An approach to the use of word embeddings in an opinion classification task

Fernando Enríquez [*], José A. Troyano, Tomás López-Solaz

University of Seville, Department of Languages and Computer Systems, ETS Ingeniería Informática, Av. Reina Mercedes s/n, 41012 Sevilla, Spain

Keywords:
Document classification
Opinion classification
Word embedding
Bag of words
Word2vec

A B S T R A C T

In this paper we show how a vector-based word representation obtained via word2vec can help to improve the results of a document classifier based on bags of words. Both models allow obtaining numeric representations from texts, but they do it very differently. The bag of words model can represent documents by means of widely dispersed vectors in which the indices are words or groups of words. word2vec generates word level representations building vectors that are much more compact, where indices implicitly contain information about the context of word occurrences. Bags of words are very effective for document classification and in our experiments no representation using only word2vec vectors is able to improve their results. However, this does not mean that the information provided by word2vec is not useful for the classification task. When this information is used in combination with the bags of words, the results are improved, showing its complementarity and its contribution to the task. We have also performed cross-domain experiments in which word2vec has shown much more stable behavior than bag of words models.

1. Introduction

When we apply machine learning algorithms to natural language processing tasks, an initial phase of feature extraction is always necessary, allowing us to convert text into numerical vectors, which are the data handled by these type of algorithms.

For example, in a document classification task the bag of words (BOW) technique is the most frequently used to make this transformation from text to numbers. In the BOW model the documents are represented by vectors in which each dimension corresponds to a word or group of words, generating vectors with a very high dimensionality. It is an exclusively lexical representation that relies on metrics based on frequency to determine the values associated with each dimension of the vector.

However, there are other alternatives for accomplishing this transformation from text to numbers. In general, the term *word embedding* refers to those techniques that allow us representing the words of a certain vocabulary in a continuous vector space of a dimension substantially lower than the size of the vocabulary. These techniques manage to extract a profile of the words in an unsupervised way taking into account simply the contexts in which these words appear. Unlike models based on bags of words, word embedding techniques cross the lexical frontier because the representations of words capture syntactic and semantic aspects of them.

One of the word embedding tools that is gathering more attention among the scientific community in recent years is word2vec. It is a model based on neural networks and related to some of the elements that are behind the recent rise of the so-called deep learning.

In this paper the interest is focused on evaluating the usefulness of the representations provided by word2vec in the classification of documents. Our main goal is to determine if the knowledge offered by word2vec about words can complement the information provided by a very stable model for the classification task, such as the BOW model. We conducted our experiments in a polarity classification task using a collection of texts from Amazon that covers eleven different domains. The results of these studies show that although the representation of word2vec is not able to improve the BOW results on its own, there is a contribution of complementary information. That is why in models where word2vec and BOW representations are integrated, there is an improvement in the results compared to a classifier based solely on bags of words. We also tested the adaptability of our algorithms by doing cross-domain classification experiments (training with texts of one domain and evaluating on texts from other domains). In this case, the contribution of word2vec is even more interesting, showing

* Corresponding author.
E-mail addresses: fenros@us.es (F. Enríquez), troyano@us.es (J.A. Troyano), tlopez2@us.es (T. López-Solaz).

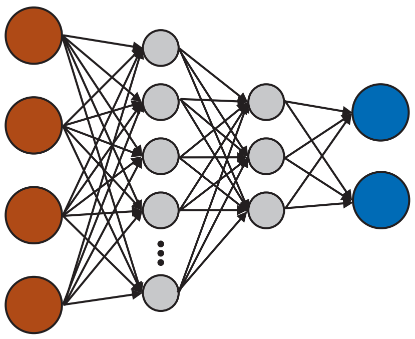

Fig. 1. Neuronal network.

the usefulness of a more semantic representation in such a more demanding situation.

The rest of the paper is organized as follows. Section 2 presents the basic concepts of WORD2VEC, Section 3 introduces the different document representations that we used in our experiments, Section 4 includes the experimental design and the results we have obtained from our experiments. Finally, Section 5 contains the conclusions of our work.

