

Twitter Sentiment Analysis Using Deep Convolutional Neural Network

Dario Stojanovski^(✉), Gjorgji Strezoski, Gjorgji Madjarov,
and Ivica Dimitrovski

Faculty of Computer Science and Engineering,
Saints Cyril and Methodius University,
Rugjer Boshkovikj 16, 1000 Skopje, Macedonia
{stojanovski.dario,strezoski.g}@gmail.com,
{gjorgji.madjarov,ivica.dimitrovski}@finki.ukim.mk

Abstract. In the work presented in this paper, we conduct experiments on sentiment analysis in Twitter messages by using a deep convolutional neural network. The network is trained on top of pre-trained word embeddings obtained by unsupervised learning on large text corpora. We use CNN with multiple filters with varying window sizes on top of which we add 2 fully connected layers with dropout and a softmax layer. Our research shows the effectiveness of using pre-trained word vectors and the advantage of leveraging Twitter corpora for the unsupervised learning phase. The experimental evaluation is made on benchmark datasets provided on the SemEval 2015 competition for the Sentiment analysis in Twitter task. Despite the fact that the presented approach does not depend on hand-crafted features, we achieve comparable performance to state-of-the-art methods on the Twitter2015 set, measuring F1 score of 64.85%.

Keywords: Twitter · Sentiment analysis · Convolutional neural networks · Word embeddings · Deep learning

1 Introduction

Sentiment analysis is an area of Natural Language Processing (NLP) that focuses on understanding human emotion in text. With the spread of social media and microblogging websites, sentiment analysis in social networks has gained increasing popularity amongst scientific researchers. Users on these services share their ideas and opinions on various topics, events and products. As of 2014 Twitter has over 284 million monthly active users and about 500 million messages are sent on a daily basis¹, which positions Twitter as the focal point for research in sentiment analysis. This motivates companies to poll data from social networks to get a better understanding of the reactions their products and services get.

¹ <https://about.twitter.com/company>.

Sentiment analysis in social networks and microblogs is more challenging due to the informal nature of the language. Unlike text from movie or product reviews, tweets have limitation of 140 characters and users tend to use a lot of abbreviations, slang and URLs along with Twitter specific terms such as user mentions and hashtags.

Deep learning techniques have recently shown great improvements over existing approaches in computer vision and speech recognition. In the field of NLP, deep learning methods are primarily used for learning word vector representations [1, 2], part-of-speech tagging, semantic role labeling, named entity recognition [3] etc. Traditional NLP methods are based on hand-crafted features which is both time-consuming and leads to over-specified and incomplete features. Feature generation is inherently built into deep neural networks and they enable the model to learn increasing levels of complexity. Recently there have been attempts of using deep learning for sentiment analysis, primarily through utilizing deep convolutional neural networks. Deep CNNs have one key advantage over existing approaches for sentiment analysis that rely on extensive feature engineering. CNNs automate the feature generation phase and learn more general representations, thereby making this approach robust and flexible when applied to various domains.

In this paper, we tackle the problem of sentiment analysis on Twitter messages by using a deep CNN architecture. The architecture is based on the model proposed in [4] which reported state-of-the-art performance on 4 out of 7 sentence classification tasks. Unlike the aforementioned network that has only one fully connected layer, we employ a more deep architecture. We have two fully connected non-linear layers with dropout and a softmax layer. Our model resembles the architecture of the network proposed in [3] and [5], the difference being that we use multiple filters in the convolutional layer. Additionally, our model has multiple layers of non-linearity which enables for learning more complex representations.

Though some of the work so far have relied on emoticons for labels in order to obtain a larger training set, we only utilize manually labeled tweets. Training and test sets were provided by the organizers of the SemEval competition, but we additionally extend our train set with other available manually annotated Twitter data. Evaluation metrics that are used are accuracy and average macro-F1 score on positive and negative labels.

The main contributions of this paper are three fold:

- We evaluate several pre-trained word vectors and the usefulness of leveraging Twitter corpora as training data when applied to the task of Twitter sentiment analysis.
- We present an architecture for deep convolutional neural network that, to our knowledge, has not been used for sentiment analysis on Twitter data.
- We report our results on the benchmark test sets on the Twitter Sentiment Analysis Track in SemEval 2015.

