Classificaiton of Social Media Comments as Hate Speech – Comparative study

Jer Pelhan[1]

[1]*Faculty of Computer Science Ljubljana*
E-mail: jp4861@student.uni-lj.si

Abstract

The presence of hate speech on the social media platforms is becoming one of the strongest society problems. The vast amount of content that is created every passing minute cannot be manually checked, thus we need to empower algorithms for classification of toxicity. In this article we test a statistical method, logistic regression as a baseline, and three deep learning models: (LSTM) long short-term memory, LSTM with convolutional layers, and bidirecitonal LSTM for classification of the comment toxicity on Toxic Comment Classification Challenge dataset. LSTM outperforms logistic regression, LSTM with convolution and Bidirectional LSTM for 8.4, 0.9 and 2.4 percent of the original score in F1, respectively. Furthermore, we present a novel pre-processing approach for classification of offensive speech. Instead of removing all punctuation akin to related works, we leave exclamation marks and words written in all caps, as we detected a correlation to the offensiveness of a comment on a selected dataset. Using the best performing model, proposed pre-processing outperforms commonly used pre-processing approaches by 0.87 percent of the original score.

1 Introduction

With the start of the twenty-first century many social media platforms arose. Social media platforms are a medium that enables the communication – registered user can generate content, e.g. post text, photos or videos. Currently more than 50% of people worldwide use social media platforms daily, that consequently means that the impact of social media on our mental well-being is immense and extensive. All-though, many natural language processing algorithms are successfully employed on different spheres, the problem of detecting hate speech is limited due to absence of public laws that classify hate speech.

Nonetheless, with all the assets, the social media platforms are a handy medium for sharing negativity and hate speech, i.e., insulting speech targeted at a specific person or marginalized community on the basis of characteristics of that person or group, for example, race, origin, disability, gender identity, sexual orientation, etc. Given that there is an average of more than 10.000 tweets made every second, all the content cannot be manually checked. In addition, some European countries already enforced laws demanding for social media platforms to remove hate speech within twenty-four hours. Hence, artificial intelligence methodology, *i.e.*, natural language processing (NLP), is required to tackle with colossal amount of possibly hateful content created every second.

The main problem of toxic speech on a social media web-page is the myriad of forms it comes in. There is also clearly a lack of the definition what hate speech is and does it result in cyberbullying, which is punishable in many European countries. In the last decade many different approaches or methods of hate speech detection developed. Such as methods for binary classification, and with their advancement multi-class approaches in the last five years. In our work, we focus on the classification of toxicity – multi-class approaches. We propose a new pre-processing technique, that outperforms existing pre-processing techniques, and test it with respect to different models.

The remainder of the articles is composed of four sections. In Section 2 we briefly describe related works on toxic comment classification. In Section 3, we present and analyze the Toxic Comment Classification Dataset that is used in this study. Methods that we employ in this comparative study are described in Section 4, where we also describe used metrics, pre-processing pipelines, used models, etc. Experimental results are reported in Section 5. In Section 6, we conclude the thesis and discuss future work.

2 Related Work

Current methods that work with hate speech detection and classification are divided into two groups: (i) deep learning models that use and extract their own features, and (ii) feature based machine learning models that need extracted thoroughly cleaned text within features as input data.

Support vector machine (SVM) is a popular machine learning method on this field [1] that has achieved successes with simplicity and performance of the model.
Greevy *et. al* [1] presented a classification method for racist text using SVM in combination with bag of words (BOW) as feature representation. BOW is a technique of representing texts in vectors of fixed length. It does not consider word order, as it is only a collection of words that appear in the text. In [2] they show, that employing BOW classifier is not sound, as it classifies comment

as toxic if it includes a word that is marked as hateful in feature space, all though it is not in the context of this comment. Nonetheless, this results in high recall, but also a high rate of false positives, as many comments get marked as hate speech, only due to containing a marked word.

Global Vectors for Word Representation (GloVe) [3] is a word embedding method published by Stanford researchers that does not explicitly model word order, but considers the words in the representational matrix based on their distance from the target word. The creators of fastText [4] from Facebook tackle with word order in text as this is important part of text classification. In this method a text is represented by a vector that in a way as with BOW, but it also uses vectors to represent word ngrams and this way representing local word order.

With development of Neural Networks, Reccurent Neural Networks (RNN), especially Long short-term memory (LSTM) [5, 6] became popular NLP in general, but also within the task of classifying hate speech in texts. Chakrabarty *et. al* [7] state that the best reported score on classification of abuse on Twitter is achieved using LSTM. In [8] they show that neural networks in combination with RNN, achieve good performances. As neural networks can in general detect interesting connections between features, that can be extracted using RNN such as LSTM.