2. Word embeddings and WORD2VEC

WORD2VEC was developed by Mikolov, Chen, Corrado, and Dean (2013a); Mikolov, Sutskever, Chen, Corrado, and Dean (2013b); Mikolov, Yih, and Zweig (2013c) for representing words as vectors and implemented by Google, which released a version of the code for academic purposes. Although WORD2VEC is not a deep learning technique, it is usually linked with the term *deep learning*. This is due to the fact that some of the elements related to deep learning are part of the training scheme of WORD2VEC. In this section we summarize the main ideas behind WORD2VEC and its relationship with neural networks and deep learning.

2.1. Neural networks

Neural networks, also known as artificial neural networks, are a mathematical tool based on a directed graph structure, which takes as input a series of numerical values and produces as output another series of numerical values. The basic element of neural networks is the perceptron, which can be considered as the mathematical modeling of neurons. The perceptron takes as input several numerical values and applies a function on them to produce an output. This function is called activation function and can be non-linear allowing the network to learn how to solve problems that are not linearly separable. Neural networks are organized in layers of perceptrons as shown in Fig. 1. Each connection has an associated weight that is used to multiply the output value of neurons. Each neuron will add all values received as input and apply the activation function to produce its output.

One of the most common training methods for neural networks is called *backpropagation* and estimates the values of the output weights of the network neurons. It involves applying an input to the network and compare the obtained output with the expected output. The difference between the two will be considered the network error or loss and is propagated backwards from the output to correct the weights of each neuron. In very deep networks (with many layers) the corrections made to the weights keep decreasing as we get further from the output and thus only the final layers are well trained, while the first ones hardly suffer changes during the process. Indeed one of the reasons why the concept of deep learning has gained so much attention in recent years is because it provides a solution to this limitation of backpropagation learning.

2.2. Deep learning and autoencoders

One of the most important ideas related to deep learning is the use of autoencoders to train layers of neurons. The autoencoders are neural networks that generally have a single hidden layer that are trained to produce the same output as the input they receive. This feature, that seems so useless at a first glance, has two compensatory effects that make autoencoders particularly interesting. First, it opens the door to unsupervised learning because to train an autoencoder there is no need for labeled data, as the expected output is the same as the input. For example, to train an autoencoder to process images to later identify if there are faces in it or not, it is not necessary to annotate what images contain faces. This allows training the autoencoders with very large datasets as the amount of unlabeled data on Internet (texts, images,...) is immense.

The second benefit of autoencoders has to do with what they learn: they learn to generate a different representation of the inputs that they process. If we train an autoencoder with images, the resulting hidden layer will consist of neurons specialized in identifying some aspect often repeated in the input images. Discarding the output layer, the autoencoder becomes an encoder that provides an alternative representation that has been generated from the observation of large amounts of data.

The idea of deep learning using autoencoders takes advantage of these two characteristics mentioned. Autoencoders are trained in an unsupervised way (with large amounts of unlabeled data) one by one, the output layers are discarded and they are stacked to build the first layers of a neural network. Once these layers are trained, a smaller set of labeled data is used for supervised training of the network layers that are closest to the output.

2.3. Representation provided by WORD2VEC

WORD2VEC works in a similar way to autoencoders, which means it can be trained using large amounts of unlabeled text. Given a vocabulary V, inputs and outputs of the autoencoder are vectors with $|V|$ dimensions that represent words and contexts of words through the *one hot encoding* technique (1 means that the word for that dimension is present and 0 that the word is not present in the text).

Unlike autoencoders, WORD2VEC is not trained to produce the same output as the input. In this case, words are faced with their context. The two ways to do this pairing produce two training models: CBOW and Skipgram. Figs. 2 and 3 show the difference between the two models. In CBOW a word is used as the output and its context as input, while in the Skipgram model it is done the other way around.

Both models learn the underlying relationship between words and their context. Skipgram works better for smaller amounts of training texts, while the advantages of CBOW are the increased training speed and a higher quality of the representation for frequent words.

As for autoencoders, the most valuable element of WORD2VEC is the middle layer. It contains many fewer neurons than the size of the vocabulary and provides a vector encoding mechanism for words.

