Available online at https://www.asjp.cerist.dz/en/PresentationRevue/134

RIST

Information Processing at the Digital Age Journal

Organization of Knowledge and Advanced Technologies (OCTA)

https://multiconference-octa.loria.fr/

Chapter 1: Classification of Hate Speech Using Deep Neural Networks.

Ashwin Geet D'Sa, Irina Illina, Dominique Fohr

Université de Lorraine, CNRS, Inria, LORIA, F-54000, Nancy, France

Abstract

In the Internet age where the information flow has grown rapidly, there is an increase in digital communication. The spread of hatred that was previously limited to verbal communications has quickly moved over the Internet. Social media and community forums that allow people to discuss and express their opinions are becoming platforms for the dissemination of hate messages. Many countries have developed laws to prevent online hate speech. They hold the companies that run the social media responsible for their failure to remove hate speech. However, manual analysis of hate speech on online platforms is infeasible due to the huge amount of data as it is expensive and time consuming. Thus, it is important to automatically process the online user contents to detect and remove hate speech from online media. Through this work, we propose some solutions for the problem of automatic detection of hate messages. We perform hate speech classification using embedding representations of words and Deep Neural Networks (DNN). We compare fastText and BERT (Bidirectional Encoder Representations from Transformers) embedding representations of words. Furthermore, we perform classification using two approaches: (a) using word embeddings as input to Support Vector Machines (SVM) and DNN-based classifiers; (b) fine-tuning of a BERT model for classification using a task-specific corpus. Among the DNN-based classifiers, we compare Convolutional Neural Networks (CNN), Bi-Directional Long Short Term Memory (Bi-LSTM) and Convolutional Recurrent Neural Network (CRNN). The classification was performed on a Twitter dataset using three classes: *hate*, *offensive* and *neither* classes. Compared to the feature-based approaches, the BERT fine-tuning approach obtained a relative improvement of 16% in terms of macro-average F1-measure and 5.3% in terms of weighted F1-measure.

Keywords: Natural language processing; classification; deep neural network; embedding; hate speech;

1. Introduction

Hate speech expresses an anti-social behavior. The topics of the hate can be gender, race, religion, ethnicity, etc. (Delgado and Stefancic, 2014). There is no clear definition of the term *hate speech*. The Council of European Union defines hate speech as: "All forms of expression which spread, incite, promote or justify racial hatred, xenophobia, antisemitism or other forms of hatred based on intolerance, including intolerance expressed by aggressive nationalism and ethnocentrism, discrimination and hostility towards minorities, migrants and people of immigrant origin."[1]. In the following of this paper, we will consider hate and offensive speech. There are few examples of hate speech:

She look like a tranny.
You Asian, they will deport you when they see your eyes.
I'm not going to believe any of the stupid rumors I hear about jews being friends of Christians.
We hate niggers, we hate faggots and we hate spics

Hate speech can be expressed in different forms. Explicit hate speech contains offensive words such as *'fuck', 'asshole'*. Implicit hate speech can be realized by a sarcasm and irony (Waseem *et al*., 2017; Gao *et al*., 2017). While explicit hate speech can be identified using the lexicons that forms the hate speech, implicit hate speech is often hard to identify and requires semantic analysis of the sentence. There are few examples of implicit hate speech:

Affirmative action means we get affirmatively second rate doctors and other professionals.
I will remove all your organs in alphabetical order.
She looks like a plastic monkey doll!

Hate content on the Internet platform can create fear, anxiety and threat to the individuals. In the case of a company or online platform, company or platform may lose its reputation or the reputation of its product. Failure to moderate these contents may cost the company in multiple ways: loss of users, drop in stocks[2], penalty from legal authority[3], etc.

Most of the online platforms such as social media or the forums, generally cannot be held responsible for the propagating of hate speech. However, their inability to prevent its use is the reason for the spread of hate. A report from the news article states that during the recent crisis of COVID-19, there has been a 900 percent surge in the hate speech against people from China and other Asian origins on Twitter.[4]

In many countries, online hate speech is an offense and it is punishable by the law. In this case, the social medias are held responsible and accountable if they do not remove hate speech content promptly.

The manual analysis of such content and its moderation are impossible because of the huge amount of data circulating on the Internet. An effective solution to this problem would be to automatically detect and moderate the hate speech comments.