The rest of the paper is organized as follows. Section 2 outlines current approaches on sentiment analysis, with emphasis on Twitter sentiment analysis and deep learning methods. Section 3 presents the details of the model proposed in this

paper, the pre-processing phase and an overview of the pre-trained word vectors used in our work. In Sect. 4, the experimental setup is explained in detail along with the datasets being used. We present and elaborate on the performance achieved using our approach and provide insight on the findings of our research in Sect. 5. Finally, we conclude our work and discuss future directions in Sect. 6.

2 Related Work

There has been a lot of work done in the field of sentiment analysis in natural language and social network posts. The research ranges from document level classification [6], contextual polarity disambiguation to topic based sentiment classification. Twitter messages have 140 character limitation which makes the task of Twitter sentiment analysis closest to sentence level sentiment detection.

Most of the work done so far in Twitter sentiment analysis have revolved around extensive feature engineering which is both labor intensive and is likely to be too domain specific. In 2013, the best performance on classifying tweets polarity was reported by Mohammad et al. [7]. The features used in their method are word and character ngrams, the number of words with all characters in upper case, the number of hashtags, the number of contiguous sequences of exclamation marks, the presence of elongated words, the number of negated contexts etc. The system also depends on the use of several lexicons to determine the sentiment score for each token in the tweet, part-of-speech tag and hashtag.

The authors of [8] developed around 100 features and compared the effectiveness against unigram features. Their approach separates the features in multiple categories based on whether they are generated using POS tagging, carry polarity information and their type(integers, booleans or real values).

Classifiers that are mostly used by the aforementioned methods are Naive Bayes, Maximum Entropy and Support Vector Machines (SVM). In the work of [9], these commonly used classifiers were evaluated on unigram and bigram features and showed that all classifiers perform similarly. While most of the methods use hand-crafted features, some approaches rely on lexicons with words and their polarity score. These approaches map the words polarity score and compute the sentiment of the tweet.

On the other hand, deep convolutional neural networks do not depend on extensive manual feature engineering and extract the features automatically. They have the advantage of inherently taking into account the ordering of the words and by using word vectors they encompass syntactic and semantic meaning of words. Socher et al. [10] introduced a Recursive Neural Tensor Network that maps phrases through word embeddings and a parse tree. Afterwards, vectors for higher nodes in the tree are computed and tensor-based composition function for all nodes is used. The method pushed state-of-the-art results on fine-grained and positive/negative sentiment classification of movie reviews. However, RNTN depends on the syntactic structure of the text as input.

CNNs have already started to give better results on several tasks in NLP. Kalchbrenner et al. [11] showed that their Dynamic Convolutional Neural Network outperforms other unigram and bigram based methods on classification

of movie reviews and tweets. However, we can't directly compare with the aforementioned method as they only report the accuracy their model achieves and not the F1 score. The authors of [12] proposed a neural network architecture that exploits character-level, word-level and sentence-level representations. Character-level features proved to be useful for sentiment analysis on tweets, because they capture morphological and shape information.

Using pre-trained word representations for sentiment analysis has one obvious disadvantage. Since the training is done in an unsupervised manner, there is no sentiment information encoded in the word vectors. Tang et al. [13] attempt to resolve this issue by learning sentiment specific word embeddings (SSWE) from massive distant-supervised tweets. The proposed method uses noisy-labeled tweets where labeling is based on the presence of positive and negative emoticons. Besides these word embedding features, the system they proposed at SemEval 2014 Task 9 [14] also consists of hand-crafted features which are based on the SemEval 2013 winning system [7].

3 System Architecture

3.1 Deep Convolutional Neural Network

In this work, we propose a deep convolutional neural network for classification of tweets into positive, negative and neutral classes. Our approach is based on the approaches proposed by Kim [4] and Collobert et al. [3], incorporating elements from both architectures. Kim [4] propose an architecture that uses multiple filters with varying window sizes that are applied on each given sentence. We modify the aforementioned model by adding two fully connected layers with dropout and a softmax layer to the architecture. The first layer consists of sigmoid activated units since using a linear layer showed worse performance. The second layer is a hyperbolic tangent layer to which a standard softmax layer is appended.