3 Dataset

For the comparative study of methods for classification of social media comments we chose the Toxic Comments Classification Challenge (TCCC) dataset published on Kaggle by Google and Jigsaw. The dataset is composed of 159.571 human annotated comments collected from Wikipedia used for training, and 63.978 annotated comments for testing the algorithm. All the comments were labeled by the type of toxicity: (i) toxic, (ii) severe toxic, (iii) obsene, (iv) threat, (v) insults and (vi) identity hate. Single comment of the dataset can be associated with multiple labels, thus the task on this dataset is multi-label classifciation problem. Three most represented classes of comments are toxic, obscene and insults. The dataset is featured of a very unbalanced distribution with only just above 12% of comments labelled as at least one or multiple categories of toxicity.

With correlation matrix we observed correlations between classes of TCCC dataset. Interestingly, 'toxic' and 'severe toxic' classes are correlated with a small correlation factor of 0.31. We further analyzed this correlation and came to conclusion that 'severe toxic' class is a subset of the 'toxic' class and the correlation factor is small only because the 'severe toxic' is represented in minority in comparison with 'toxic' class. A strong correlation exists between 'toxic and 'obscene', but the strongest correlation is between 'obscene' and 'insult' class.

We thoroughly analyzed different features as comment length, number of punctuation, word length, number of repetitions of characters in words, as useful features can be removed due to not paying enough attention to the dataset. Both median and mean of clean comment lengths are larger for circa 100 characters in comparison with hateful comments. Intuitively, hateful comments are more likely to have higher percentage of capitalized characters. This also holds for average percentage of TCCC dataset capitalizations, but the median percentages are very similar. Upper-cased words are correlated with number of exclamation marks used in the comment. Clean comment holds less than one exclamation mark in average, but hateful comment in average holds almost three exclamation marks. We do not explore any other strong correlations with other punctuation, or other features.

4 Methods

4.1 Dataset Pre-processing

Comments in the datasets for classification of hate speech contain special words and characters, *e.g* punctuation, capitalization, new line symbols, stop words, emoticons, links and even IP addresses. All of that introduces sometimes unnecessary additional dimensionality into feature space which affects the performance of the model. Pre-processing is a set of techniques that change text data in a way of making it more feature intense without any information losses. This step enables better classification, as it strongly reduces dimensionality of text space. It is important not only which pre-processing techniques we choose, but also the order in which we apply them. In [9] they extensively test pre-processing methods and their order. They point out that it has a strong impact on the end performance of not only traditional machine learning models, e.g. linear regression, SVM, but also deep learning models.

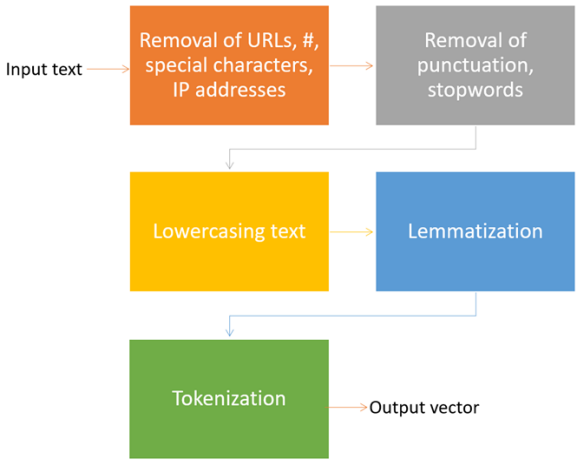

Figure 1: Dataset preprocessing. We introduce novelties in second and third step.

First step in our pre-processing of the dataset is removal of URLs, hashtag symbols and any other HTML elements as seen in Figure 1. In the dataset that we use, we also remove IP addresses as they are only adding noise to the text. In the second step we joined the removal of punctuation, and stop words. In contrary with [9], we do not remove all punctuation, but we leave the exclamation marks. Furthermore, we only strip capitalization

of words, that are not consisted of all capitalized characters. We leave the completely uppercased words in the dataset. Both described steps exploit the fact that Mikolov *et.al.* [10] employ no complex data normalization or pre-processing in the training process of fastText on large text corpora from Wikipedia and Web Crawl which we use as a word embedding. As the last step we employ lemmatization, *i.e.*, changing all the words of a text to the uninflected forms, so all different forms of a word can be analysed as a single item.

4.2 Metrics

The official metric of the challenge [11] is mean column-wise ROC/AUC, *i.e.* the mean area under a receiver operating characteristic curve by all the classes. ROC curve is a plot that presents the performance of a binary classification model at all classification thresholds. The curve is calculated from the true positive rate or sensitivity against the false positive rate or (1 - specificity). In addition, we decide to use F1 score, as F1 score penalizes models that just predict everything as a negative class. Since F1 score is harmonic mean of recall and precision, we also consider both.