The automatic detection of hate speech is a challenging problem in the field of *Natural Language Processing* (NLP). The approaches proposed for automatic hate speech detection are based on the representation of the text in the numerical form and on the use of classification models on these numerical representations. In the state-of-the-art on this field, lexical features such as word and character n-grams (Nobata *et al*., 2016), *Term Frequency-Inverse Document Frequency* (TF-IDF), *Bag of Words* (BoW), polar intensity, noun patterns (Wiegand *et al*., 2018) are used as input features. Recently, word embeddings have been used as an alternative to these lexical features. Multi-features-based approach combining various

[1]https://www.article19.org/data/files/medialibrary/3548/ARTICLE-19-policy-on-prohibition-to-incitement.pdf
[2]https://www.telegraph.co.uk/technology/2018/07/27/twitter-stock-sinks-reporting-decline-active-users/
[3]https://www.cnet.com/news/german-hate-speech-law-goes-into-effect-on-1-jan/
[4]https://news.yahoo.com/coronavirus-huge-surge-hate-speech-toward-chinese-twitter-204335841.html

lexicons and semantic-based features is presented by (Almatarneh *et al*., 2019). (Liu *et al*.,2019) used fuzzy methods to classify ambiguous instances of hate speech.

The notion of word embedding is based on the idea that, semantically and syntactically similar words must be close to each other in an n-dimensional space (Mikolov *et al*., 2013). The embeddings trained on huge corpus of data captures generic semantics of the words. Word2Vec embeddings and character n-gram features as input to CNN has been compared by (Gambäck *et al*., 2017). (Djuric *et al*., 2015) proposed low dimension sentence representation using paragraph vector embeddings (Le and Mikolov, 2014). *Global Vectors for word representation* (GloVe) (Pennington et al., 2014) and random embeddings as input to DNN classifiers has been compared by (Badjatiya *et al*. 2017). Recently, sentence embeddings (Indurthi *et al*., 2019) and *Embeddings from Language Models* (ELMo) (Bojkovský *et al*., 2019) were used as input to classifiers for hate speech comment classification.

Deep-learning techniques have shown to be very powerful in classifying hate speech (Mohaouchane *et al*., 2019; Del Vigna *et al*., 2017). The performance of the deep-learning based approaches has outperformed the classical machine learning techniques such as Support Vector Machines (SVM), Gradient Boosting Decision Trees (GBDT) and Logistic Regression (Badjatiya *et al*., 2017). Among deep-learning based classifiers, Convolutional Neural Network (CNN) captures the local patterns in the text (Kim, 2014). The deep-learning based Long Short Term Memory (LSTM) model (Baruah *et al*., 2019) or Gated Recurrent Unit (GRU) model (Cho *et al*., 2014) captures the long-range dependencies. Such properties are important for modelling hate speech (Bodapati *et al*., 2019). (Park *et al*., 2017) designed hybrid CNN by combining word CNN and character CNN to classify hate speech. (Zhang *et al*.,2018) designed Convolutional Recurrent Neural Networks (CRNN) by passing the inputs of CNN to GRU for hate speech classification. (Del Vigna *et al*., 2017) showed that LSTMs performed better than SVM for hate speech detection on Facebook. (Founta *et al*.,2018) used an attention layer along with the Recurrent Neural Network (RNN) to improve the performance of hate speech classification on longer sequence of text.

In this article, we propose a multiclass classification approach for hate speech detection using two powerful word representations: fastText and BERT embeddings. fastText is based on the skip-gram model, where each word is represented as a bag of character n-grams. Thanks to this it is possible to model a large vocabulary and take into account rare words. The BERT's key innovation is to apply the bidirectional training of Transformer, a popular attention model, to language modelling. This is in contrast to previous efforts which looked at a text sequence either from left to right or combined left-to-right and right-to-left training. A model which is bidirectionally trained can have a deeper sense of language context and flow than single-direction language models. In our work, we compare these two embeddings for the task of hate speech multiclass classification. These representations are used as inputs to DNN classifiers, namely CNN, Bi-LSTM and CRNN. Among these DNNs CNN captures local patterns, Bi-LSTM captures long range dependencies within a sentence, and CRNN provides a way to combine the advantages of CNN and Bi-LSTM. The multiclass classification of hate speech involves fine-grained classification between *hate, offensive and ordinary speech.* Moreover, we explore the capabilities of BERT fine-tuning. This work represents the extension of the work presented in (D'Sa *et al*., 2020). Compared to (D'Sa *et al*., 2020) we also study the SVM and CRNN. We evaluate the proposed approaches on a Twitter corpus.

