NLP at SemEval-2019 Task 6: Detecting Offensive language using Machine Learning models and Simple Transformer models

Aleksandra Pushkina
University of Tübingen
aleksandra.pushkina@student.uni-tuebingen.de

Abstract

Offensive language is commonly defined as any communicative interaction that disparages a person or a group on the basis of some characteristic such as race, ethnicity, gender, sexual orientation, nationality, religion, or other characteristics. According to the Twitter and Facebook policy, offensive language is also considered as 'threats of violence towards others. Given the huge amount of user-generated content on Twitter, the problem of detecting, and therefore possibly opposing the HS expansion, is becoming fundamental, for instance for struggling against homophobia, misogyny and xenophobia.

1 Introduction

Offensive language or hate speech (further, HS) phenomenon is widely explored by many authors (Eadicicco, 2014; Kettrey & Laster, 2014; Watch, 2014; Moulson, 2016; Zeerak and Hovy, 2016; ElSherief et al., 2018; Basile et al., 2019). Drawing upon the various definitions, HS can be defined as language that is used to express hatred towards a targeted group or is intended to be derogatory, to humiliate, or to insult the members of the group (Davidson et al., 2017).

In many countries, including the United Kingdom, Canada, France and Russia there are laws prohibiting hate speech. In these countries, it tends to be defined as speech that targets groups in a way that could promote violence or social disorder. People convicted of using hate speech can face imprisonment. These laws extend to the internet and social media, leading many sites to create their own provisions against hate speech.

There is no doubt that HS should be detected and moderated. One of the reasons for that is that there is possible connection between hate speech and actual hate crimes (Hate Speech Watch, 2014).

The concept of hate speech is really sketchy and there is no general definition. For that reason, NLP research on HS is also vague and has been limited mostly to an investigation of the most effective features (Waseem & Hovy, 2016).

A key challenge for automatic hate-speech detection on social media is the separation of hate speech from other instances of offensive language. But not always offensive language is expressed with obscene vocabulary – it could be a comparison with animals or pasquinade. Also, people often use terms that are highly offensive to certain groups but in a qualitatively different manner (Davidson et al., 2017). For example, some African American use the term *n*gga* in everyday language; some trap artists often use words like *b*tch*, and gamers use some words that can be easily considered as hatred, like *f*g*. But in all of these cases people don't mean to insult any group. Therefore, it is very important to look at the context of a communication (who is the author of message and etc.) before claiming something as HS. Moreover, with the increase of restricting policies in social media, people find other ways to express HS in their messages: for example, it is now popular among hate propagators to use triple parenthesis around name to indicate that the person is a Jew in a negative context, like *(((person name)))*.

The SemEval 2019 - OffensEval Shared Task (Zampieri et al., 2019b) focuses on the above problem by querying systems capable of detecting offensive language in social media data. The Offensive Language Identification Dataset (OLID)[1] contains a collection of 14,200 annotated

[1] https://sites.google.com/site/offensevalsharedtask/olid

English tweets using an annotation model that encompasses following three levels:

Sub-task A: Offensive language identification (104 participating teams)

Sub-task B: Automatic categorization of offense types (71 participating teams)

Sub-task C: Offense target identification (66 participating teams).

I analyze the impact of various lexical features in conjunction with sentiment analysis for hate speech detection. I also present deep learning classifier for detecting HS.

2 Related work

Using computational approaches to identify offense, hostility, and hate speech in user-generated content is one of the most effective strategies for addressing the identification of offensive language (e.g. posts, comments, microblogs, etc.). As indicated by articles over the last years, this topic has gotten a lot of attention recently: Schmidt and Wiegand (2017) present an overview of existing research in this field as well as a list of features that are used machine learning and deep learning models.

In their studies, Waseem and Hovy (2016), found that using a character n-gram based approach provides a solid foundation. Nevertheless, in their research they used additional information about the authors of tweets, like gender and demographic information. This gave additional surplus in f score in 10 points.

Thomas Davidson, Dana Warmsley, Michael Macy and Ingmar Weber (2017) in their research used Tf-Idf 1–3-word n-grams with additional sentiment scores to define hate speech. In combination with SVM they got sufficient results. They also claimed that lexical methods are effective ways to identify potentially offensive terms but are inaccurate at identifying hate speech (Davidson et al., 2017: 3).

In their article "Hate speech detection: Challenges and solutions", Sean MacAvaney, Hao-Ren Yao, Eugene Yang, Katina Russell, Nazli Goharian and Ophir Frieder (2019), proposed multi-view SVM classifier with character n-grams as initial features: It applies a multiple-view stacked Support Vector Machine (mSVM). Each type of feature (e.g., a 'word'-Tf-Idf unigram) is fitted with an individual Linear SVM classifier (inverse regularization constant C = 0.1), creating a view-classifier for those features. They further combined the view classifiers with another Linear SVM (C = 0.1) to produce a meta-classifier.

3 Experiment Details

In this section, I firstly present data and its distribution. I will further discuss several methods used in the experiments and implement different machine learning and deep learning models for offensive detection task after introducing evaluation metrics. Comparisons and conclusions will be derived at the end.