By capturing the relationships between words and their context, WORD2VEC is able to represent words that are semantically close with vectors that are also close to each other. Another interesting feature of the resulting vectors is that their spatial properties are consistent with the semantics of the words they represent. Therefore, applying arithmetic operations to vectors of some words

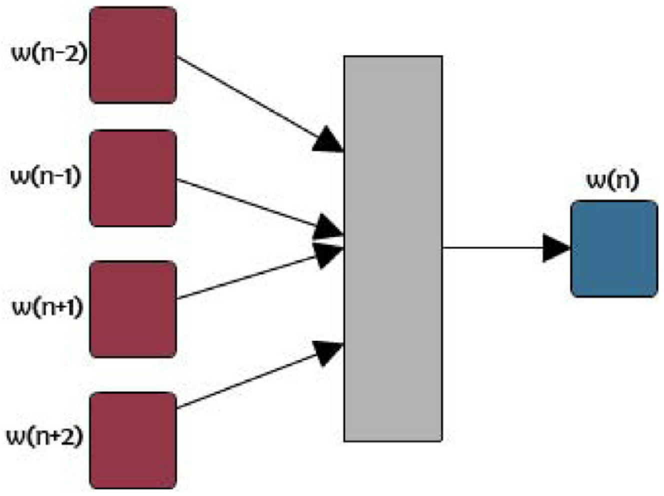

Fig. 2. CBOW model.

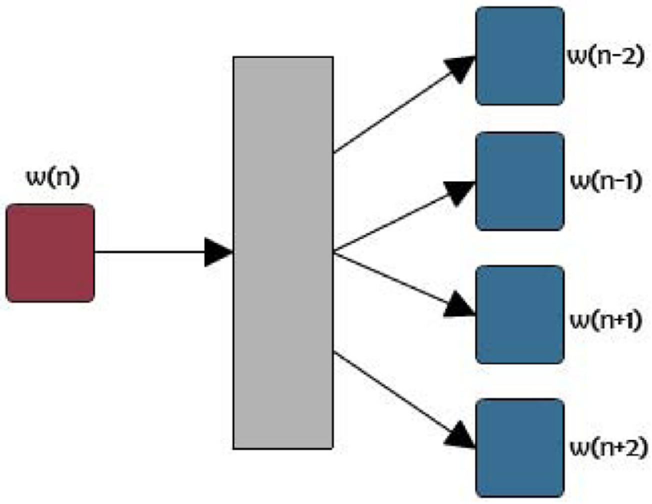

Fig. 3. Skipgram model.

we obtain vectors that are close to other words that are related to the first ones. For example, the difference between the vectors of the words *king* and *man* is very similar to the difference between the vectors of the words *queen* and *woman*. This ability to capture the semantics of words and the relationships between them is the reason why more and more researchers in the field of natural language processing include in their systems knowledge extracted from this type of tools.

Although WORD2VEC has become very popular, it is worth noting that many researchers develop their own word embedding methods, for example to integrate other types of information, like metadata or author features that are relevant to their task (Kiros, Zemel, & Salakhutdinov, 2014b; Yang & Mao, 2016). Word embeddings have also been applied to user segmentation (Boratto, Carta, Fenu, & Saia, 2016), knowledge representation (Bordes, Usunier, Garcia-Duran, Weston, & Yakhnenko, 2013; Socher, Chen, Manning, & Ng, 2013), data stream mining (Djuric, Wu, Radosavljevic, Grbovic, & Bhamidipati, 2015), information retrieval (Grbovic, Djuric, Radosavljevic, Silvestri, & Bhamidipati, 2015; Kiros, Salakhutdinov, & Zemel, 2014a), question answering (Yang, Lee, Park, & Rim, 2015) and social network analysis (Perozzi, Al-Rfou, & Skiena, 2014) among other tasks.

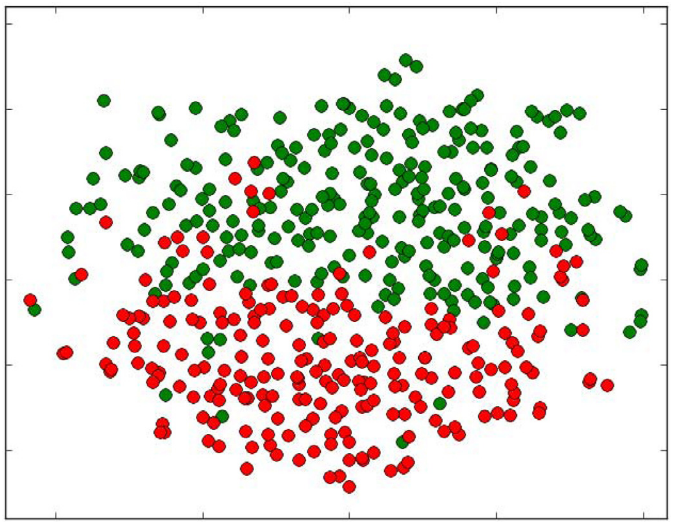

Fig. 4. Positive words are represented by green points and negative words by red points.