The contributions of our paper are as follow:

- We use fastText embeddings and BERT embeddings as input features to SVM, CNN, Bi-LSTM and CRNN classifiers.
- We perform fine-tuning of the pre-trained BERT model.
- We investigate the multiclass classification of comments**,** we use three classes *hate speech, offensive speech* and *neither* respectively.

The rest of the paper is organized as follows: Section 2 describes the word embeddings. Section 3 presents proposed methodology. Section 4 describes the data and preprocessing. The results are discussed in section 5.

2. Word embeddings

The main idea of word embeddings is to project words in a continuous vector space. In this space, semantically or syntactically related words should be located in the same area. An important advantage of word embedding is that their training does not require a labeled corpus.

The embeddings are generally learned from a very huge unlabeled corpus. This training is time consuming and often requires high-level technical conditions (big GPU, large memory, etc.). Pre-trained word embeddings are made available via Internet and can be used by researchers from around the world for different NLP tasks. For example, Facebook provided fastText model, Google provided several BERT models for different languages. In this paper, we propose to use these pre-trained embeddings. In the following of this section, we will describe the embeddings used in this study.

fastText embedding: It is an extension of Mikolov's embedding (Mikolov *et al*., 2013). The fastText approach is based on the skip-gram model, where each word is represented as a bag of character n-grams (Joulin *et al*., 2016; Bojanowski *et al*., 2017). A vector representation is associated to each character n-gram; words being represented as the sum of these representations. The word representations are learned by considering a window of left and right context words. Unlike Mikolov's embeddings, fastText is able to provide an embedding for misspelled word, rare words or words that were not present in the training corpus, because fastText uses character n-gram word tokenization.

BERT embedding: Currently BERT is one of the most powerful context and word representations (Devlin *et al*., 2019). BERT is based on the methodology of *transformers* and uses *attention* mechanism. Attention is a way to look at the relationship between the words of a given sentence (Vaswani *et al*., 2017). Thanks to that, BERT takes into account a very large left and right context of a given word. It is important to note that the same word can have different embeddings according to the context. For example, the word *bank* can have one embedding when it occurs in the context *the bank account* and a different embedding when it occurs in the context *the bank of the river*. Moreover, BERT model uses word-piece tokenization. For instance, the word *singing* can be represented as two word-pieces: *sing* and *##ing*. Thus, it is possible to have embeddings for rare words, like in fastText.

BERT model can be used in two ways:

- for generating the embeddings of the words of a given sentence. These embeddings are further used as input for SVM and DNN classifiers.
- for fine-tuning a pre-trained BERT model using a task-specific corpus to perform the classification.

3. The proposed methodology

We propose two approaches: ***feature-based*** and ***fine-tuning*** (see Figure 1).

- In the feature-based approach, two steps are performed. First, each comment is represented as a sequence of words or word-pieces and for each word or word-piece, an embedding is computed using fastText or BERT. Secondly, this sequence of embeddings will form the input to the SVM or DNN classifiers which takes the final decision. We use CNN, Bi-LSTM and CRNN models as the DNN-based classifiers.
- in the fine-tuning approach, everything is done in a single step. Each comment is classified by a fine-tuned BERT model.

We classify each comment as o*ffensive*, *hate speech* or *neither*.