3.1 Target Task and Data

The 2019 edition of OLID shared Task with the title "OffensEval 2019" is based on the assumption that the target of offensive messages is an important variable that allows us to distinguish between, for example, hate speech, which often consists of insults directed at a group, and cyberbullying, which is usually directed at individuals. Zampieri et al. (2019b, 75) describe the OLID dataset as following: "OLID is annotated following a hierarchical three-level annotation schema that takes both the target and the type of offensive content into account".

The data consists in a testsets per sub-task (A, B, C) and CSV files with labels. I only worked with Offensive language identification Task within the framework of this project. Each testset consists of ID of the tweet and the tweet itself. The CSV files consist of ID of the tweet and the annotated label: OFF when the tweet contains offensive speech and NOT if it does not.

The number of tokens in each sentence is rather unbalanced. The histogram of the number of tokens of the dataset is shown in the Figure 1. The distribution shows that most of the sentences are 5-20 words long with average ≈ 20 tokens per sentence.

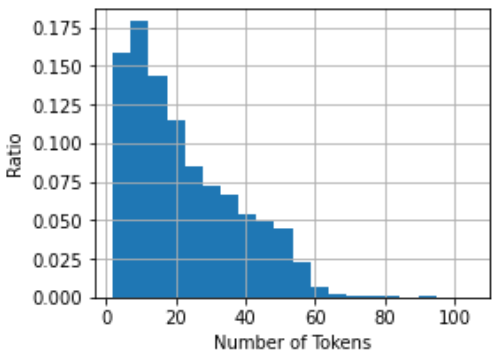

Figure 1. Number of Tokens in Sentences Distribution.

3.2 Data Distribution

In terms of class distribution, the labels (OFF/NOT) were distributed unbalanced across the dataset (240 tweets with offensive language and 620 neutral tweets).

3.3 Methods

While there is a large body of work devoted to detecting offensive language and hate speech (Schmidt & Wiegand, 2017), to accurately detect these phenomens is still extremely challenging (CNN Tech, 2016). To complete this task, I decided to use one existing method, which showed good results when finding a hate speech in tweets, namely an **N-gram Method** by Davidson et al. (2017).

As for N-gram Method, I decided to use not only word n-grams as described in the method of Davidson et al. (2017), but also character n-grams, since Waseem & Hovy (2016) found that using character n-grams outperforms using word n-grams by at least 5 F_1-points.

It seems important to check how already existing methods can be used to identify offensive language in the tweets.

3.4 Feature Extraction

After bringing the dataset into the desired form, it is necessary to extract relevant features by converting textual documents to numerical vectors and afterwards concatenate all the vector arrays in the matrices. This should be done in order to be able to use the machine learning models.

In this term paper, following settings were used for scikit-learn vectorizers (Pedregosa et al., 2011) in **N-gram Method**:

• Word representation: Tf-Idf Vectorizer

In comparison to another popular vectorizer, Count Vectorizer, Tf-Idf doesn't only give number of frequencies with respect to index of vocabulary, but weighs a keyword in any content and assigns importance to that keyword based on the number of times it appears in the document. As explained on the Wikipedia: "The tf–idf is the product of two statistics, term frequency and inverse document frequency."[2]

$$\text{tf-idf}(t,d) = \text{tf}(t, d) \times \text{idf}(t)$$

where tf (t, d) is the raw count of a term in a document, i.e., the number of occurrences of t in document d, and idf (t) is defined as follows:

$$\text{idf}(t) = \log \frac{n}{1+\text{df}(t)}$$

where: n is the total number of documents in the corpus; df (t) is the number of documents containing the term t.

• N-grams:

-unigram, bigram & trigram for 'word'-analyzer

- characters 3-6-gram for 'char'-analyzer

• Minimum document frequency: 5

• All characters were converted to lowercase

• All stop words were removed for machine learning models[3], however not for BERT models: I removed not only the list of English stop words, but also all words with more than thousand occurrences within the dataset.

• All emojis, emoticons, symbols and pictographs were also removed from the dataset.

Following Davidson and colleagues' approach I also want to capture information about the syntactic structure and for that I construct Part-of-Speech (POS) tag unigrams, bigrams, and trigrams. All other features of their approach (Davidson et al., 2017: 2) seemed to be redundant as I didn't see any improvement of the results applying them.

4 Experiments and Results

In this section, I will implement machine learning and deep learning models for hate spreader prediction task on OLID dataset.

[2] Tf-Idf Vectorizer: https://en.wikipedia.org/wiki/Tf–idf

[3] Recently, it has become a common opinion that it is better not to remove the stop words, as they carry additional information, which, if removed, can badly affect the performance of the model. That is why I also tested the two best models (neural network model and random forest model) without removing the stop words. Unfortunately, I did not see any improvement in the results. Confusion matrices of both models are presented in the appendix.

Evaluation metrics will be introduced first following by five machine learning models and three BERT models. Comparisons and conclusions will be derived at the end.

