Sentiment Analysis Using Convolutional Neural Networks

Evangelia Gogoulou Gogoulou@kth.se
Thomas Peterson Thpeter@kth.se
Magdalini Papaioannou Magpap@kth.se

Kungliga Tekniska Högskolan

Abstract. Sentiment analysis has quickly become a rapidly growing field of study and has proved to be applicable in numerous cases. For example for task related to business, marketing and political analysis. In this work, sentiment analysis was conducted with convolutional neural network based architecture. The architecture presented in Yoon Kim's master thesis was used as a reference point [1]. The goal of the work was to explore different network architecture modifications that could lead to accuracy improvement. Experiments were conducted using different filter heights and numbers of convolutional filters, on datasets of Twitter tweets and movie reviews. The non-static word embedding option was also explored. Better accuracy than Kim's was obtained with minor architecture adjustments [1]. The results obtained suggests that filter height plays an important role in the network accuracy, while filter depth does not. It is important to note that different architectures seem to work better with different datasets.

Keywords: CNN, Word2Vec, Sentiment Analysis, NLP, SGD

1 Introduction

Machine learning has been a hot topic during the last couple of years. As more computational resources are readily available, the advent of deep neural networks, networks with multiple hidden layers, has been a big success[2]. Deep neural networks can be used for numerous tasks, one of these tasks is sentiment analysis.

Sentiment analysis is the task of identifying how sentiments are expressed in texts and whether the expressions indicate positive or negative opinions towards a subject. In general, it assigns some metric to a text which indicates how positive or negative it is with respect to some positivity scale.

Although sentiment analysis is a subjective matter, the accuracy of which can never be measured with 100% certainty, it has quickly become a rapidly growing field of study and has proved to be applicable in numerous cases, as it constitutes of a way to quantitatively analyze qualitative data[3]. First of all, it is used as a way of labeling unlabeled text data in order for this data to be used in machine learning systems such as recommender systems. Furthermore, paired with statistics it has been used with substantial benefits in different fields such as business, marketing and political analysis[4].

In this work a deep neural network is used for sentiment analysis. It classifies text samples into positive and negative rather than using a continuous metric. The input were extracts from movie reviews and twitter tweets with their corresponding labels. The goal of the work is to find a good architecture in terms of filter heights, number of filters and non-static vs static word representations.

2 Background

This chapter provides a background for understanding the approach taken in the work. The chapter starts by providing an overview over various techniques used for Sentiment Analysis and thereafter treats related work. After this, a section describes different word embeddings for Natural Language Processing. Finally, a section provides a high level description of Convolutional Neural Networks(CNNs) since they are a part of the network architecture later described in the approach chapter.

2.1 Types and Techniques for Sentiment Analysis

There are two levels of sentiment analysis; document level and sentence level[5]. Document level sentiment analysis attempts to classify a document as positive or negative while sentence level sentiment analysis classifies each sentence as positive or negative.

Approaches to the problem of sentiment analysis can be categorized into 4 categories[6]. These are keyword spotting, lexical affinity, statistical methods, and concept-based techniques. Keyword spotting works by searching for keywords from related to different emotions. This method is weak in the sense that certain sentences can be given the same score even if they mean opposite things. For example, it is likely that “today was a happy day”and “today wasn’t a happy day at all”receives the same positive score[6]. The second typical approach, lexical affinity ,builds on top of keyword spotting by assigning probabilities to every non-keyword where this probability defines if it is positive or not[6]. Albeit being more effective than keyword spotting, this approach can still not handle cases with negation and tricky sentences. An example of the latter is the sentence ”I met my girlfriend by accident”.

The third common approach is statistical methods. These includes usage of machine learning techniques such as support vector machines and Bayesian inference[6]. These methods are semantically weak and thus only work well with large amount of input data[6]. The last category of frequent approaches are concept-based approaches. These approaches are the usage of Web ontologies and semantic networks. These type of approaches have the advantage of being able to recognize subtly expressed sentiments [6].

