2017

Abusive Text Detection Using Neural Networks

Hao Chen
Technological University Dublin

Susan McKeever
Technological University Dublin, susan.mckeever@tudublin.ie

Sarah Jane Delany
Technological University Dublin, sarahjane.delany@tudublin.ie

Abusive Text Detection Using Neural Networks

Hao Chen, Susan McKeever, Sarah Jane Delany

School of Computer Science, Dublin Institute of Technology, Ireland
hao.chen@mydit.ie, susan.mckeever@dit.ie, sarahjane.delany@dit.ie

Abstract. Neural network models have become increasingly popular for text classification in recent years. In particular, the emergence of word embeddings within deep learning architectures has recently attracted a high level of attention amongst researchers. In this paper, we focus on how neural network models have been applied in text classification. Secondly, we extend our previous work [4, 3] using a neural network strategy for the task of abusive text detection. We compare word embedding features to the traditional feature representations such as n-grams and handcrafted features. In addition, we use an off-the-shelf neural network classifier, FastText[16]. Based on our results, the conclusions are: (1) Extracting selected manual features can increase abusive content detection over using basic ngrams; (2) Although averaging pre-trained word embeddings is a naive method, the distributed feature representation has better performance to ngrams in most of our datasets; (3) While the FastText classifier works efficiently with fast performance, the results are not remarkable as it is a shallow neural network with only one hidden layer; (4) Using pre-trained word embeddings does not guarantee better performance in the FastText classifier.

1 Introduction

Text classification is an essential component in many applications, such as sentiment analysis[27, 29], news categorization[16], and in our research domain of interest, abusive text detection[4, 3]. One of the fundamental tasks in text classification is feature representation - finding appropriate approaches to represent text content. The traditional approach is based on the occurrence-model, counting the frequency of words (e.g. BoW) in text content. This largely ignores word orders and thus the problem of capturing semantics between words still remains. Adding extra features that are identified by experts based on specific task requirements can alleviate the drawback of traditional features. However, this takes time and human effort and introduces domain specific dependencies into the model. One solution for feature extraction without hand-crafting is to use deep learning methods. In particular, this trend is sparked by the emergence of word embedding techniques, such as word2vec[22] and glove[26]. Word embedding is a distributed representation at word level which has been proven to be capable of learning word semantics. To generate a distributed feature representation at sentence level, one of the straightforward approaches is averaging the pre-trained word embeddings. However, this reduces context information such

as the sensitivity of word orders, which limits semantic knowledge. To address this issue, combining word embeddings with deep neural networks is a promising approach, and it has attracted increasing attention in recent research.

Originating from neural network structures, deep neural networks aim to automatically abstract feature representations for data based on hierarchical layers. It produces the state-of-the-art results in many text classification tasks[27, 25, 14]. In this paper, we first present a review of the recent deep neural networks that are widely used in the text classification. Afterwards, we carry out some preliminary experiments on abusive text detection using fundamental neural network techniques. We compare word embeddings to the more traditional feature representations using SVM classifiers. In addition, we investigate an off-the-shelf neural network based classifier.

The structure of the rest of paper is as follows: Section 2 discusses two modes of using neural networks for text classification and how these have been used in abusive text detection. In Section 3, we present experimental results. Finally, conclusions and future work are presented in Section 4.

2 State-of-the-Art Neural Networks

In this section, we investigate the current deep neural networks that have been used in general text classification tasks. We consider the use of deep neural networks in two modes: unsupervised for feature representation, and supervised for text classification. Furthermore, we review some deep neural networks that have been used in the abusive text detection domain.

2.1 Unsupervised Mode for Feature Representation

The cornerstone of using neural networks in unsupervised mode is word2vec[22] which is an approach to generate a distributed representation for words in a vector space with lower-dimension. These word vectors, also called word embeddings, are learned based on the concept that the words with similar meaning should have the similar surrounding words. Mikolov et al.[22] introduced two models, skip-gram and continuous bag of words (CBOW). The framework of skip-gram is shown in Figure 1. The architecture is very straightforward. In the training process, the model generates a representation of current word W_t based on predicting the nearby words (W_{t-2},W_{t-1},W_{t+1},W_{t+2}) in a window. After training, the vector of weights from the hidden layer is the representation of the word, the word embedding. CBOW is a similar model to skip-gram except it swaps the input and output, using nearby words to predict the current word.