4.1 Evaluation Metrics

In order to evaluate machine learning and deep learning models in a fair way, the following evaluation metrics will be used:

• Accuracy classification score: is the fraction of predictions the model got right.

$$\text{Accuracy} = \frac{Number\ of correct\ predictions}{Total\ number\ of\ predictions}$$

• Weighted averaged F_1 score:" calculates the F_1 score for each class independently but when it adds them together uses a weight that depends on the number of true labels of each class"[4].

$$F_{1\ class1} \times W_1 + F_{1\ class2} \times W_2 + \cdots + F_{1\ class\ n} \times W_n$$

where F1 is defined as follows:

$$F_1 = \frac{2 \times precision \times recall}{precision + recall}$$

For an acceptable classifier, I would prefer high precision and recall values. Classification reports are used to obtain the values of these metrics in a text format per class. For that reason, it is essential to look at weighted averaged F_1 to make sure that one has a good model.

• Confusion matrix: "table for visualizing how an algorithm performs with respect to the human gold labels, using two dimensions (system output and gold labels), and each cell labeling a set of possible outcomes" (Jurafsky & Martin, 2009: 11).

Date splitting was performed using train test split function of the scikit-learn library (Pedregosa et al., 2011). All features were split in proportion 70 to 30. The random state remained the same (= 42) to maintain consistency of the result.

A general overview of the results and evaluation of all models can be seen in Table 1.

Model (Tf-Idf 'word')	Accuracy	Macro avg F_1 Score
Linear Regression	0.6815	0.6482
Random Forest	0.6715	0.6467
Neural Network	0.6891	0.6540
K-Neighbor Classifier	0.6901	0.4849
Model (Tf-Idf 'char')	**Accuracy**	**Macro avg F_1 Score**
Linear Regression	0.7014	0.6623
Random Forest	0.7583	0.7102
Neural Network	0.7024	0.6620
K-Neighbor Classifier	0.6916	0.4619
Model (Transformer)	**Accuracy**	**Macro avg F_1 Score**
Small BERT 32	0.8273	0.7905
Small BERT 64	0.8077	0.7697
Small BERT 128	0.8079	0.7750
Large BERT	0.8049	0.7653
RoBERTa 32	0.7850	0.7454
RoBERTa 64	0.7998	0.7693
RoBERTa 128	0.8039	0.7765

Table 1. Model Evaluations on the Testing Set

4.2 Machine Learning Models

This section will present the models I used i.e., Logistic Regression (further, LR), Random Forest (further, RF), Neural Network (further, NN) and K-Neighbors Classifier (further, KNN) for the author prediction task. First, the results will be presented using Tf-Idf-word-vectorizer, then Tf-Idf - char-vectorizer. All applied classifiers are from scikit-learn library (Pedregosa et al., 2011). In order to save space, only a confusion matrix of the best model of each type of model will be presented in the upcoming section. Confusion matrices of other models can be found in the Appendices (see below).

4.2.1 Logistic Regression

The first applied machine learning model is Logistic Regression (Hastie et al. 2001) with L2 penalty and L-BFGS solver. The accuracy and weighted F_1 score on the testing set of all methods are shown in the Table 1, and the confusion matrix

[4] F_1-Score: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html

using Tf-Idf 'char'-Vectorizer is shown in the Figure 2.

Tf-Idf 'char' method with Logistic Regression model shows the second-best result together with Neural Network model with 70% accuracy and macro averaged F1 equals 0.66. Both 'char'-Tf-Idf and 'word'-Tf-Idf methods outperform the result reported by Davidson et al. (2017: 3). In their experiment, almost 40% of hate speech is misclassified: the reported precision and recall scores for the hate class are 0.44 and 0.61 respectively.

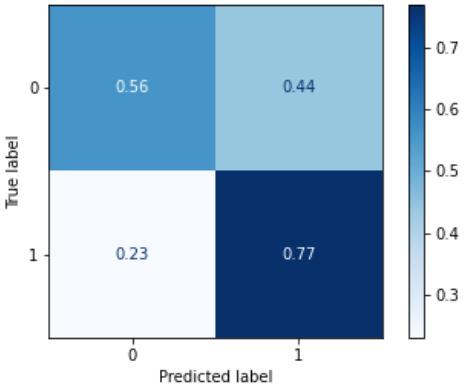

Figure 3. Confusion Matrix for Logistic Regression on the Testing Set using TF-IDF 'char'-Vectorizer

4.2.2 Random Forest

The second applied machine learning model is Random Forest (Hastie et al., 2001) with n_estimators equal 500, minimum samples leaf 10, and extra criterion - Gini impurity. The accuracy and weighted F1 score on the testing set of all methods are shown in the Table 1. The confusion matrix presented in Figure 3 demonstrates the result of the RF model with the features of the 'char'-Tf-Idf method.

Tf-Iidf 'char' method with Random Forest model outperforms all other methods with 75% accuracy and 0.71 of the macro averaged F1 score.

