by the European Regional Development Fund.

References

[1] Barla, M., Bieliková, M.: On Deriving Tagsonomies: Keyword Relations Coming from Crowd. In: *Proc. of the 1st Int. Conf. on Computational Collective Intelligence. Semantic Web, Social Networks and Multiagent Systems*, Springer-Verlag, 2009, pp. 309–320.

[2] Collobert, R., Weston, J.: A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning. In: *Proc. of the 25th International Conference on Machine Learning*, ACM, 2008, pp. 160-167.

[3] Fara. D. G., Russell, G.: The Routledge Companion to Philosophy of Language, Routledge, 2013, pp. 92, ISBN: 978-0-203-20696-6.

[4] Giesbrecht, E.: In Search of Semantic Compositionality in Vector Spaces. In: *Proc. of the 17th International Conference on Conceptual Structures: Conceptual Structures: Leveraging Semantic Technologies*, Springer-Verlag, 2009, pp. 173–184.

[5] Harris, Z. S.: Distributional structure, In: *Word*, Vol. 10, No. 23, 1954, pp. 146-162.

[6] Hinton, G. E., McClelland, J. L., Rumelhart, D. E.: Distributed representations. In: *Parallel distributed processing: Explorations in the microstructure of cognition*, Vol. 1: Foundations, MIT Press, 1986, pp. 77-109.

[7] Kramár, T., Barla , M., Bieliková, M.: Personalizing search using socially enhanced interest model, built from the stream of user's activity. In: *Journal of Web Engineering*, Vol.12, No. 1-2, 2013, pp. 65-92.

[8] Lan, M., Tan, C., Low, H.: Proposing a New Term Weighting Scheme for Text Categorization. In: *Proc. of the 21st national conference on Artificial intelligence - Volume 1*, AAAI Press, 2008, pp. 763-768.

[9] Larochelle, H., Bengio, Y.: Classification using Discriminative Restricted Boltzmann Machines. In: *Proc. of the 25th International Conference on Machine Learning*, ACM, 2008, pp. 536-543.

[10] Li, B., Vogel, C.: Improving Multiclass Text Classification with Error-Correcting Output Coding and Sub-class Partitions. In: *Proc. of the 23rd Canadian Conference on Advances in Artificial Intelligence*, Springer-Verlag, 2010, pp. 4-15.

[11] Van der Maaten, L.J.P., Hinton, G.E.: Visualizing High-Dimensional Data Using t-SNE. In: *Journal of Machine Learning Research*, Vol. 9 (Nov), 2008, pp. 2579-2605.

[12] Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J.: Distributed Representations of Words and Phrases and their Compositionality. In: *Advances in Neural Information Processing Systems 26*, Curran Associates, 2013, pp. 3111-3119.

[13] Mikolov, T., Yih, W., Zweig, G.: Linguistic Regularities in Continuous Space Word Representations. In: *Proc. of NAACL HLT*, ACL, 2013, pp. 746-751.

[14] Mitchell, J. and Lapata. M.: Vector- based Models of Semantic Composition. In *Proc. of the 46th Annual Meeting of the Association for Computational Linguistics*, ACL, 2008, pp. 236–244.

[15] Socher, R., Bauer, J., Manning, C. D., Ng, A. Y.: Parsing With Compositional Vector Grammars. *Proc. of the 51st Annual Meeting of the Association for Computational Linguistics*, ACL, 2013, pp. 455-465.

[16] Šajgalík, M., Barla, M., Bieliková, M.: From Ambiguous Words to Key-Concept Extraction. In: *Proc. of the 24th International Workshop on Database and Expert Systems Applications*, IEEE, 2013, pp. 63-67.

[17] Vu, T., Aw, A. T., Zhang, M.: Term Extraction Through Unithood And Termhood Unification. In: Proc. of the Third International Joint Conference on Natural Language Processing, ACL, 2004, pp. 631–636.

[18] Wang, D., Zhang, H.: Inverse-category-frequency based supervised term weighting scheme for text categorization. arXiv preprint arXiv:1012.2609, 2010.