3. Representation of documents

The method most frequently used to represent text in a vector form is the bag of words or BOW, mentioned above. The resulting vector for each analyzed text, which can be a document, a paragraph or a sentence according to the purpose of the system, is based on a dictionary. Each word in the corpus receives an *id* that corresponds with a position in the vector representing that dictionary. The encoding of the value of each dimension is based on the number of occurrences of the corresponding word in the text. At this point, there are multiple ways to carry out this encoding, ranging from assigning a one in case the word appears and zero in the opposite case, to more complex methods that take into account, for example, the relative frequency of the word.

In our experiments the *tf-idf* measure has been applied, whose value increases with the number of occurrences of the word in the text, but decreases with the number of occurrences in the whole corpus. Therefore, the relevance of each term is better captured considering there are words that appear much more frequently than others.

Besides the bag of words classical representation, we have the encoding of words provided by WORD2VEC. In this method of unsupervised learning we start choosing between the *Skipgram* and *CBOW* algorithms (we selected *Skipgram* as explained in Section 4.2), setting the number of features and vector components, and generating the model. After the training phase, words that share the same semantic orientation are close in the vector space as we can see in Fig. 4. In this figure, generated by the t-distributed stochastic neighbor embedding (t-SNE) visualization algorithm, vectors generated by WORD2VEC are represented in a two dimension plot for a set of positive and negative words.

Using WORD2VEC for document-level tasks requires a method to unify all word vectors generating a single vector representing the entire document. After testing several aggregate functions, the arithmetic mean was chosen for this task. Thus, the final representation was obtained by adding all the word vectors and dividing by the number of words contained in the document (Eq. 1).

$$\vec{V_d} = \frac{\sum_{i=0}^{n} \vec{v_i}}{n} \qquad (1)$$

4. Experimental design and results

In order to evaluate the use of the WORD2VEC vector representation for document classification, we designed an experimental phase in which we compared the results of two base classifiers trained respectively with the BOW and WORD2VEC representations, with a third classifier obtained through the combination of the first two. A logistic regression model was trained for each base classifier. In the following sections the details of the experiments and their results are described.

4.1. Corpus

The corpus chosen to test our system is a set of user-generated reviews referring to articles from different domains as explained in McAuley, Targett, Shi, and van den Hengel (2015b) and McAuley, Pandey, and Leskovec (2015a). It is a collection used in many papers, such as Li and Zong (2008), Bollegala, Weir, and Carroll (2013) and Franco-Salvador, Cruz, Troyano, and Rosso (2015). The dataset is organized into different domains, from which we have used eleven, the ones providing a minimum of 2500 positive reviews and another 2500 negative, all randomly selected. While all domains contain texts of the same nature, there are differences in the type of terms used that affect the performance of the classifiers as discussed later. When assigning polarity to the opinions of this dataset, and taking into account that the opinions are classified with stars, we will consider as positive reviews those with three or more stars and the ones with less than three will be marked as negative.

4.2. WORD2VEC model

To obtain the vector representation of words by the WORD2VEC method, the Skipgram model has been chosen and trained with a Google News dataset in English (Mikolov et al., 2013b) that Google makes available to the community on the project website. It has been done with a WORD2VEC implementation developed in Python (Řehůřek & Sojka, 2010), using the default values for almost all parameters, with vectors with 300 dimensions and 25 training epochs starting with a learning rate of 0.25.

The classification of documents is done with a supervised scheme, as it is performed by the BOW method, although using in this case the average of the WORD2VEC vectors for every word in the analyzed text. Other aggregation functions were tested, like the sum of the vectors or a weighted average considering the *tf-idf* value of each term, but none of them improved the results.

4.3. Classification

All the vectors representing the documents of the corpus are grouped together, resulting in two datasets, one containing the vectors provided by the BOW method, and another one with the WORD2VEC vectors. These datasets, and the correct tag given by the corpus for each document, will be the input for a classification algorithm that will generate a model for each method. We used the scikit-learn (Pedregosa et al., 2011) python library implementation of the logistic regression algorithm with the default parameters. Ten-fold cross-validation was applied in both cases, and also in the combination method that will be explained next.