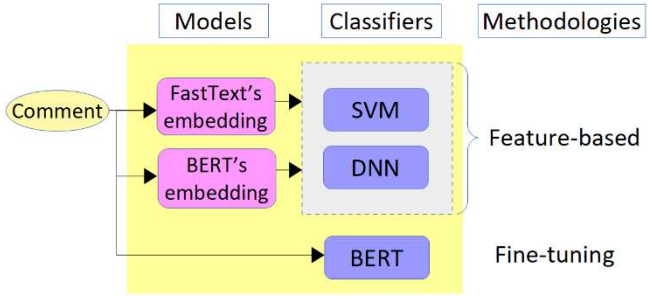

Fig. 1. Proposed methodologies

3.1. Feature-based approaches

For feature-based approaches, we used pre-trained fastText and BERT models to obtain the sequence of embeddings for a given comment. This sequence of embeddings is used as input features to the SVM and DNN classifiers. The sequence should have a fixed size. For this, we extend the short input comments by zero padding.

fastText model: We use pre-trained fastText embedding model and apply this model to generate one embedding for each word of a given comment. Thanks to the bag of character n-grams model of fastText, every word in a given comment will have an embedding, even rare words.

BERT model: Word-piece tokenization is performed on the comment and then used as input to a pre-trained BERT model. BERT model provides contextual embedding for the word-pieces.

The obtained embeddings from either fastText or BERT models are then used as input to the SVM and DNN classifiers.

3.2. Classifiers

For the purpose of multiclass classification, we use SVM and DNN-based classifiers (CNN, Bi-LSTM and CRNN):

- Support Vector Machine (SVM) (Cortes *et al.*, 1995) is a supervised training model. This model projects the input feature vector into non-linear higher dimensional space. Then a linear decision boundary that maximizes minimum separation between training instances is constructed. SVM has been one of the classification algorithms used in many NLP tasks (Salminen *et al.*, 2020).
- CNN were traditionally used in the domain of image processing, and are effective at capturing patterns. (Kim, 2014) demonstrated the efficient use of CNN for NLP on various benchmark tasks.
- Bidirectional LSTM (Bi-LSTM) is a class of RNN models, which overcomes the problem of vanishing gradient problem. Bi-LSTMs are used for sequential data processing and are efficient at capturing long-range dependencies.
- Convolutional Recurrent Neural Networks (CRNN) combines the advantages of CNN and RNN models (Zhang *et al.*, 2018). The input sequence is processed by CNN layers, and the output of the CNN layers are passed through Gated Recurrent Unit (GRU) layers.

3.3. BERT fine-tuning

A BERT pre-trained model can be fine-tuned to a specific task. This consists in adapting of the pre-trained BERT model parameters to a specific task using a small corpus of task specific data. Since BERT is contextual model and pre-trained BERT model is trained on a very huge corpora containing few *hate speech* or *twitter data*, it will be interesting to fine-tune this model with a twitter dataset containing hate speech. For the purpose of classification task, a neural network layer is added on top of the BERT model. So, the weights of this layer and the weights of the other layers of the BERT model are fine-tuned using the task specific dataset.

4. Experimental setup

4.1. Data set

For the purpose of hate speech classification, we used a Twitter corpus (Davidson *et al*., 2017). The tweets are collected based on keywords from *hatebase.org* lexicon. The data set contains 24883 tweets and annotations performed by CrowdFlower. Each tweet is annotated by at least 3 annotators. The annotator agreement is 92%. The labels correspond to three classes: *hate speech*, *offensive language* and *neither*, representing 5.7%, 77.1% and 16.7% respectively. Thus, this data set is an unbalanced data set. Table 3 gives the statistics of the data set after pre-processing.

We followed the 5-fold cross validation procedure as in (Davidson *et al*., 2017). We used 70% of data as training, 20% as test set and 10% as development set. The development set is used to tune the hyper-parameters. The test set is used to evaluate the performance of the proposed approaches.

In our experiments, we use the three classes and the labels provided with the data set: *hate speech*, *offensive speech* and *neither*.

4.2. Text pre-processing

The way the input text is pre-processed plays an important role. For both, fastText and BER, we decided to remove the numbers and all the special characters except *'!', '?', ',', '.'* and apostrophe.

We also performed tweet specific pre-processing. We removed user names (words beginning with symbol '@'). and the word 'RT', indicating *re-tweet*. We split hast-tags in multiple words. For example, *#KillThemAll* is split into *Kill Them All*.

Table 1. Statistics of Twitter data set after pre-processing. K denotes thousand.