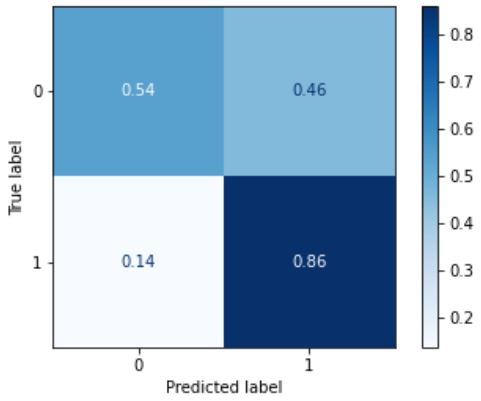

Figure 2. Confusion Matrix for Random Forest on the Testing Set using TF-IDF 'char'-Vectorizer

But at the same time, when we look at the Figure 3, we see that recall of class 0 remained rather law, 46% of the class 0 was misclassified. The better result can be observed in the other method: 'word'-Tf-Idf (recall of class 0 = 61% vs. 39% misclassified). It seems that the features which require the model based on Tf-Idf 'char' method to take proper decisions are missing. Precision score of both methods is also rather unbalanced: 'word'-Tf-Idf (class 0 = 0.50; class 1 = 0.77); 'char'-Tf-Idf (class 0 = 0.65; class 1 = 0.79);

4.2.3 Neural Network

The third applied machine learning model is Neural Network (Hastie et al. 2001) with hidden layer sizes 150, 100 and 50, Adam solver as the optimization algorithm and ReLU as activation function. The accuracy and weighted F_1 score on the testing set of all methods are shown in the Table 1, and the confusion matrix using Tf-Idf char-Vectorizer is shown in the Figure 4.

The NN model achieves the highest accuracy of 70% and the highest macro avg F_1 score of 66 % with features of Tf-Idf 'char'-Vectorizer. As in all other models Tf-Idf 'char' outperforms Tf-Idf 'word' method by 0.2 in accuracy and 0.1 in macro avg F-score (see Table 2 for the evaluation of Tf-Idf 'word'-Vectorizer).

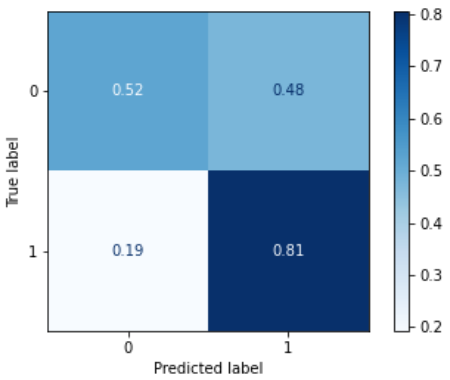

Figure 4. Confusion Matrix for Neural Network on the Testing Set using TF-IDF 'char'-Vectorizer

Just as in the RF model precision and recall of the different classes remain rather unbalanced.

4.2.4 K-Neighbors

The fourth applied machine learning model is (Hastie et al. 2001) with number of neighbors 18. In contrast to all other models, the K-Neighbors Classifier showed no difference between the 'word'-Tf-Idf and 'char'-Tf-Idf methods. The accuracy and macro avereged F_1 score on the testing set of all methods are shown in the Table 1, and the confusion matrices using both Vectorizers are shown in the Figures 5 and 6 since the results do not differ significantly.

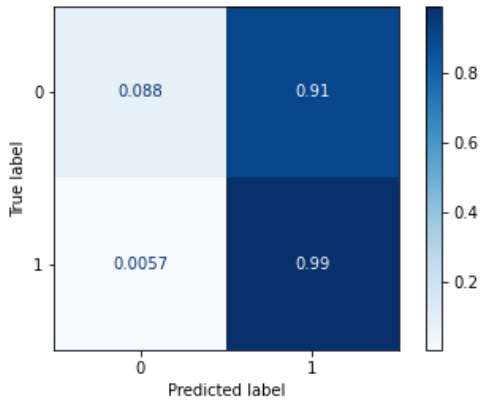

Figure 5. Confusion Matrix for K-Neighbors Classifier on the Testing Set using TF-IDF 'word'-Vectorizer

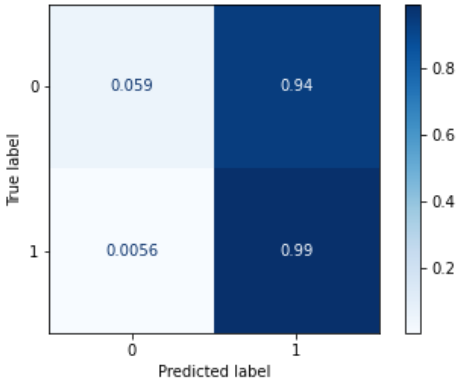

Figure 6. Confusion Matrix for K-Neighbors Classifier on the Testing Set using TF-IDF 'char'-Vectorizer

As one can see in the figures 5 and 6, the model based on the K-Neighbors Classifier showed the worst result. One of the classes, namely class 0, was completely misclassified on both the training set and the test set, which can be seen on the recall that remained below 1 (see appendix and Figures 5 and 6). When I tried more fine-tuning the hyperparameters, such as adjusting the random state parameter and the number of neighbors, the result did not improve. From this it can be concluded that the K-Neighbors Classifier is the worst suited for solving the problem of identifying offensive language on the Twitter social network.