4.4. Combination

Once the classifiers based on the BOW and WORD2VEC methods were built, we developed a classification system combining both approaches with the aim of exploiting the complementarity we believe exists between the information encoded in the BOW vectors and in the WORD2VEC vectors. The combination method consists of a simple voting system based on the confidence values returned by each method (see Fig. 5). Since these values are between zero and one, the average of both is calculated to determine if it is labeled as positive in the case of being between 0.5 and 1 and negative otherwise. Besides this combination method, we also carried out experiments using stacked generalization (Wolpert, 1992), but the weighted voting method obtained the best results.

Table 1
Results of the classifiers.

	BOW	W2V	Comb	Δ
Apparel	77.04	75.06	**78.08**	1.04
Books	73.52	72.74	**74.64**	1.12
Camera	**87.58**	72.90	86.06	−1.52
DVDs	75.04	73.30	**76.12**	1.08
Electronics	**89.90**	73.56	88.52	−1.38
Kitchen	77.92	75.34	**79.32**	1.40
Music	73.00	71.82	**73.86**	0.86
P. Care	76.12	71.08	**76.66**	0.54
Sports	73.90	71.42	**75.04**	1.14
Toys	78.10	72.58	**78.22**	0.12
Video	80.66	72.14	**81.02**	0.36
Mean				0.43

4.5. Cross-domain classification

In addition to assessing the complementarity between the two vector encodings, we conducted experiments to evaluate the ability of the classifiers to adapt to a new domain.

In these experiments we process texts with a classifier that has been trained with texts of a different domain. Given that we have 11 domains, for each of them we have another 10 domains for training every classifier. We consider as the final result for that domain the average of the results obtained by those 10 classifiers of the same type that have been trained with a domain that differs from the target one.

4.6. Results

In Table 1 we can see the results in terms of the percentage of accuracy obtained by the three versions of classifiers for each domain. The classic version of the classifier based on bag of words (BOW) outperforms the one based on the WORD2VEC method (W2V), but as shown in the penultimate column (Comb), the combination of the two classifiers gives the best results in 9 of the 11 domains. That is, all except 'Camera' and 'Electronics', where the performance difference between the BOW and W2V classifiers looks as if it were too big to be compensated by a simple combination method based on voting. Anyway, it seems logical to think that there exists complementarity in the information contained in each type of vector encoding, and that is the reason why the combination achieves good results. Finally, it has been added to the table one last column with the improvement obtained by the combination method, including the average of this value considering all analyzed domains.

Noting the big loss accumulated by the classifier based on WORD2VEC in comparison with the BOW classifier in the 'Camera' and 'Electronics' domains, we decided to further analyze the contents of these texts. We found that there is a large number of infrequent words in these texts that are very informative for the BOW classifier to correctly infer the output class. To test the impact of such terms we performed an experiment where several runs were made varying progressively the percentage of infrequent words, resulting in Fig. 6. It shows how the performance of the BOW classifier decreases to a greater extent in the 'Camera' and

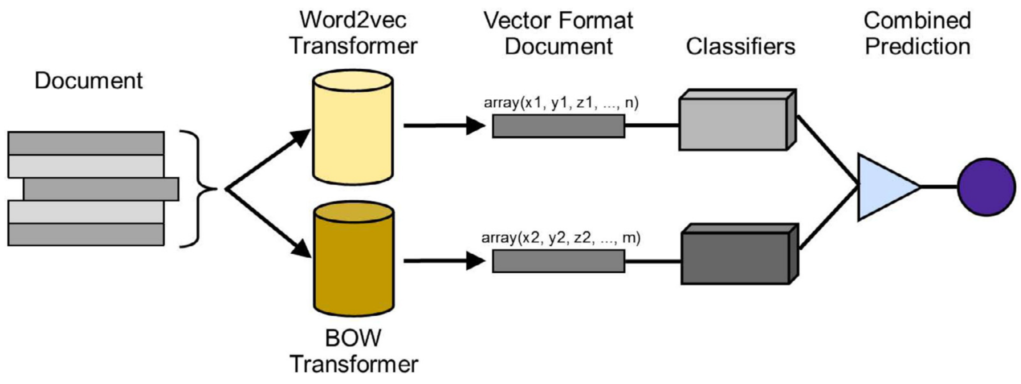

Fig. 5. Combined model using a voting system.