	Hate speech	**Offensive speech**	**Neither**	**Total**
Number of tweets	1430	19190	4163	24783
Corpus size (word count)	19.6K	259.5K	62.1K	341.2K
Number of unique words	3.7K	16.2K	9.9K	21.2K
Average number of words per tweet	13.7	13.5	14.9	13.8

4.3. Embedding models

- **fastText embedding:** the model is provided by Facebook[5] and pre-trained on *Wikipedia 2017*, *UMBC web-base* and *statmt.org news* datasets with total 16B tokens. The embedding dimension is 300, the size of the vocabulary is 1M.
- **BERT model:** In our work, we used BERT-base-uncased word-piece model (for English), provided by Google[6] and pre-trained on *BookCorpus* and *Wikipedia* corpora. The model has 12 stacked transformer encoder layers, with 24 attention heads. The embedding dimension is 768, the number of word-pieces is 30K.

4.4. Model configurations

We perform the classification experiments with different hyper-parameters and choose the final configuration based on the best performance obtained on the development set. The best model configurations are detailed below.

For SVM, we use linear SVM classifier with one-versus-rest classification, the squared-hinge loss and L2 regularization. For Bi-LSTM, we used one or two bidirectional LSTM layers with varying LSTM units (between 50 and 128) followed by one or two dense layers with 64 and 256 dense units in the first dense layer and 16 and 64 dense units in the second layer. For CNN, we have used either one or two layers (filter size between 3 and 5), and used between 16 and 64 units, followed by two dense layers having 64 and 256 dense units in the first dense layer and 16 and 64 dense units in the second layer. For CRNN, we have used either one or two layers of CNN (filter size between 3 and 5), followed by one or two layers of GRU (between 50 and 100 units) followed by dense layers. The dense units use *Rectified Linear Unit* activation (ReLU), while the final output neuron uses *sigmoid* activation. We use a varying dropout up to 0.2. We use l2 regularization. The models are trained using Adam optimizer with learning rate of 0.001. For BERT fine-tuning we used maximum sequence length 256, batch size 16, learning rate $2 \cdot 10^{-5}$ and 3 epochs.

We evaluate the performance of our approaches in terms of macro-average F1-measure and weighted-average F1-measure. **F1-measure** is a statistical measure to analyze classification performance. This value ranges between 0 and 1, where 1 indicates the best performance. F1-measure is calculated as follow:

$$F1 = \frac{2*(precision*recall)}{(precision+recall)}$$

where, ***precision*** is the ratio between number of samples correctly predicted as class A and total number of samples predicted as class A by the classifier; ***recall*** is the ratio between number of samples correctly predicted as class A and total number of samples that should be predicted as class A.

The ***Macro-average F1-measure*** computes the arithmetic mean of F1-measures of all classes:

$$macroF1 = \frac{1}{C}\sum_{i=1}^{C} F1_i$$

where C is the total number of classes. For each experiment, we compute an average macro-average F1-

[5]https://fasttext.cc/docs/en/english-vectors.html
[6]https://github.com/google-research/bert

measure obtained from the 5-folds test sets.

The ***weighted-average F1-measure*** computes the weighted arithmetic mean of F1-measures of all classes, weighted by the support count of each class:

$$weightedF1 = \frac{\sum_{i=1}^{C} w_i * F1_i}{\sum_{i=1}^{C} w_i}$$

where C is the total number of classes and w is the support count for each class.

The weighted-average F1-measure gives more importance to majority class. Macro-average F1-measure does not use weights for the aggregation. This results in a greater penalty when a model makes mistakes for the minority class. This measure is often used for an imbalanced dataset. As shown in the table 1, it is the case for our dataset.

5. Results and discussion

Table 2 gives the macro-average F1 and the weighted-average F1 results for the multiclass classification task using SVM, CNN, Bi-LSTM and CRNN classifiers with fastText and BERT embeddings as input features.

From table 2, we observe that both fastText and BERT embeddings provide nearly the same results. Among the classifiers, DNN-based classifiers (CNN, Bi-LSTM and CRNN) perform better than SVM. CNN gives the same level of performance as CRNN, and Bi-LSTM performs slightly better than CNN and CRNN.