4.2.5 Adding new Features

Exactly as Davidson et al. (2017) I also added new features to existing vectorizers, such as modified Flesch-Kincaid Grade Level and Flesch Reading Ease scores to capture the quality of each tweet. The aforementioned features were added to the char vectorizer because the models showed the best results based on it. The three best performing models were also used, namely LR, RF and NN. The accuracy and macro averaged F_1 score on the testing set with the new features are shown in Table 2. As a result, it can be said that the addition of new features has not led to improved findings.

Model (Tf-Idf 'char' with new features)	Accuracy	Macro avg F_1 Score
Linear Regression	0.7155	0.6856
Random Forest	0.7533	0.7072
Neural Network	0.7080	0.6668

Table 2. Machine Learning Model Evaluations on the Testing Set with adding of new features

4.3 Transformer Based Models

NLP industry has completely changed after Vaswani and colleagues' (2017) study, where they introduce transformer-based models. These models outperform almost all the existing state-of-the-art methods. By analogy with recurrent neural networks (RNNs), transformers are designed to process sequences such as natural language text and solve problems such as machine translation and automatic summarization. Unlike RNS, transformers do not require sequence processing. Transformers provide general-purpose architectures (BERT, RoBERTa, XLM, DistilBert, XLN and so on) for Natural Language Generation (NLG) with over 32+ pretrained models in more than 100 languages.

In the second part of experiment, two transformer-based models are used on the SemEval 2019 task, including BERT (Vaswani et al., 2017) and RoBERTa (Liu et al., 2019). For this purpose, I will use Simple Transformers library[5] because it was conceived to make Transformer models easy to use and also due to the limits of computational power of my laptop. This library allows Transformer models to be applied to sequence classification problems with just a few lines of code.

4.3.1 BERT

The first applied transformer-based model is BERT.It is a multilayer transformer (Vaswani et al., 2017) which takes as input a sequence of subtokens, obtained by using WordPiece tokenization (Wu et al., 2016), and produces a sequence of context-based embeddings of these subtokens. I adopt BERT base as the basis for all experiments of this section[6]:

- small base BERT (cased) with 12-layers, 768-hidden, 12-heads, 109M parameters, trained on cased English text.
- large base BERT with 24-layers, 1024-hidden, 16-heads, 335M parameters, trained on cased English text.

The results of the two aforementioned models, namely BERT-base-cased and large- BERT-cased, for the test dataset are presented in Table 1. The results of the uncased model are not presented there, since it did not predict any offensive language at all. This conforms to the usual expectations, since uncased BERT model was trained on lower-cased English text and upper-case words could have different meanings than lower-case ones. Both other models performed very well with the accuracy higher than 0.80 and macro averaged F_1 score higher than 0.76.

All BERT models were applied with the following hyperparameters:

• Architecture: base model pretrained on uncased (lower-case) or cased English text,

• Training batch size: 16,

• Number of training epoch: 1,

• Evaluation during training steps: 20000

I varied the maximum sequence level in the base BERT cased model to get a better performance. Parameters 32, 64, and 128 were chosen for simplicity and representativeness (a higher sequence level would have been out of memory in Google Colab). The results on the testing dataset are shown in the Table 3.

Max Sequence Level	Accuracy	Macro avg F_1 Score
32	0.8273	0.7905
64	0.8077	0.7697
128	0.8079	0.7750

Table 3. Evaluation for Different Maximum Sequence Levels with Cased Base BERT on the Testing Set

The confusion matrices for the small BERT with max seq level 32 (since it demonstrated the best result) and Large BERT are presented in Figures 7 and 8.

[5] All the details including supported model types can be found here: https://simpletransformers.ai/docs/ner-specifics/

[6] All the details of models on this and further pages were retrieved from https://huggingface.co/transformers/pretrained_models.html

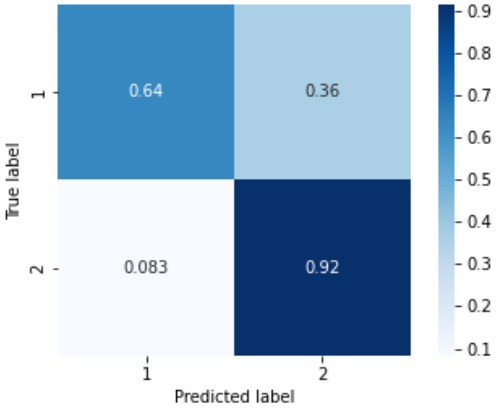

Figure 7. Confusion Matrix for Cased Small BERT with max sequence level 32 on the Testing Set;

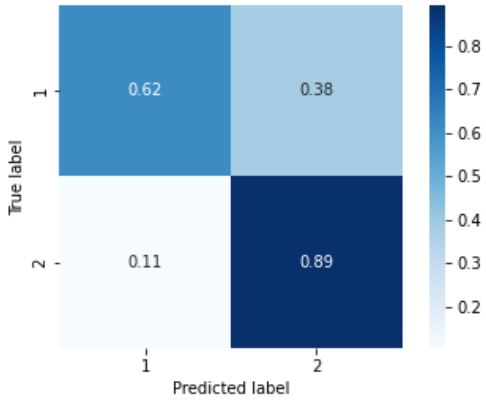

Figure 8. Confusion Matrix for Cased Large BERT on the Testing Set;

One would expect that with an increase in the maximum sequence level, the result would also improve, but in this case the model with the maximum sequence level of 32 turned out to be the best, followed by 128 and 64. The Large BERT also did not show a better result, but a result on par with the poorest result of Small BERT with max seq level 64.