In this paper a novel method, inspired by Yoon kim, is explored [1]. This approach is based on deep learning and can thus not be classified as one of the 4 common approaches.

2.2 Related Work

Multiple scientists and researchers have explored sentiment analysis with similarities to our work. For example, Agarwal et al conducts sentiment analysis on a twitter dataset by creating kernel trees(Trees representing tweets)[7]. An interesting aspect of their work is that they use an emoticon and acronym dictionary. This translates emoticons and acronyms to emotions. Furthermore, Ghorbel et al conducts sentiment analysis on french movie reviews by using POS tagging, chunking and simple negation forms[8]. They also make use of word semantic orientation acquired from the lexical resource SentiWordNet to improve classification accuracy. However, since SentiWordNet was created for the english language, they translate the french input into english before extracting the semantic orientation.

Other related work includes the work by Kumar et al where they not only do sentiment analysis but also spam detection[9].This is conducted with the use of machine learning techniques such as Binomial classification, CART and random forests. This work can thus be classified as the third common approach in section 2.1. Futher related work is that of Severyn et al, where they conduct analysis of tweets by implementing a deep convolutional neural network[10]. However, their work is different from ours in the sense that it focuses on the parameter initialization rather than the network architecture itself.

The experiments conducted in this paper were heavily inspired by those of Yoon kim [1]. Kim applies multiple convolutional filters with varying filter heights to the input. Thereafter, he stacks the feature maps produced by this operation on top of each other and feeds the result into a dense layer. He applies the softmax operator to the results. He uses word2Vec to encode the words of the sentences that are used as input to the network. Additionally, he studies 3 different implementations. The first one is a 1-channel implementation with static word vectors. The second one is a 1-channel implementation with non-static word vectors, that get trained with SGD alongside the network. And the third one is a 2-channel implementation with both static and non-static word vectors.

2.3 Word Embeddings

The concept of word embedding, proposed by Mikolov et al, refers to the mapping of discrete objects, in this case words, to vectors of real numbers, which can easily be processed by a network. Additionally, the dense embedding representation overcomes the problem of sparse representations of high dimensional objects. More importantly, word embeddings are used by Vector Space Models for a topological mapping of input words to a high dimensional vector space [11], where semantically related words are close to each other. One of the most famous models for learning the vector space of a corpus is word2Vec [12]. In the proposed architecture, the input sentences are mapped to word embeddings produced by the word2vec model, instead of random vectors. In that way, the network input contains linguistic regularities and patterns, which are exploited by the network.

3 Approach

This chapter describes how we addressed the challenge of finding a good CNN architecture for sentiment analysis. It starts with a description of how the Twitter and Movie Review datasets were preprocessed. Thereafter, the network architecture is describedv. Finally, the parameter exploration for the network architecture is explained.

3.1 Preprocessing

The network was trained using two different datasets. The first one, containing labeled movie reviews from the website https://www.rottentomatoes.com and the second one containing twitter tweets [1]. These text passages were cleaned in the same way as in the work conducted by Yoon Kim. Namely, from every text passage only alphanumeric characters as well as the symbols ”,(,),!,?, ’ and ‘ were kept. Additionally, all characters were lowercased.

In order to convert the text datasets into a numerical format that can be used by a neural network, the word vectors trained with the word2vec model, proposed by Mikolov et al, were used [13]. This version of the word2vec model was trained on 100 billion words of Google News [14]. In this work, the word2vec model was fitted on the vocabularies extracted from the two datasets.

3.2 Network Architecture

In this project a convolutional neural network architecture was implemented, to classify text as having a positive or a negative connotation.

The network follows the architecture that can be seen in Figure 1, an adapted version of Yoon Kim’s architecture[1]. It involves a convolutional layer followed

[1] The movie review and Twitter datasets are available online at https://www.cs.cornell.edu/people/pabo/movie-review-data/rt-polaritydata.tar.gz and http://alt.qcri.org/semeval2017/task4/data/uploads/codalab/4a-english.zip respectively.