Finally, BERT fine-tuning achieved the best performance. Compared to feature-based approaches, we obtained of absolute improvement of 11.6% in terms of macro-average F1-measure (84% versus 72.4%) and absolute improvement of 4.8% in terms of weighted F1-measure (94.4% versus 89.6%). In terms of relative improvement, it represents 16% for macro-average F1-measure and 5.3% for weighted F1-measure. One reason may be that in the feature-based approach the embedding models have not been trained on hate speech or offensive data. On the contrary, the BERT fine-tuning approach is fine-tuned on twitter data to distinguish *hate*, *offensive* and *neither* and this allows to create an accurate model for the task.

Comparing macro-average F1 and weighted-average F1 in table 2, we obtain at least 10% higher weighted-average F1 than the macro-average F1 because the performance of the classification is better for the *offensive speech* class, which has the higher number of samples.

Table 3 shows the confusion matrix for the classification using Bi-LSTM with fastText and BERT embeddings. The confusion matrix obtained by BERT fine-tuning is presented in table 4. From table 3 and table 4, we can notice that the main confusions occur between *hate speech* and *offensive speech* where most of the samples labeled as *hate speech* are predicted as *offensive speech.* This suggest that the model is biased towards classifying tweets as less hateful than the human annotators. This may be due to the imbalance in class distribution within the dataset. The feature-based approach is able to correctly predict up to 31% of the hate speech tweets (table 3), while BERT fine-tuning achieved 53% (table 4).

Table 2. Macro-average and weighted-average F1-measure for different classifiers and different embeddings.

A. Feature-based approaches								
	SVM		**CNN**		**Bi-LSTM**		**CRNN**	
	macro F1	**weighted F1**	**macro F1**	**weighted F1**	**macro F1**	**weighted F1**	**macro F1**	**weighted F1**
fastText embedding	65.8	84.6	70.9	89.2	72.3	89.6	72.0	89.5

BERT embedding	62.01	83.0	71.9	88.9	72.4	89.6	70.9	88.9

B. BERT fine-tuning

	macro F1	**weighted F1**
BERT fine-tuning	**84.0**	**94.4**

Table 3. Confusion matrix for feature-based Bi-LSTM with fastText embeddings in %. The results for Bi-LSTM with BERT embedding are given in parentheses.

True Label	hate	27 (31)	61 (60)	12 (9)
	offensive	2 (2)	95 (95)	3 (3)
	neither	1 (3)	8 (10)	91 (87)
		hate	offensive	neither
		Predicted Label		

Table 4. Confusion matrix for BERT fine-tuning in %

True Label	hate	53	43	4
	offensive	1	98	1
	neither	1	4	95
		hate	offensive	neither
		Predicted Label		

We analyzed some errors produced by the BERT fine-tuning classifier. We chose this model because it achieved the best results among the classification techniques. We see that the hate tweets containing hate words like *pussy*, *bitch*, *cunt*, etc. have often been misclassified as offensive speech. Below are examples of a few hate tweets misclassified as offensive speech:
The fox says you're a cunt.
Feminist aka a bitch
She's a cunt.
you are a pussy without your guns bitch as spic.

Hate speech tweets with words like *trash*, t*eabagger* and tweets with sarcasm have been misclassified as *neither.* The examples of a few hate tweets misclassified as *neither* are given below*:*
Move out of our country teabaggers!
California is home to skater trash
trash eating trash
The reason you called Jesus a monkey is because you knew he was not your color

The tweets belonging to *offensive speech* have been misclassified as *hate speech* in the presence of words like *faggots*, *nigger*, *queer*. For example:
Facebook is for fags
fucking faggot
Come over queer
Alright , amateur grease monkey , show me how its done . Wait , that sounds racist

Moreover, *offensive speech* tweets without hate words and tweets with words like *trash* have been misclassified as *neither*. For example:
Xbox live fags
Truth is ate yellow snow as a child
lol cracker
Take that and shove it up your ass

A very few tweets belonging to *neither* have been mistakenly classified as *hate speech*. Examples are:
I see. So you feel betrayed, your racism comes out.
Weekend is here. What an amazing week this has been. Let's use this extended weekend to celebrate our successes my fellow queer folk.
He's a pretty damn good actor. But as a gay man it's awesome to see an openly queer actor given the lead role for a major superhero film.