There are also approaches that use neural networks to generate the distributed representation for blocks of text (sentence, paragraph or document). Here, we introduce three typical unsupervised models, paragraph2vec[20], Skip-Thought[18] and autoencoder[6]. Adapted from the word2vec architecture, Mikolov et al.[20] proposed a method called paragraph2vec which can learn comment representation from variable length text. Figure 2 shows the framework. Looking

at the base of the diagram, the input layer has two elements, a unique vector D represents paragraph id, and a set of vectors Ws representing words in a window which slides over the text. The output layer is the prediction of the next word in the context. During the training process, each weights of each word vector Ws is updated over each window. However, the weights of the paragraph vector are updated only when the window is in the paragraph. At the end of training process, the paragraph vectors D can be used as the text feature representation. In paper[20], the error of classification when using paragraph vectors decreased by approximately 39% compared to the traditional Bag-of-Words feature representation.

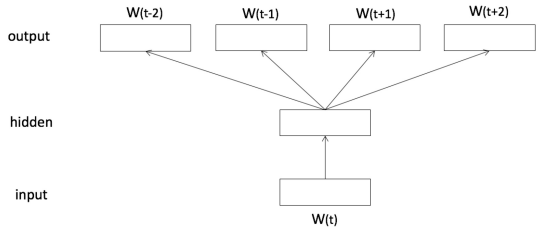

Fig. 1: The skip-gram architecture of word2vec proposed by Mikolove et al.[22]

Skip-Thought[18] is an another unsupervised approach to generate feature representations at sentence level also inspired by word2vec. Given a tuple of (S_{i-1}, S_i, S_{i+1}) sentences, the words in S_i are mapped as input, through the neural network model, S_i is converted into the vector, and then the vector is converted back to a set of words that are appear in the nearby sentences(S_{i-1} for previous sentence, and S_{i+1} for next sentence). After the training process, the resulting weights vector can be used as the representation. Instead of reconstructing context information such as surrounding words (paragraph2vec) or surrounding sentences (Skip-Thought), reconstructing the text content itself is also a popular way to generate feature representation. This approach is named autoencoder where the model starts by encoding the sentence into a lower-dimensional embedding, and then converts it back to the original input. After training, the lower-dimensional embeddings can be used as the feature representation [6, 32].

2.2 Supervised Mode for Text Classification

Neural network architectures have also been used directly for text classification tasks. For example, the simple model named FastText, proposed by Joulin et al.[16], is an efficient classifier. As shown in Figure 3, a sentence with a set of N ngrams features $(X_1, X_2, ..., X_{N-1}, X_N)$ are embedded and then averaged to the middle layer, the output layer is the softmax function that computes the probability distribution over the pre-defined classes. The advantage of the

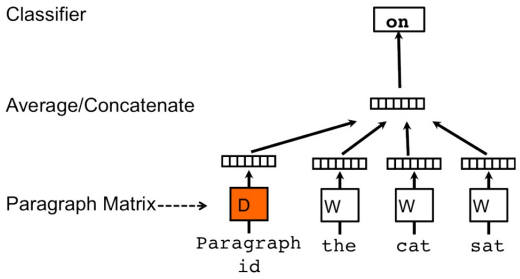

Fig. 2: The architecture of Paragraph2vec proposed by Mikolove et al.[20]

FastText model is its fast execution time. However, the performance of using a shallow neural network in supervised classification is rudimentary (see the results of experiments in Section 3).

A range of more complex neural network architecutures have been used in recent years for text classification. Two particular architectures that are widely used are convolutional neural networks and recurrent neural networks. The following subsections expand on details of these two models respectively.

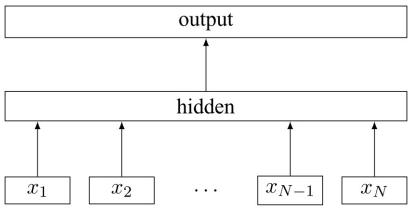

Fig. 3: The architecture of FastText proposed by Joulin et al.[16]

Convolutional Neural Network The CNN model uses multiple layers with convolving filters that aim to capture 'local' features. It was originally used in computer vision[19]. Subsequently, CNN was adopted in natural language processing and produced impressive results for many text classification tasks. The basic framework of CNN is shown in Figure 4. The sentence is represented by a set of word embeddings which are then mapped through a variety of convolutional filters of different sizes. Afterwards, the structure applies max-pooling to reduce the dimensionality of the features in order to reduce the complexity of the model and prevent overfitting. The final layer is the probability distribution over classes.