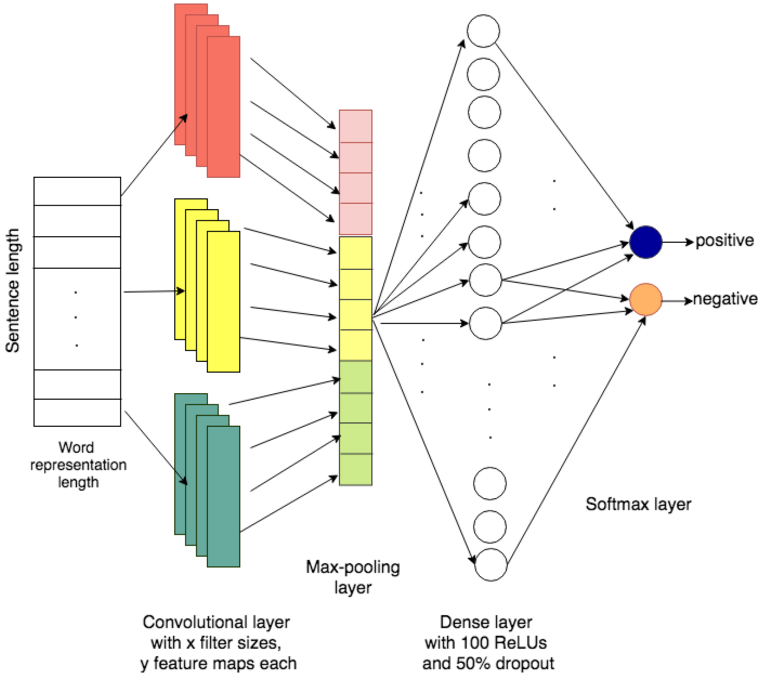
Sentence length
Word representation length
Convolutional layer with x filter sizes, y feature maps each
Max-pooling layer
Dense layer with 100 ReLUs and 50% dropout
Softmax layer
positive
negative

Fig. 1: Model architecture for an example sentence.

by a max-pooling layer, a dense layer of 100 ReLUs and 50% dropout, and finally a softmax layer of two units, extracting the probabilities of the input sentence to be positive or negative.

The convolutional layer consisted of one or three filters, producing multiple feature maps each. The filter width is the same as the width of one word-vector, because word vectors are not meant to be split in half, but the optimal combination of filter heights depends on the specific dataset too. Zero padding was used in order for each feature map to have the same dimensions as the input. The activation function used was hyperbolic tangent.

The feature maps were max-pooled in order to select the most important features of each one of them. Unlike traditional convolutional neural networks, where the widely known one-max-pooling technique is used, max-over-time-pooling seems to perform better in CNNs dealing with natural language tasks [15]. Proposed in Collobert 2011, this techniques captures the most important value per feature in the sentence [15]. In this case, iterating over the words of a sentence means iterating through a time sequence. After applying the max-over-time-pooling operator, a global feature vector is constructed for each sentence, with size independent of the length of each sentence.

After the stage of max-over-time pooling, the global feature vectors produced for each filter were concatenated, forming a global feature matrix.

The max-pooling layer outputs were fed into a fully-connected layer of 100 rectified linear units, widely known as ReLUs. In order to prevent overfitting, dropout was applied on the weights of this layer.

Finally, a softmax layer of two units was used as the output layer. Depending on which of the two nodes exhibits higher probability, we can decide whether the input sentence should be considered positive or negative.

The whole network is trained using stochastic gradient descent, enhanced with momentum optimizer. Furthermore, the learning rate was set to decay exponentially. Both techniques were used as a way of faster and smoother convergence [16] [17].