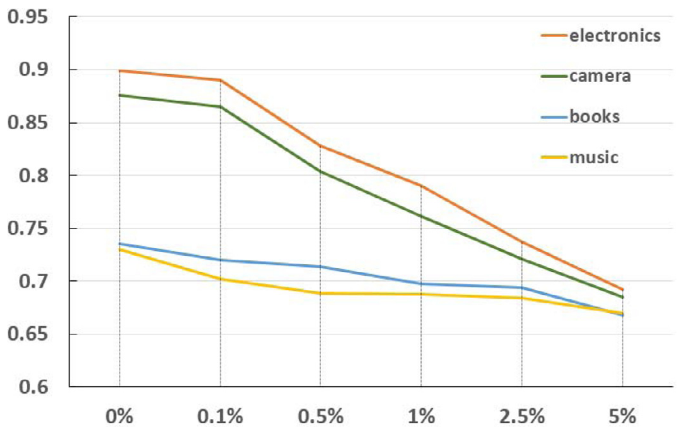

Fig. 6. BOW performance with varying percentage of infrequent terms filtered.

Table 2
Cross-domain results.

	BOW	W2V	Comb	Δ
Apparel	71.64	71.78	**73.48**	1.7
Books	66.90	68.74	**69.49**	0.75
Camera	67.50	64.91	**67.64**	0.14
DVDs	70.16	69.98	**72.07**	1.92
Electronics	**67.91**	64.41	67.59	−0.31
Kitchen	72.33	70.92	**73.50**	1.17
Music	67.94	68.53	**69.88**	1.36
P. Care	69.38	67.44	**70.05**	0.67
Sports	68.34	67.52	**69.62**	1.28
Toys	70.78	70.77	**72.40**	1.62
Video	69.05	68.36	**70.39**	1.33
Mean				1.06

'Electronics' domains as we go progressively excluding a greater number of infrequent words. In the graph we show the results of 'Camera' and 'Electronics' along with the 'Books' and 'Music' domains, which are the domains where the difference between the BOW and W2V methods is smaller. The conclusion we can draw is that for 'Camera' and 'Electronics' BOW behaves particularly well because there are many domain specific terms (such as model names or technical characteristics of the products) and that are not included in a generic corpus as the one that was used to build the W2V model. These terms are filtered out when we remove the low frequency terms and this is the reason why the performance of the BOW classifier decreases so significantly in these domains, while in other domains with a more general vocabulary, like 'Books' and 'Music', the filtering of infrequent words has a very low impact on the results.

Regarding the experiments conducted in the field of *cross-domain*, the results obtained are shown in Table 2. The combination method (COMB) obtains the best percentages of accuracy in 10 of 11 domains. In this case it is able to surpass the BOW classifier in the 'Camera' domain, leaving 'Electronics' as the only one where the BOW method obtains the best result. This confirms that our classification scheme is more robust than the BOW method alone, providing better performance in unfavorable situations where the lack of resources prevents us from training with annotated texts of the target domain.

5. Conclusions

Vector representations of words offer the possibility of applying machine learning algorithms for text classification. The rise of deep learning in recent years and, more specifically, the encoding that makes WORD2VEC by autoencoders, opens a new way to explore the utility of these type of representations in front of the 'classic' method based on bags of words (BOW).

In this paper we wanted to study these two types of vector representations focusing on the complementary nature of the information provided by them and their combination through a voting system. We have experimented with a set of opinion texts grouped into eleven different domains. The results confirm that both representations have complementary information. In nine of the eleven domains the combined classifier obtains the best accuracy compared to the two individual classifiers alone, where the BOW method clearly outperforms the classifier based on WORD2VEC.

The 'Camera' and 'Electronics' domains, the only ones in which the combined classification does not achieve the best results, have been studied in more detail. A content analysis of their texts reveals particularities concerning the infrequent words that can be found in them, favoring in our opinion the purely lexical approach of the BOW classifier. After executing this classifier using a filter of infrequent words, we obtained a graph showing how the results of this classifier vary depending on the degree of tolerance of the filter, evidencing the disparity compared to other domains in the corpus.