Several researchers have adapted the CNN architecture to perform text classification. Ren et al. [27] proposed a context-based CNN for sentiment analysis in a Twitter dataset, incorporating context information from relevant tweets into the model in the form of word embedding vectors; Yang Wang et al.[30]

designed a hybrid CNN to integrate metadata with text content for fake news classification. For sarcasm detection in social media, Amir et al.[1] introduced a CUE-CNN model which learns embeddings which represent both text content and user information.

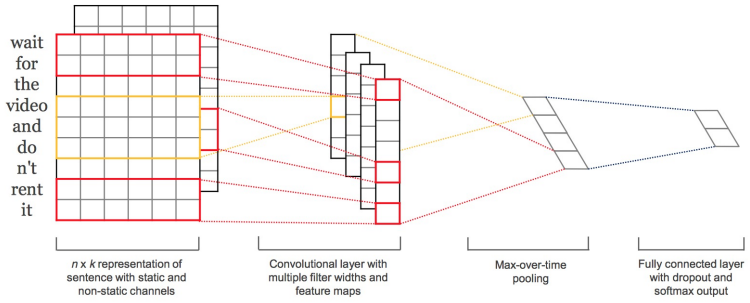

Fig. 4: The CNN model proposed by Kim[17]

The superiority of the CNN model is its ability to mine the relations in the contextual windows and capture the 'local' information such as semantic clues through the convolutional filters. However, given multiple filters with a large number of trainable parameters, the CNN is a heavy data model which usually requires a huge amount of training data. In addition, a critical issue of CNNs is the inability to handle sentences of variable length as input due to the restriction of a fixed input size. To address this particular issue, research has focused on the recurrent neural network.

Recurrent Neural Network The RNN is an extension of a deep neural model that has ability to handle variable-length sequence input. Instead of learning features by the traditional feedforward structure, the RNN involves recurrent units which can use the information of the previous states. In addition, this architecture can effectively address the issue that the input of text content is in fixed size. Figure 5 illustrates the basic RNN framework and unfolded into a graph of the timesteps at time t. The input x_t is fed to the model at timestamp t, S_t is the hidden layer that captures input information x_t and previous state S_{t-1} at timestamp $t-1$, o_t illustrates the output of the model.

Although the RNN model can capture the information from previous states, the key drawback is the vanishing gradients problem which make is difficult to learn and tune parameters from earlier states in the network. The limitation is addressed by two advanced models, gated recurrent units (GRU) and long short-term memory (LSTM). As shown in Fig 6, the normal recurrent unit is replaced by the these two variant units with multiple useful gates. For GRU unit, the gates r and z are designed to control long-term and short-term dependencies which can mitigate against the vanishing gradients problem; For the LSTM unit,

in addition to updates and reset gates, it adds one more cell c as memory for the previous state, and the gate o stands for how much information should the cell output.

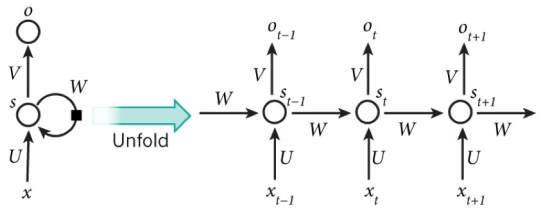

Fig. 5: The RNN structure by Young et al.[33]

To date, RNN based models have been widely applied in text classification. Tang et al.[29] employed a gated RNN architecture in sentiment analysis, which showed a superior performance over a standard RNN model. Wang et al.[31] applied LSTM to predict polarities of tweets and gained 1% better accuracy comparing to the standard RNN model. Typically, the standard LSTM is a single direction structure that can only capture textual information from one directional sequence. A bidirectional LSTM, consisting of two LSTMs that are run in parallel, has proved to be useful in text classification[34]. In addition, Tai et al.[28] developed a variant LSTM model that is based on a tree topology. Rather than the traditional LSTM unit composed from the current timestamp input and previous state, the tree-LSTM unit composes from the current timestamp and previous tree-based state. This model shows superiority for sentiment classification than the standard LSTM.