CNNs with pooling operation deal naturally with variable length sentences and they also take into account the ordering of the words and the context each word appears in. This solves the problem of negations which may appear in different places in a sentence. For simplicity, we consider that each tweet represents one sentence. The architecture of the model is depicted in Fig. 1 and is somewhat similar to the one presented in [3].

Let's consider a tweet t with length of n tokens with the appropriate padding at the beginning and at the end of the tweet. Padding length is defined as $h/2$ where h is the window size of the filter. The first step is mapping tokens to the corresponding word vectors from a lookup table $L \in R^{k \times |V|}$, where k is the dimension of the word vectors and V is a vocabulary of the words in the lookup table (more details on the lookup table are provided in Sect. 3.2). Each word or token is projected to a vector $w_i \in R^k$. After the mapping, a tweet is represented as a concatenation of the word embeddings

$$x = \{w_1, w_2, \dots, w_n\}. \quad (1)$$

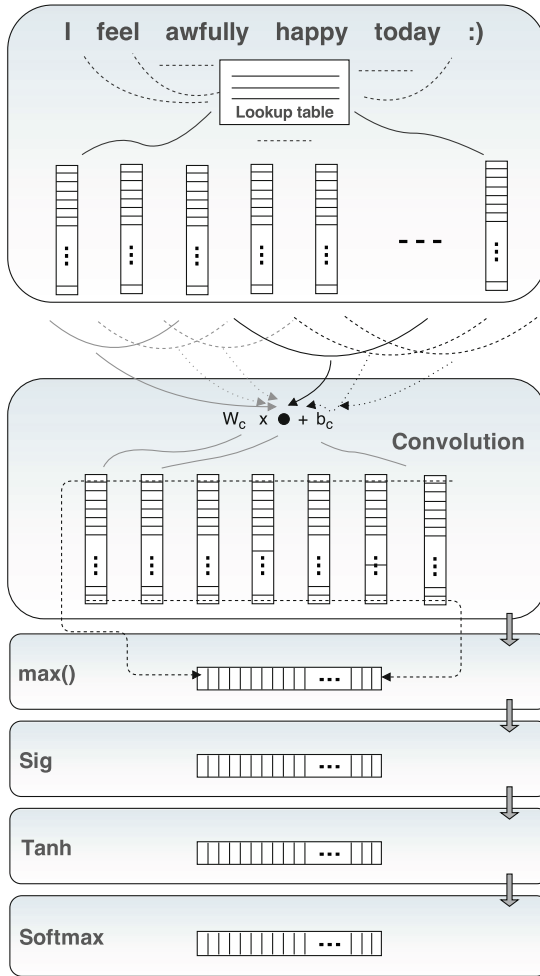


Fig. 1. Deep convolutional neural network architecture.

The next step is the convolution operation in which we apply multiple filters with varying window sizes h . Filters are applied to every possible window of words in the tweet and a feature map is produced as a result. For each of the filters, a weight matrix $W_c \in R^{h_u \times h_k}$ and a bias term b_c are learned, where h_u is the number of hidden units in the convolutional layer. The weight matrix is used to extract local features around each word window. The convolution operation can be formally expressed as

$$x'_i = h(W_c \cdot x_{i:i+h-1} + b_c), \tag{2}$$

where $h(\cdot)$ is the hyperbolic tangent function and $x_{i:i+h-1}$ is the concatenation of word vectors from position i to position $i + h - 1$.

We then apply a max-over-time pooling operation on the feature map x' computed from the convolution operation (3).

$$x' = \max\{x'_1, x'_2 \dots x'_{n-h+1}\}. \quad (3)$$

As a result, we get a fixed size vector for the tweet and extract the most important features for each feature map. The size of the vector is a hyper-parameter to be determined by the user and corresponds to the number of hidden units in the convolutional layer. This is the process for generating one feature for one filter. In order to utilize more filters, the fixed size vectors generated by the max-over-time pooling operation are concatenated and fed to the first layer of the network architecture.

The rest of the architecture is a classical feed forward neural network consisting of three separate layers. The first layer contains units with sigmoid activation function. We have experimented with a linear layer as was done in [3], but using sigmoid function yielded better results.

$$x^1 = f(W^1 \cdot x' + b^1), \quad (4)$$

where f is the sigmoid activation function. The sigmoid layer is followed by another non-linear layer for which we chose a hyperbolic tangent function. The final layer of the architecture is a softmax layer that gives a probability distribution over the labels. The network is trained using stochastic gradient descent over shuffled mini-batches with the Adadelta update rule [15].