Finally we have conducted experiments in the field of cross-domain, executing the classifiers on texts of a different domain of those used in the training phase. Again, the classifier that combines BOW and WORD2VEC gets the best results, this time in ten out of the eleven domains, excluding only the 'Electronics' domain. Surpassing the BOW method in the 'Camera' domain the combined classifier demonstrates that it manages to reduce the advantage that BOW obtains in some domains because of the appearance of infrequent words that are highly correlated with the class. Moreover, the combined method achieves more than one point of improvement in average, showing itself as the most robust option for cases where limited resources do not allow training with labeled texts of the same domain.

Acknowledgements

This work has been supported by the project AORESCU (P11-TIC-7684 MO).

References

Bollegala, D., Weir, D., & Carroll, J. (2013). Cross-domain sentiment classification using a sentiment sensitive thesaurus. *Knowledge and Data Engineering, IEEE Transactions on, 25*(8), 1719–1731.

Boratto, L., Carta, S., Fenu, G., & Saia, R. (2016). Using neural word embeddings to model user behavior and detect user segments. *Knowledge-Based Systems*.

Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., & Yakhnenko, O. (2013). Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems* (pp. 2787–2795).

Djuric, N., Wu, H., Radosavljevic, V., Grbovic, M., & Bhamidipati, N. (2015). Hierarchical neural language models for joint representation of streaming documents and their content. In *Proceedings of the 24th international conference on world wide web* (pp. 248–255). ACM.

Franco-Salvador, M., Cruz, F. L., Troyano, J. A., & Rosso, P. (2015). Cross-domain polarity classification using a knowledge-enhanced meta-classifier. *Knowledge-Based Systems*.

Grbovic, M., Djuric, N., Radosavljevic, V., Silvestri, F., & Bhamidipati, N. (2015). Context-and content-aware embeddings for query rewriting in sponsored search. In *Proceedings of the 38th international acm sigir conference on research and development in information retrieval* (pp. 383–392). ACM.

Kiros, R., Salakhutdinov, R., & Zemel, R. S. (2014a). Multimodal neural language models. In *Icml: 14* (pp. 595–603).

Kiros, R., Zemel, R., & Salakhutdinov, R. R. (2014b). A multiplicative model for learning distributed text-based attribute representations. In *Advances in neural information processing systems* (pp. 2348–2356).

Li, S., & Zong, C. (2008). Multi-domain sentiment classification. In *Proceedings of the 46th annual meeting of the association for computational linguistics on human language technologies: Short papers* (pp. 257–260). Association for Computational Linguistics.

McAuley, J., Pandey, R., & Leskovec, J. (2015a). Inferring networks of substitutable and complementary products. In *Proceedings of the 21th acm sigkdd international conference on knowledge discovery and data mining* (pp. 785–794). ACM.

McAuley, J., Targett, C., Shi, Q., & van den Hengel, A. (2015b). Image-based recommendations on styles and substitutes. In *Proceedings of the 38th international acm sigir conference on research and development in information retrieval* (pp. 43–52). ACM.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013a). Efficient estimation of word representations in vector space. arXiv:1301.3781.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems* (pp. 3111–3119).

Mikolov, T., Yih, W.-t., & Zweig, G. (2013c). Linguistic regularities in continuous space word representations. In *Hlt-naacl* (pp. 746–751).

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., … Duchesnay, E. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research, 12*, 2825–2830.

Perozzi, B., Al-Rfou, R., & Skiena, S. (2014). Deepwalk: Online learning of social representations. In *Proceedings of the 20th acm sigkdd international conference on knowledge discovery and data mining* (pp. 701–710). ACM.

Řehůřek, R., & Sojka, P. (2010). Software framework for topic modelling with large corpora. In *Proceedings of the lrec 2010 workshop on new challenges for nlp frameworks* (pp. 45–50). Valletta, Malta: ELRA. http://is.muni.cz/publication/884893/en

Socher, R., Chen, D., Manning, C. D., & Ng, A. (2013). Reasoning with neural tensor networks for knowledge base completion. In *Advances in neural information processing systems* (pp. 926–934).

Wolpert, D. H. (1992). Stacked generalization. *Neural networks, 5*(2), 241–259.

Yang, M.-C., Lee, D.-G., Park, S.-Y., & Rim, H.-C. (2015). Knowledge-based question answering using the semantic embedding space. *Expert Systems with Applications, 42*(23), 9086–9104.

Yang, X., & Mao, K. (2016). Learning multi-prototype word embedding from single-prototype word embedding with integrated knowledge. *Expert Systems with Applications, 56*, 291–299.