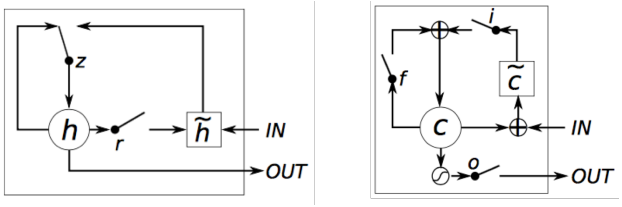

Fig. 6: The structures of GRU(Left) and LSTM(Right) by Chung et al.[7]

2.3 Application in Abusive Detection

Early research in text classification on addressing abusive social media comments focused on exploring useful information such as lexical features[5], the users' profile[8] and historical activities[9]. Djuric[11] et al. are the forerunners of implementing a neural network architecture to generate a distributed feature

representation for hate speech detection. They used paragraph2vec[20] for the modelling of comments. Compared to the BoW representation, the classification accuracy increased from 0.78 to 0.80 compared with a logistic regression classifier. Subsequently, Nobata et al.[23] also conducted a set of comprehensive experiments to evaluate the performance of a variety of representations for abusive comments. They compared the paragraph2vec[20] to a number of feature representations including n-grams, linguistic and syntactic. Using an SVM classifier, the results indicated that using paragraph2vec to generate comment embeddings outperformed the linguistic and syntactic handcrafted features. In addition, they also show the performance of simply using an averaging strategy over the pre-trained word embeddings is better to the ngrams feature representation in most of datasets.

Furthermore, there are an increasing number of researchers who started to work on complex deep neural networks for tackling the problem of abusive text detection. Badjatiya et al.[2] investigated CNNs for hate speech detection in tweets, which significantly outperformed the traditional methods such as Logistic Regression and SVM; Gamback et al. [12] also conduct CNN architecture to classify tweets into four categories include racism, sexism, both (racism and sexism) and neither, they modified traditional CNN input with word embedding by concatenating character ngrams; Park et al.[24] proposed an improved CNN model that combined word embeddings and character embeddings as well. Mehdad et al. [21] implemented RNN using characters as input instead of words, which achieved an increase of approximately 8% in average class accuracy. An advanced RNN model, bi-directional LSTM with attention mechanism which adds weights for importance of each input, was proposed by Gao et al.[13] and Del Vigna et al.[10]. Both of them achieved better performance compared to the one-directional LSTM. In addition, Pavlopoulos et al.[25] also showed the attention mechanism improves the performance of the RNN model when dealing with abusive comments in the Greek language.

3 Experiments - Abusive UGC Detection

From our previous paper[4, 3], we have implemented traditional text classification techniques to tackle abusive detection. In this section, we attempt to use the fundamental neural network strategy to deal with this issue. We first compare word embeddings to a variety of traditional text feature representations include ngrams at word level, ngrams at character level, and handcrafted features. In addition, we investigate the performance of using a recent neural network classifier. The structure of this section is as follows: we describe the datasets that are used in our experiments; we then detail the methodologies of our experiments; finally, the experimental results are presented and discussed.

3.1 Datasets

All datasets used for abusive detection in this paper were extracted from social media websites. Although these websites include a variety of user content sources

including forums, micro-blog, media-sharing, news article discussion, chat and Q&A, they share the common characteristic that they allow online users to freely post their comments. We identified 8 published datasets[3] that have been gathered from different social media platforms: Twitter, MySpace, Formspring, YouTube, SlashDot. Given that these are published datasets in the research domain, we have assumed that their labelling strategies are correct and that the label results are reliable. We also used our own abusive content dataset, collected from a news site and labelled using crowd sourced labelling[4]. The following Table 1 is an overview of each dataset, showing basic information including the source type, the number of instances, average number of words across instances, and the proportion of positive (abusive) and negative instances.