Deep neural networks suffer from overfitting due to the high number of parameters that need to be learned. To counteract the problem of large number of hidden units and connections between them we utilize dropout regularization. The idea behind this technique is that during the training process random units along with their connections are dropped (set to zero). The proportion of units to be dropped is a hyper-parameter to be determined by the user. Dropout regularization is applied to the fully connected layers.

3.2 Pre-training

Pre-processing. Preprocessing is a common stage in any task involving Twitter data because of the language irregularities that are present in tweets. We do several pre-processing actions in order to clean the input data from noise. As URLs carry no information for the sentiment of a tweet we remove all instances of URLs. We clean tweets from all HTML entities, user mentions and punctuation with the exception of exclamation and question marks. We keep hashtags and emoticons as emoticons are probably the strongest indicator for the sentiment a tweet carries.

Users on social media tend to elongate words in order to emphasize the emotion they are trying to convey. The length of one word (the number of character repetitions) may differ in different tweets, but they essentially carry the same information. One simple example is that a tweet can contain the token “haaaapy”

and another the token “haaaaapy”. In order not to differentiate between these variations, we shorten the repeated characters to maximum 3 repetitions.

We tried replacing abbreviations with their actual meaning, but in our experiments this approach didn’t bring the expected performance improvements. We contribute this to the fact that the network is well suited to learn an appropriate word vector for these abbreviations even if they are not present in the lookup table. Consequently, we decided to leave out this pre-processing step.

Pre-trained Word Vectors. Learning word representations from massive unannotated text corpora have recently been used in many NLP tasks. Leveraging large corpora for unsupervised learning of word representations enables capturing of syntactic and semantic characteristics of words.

One possibility is to randomly initialize the word vectors and let the model learn appropriate representation for each word. Kim [4] and Santos et al. [12] reported better results by initializing word vectors with ones obtained from an unsupervised neural language model. Therefore, we decided not to explore the effect of random initialization since this has already been studied and proven considerably less effective than using pre-trained vectors.

In our research we evaluate three different methods for generating word vectors and their applicability to sentiment analysis. Apart from the widely popular *word2vec*² [1], we also use global vectors for word representation [2], referred throughout the paper as GloVe word embeddings³ and the semantic specific word embeddings (SSWE) from [13]. Details of the word embeddings are presented in Table 1. Note that corpora size is expressed in token count with the exception of SSWE where the authors only report the number of tweets on which vectors were learned. GloVe Crawl embeddings were trained on web data, while for *word2vec* vectors, data from Google News was used.

Table 1. Details about the word embeddings used for the lookup table. Corpora size is expressed in token count with the exception of SSWE where only the number of tweets is provided.

	Dimension	Corpora	Vocabulary
SSWE	50	10 M	137 K
<i>word2vec</i>	300	100 B	3 M
GloVe Crawl	300	840 B	2.2 M
GloVe Twitter	200	20 B	1.2 M

Unsupervised learning of word embeddings has the drawback of not having sentiment information encoded in their representation. A simple example is that

² <https://code.google.com/p/word2vec/>.

³ <http://nlp.stanford.edu/projects/glove/>.

“good” and “bad” are neighboring words based on cosine similarity. This stems from the fact that these words appear in similar context in the large text corpora that is used for the training process.

Our model tackles this issue differently than [13]. Instead of doing the pre-training phase ourselves, we use available word vectors and by backpropagation, during the training of the network, update them in order to adapt to the specific task at hand. Kim [4] showed that by using non-static word vectors, their approach was able to capture sentiment regularities in words. They showcase this by comparing top-4 neighboring words of “good” before and after training. Non-static word vectors managed to learn that “bad” is not similar to “good”. One could see the benefit from this method if sentiment detection system is put into production use. In time, the model will adapt to changes in language that may arise and which in fact occur frequently in microblogging environments.

For words that are not present in the vocabulary of word vectors, we use random initialization. Kim [4] suggest to use a range of $[-a, a]$ where a is set so that random initialized words have the same variance as the pre-trained ones. In our case, we set a to 0.25.