Besides, the tweets of *neither* class containing hateful words such as *pussy*, *retard* used in non-hateful content have been erroneously classified as *offensive speech*. Below are examples of some *offensive speeches* misclassified as *neither*:
I'm such a retard sometimes.
My flow retarded.
I love how we can marry and still be the same queer, messy, funny, fabulous, dramatic community we've always been. Why I love us.
momma said no pussy cats inside my doghouse.

Through the above examples, we see that tweets can be misclassified as hate speech or offensive speech based on the dominant hate words present in the tweet. Some *hate speech* tweets can be misclassified as *neither* due to the absence of hate words or due to the presence of ordinary words which was used in hateful context (implicit hate speech). In future work we will perform a deeper analysis of these errors.

6. Conclusion

In this article, we investigated the multiclass classification of hate speech using embedding representation of words and DNNs. The classification was performed on a Twitter data set using a classification in three classes: *hate*, *offensive* and *neither* classes. We have proposed feature-based and fine-tuning approaches for the hate speech classification.

In the feature-based approach, a sequence of word embeddings is used as input for the classifiers. As a word embedding, we investigated fastText embedding and the BERT embedding. Within the framework of feature-based approach, the performances of these two types of embeddings are almost identical. Among the classifiers, SVM, CNN, Bi-LSTM and CRNN were compared. DNN-based classifiers (CNN, Bi-LSTM and CRNN) performed better than the SVM classifier. Among the DNN-based classifiers, CNN and CRNN provided similar results. Bi-LSTM classifier performed slightly better than CNN and CRNN.

The fine-tuning approach is a one-step approach, where the pre-trained BERT model is fine-tuned for our classification task of hate speech. Compared to the feature-based approaches, the BERT fine-tuning approach obtained a relative improvement of 16% in terms of macro-average F1-measure and 5.3% in terms of weighted F1-measure.

The confusion matrices of these approaches show that BERT fine-tuning classified *hate speech* better than the feature-based approaches. The confusion between *hate speech* and *offensive speech* is widespread. Manual analysis of some misclassified examples revealed that most classification errors were due to the presence of specific words. A further study of this problem will be performed in the future.

Acknowledgements

This work was funded by the M-PHASIS project supported by the French National Research Agency (ANR) and German Research Foundation (DFG) under contract ANR-18-FRAL-0005.

References

Delgado, R. and Stefancic, J., 2014. Hate speech in cyberspace. Wake Forest L. Rev., 49, p.319.

Waseem, Z., Davidson, T., Warmsley, D. and Weber, I., 2017. Understanding Abuse: A Typology of Abusive Language Detection Subtasks. In Proceedings of the First Workshop on Abusive Language Online (pp. 78-84).

Gao, L., Kuppersmith, A. and Huang, R., 2017, November. Recognizing Explicit and Implicit Hate Speech Using a Weakly Supervised Two-path Bootstrapping Approach. In Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers) (pp. 774-782).

Nobata, C., Tetreault, J., Thomas, A., Mehdad, Y. and Chang, Y., 2016. Abusive language detection in online user content. In Proceedings of the 25th international conference on world wide web (pp. 145-153).

Wiegand, M., Ruppenhofer, J., Schmidt, A. and Greenberg, C., 2018. Inducing a Lexicon of Abusive Words–a Feature-Based Approach. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers) (pp. 1046-1056).

Almatarneh, S., Gamallo, P., & Pena, F. J. R., 2019. CiTIUS-COLE at semeval-2019 task 5: Combining linguistic features to identify hate speech against immigrants and women on multilingual tweets. In Proceedings of the 13th International Workshop on Semantic Evaluation (pp. 387-390).

Liu, H., Burnap, P., Alorainy, W. and Williams, M.L., 2019. A fuzzy approach to text classification with two-stage training for ambiguous instances. IEEE Transactions on Computational Social Systems, 6(2), pp.227-240.

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

Gambäck, B. and Sikdar, U.K., 2017. Using convolutional neural networks to classify hate-speech. In *Proceedings of the first workshop on abusive language online* (pp. 85-90).

Djuric, N., Zhou, J., Morris, R., Grbovic, M., Radosavljevic, V. and Bhamidipati, N., 2015. Hate speech detection with comment embeddings. In Proceedings of the 24th international conference on world wide web (pp. 29-30).

Le, Q. and Mikolov, T., 2014. Distributed representations of sentences and documents. In International conference on machine learning (pp. 1188-1196).