Table 1: Datasets Summary Statistics, D1-D8[3], D9[4]

Dataset with Reference	Dataset Style	#of Instances	Avg Len (words)	Class Dist. (Pos./Neg.)%
D1	Micro-Blog	3110	15	42/58
D2	Video-Sharing	3466	211	12/88
D3	Forum	1710	337	23/77
D4	Q&A	13153	26	6/94
D5	Chat	4802	5	1/99
D6	Forum	4303	94	1/99
D7	Forum	1946	56	3/97
D8	Micro-Blog	1340	13	13/87
D9	News Discussion	2000	59	21/79

For each dataset, we carried out pre-processing operations as follows: all letters were changed to the lowercase; the mentioned names started with '@' symbol were replaced by the anonymous text "@username"; links following with "http://" or "https://" were replaced by the generic term. Considering the comments are typically short, we did not remove stop-words nor implement word stemming.

3.2 Methodology

We employed Support Vector Machines (SVMs) with a linear kernel, one of the most efficient classifiers[15], as our classification algorithm. We established baseline results by using ngrams feature representation, implemented in two ways: 1-4 word level and 2-4 character level. Based on our previous work[3], we normalized features values, and used document frequency (1% as threshold) to reduce the features where the most and least frequent 1% of terms are excluded.

To validate our results, we applied stratified 10-fold cross validation on each dataset. In addition, Table 1 shows that most of the datasets are imbalanced, with the positive instances (abusive) far less in number than the negative instances (non-abusive). We used resampling to randomly oversample the minority class of our training data and averaged results over three iterations.

The results of our experiments are reported using a standard classification measure recall, which measures ability to find all instances of a specific class. Our working assumption is that the consequence of failing to detect abusive content is more serious than the non abusive content being predicted as abusive. Therefore, we focus on abusive recall rather than non-abusive recall. Equation 1 shows the calculation where TruePositive is the proportion of abusive comments correctly classified as abusive, and FalseNegative is the proportion of abusive comments were wrongly classified as non-abusive. Average recall is also reported.

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives} \tag{1}$$

3.3 Experiments & Results

We present the results in Table 2. User generated comments are usually free-format in style and ngrams at word level capture word misspellings and abbreviations. Therefore, the performance of using ngrams in word level is normally worse than the ngrams at character level. In addition, based on the results of character-level ngrams, extracting additional features that capture syntactic and semantic information[4] achieves better results in abusive detection across 9 datasets. We applied the paired t statistical test (p=0.0353) to confirm the difference.

Next, we analyzed the embedding representation which is averaging the word embeddings that are appear in the comment. The existing word embeddings we used in our paper are collected from Glove across 4 different dimensions (50,100,200,300) [1] which were pre-trained on the Wikipedia data corpus. Although averaging is one of the most naive approaches in using word vectors, it shows better performance to ngrams for most of the datasets. In general, using higher dimension word embedding achieves better performance since the higher dimension vectors contain in theory more information than the lower dimension ones. However, in our experiments, some datasets show the opposite results. For example, in D5, the performance of using 300-dimension vectors slumped 10% when compared to the 50-dimension vectors; In D6 also, abusive recall rate decreased as higher dimension vectors were used. The selection of appropriate word embeddings appears to play an important role when dealing with the specific classification task. One reason for this counter intuitive result is the difference between the training corpus language and the language of our posts. In this paper, the Glove word embeddings used are trained on the Wikipedia text corpus which generally uses a formal language style. However, social media datasets typically contain casual expressions, meaning that the word context information captured in Wikipedia-trained word embeddings may be different from that used in the abusive datasets. In the future, we will attempt to use pre-trained word embedding on a source corpus similar to the experimental dataset to boost the classifier accuracy.

FastText[16] is a one-hidden layer neural network text classifier. We analyzed this model in two ways: with and without using pre-trained word embeddings. As

[1] https://nlp.stanford.edu/projects/glove/

shown in Table 2, we note that the performance of FastText in abusive content detection is in fact worse than the other feature representations with an SVM classifier. However, FastText is an efficient classifier with a much faster execution time than the SVMs. We attribute the classification results to the straightforward structure of FastText. In addition, using pre-trained word embeddings in FastText does not guarantee better results, which also may be due to the data source used for the pre-trained word embeddings.

Table 2: Results of Abusive Detection across 9 Datasets. The results are shown as percentage recall, the abusive recall is outside of thebrackets, average recall is inside brackets. The highest abusive recall is displayed in bold.