3.3 Static and non-static CNN

The term static or non-static characterizes the word vector representations of the input. In the case of static CNN network, the word vector representation of each sentence remains unchanged during the training phase. On the other hand, in non-static CNN architectures the word vector representations get updated after each training epoch. In that way, the new word vectors capture the sentimental polarity of each word in a more accurate way, specific to the task at hand.

From the perspective of network architecture, the only difference between a non-static and static CNN is the addition of an embedding lookup layer before the convolutional layers. The role of the embedding layer is to map the input sentence to the respective embedding representation. Since the embedding layer constitutes a part of the network, it is optimized alongside with the other network parameters. In that way, it encapsulates the polarity information learned by the network. The architecture of a non-static CNN network is illustrated in figure 2

3.4 Parameter Exploration

There is an abundance of parameters that need to be tuned in a convolutional neural network. Taking into account how much time a CNN needs to be trained, running experiments on each one of them would be very computationally expensive. After exploring relevant research results, such as those of Zhang and Wallace, the parameters most worth tuning were established [18]. They include filter heights and number of feature maps per filter in the convolutional layer.

In order to find the best performing filter height combination for each dataset a method described in Zhang and Wallace was applied [18]. The network architecture was modified to use a single filter. In this setting, different filter heights

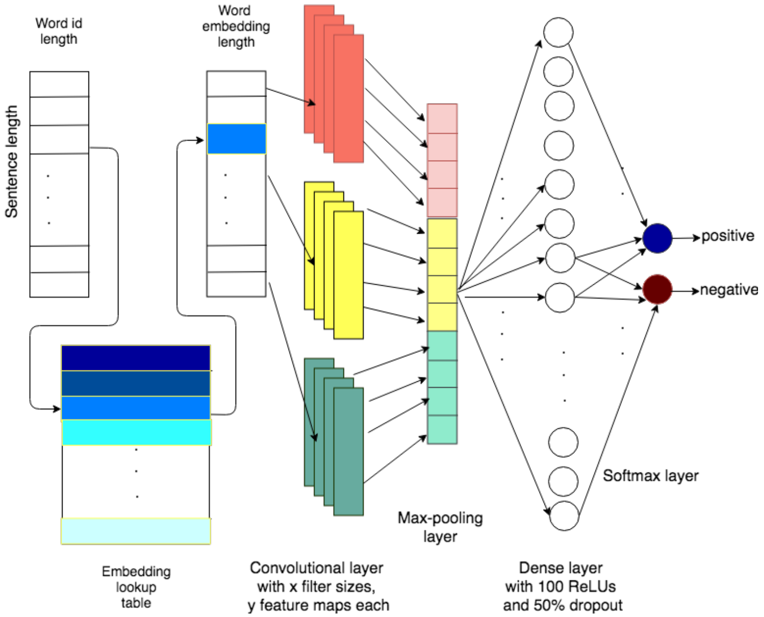

Fig. 2: Model architecture for an example sentence.

ranging from 1 to 10 were tested, in order to determine the best performing heights. Then different combinations of filter heights were explored, around the optimal heights, such as $optimal \times 3$ and $optimal - 1, optimal, optimal + 1$.

After determining the optimal filter height combination per dataset, the optimal number of feature maps per filter had to be determined as well. Different numbers in the range of 100 to 600 were tested. In general, more features improve the accuracy, on the cost of computational power and time. Some times too many filters also tend to overfit the dataset [18]. This trade off was considered worth exploring, so there is a relevant graph in the 4 section of this paper.

Last but not least, the learning rate had to be determined in correlation with the training epochs. In general, the bigger the learning rate the less the epochs that are necessary for a network to converge, and vice versa. On the other hand, bigger learning rates can prevent the network from converging, while smaller learning rates can lead the network to get trapped in some suboptimal local minimum [16]. As a consequence, the learning rate was kept as big as possible, as long as convergence was achieved within the range of the training epochs.