Pennington, J., Socher, R. and Manning, C.D., 2014. Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 1532-1543).

Badjatiya, P., Gupta, S., Gupta, M. and Varma, V., 2017. Deep learning for hate speech detection in tweets. In Proceedings of the 26th International Conference on World Wide Web Companion (pp. 759-760).

Indurthi, V., Syed, B., Shrivastava, M., Chakravartula, N., Gupta, M., & Varma, V., 2019. Fermi at semeval-2019 task 5: Using sentence embeddings to identify hate speech against immigrants and women in twitter. In Proceedings of the 13th International Workshop on Semantic Evaluation (pp. 70-74).

Bojkovský, M., & Pikuliak, M., 2019. STUFIIT at SemEval-2019 Task 5: Multilingual Hate Speech Detection on Twitter with MUSE and ELMo Embeddings. In Proceedings of the 13th International Workshop on Semantic Evaluation (pp. 464-468).

Mohaouchane, H., Mourhir, A. and Nikolov, N.S., 2019. Detecting Offensive Language on Arabic Social Media Using Deep Learning. In 2019 Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS) (pp. 466-471). IEEE.

Del Vigna12, F., Cimino23, A., Dell'Orletta, F., Petrocchi, M. and Tesconi, M., 2017. Hate me, hate me not: Hate speech detection on facebook. In *Proceedings of the First Italian Conference on Cybersecurity (ITASEC17)* (pp. 86-95).

Kim, Y., 2014. Convolutional Neural Networks for Sentence Classification. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP) (pp. 1746-1751).

Baruah, A., Barbhuiya, F., & Dey, K., 2019. ABARUAH at SemEval-2019 Task 5: Bi-directional LSTM for Hate Speech Detection. In Proceedings of the 13th International Workshop on Semantic Evaluation (pp. 371-376).

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y., 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP) (pp. 1724-1734).

Bodapati, S., Gella, S., Bhattacharjee, K., & Al-Onaizan, Y., 2019. Neural Word Decomposition Models for Abusive Language Detection. In Proceedings of the Third Workshop on Abusive Language Online (pp. 135-145).

Park, J.H. and Fung, P., 2017. One-step and Two-step Classification for Abusive Language Detection on Twitter. In *Proceedings of the First Workshop on Abusive Language Online* (pp. 41-45).

Zhang, Z., Robinson, D. and Tepper, J., 2018. Detecting hate speech on twitter using a convolution-gru based deep neural network. In European semantic web conference (pp. 745-760). Springer, Cham.

Founta, A.M., Chatzakou, D., Kourtellis, N., Blackburn, J., Vakali, A. and Leontiadis, I., 2019. A unified deep learning architecture for abuse detection. In *Proceedings of the 10th ACM Conference on Web Science* (pp. 105-114).

D'sa, A. G., Illina, I., and Fohr, D., 2020. BERT and fastText Embeddings for Automatic Detection of Toxic Speech. In Proceedings of OCTA 2020 - Organization of Knowledge and Advanced Technologies.

Joulin, A., Grave, E., Bojanowski, P., Douze, M., Jégou, H. and Mikolov, T., 2016. Fasttext. zip: Compressing text classification models. arXiv preprint arXiv:1612.03651.

Bojanowski, P., Grave, É., Joulin, A. and Mikolov, T., 2017. Enriching Word Vectors with Subword Information. Transactions of the Association for Computational Linguistics, 5, pp.135-146.

Devlin, J., Chang, M.W., Lee, K. and Toutanova, K., 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers) (pp. 4171-4186).

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. and Polosukhin, I., 2017. Attention is all you need. In Advances in neural information processing systems (pp. 5998-6008).

Cortes, C. and Vapnik, V., 1995. Support-vector networks. Machine learning, 20(3), pp.273-297.

Salminen, J., Hopf, M., Chowdhury, S.A., Jung, S.G., Almerekhi, H. and Jansen, B.J., 2020. Developing an online hate classifier for multiple social media platforms. Human-centric Computing and Information Sciences, 10(1), p.1.

Davidson, T., Warmsley, D., Macy, M. and Weber, I., 2017. Automated hate speech detection and the problem of offensive language. In Eleventh international aaai conference on web and social media.