	D1	**D2**	**D3**	**D4**	**D5**	**D6**	**D7**	**D8**	**D9**
Ngrams (1-4 Word)	70(75)	35(62)	**91(93)**	62(77)	58(78)	12(56)	18(58)	65(78)	33(60)
Ngrams (2-4 Char)	**75(77)**	33(62)	89(93)	66(80)	57(78)	18(59)	18(58)	70(83)	35(62)
Ngrams+Feat.	**75(77)**	40(64)	89(93)	**67(80)**	57(78)	18(59)	22(60)	73(85)	40(64)
Avg. of Glove50	62(68)	13(53)	56(70)	49(68)	60(76)	**61(74)**	50(68)	75(82)	36(61)
Avg. of Glove100	65(71)	30(59)	66(76)	59(74)	58(76)	51(71)	48(68)	77(85)	48(66)
Avg. of Glove200	66(72)	31(60)	78(82)	62(76)	**60(78)**	44(69)	48(69)	**78(85)**	48(66)
Avg. of Glove300	68(72)	33(60)	81(85)	65(77)	54(76)	43(69)	**50(70)**	74(83)	**50(68)**
fastText	72(75)	**42(61)**	70(62)	65(76)	58(79)	31(63)	23(58)	47(71)	30(56)
fastText+Glove100	73(76)	37(62)	60(76)	46(71)	58(79)	21(60)	21(60)	62(79)	41(65)

At this stage, we have not found a general model that performs best across all 9 datasets. The ngrams with syntactic & semantic information achieves significantly better results to the standard ngrams in 7 of our 9 datasets. Averaging word embedding is considered as the most straightforward approach to generate sentence vectors and achieves good results in our experiments. In addition, the FastText simple neural network shows poor performance in the abusive detection task. We suggest that an advanced structure needs to be designed for abusive detection in future work.

4 Conclusions & Future Work

The purposes of this paper were two fold: (1) to investigate how off-the-shelf deep neural networks have been used across the two tasks within text classification - feature representation and the classification tasks itself and (2) to run preliminary experiments in abusive detection across 9 different social media datasets. We highlight the following aspects from our work: Firstly, we systematically summarized current neural models and categorized them into two modes, unsupervised approaches for generating distributed feature representations and supervised approaches for classification. Secondly, we attempted to compare the classification performance of using traditional feature representation to word embedding feature. Simply averaging pre-trained word embeddings has better results to ngrams feature representation in most of datasets. In addition, we employed a recent neural network, FastText. Due to its shallow architecture,

the performance is unexceptional. Using pre-trained word embeddings can not guarantee better results possibly probably due to the characteristics of corpus used for pre-training the embeddings being different from our datasets. We will validate this assumption in our subsequent experiments.

The ultimate goal of our research is to develop a powerful classification model that can assist social media moderators to detect abusive comments efficiently and effectively. The path to the goal can be divided into two directions, designing appropriate features for abusive comments and designing an outstanding model for detection. In future work, we will focus our research on both directions by exploring deep neural networks in unsupervised mode and supervised mode.

References

1. Amir, S., Wallace, B.C., Lyu, H., Silva, P.C.M.J.: Modelling context with user embeddings for sarcasm detection in social media. arXiv preprint arXiv:1607.00976 (2016)
2. Badjatiya, P., Gupta, S., Gupta, M., Varma, V.: Deep learning for hate speech detection in tweets. In: Proceedings of the 26th International Conference on World Wide Web Companion. pp. 759–760 (2017)
3. Chen, H., Mckeever, S., Delany, S.J.: Harnessing the power of text mining for the detection of abusive content in social media. In: Advances in Computational Intelligence Systems, pp. 187–205. Springer (2017)
4. Chen, H., Mckeever, S., Delany, S.J.: Presenting a labelled dataset for real-time detection of abusive user posts. In: Proceedings of the International Conference on Web Intelligence. pp. 884–890. ACM (2017)
5. Chen, Y., Zhou, Y., Zhu, S., Xu, H.: Detecting offensive language in social media to protect adolescent online safety. In: Privacy, Security, Risk and Trust (PASSAT), 2012 International Conference on and 2012 International Confernece on Social Computing (SocialCom). pp. 71–80. IEEE (2012)
6. Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078 (2014)
7. Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555 (2014)
8. Dadvar, M., de Jong, F.M., Ordelman, R., Trieschnigg, R.: Improved cyberbullying detection using gender information (2012)
9. Dadvar, M., Trieschnigg, D., Ordelman, R., de Jong, F.: Improving cyberbullying detection with user context. In: ECIR. pp. 693–696. Springer (2013)
10. Del Vigna12, F., Cimino23, A., Dell'Orletta, F., Petrocchi, M., Tesconi, M.: Hate me, hate me not: Hate speech detection on facebook (2017)
11. Djuric, N., Zhou, J., Morris, R., Grbovic, M., Radosavljevic, V., Bhamidipati, N.: Hate speech detection with comment embeddings. In: Proceedings of the 24th International Conference on World Wide Web. pp. 29–30. ACM (2015)
12. Gambäck, B., Sikdar, U.K.: Using convolutional neural networks to classify hate-speech. In: Proceedings of the First Workshop on Abusive Language Online. pp. 85–90 (2017)
13. Gao, L., Huang, R.: Detecting online hate speech using context aware models (2017)