4 Experimental Design

4.1 Datasets

We train and test our model on the benchmark sets from the SemEval Task 10 challenge. The sets were manually annotated by the organizers of the challenge with three labels, positive, negative and neutral. Unfortunately, we were not able to retrieve all of the tweets because some of them were most likely removed or had altered privacy status. The overall number of tweets was somewhat smaller than what some teams reported in this year’s challenge. This year the organizers of SemEval Task 10, decided not to constrain competitors by allowing them to use data outside of the one provided by them. Therefore, we extended our training set with another manually labeled set of tweets⁴, which were collected with respect to 4 topics. The results presented in this paper are on the latest Twitter2015 test set along with the test sets from Subtask B from previous years. Classes distribution are depicted in Table 2.

Table 2. Distribution of labels on the training and test sets

	Positive	Negative	Neutral
Train	4000	2101	6653
TW13	1572	601	1640
TW14	982	202	669
TW15	1038	365	987

⁴ <http://www.sananalytics.com/lab/twitter-sentiment/>.

4.2 Experimental Setup

In our experiments, we reused some of the hyper-parameters reported in [4] such as a mini-batch size of 50, l_2 constraint of 3 and filter windows of 3, 4 and 5. We experimented with other combinations of filter windows, but this proved to be a suitable combination because of the limitation of Twitter messages. In our case, we observe that using hyperbolic tangent activated units performs better than rectified linear units for the convolutional layer. We set the learning rate to 0.02 and apply a regularization with a dropout rate (0.7, 0.5) on the fully connected layers in the network respectively.

We started with 100 hidden units in each layer and 100 feature maps for the different filter windows. However, results were slightly improved by using 500 hidden units in the first layer and 300 in the second, while increasing feature maps size to 300.

5 Results and Discussion

In this paper we present performance achieved on the test set of this year’s SemEval Task 10. Results are presented in Tables 3 and 4. We use two evaluation metrics, accuracy and average macro-F1 score on positive and negative labels.

We observe that using GloVe word embeddings gives better performance than other approaches. It is interesting to see that GloVe Twitter and GloVe Crawl

Table 3. Performance using different word vectors on Twitter2015

	SSWE	word2vec	GloVe Crawl	GloVe Twitter
Accuracy	61.59	66.44	68.77	66.99
Macro-F1	58.76	60.26	64.72	64.85

Table 4. Macro-F1 score on positive, negative and neutral classes. TW13, TW14 and TW15 are Twitter2013, Twitter2014 and Twitter2015 respectively. Table 4 is structured into three main sections. The first section contains top performing approaches using hand-crafted features and SVM, followed by methods using a combination of deep learning and manual feature engineering. The last section contains pure deep learning approaches.

Team	TW13	TW14	TW15
Webis	68.49	70.68	64.84
Isislif	71.34	71.54	64.27
CIS-positiv	64.82	66.05	59.57
Spluplus	72.8	74.42	63.73
unitn	72.79	73.6	64.59
Finki	69.07	72.75	64.85

perform similarly, even though GloVe Twitter embeddings have a dimension of 200 in comparison to the 300-dimensional GloVe Crawl and *word2vec* word vectors. Using SSWE vectors did not produced comparable results, and it would take further research to determine whether this is due to the lower dimensionality and smaller corpora size. On the other side, *word2vec* word vectors were not as effective as GloVe embeddings. Whether the reason for the difference in performance is the type of corporas used or the methods themselves requires further examination.

We compare our approach with the submissions from the latest SemEval challenge. Results for the Twitter2015 test set were considerably worse than for the Twitter2013 and Twitter2014 sets, with almost 10 % margin between the top results. Our system, Finki, performs well on Twitter2013 and is in top-3 on Twitter2014, while outperforming other approaches on the Twitter2015 test set. It is obvious from Table 4, that deep learning techniques provide more consistent results across datasets.

As was mentioned before, the training set was extended with an additional manually labeled set. Although the added training set that we use has limited domain, combining the sets improved performance. This only confirms the intuition that deep learning techniques are more flexible than approaches with hand-crafted features and can greatly benefit from a larger training set. From the presented results, we can also observe the significance word embeddings have on performance in Twitter sentiment analysis, especially ones that are trained on corpora originating from Twitter.