Throughout our experiments, some parameters were kept constant. These parameters were: batch size equal to 50, number of fully connected layer nodes equal to 100, and dropout rate of 50% in the fully connected layer. The SGD used to train the network used a momentum of 0.9, and initial learning rate of 0.01 exponentially decaying every 100 steps at a rate of 0.95.

4 Results and Conclusions

In this chapter we present the results of the experiments as well as the conclusions. The first section presents the results from the experiments with the static CNN. Thereafter, the results from the experiments with the non-static CNN are provided. Finally, a section describes future work that could be conducted.

4.1 Static CNN

The experiments were conducted with the parameter settings detailed in section 3.4. The results for the experiment on the static architecture with only one filter can be seen in table 1. The bold accuracies correspond to the best 3 filter heights for each dataset. Note, however, that the MR dataset had 2 filter heights with the same accuracy and that both of these two(3 and 5) thus share the third place. Thus, the 3 filter search, described later, was conducted with 3 filter-height-configurations for the twitter set and 4 filter-height-configurations for the MR dataset. The best accuracies for the Twitter and MR dataset was 95.02% and 88.7456% respectively. Interestingly, the acquired accuracy for the MR dataset was higher than the accuracy mentioned by Yoon Kim[1]. This is believed to be a consequence of the experiments running for more update steps and were run with a lower learning rate than Kim's.

Filter height	MR Accuracy	Twitter Accuracy
1	85.3458%	92.9612%
2	87.5733%	94.4175%
3	**88.2767%**	**95.0243%**
4	87.8077%	**94.9029%**
5	**88.2767%**	**95.0243%**
6	88.1594%	94.9029%
7	**88.7456%**	94.7816%
8	**88.5111%**	94.0534
9	88.0422%	93.5680%
10	87.8077%	93.4466%

Table 1: Validation accuracies when applying different filter sizes for the static architecture on both of the datasets. The 3 best runs are marked in bold. Note that the best run for the Twitter dataset as well as the third best run for the MR dataset are tied

After experimenting with only one filter height, filter heights in groups of 3 were utilized. These were selected in the way described in section 3.4. The resulting

MR		Twitter	
Filter Heights	Validation Accuracy	Filter Heights	Validation Accuracy
3,3,3	88.9801%	3,3,3	94.2961%
5,5,5	88.5111%	4,4,4	93.9320%
7,7,7	88.6284%	5,5,5	93.4466%
8,8,8	88.8628%	2,3,4	93.9320%
2,3,4	88.8628%	3,4,5	**94.4175**%
4,5,6	88.8628%	4,5,6	68.9320%
6,7,8	**89.2145**%		
7,8,9	88.2767%		

Table 2: Validation accuracies when applying different filter heights in combinations of three for the static architecture on both of the datasets. The best accuracies are marked in bold

accuracy can be observed in table 2. The accuracy for the Twitter dataset did in general decrease. However, for the MR dataset, the opposite was true. It is belived that this increase and decrease respectively might be a result of the intrinsic properties of the datasets. Another speculation is that the decrease in accuracy for twitter might be a consequence of the loss not having enough update steps to fully converge which is thought to be the case since more filters increase the model's capacity and thus may increase its training time. However, due to lack of computational resources, this could not be confirmed.

After experimenting with different filter heights as explained above, we concluded that our networks perform best when using a single filter height, of height 3 (or 5) for the Twitter dataset, and height 7 for the MR dataset. We plotted the loss of our network as a function of the training steps, for these best performing filter heights, and filter depth of 100. These plots can be seen in figure 3 and 4.