4.3.2 RoBERTa

The second applied transformer-based model is RoBERTa. Liu et al. (2019) describe it as: ”robustly optimized method for pretraining natural language processing (NLP) systems that improves on Bidirectional Encoder Representations from Transformers, or BERT [...].” For this project, I use pretrained roberta-base architecture: 12-layer, 768-hidden, 12-heads, 125M parameters.

Max Sequence Level	Accuracy	Macro avg F_1 Score
32	0.7850	0.7454
64	0.7998	0.7693
128	0.8039	0.7765

Table 4. Evaluation for Different Maximum Sequence Levels with Cased RoBERTa on the Testing Set

As with base BERT cased, I experimented with the maximum sequence level, the tuning results are shown in Table 4. Compared to other BERT models (see section 4.3.1), a consistent trend could be observed with the use of RoBERTa: The best result was achieved by increasing the maximum sequence level parameter.

The accuracy and macro avereged F_1 score of the best model on the testing set are shown in the Table 1. The confusion matrix for the best model with the maximum sequence level 128 is shown in the Figure 9.

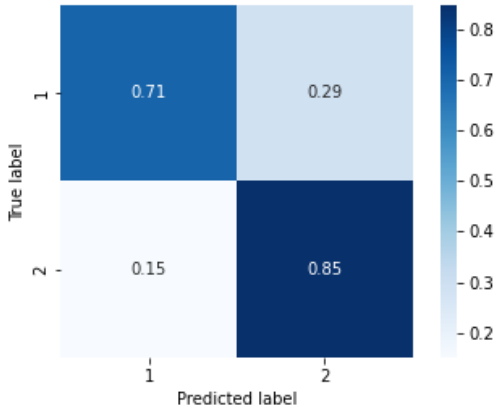

Figure 9. Confusion Matrix for Base RoBERTa with the maximum sequence level 128 on the Testing Set

The best RoBERTa model did not outperform Small BERT model and achieved the same scores as the poorest BERT model. This is quite unexpected, because RoBERTa is tuned on English texts and should have outperformed base BERT. What is also surprising is that a base RoBERTa outperforms a large BERT model with comparable running time. This suggests the RoBERTa model will be a good choice for higher prediction metrics with sufficient computational resources.

5 Conclusion

In this term paper, I tried out several machine learning approaches with different features and several types of BERT models in order to find the best solution for predicting offensive speech in tweets. The OffensEval 2019 Dataset of SemEval 2019 - Task 6 was used for this purpose.

Four machine learning models including Logistic Regression, Random Forest, Neural Network and K-Neighbours Classifier were used based on features from 'word'-Tf-Idf Vectorizer, and 'char'-Tf-Idf vectorizer. Comparisons between models and hyperparameters were done. The best model using features of 'char'-Tf-Idf Method was Random Forest, giving almost 0.75 accuracy and 0.71 F_1 score; all models within 'word'-Tf-Idf method showed rather equal results (≈ 0.68) with a maximum difference of 1.5 %.

In order to find out which feature extraction method gave the best average result, I calculated the average accuracy and the average F_1 score (see Table 4) by adding the results of all models and dividing them by the total number of models.

Method	avg. Accuracy	avg. F_1-Score
Tf-Idf 'word'	0.6830	0.6084
Tf-Idf 'char'	0.7134	0.6241

Table 4. Method Evaluations with average Scores of all Models on the Testing Set

Consistent with previous work (Waseem & Hovy, 2016), I find that using character n-grams of lengths from 3 up to 6 ('char'-Tf-Idf Method), along with POS tags as an additional feature provides the best results for Machine Leraning Models. However, it is also worth noting that the addition of new features as in Davidson et al. (2017) did not improve performance of the models (see section 4.2.5).

It cannot be said for sure that some of the results (see sections 4.2.2 - 4.2.4) show the reliability of the demonstrated models for the offensive language prediction task, since after a deeper analysis it turned out that one of the two classes of the dataset was half-misclassified almost by each ML-model. It happened likely because the dataset was unbalanced. In further research, it is necessary to provide models with a more balanced dataset for the adequate results. Thus, one will be able to exclude the possibility that one class was recognized worse, since it is less represented in the dataset.

The results show that the approach implemented using the BERT architectures, produced a model with the best results on both the training as well as the test sets. The best BERT model outperforms the best machine learning model by 7% in accuracy and 8% in macro avereged F-Score.