14. Hill, F., Cho, K., Korhonen, A.: Learning distributed representations of sentences from unlabelled data. arXiv preprint arXiv:1602.03483 (2016)
15. Joachims, T.: Text categorization with support vector machines: Learning with many relevant features. Machine learning: ECML-98 pp. 137–142 (1998)
16. Joulin, A., Grave, E., Mikolov, P.B.T.: Bag of tricks for efficient text classification. arXiv preprint arXiv:1607.01759 (2016)
17. Kim, Y.: Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882 (2014)
18. Kiros, R., Zhu, Y., Salakhutdinov, R.R., Zemel, R., Urtasun, R., Torralba, A., Fidler, S.: Skip-thought vectors. In: Advances in neural information processing systems. pp. 3294–3302 (2015)
19. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. pp. 1097–1105 (2012)
20. Le, Q., Mikolov, T.: Distributed representations of sentences and documents. In: Proceedings of the 31st ICML-14. pp. 1188–1196 (2014)
21. Mehdad, Y., Tetreault, J.R.: Do characters abuse more than words? In: SIGDIAL Conference. pp. 299–303 (2016)
22. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
23. Nobata, C., Tetreault, J., Thomas, A., Mehdad, Y., Chang, Y.: Abusive language detection in online user content. In: Proceedings of the 25th International Conference on World Wide Web. pp. 145–153 (2016)
24. Park, J.H., Fung, P.: One-step and two-step classification for abusive language detection on twitter. arXiv preprint arXiv:1706.01206 (2017)
25. Pavlopoulos, J., Malakasiotis, P., Androutsopoulos, I.: Deep learning for user comment moderation. arXiv preprint arXiv:1705.09993 (2017)
26. 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 (2014)
27. Ren, Y., Zhang, Y., Zhang, M., Ji, D.: Context-sensitive twitter sentiment classification using neural network. In: AAAI. pp. 215–221 (2016)
28. Tai, K.S., Socher, R., Manning, C.D.: Improved semantic representations from tree-structured long short-term memory networks. arXiv preprint arXiv:1503.00075 (2015)
29. Tang, D., Qin, B., Liu, T.: Document modeling with gated recurrent neural network for sentiment classification. In: EMNLP. pp. 1422–1432 (2015)
30. Wang, W.Y.: " liar, liar pants on fire": A new benchmark dataset for fake news detection. arXiv preprint arXiv:1705.00648 (2017)
31. Wang, X., Liu, Y., Sun, C., Wang, B., Wang, X.: Predicting polarities of tweets by composing word embeddings with long short-term memory. In: ACL (1). pp. 1343–1353 (2015)
32. Xu, W., Sun, H., Deng, C., Tan, Y.: Variational autoencoder for semi-supervised text classification. In: AAAI. pp. 3358–3364 (2017)
33. Young, T., Hazarika, D., Poria, S., Cambria, E.: Recent trends in deep learning based natural language processing. arXiv preprint arXiv:1708.02709 (2017)
34. Zhou, P., Qi, Z., Zheng, S., Xu, J., Bao, H., Xu, B.: Text classification improved by integrating bidirectional lstm with two-dimensional max pooling. arXiv preprint arXiv:1611.06639 (2016)