6 Conclusion and Future Work

In this paper, we present a deep convolutional neural network for sentiment analysis on Twitter posts. To our knowledge this specific architecture, with multiple filters and non-linear layers on top of the convolutional layer has never been used for classifying Twitter messages. We experimented with different word embeddings, trained on both Twitter and non-Twitter data.

Unsupervised pre-training of word embeddings on Twitter based corpora offers improvements over non-Twitter based corpora, as was made evident from our experimentations and we would like to further explore the effect of using Twitter corpora as training data. Utilizing the GloVe Twitter word vectors already provided slightly better results than GloVe Crawl, despite having 200 dimensions and being trained on a considerably smaller corpora. We would like to see the performance of *word2vec* when trained on Twitter data and the effectiveness of SSWE vectors if they would be trained on larger corpora and have a bigger dimensionality. We would also like to further investigate on why using first linear layer did not provided similar results to our current model.

We report our results on test sets from the SemEval 2015 Task 10, for 3-way sentiment classification of tweets. Our method performs comparable to state-of-the-art approaches in this challenge, despite the fact it does not depend on any hand-crafted features or polarity lexicons. The network achieved 64.85 % F1

score on the Twitter2015 set. We can finally conclude that CNN that leverage pre-trained word vectors perform well on the task of Twitter sentiment analysis.

Acknowledgments. We would like to acknowledge the support of the European Commission through the project MAESTRA Learning from Massive, Incompletely annotated, and Structured Data (Grant number ICT-2013-612944).

References

1. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Burges, C., Bottou, L., Welling, M., Ghahramani, Z., Weinberger, K. (eds.) *Advances in Neural Information Processing Systems 26*, pp. 3111–3119. Curran Associates, Inc. (2013)
2. Pennington, J., Socher, R., Manning, C.: Glove: global vectors for word representation. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp.1532–1543. Association for Computational Linguistics, Doha, October 2014
3. Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., Kuksa, P.: Natural language processing (almost) from scratch. *J. Mach. Learn. Res.* **12**, 2493–2537 (2011)
4. Kim, Y.: Convolutional neural networks for sentence classification. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1746–1751. Association for Computational Linguistics, Doha, October 2014
5. Chintala, S.: *Sentiment Analysis Using Neural Architectures*. New York University, New York (2012)
6. Pang, B., Lee, L.: Opinion mining and sentiment analysis. *Found. Trends Inf. Retrieval* **2**(1–2), 1–135 (2008)
7. Mohammad, S., Kiritchenko, S., Zhu, X.: Nrc-canada: building the state-of-the-art in sentiment analysis of tweets. In: *Proceedings of the Seventh International Workshop on Semantic Evaluation Exercises (SemEval-2013)*, Georgia, June 2013
8. 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 2011*, pp. 30–38. Association for Computational Linguistics, Stroudsburg, (2011)
9. Go, A., Bhayani, R., Huang, L.: Twitter sentiment classification using distant supervision. *CS224N Project Report*, Stanford, pp. 1–12 (2009)
10. Socher, R., Perelygin, A., Wu, J.Y., Chuang, J., Manning, C.D., Ng, A.Y., Potts, C.: Recursive deep models for semantic compositionality over a sentiment treebank. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp.1631–1642. Citeseer (2013)
11. Kalchbrenner, N., Grefenstette, E., Blunsom, P.: A convolutional neural network for modelling sentences, *arXiv preprint* (2014). [arXiv:1404.2188](https://arxiv.org/abs/1404.2188)
12. dos Santos, C., Gatti, M.: Deep convolutional neural networks for sentiment analysis of short texts. In: *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pp. 69–78. Dublin City University and Association for Computational Linguistics (2014)

13. Tang, D., Wei, F., Yang, N., Zhou, M., Liu, T., Qin, B.: Learning sentiment-specific word embedding for twitter sentiment classification. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, vol. 1, Long Papers, pp. 1555–1565. Association for Computational Linguistics (2014)
14. Tang, D., Wei, F., Qin, B., Liu, T., Zhou, M.: Coooolll: A deep learning system for twitter sentiment classification. In: Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014), pp. 208–212. Association for Computational Linguistics and Dublin City University, Dublin, August 2014
15. Zeiler, M.D.: Adadelata: An adaptive learning rate method, arXiv preprint (2012). [arXiv:1212.5701](https://arxiv.org/abs/1212.5701)