This project was done on Google Colab - a free cloud service that offers multiprocessors and GPUs. Machine learning model evaluation code is available on GitHub[7].

[7] https://github.com/stewieboomhauer/SNLP_OLIDv1.0

References

Basile V., C. Bosco, E. Fersini, D. Nozza, V. Patti, F. Rangel, P. Rosso, M. Sanguinetti. 2019. SemEval-2019 Task 5: Multilingual Detection of Hate Speech Against. *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*, p. 54–63. Minneapolis, Minnesota, USA, June 6–7, 2019.

CNN Tech. 2016. Twitter Launches New Tools to Fight Harassment. https://goo.gl/AbYbMv.

Davidson, T., D. Warmsley, M. Macy, I. Weber. 2017. Automated Hate Speech Detection and the Problem of Offensive Language. arXiv:1703.04009v1 [cs.CL]

Eadicicco, L. 2014. This female game developer was harassed so severely on twitter she had to leave her home. *Business Insider*, October 13 2014. Retrieved from: http://www.businessinsider.com/briannawu-harassed-twitter-2014-10?IR=T.

ElSherief, M., V. Kulkarni, D. Nguyen, W. Y. Wang, and E. Belding. 2018. Hate Lingo: A Target-based Linguistic Analysis of Hate Speech in Social Media. In *Twelfth International AAAI Conference on Web and Social Media* (ICWSM '18)

Hastie, T., R. Tibshirani, and J. Friedman. (2001). *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, USA.

Hate Speech Watch. 2014. *Hate crimes: Consequences of hate speech*. Retrieved from: http: //www.nohatespeechmovement. org/hate-speech-watch/focus/ consequences-of-hate-speech, June.

Jurafsky, D. and J. H. Martin. (2009). *Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition 2009*. Upper Saddle River, NJ: Pearson Prentice Hall.

Kettrey, H. and W. Nicole Laster. 2014. Staking territory in the world white web: An exploration of the roles of overt and color-blind racism in maintaining racial boundaries on a popular web site. *Social Currents* 1(3): 257–274.

MacAvaney S., H.-R. Yao, E. Yang, K. Russell, N. Goharian, O. Frieder. 2019. Hate speech detection: Challenges and solutions. *PLoS ONE* 14(8): 1-16. e0221152.

Moulson. G. 2016. *Zuckerberg in Germany: No place for hate speech on Facebook*. Retrieved from: http://abcnews.go.com/Technology/wireStory/zuckerberg-place-hatespeech-facebook-37217309.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12: 2825–2830.

Schmidt, A., and Wiegand, M. 2017. A Survey on Hate Speech Detection using Natural Language Processing. In SocialNLP'17: Proceedings of the 5th International Workshop on Natural Language Processing for Social Media.

Waseem, Z., and Hovy, D. 2016. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In SRW@HLT-NAACL, 88–93.

Zampieri, M., Malmasi, S., Nakov, P., Rosenthal, S., Farra, N., & Kumar, R. (2019). Semeval-2019 task 6: Identifying and categorizing offensive language in social media (offenseval). arXiv preprint arXiv:1903.08983.

Zeerak W. and D. Hovy. 2016. Hateful Symbols or Hateful People? Predictive Features for Hate Speech Detection on Twitter. *Proceedings of NAACL-HLT 2016*, p. 88–93, San Diego, California, June 12-17, 2016.

Appendices

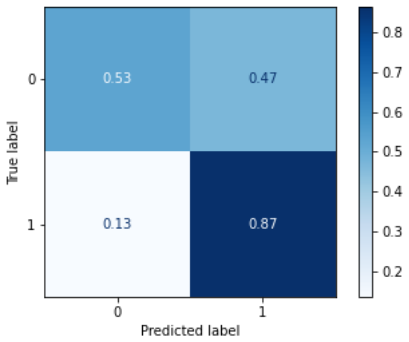

Figure 10. Confusion Matrix for Linear Regression on the Testing Set using TF-IDF 'word'-Vectorizer

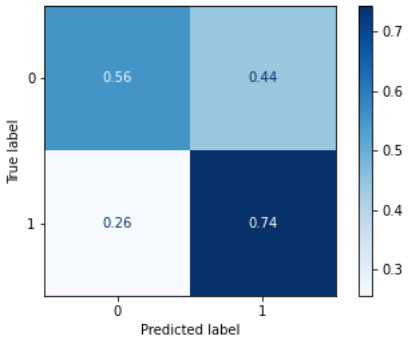

Figure 11. Confusion Matrix for Random Forest on the Testing Set using TF-IDF 'word'-Vectorizer

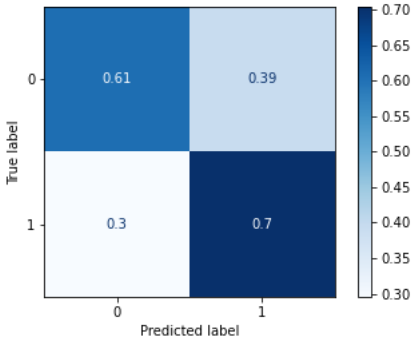

Figure 12. Confusion Matrix for Random Forest on the Testing Set using TF-IDF 'char'-Vectorizer without Stop Words