	MR		Twitter	
#Filters	Validation Accuracy	Duration(sec)	Validation Accuracy	Duration(sec)
50	87.4560%	480	92.3544%	201
100	87.5733%	855	92.2330%	272
150	86.8699%	1279	92.5971%	427
200	86.7526%	1314	93.2039%	443
250	87.2216%	1399	92.7184%	469
300	87.1043%	2406	92.3544%	738

Table 3: Validation accuracy and experiment time for different numbers of filters

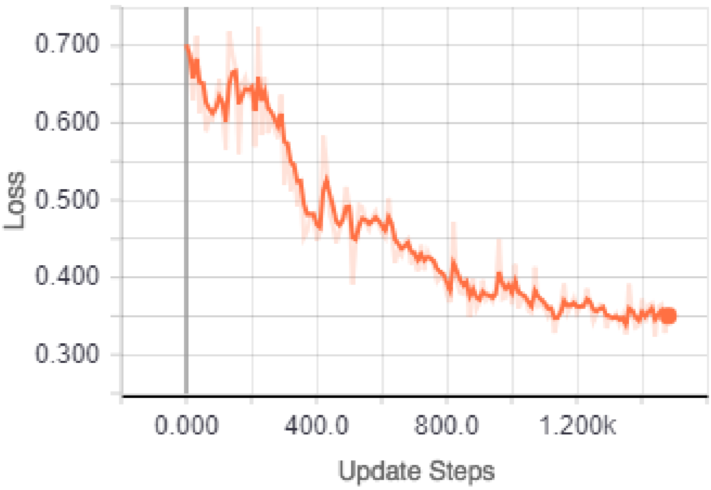

Fig. 3: SGD convergence on Twitter dataset with filter height of 3

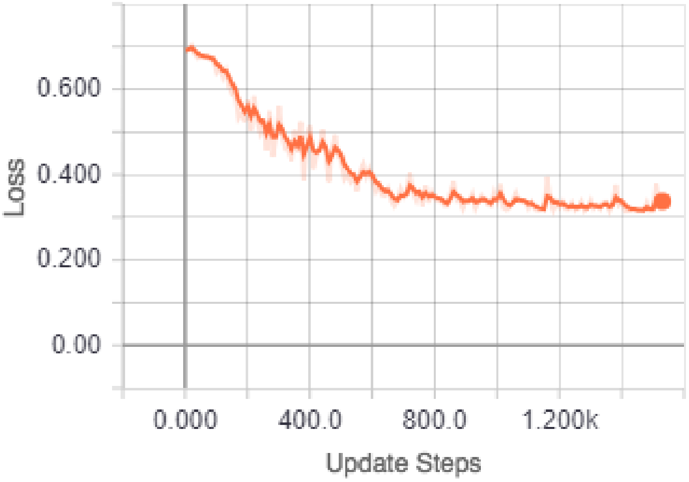

Fig. 4: SGD convergence on MR dataset with filter height of 7

We can see that while the algorithm seems to converge pretty early during the training procedure for MR dataset, the training on the twitter dataset seems to get advantage of all the training steps, and settling to a stable loss just before the training stops.

The next parameter worth exploring, as mentioned in section 3.4 was the filter depth. As the filter depth is believed to impact the time needed to train the network as well as the accuracy. The results from the conducted experiments are presented in table 3.

After running all the aforementioned experiments we chose the best possible parameter configuration for each dataset and calculated the accuracy on a previously unseen test set. The results are presented in the table 4.

Dataset	Filter Heights	Filter Depth	Test Accuracy
MR	6,7,8	100	88.52%
Twitter	3	100	93.93%

Table 4: Test accuracy of the static network

4.2 Non-static CNN

The parameters used for the experiments with non-static CNN, were chosen based on the optimal values found in the parameter exploration of the static CNN. The validation and loss accuracies of the network for both Twitter and MR datasets, with the optimal configurations, are presented in table 5.