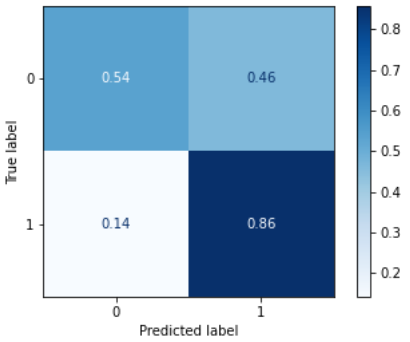

Figure 13. Confusion Matrix for Random Forest on the Testing Set using TF-IDF 'char'-Vectorizer with new features

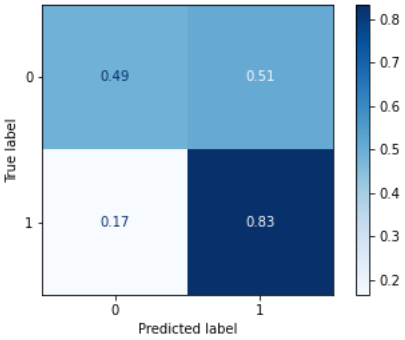

Figure 14. Confusion Matrix for Neural Network on the Testing Set using TF-IDF 'word'-Vectorizer

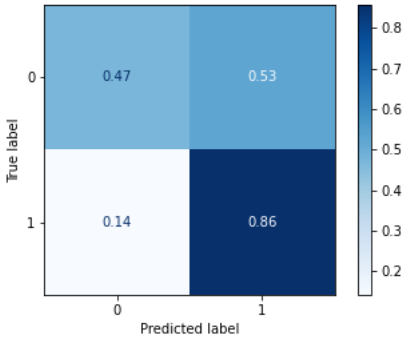

Figure 15. Confusion Matrix for Neural Network on the Testing Set using TF-IDF 'char'-Vectorizer without Stop Words

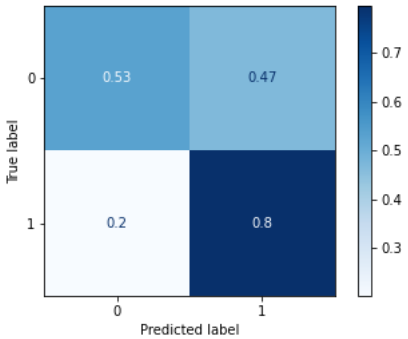

Figure 16. Confusion Matrix for Neural Network on the Testing Set using TF-IDF 'char'-Vectorizer with new features

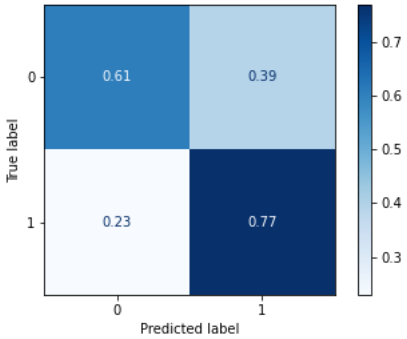

Figure 17. Confusion Matrix for Linear Regression on the Testing Set using TF-IDF 'char'-Vectorizer with new features

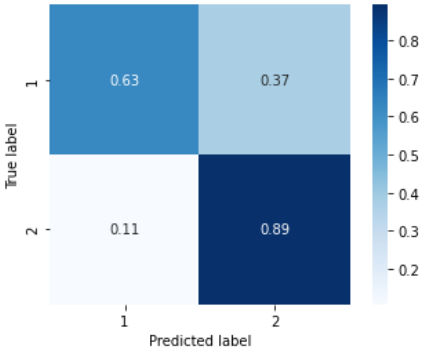

Figure 18. Confusion Matrix for Cased Small BERT with max sequence level 64 on the Testing Set;

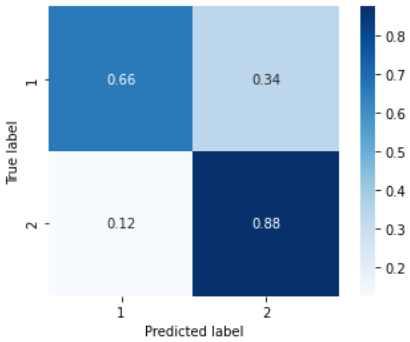

Figure 19. Confusion Matrix for Cased Small BERT with max sequence level 128 on the Testing Set;

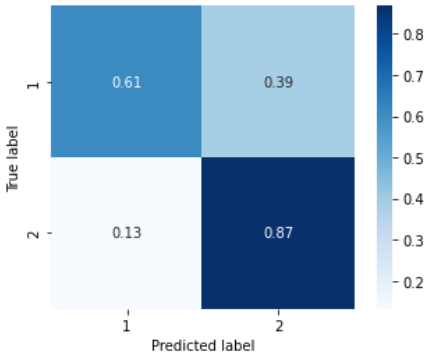

Figure 20. Confusion Matrix for Base RoBERTa with the maximum sequence level 32 on the Testing Set