Dataset	Filter Heights	Filter Depth	Test Accuracy
MR	6,7,8	100	63.061%
Twitter	3	100	67.285%

Table 5: Test accuracy of the non-static CNN network

From the table above it is clear that the performance of the non-static network is quite poor, compared to the performance of the static CNN network. The two architectures differ not only in the embedding layer, added in the non-static case, but also in the implementation. We believe that this poor performance might occur because of an undiscovered minor bug in the implementation or because the hyperparameters were not the optimal ones. That means that the vocabulary used contains stop words (such as like, as etc), as well as words which appear only once in the whole corpus of reviews. These words constitute redundant information, that affects the word embeddings learned by the network. It can be assumed that the dimensionality reduction of the vocabulary could improve the network performance.

4.3 Future Work

Continuation of these can be performed in many ways. However, we suggest that the most prominent suggestion would be to run the experiment with triple filter heights for more epochs since we were not totally convinced that the loss had had time to converge. Additionally, it could be interesting to conduct parameter exploration for the non-static architecture.

References

1. Kim, Y.: Convolutional neural networks for sentence classification. CoRR **abs/1408.5882** (2014)
2. Review, M.T.: Deep learning - with massive amounts of computational power, machines can now recognize objects and translate speech in real time. artificial intelligence is finally getting smart. (2018)
3. Karlgren, J., Sahlgren, M., Olsson, F., Espinoza, F., Hamfors, O.: Usefulness of sentiment analysis. In: Proceedings of the 34th European Conference on Advances in Information Retrieval. ECIR'12, Berlin, Heidelberg, Springer-Verlag (2012) 426–435
4. Feldman, R.: Techniques and applications for sentiment analysis. Commun. ACM **56**(4) (April 2013) 82–89
5. Shi, H., Zhang, Y., Zou, Y., Li, X.: Fine-grained sentiment analysis of reviews using shallow semantic information. In: 2017 International Conference on Progress in Informatics and Computing (PIC). (Dec 2017) 235–239
6. Cambria, E., Schuller, B., Xia, Y., Havasi, C.: New avenues in opinion mining and sentiment analysis. IEEE Intelligent Systems **28**(2) (March 2013) 15–21
7. Agarwal, A., Xie, B., Vovsha, I., Rambow, O., Passonneau, R.: Sentiment analysis of twitter data. In: Proceedings of the Workshop on Languages in Social Media. LSM '11, Stroudsburg, PA, USA, Association for Computational Linguistics (2011) 30–38
8. Ghorbel, H., Jacot, D. In: Sentiment Analysis of French Movie Reviews. Springer Berlin Heidelberg, Berlin, Heidelberg (2011) 97–108
9. Anita, Gupta, D.K., Kumar, A.: Spam and sentiment analysis model for twitter data using statistical learning. In: Proceedings of the Third International Symposium on Computer Vision and the Internet. VisionNet'16, New York, NY, USA, ACM (2016) 54–58
10. Severyn, A., Moschitti, A.: Twitter sentiment analysis with deep convolutional neural networks. In: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval. SIGIR '15, New York, NY, USA, ACM (2015) 959–962
11. Salton, G., Wong, A., Yang, C.S.: A vector space model for automatic indexing. Commun. ACM **18**(11) (November 1975) 613–620
12. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems. (2013) 3111–3119
13. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In Burges, C.J.C., Bottou, L., Welling, M., Ghahramani, Z., Weinberger, K.Q., eds.: Advances in Neural Information Processing Systems 26. Curran Associates, Inc. (2013) 3111–3119
14. Archive, G.C.: word2vec (2018)
15. Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., Kuksa, P.: Natural language processing (almost) from scratch. Journal of Machine Learning Research **12**(Aug) (2011) 2493–2537
16. Jacobs, R.A.: Increased rates of convergence through learning rate adaptation. Neural Networks **1**(4) (1988) 295 – 307
17. Sutskever, I., Martens, J., Dahl, G., Hinton, G.: On the importance of initialization and momentum in deep learning. In: Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28. ICML'13, JMLR.org (2013) III–1139–III–1147

18. Zhang, Y., Wallace, B.C.: A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. CoRR **abs/1510.03820** (2015)