Furthermore, we will use macro-averaged F1-scores, i.e., arithmetic mean of individual F-score of single class. We use macro, as we want all classes to be treated equally. Described averaging system is applied also for precision and recall.

4.3 Models

Hyperparameter tuning. We employed grid search for all the methods, to obtain as optimal hyperparatemers and consequently achieve better results. The term hyperparameter appends to all non trainable parameters of a model, that usually have big impact on the performance of the model. Grid is exhaustive search that only attempts to find the values that are optimal.

Logistic Regression As a baseline we train Logistic Regression, a statictical model that is applicable for binary classification. At the end we use liblinear as the solver function and set the inverse of regularization strength to 4 with dual formulation in combination with 1- and 2-grams. With features being extracted as term frequency–inverse document frequency (TF-IDF) that are commonly used features for text classification. We use multiple logistic regression classifiers, one for each class.

LSTM Recurrent neural networks (RNN) like Long Short-Term Memory (LSTM) can use internal memory to process considerably large sequences of inputs, thus it interprets a document as a sequence of words. Models using LSTM have a major and important novelty in comparison to traditional RNN of having ability to learn long-term dependencies between the inputs. The embedding layer transforms words to dense vectors, that are then feed to the LSTM. We use fastText [4] as embedding layer. Then we apply globalMaxPooling1D layer that down-samples representation by taking maximum value over time. After, a combination of dropout and dense layers is applied, ending with dense layer with 6 outputs, i.e., one for each category. We performed grid search for hyper-parameters and set batch size to 32, dropout parameter to 0.2, with Relu activation, binary cross-entropy for loss function for 6 epochs trained using Adam optimizer with default learning rate of 0.01.

LSTM-CNN Long Short-Term Memory with Convolutional Neural Networks is becoming popular for classification tasks as they are good at detecting specific combinations of features which RNN as LSTM can extract. After the LSTM layer we employ convolution. Hyperparameter tuning set batch size to 64, dropout parameter to 0.1, with relu activation, binary crossentropy for loss function for 8 epochs trained using Adam optimizer with standard parameters. For convolutional layer grid search found 64 as the number of convolutional filters, i.e. the third dimension of the output space with kernel size or length of the convolution window of 3 as optimal.

Bidirectional LSTM LSTM that only preserves past information, in usage of Bidirectional LSTM, the inputs are feed in two ways, one forward one backwards, preserving information not only from past but also from future. This model is composed in the same maneer as LSTM described above. The hyper-parameter tuning found the best performance of the model with batch size of 64 for 9 epochs, dropout rate of 0.1, relu activation function and sigmoid on last dense layer, binary cross-entropy as loss function and Adam optimizer with default learning rate, but with a decay of 0.0003.

Probabilistic classification. All the methods return a probability of a comment being a member of a certain class. Thus, we need to threshold the values to obtain end classifications. To find optimal threshold, we empower the precision recall curve, to find optimal threshold for both, i.e. for F1 score. We perform this for every category of the prediction, for each classifier separately.

5 Results

Table 1 summarizes the performance for all the tested methods on Toxic Comment Classification dataset. All though, on first sight we achieve low F1, recall and precision we need to take into account that we are classifying every comment into six very similar conjunctive categories on a very difficult dataset.

We decided to test the effect of pre-processing the dataset on the overall performance of the methods. We thus performed an experiment to evaluate methods with no pre-progressing, the pre-processing described in [9] and our suggested method. The results are presented in the first, second and third column of Table 1, respectively.

Linear regression is the method with the biggest performance increase is seen from no pre-processing to standard pre-processing proposed in [9]. The performance increase measured in AUC, F1, recall and precision is 3.7%, 8.7%, 15.3% and 2.45%, respectively. The increase of performance is large as linear regression is feature based machine learning model. Machine learning models in contrary to deep learning models need extracted cleaned text within feature space as input data.

Convolutional neural networks are more commonly used in image processing tasks, but we show that they can

	No pre-processing				Standard pre-processing [9]				Proposed pre-processing			
Model	AUC	F1	Re	Pr	AUC	F1	Re	Pr	AUC	F1	Re	Pr
Log. Regr	92.18	58.25	56.05	60.63	**95.62**	**63.35**	**64.63**	**62.12**	95.51	62.61	63.23	62.00
LSTM	97.22	68.30	72.77	64.36	97.39	68.66	**72.88**	64.92	**97.60**	**68.70**	72.26	**65.48**
LSTM-CNN	97.04	66.04	70.84	61.87	**97.05**	**68.08**	**72.20**	**64.41**	96.82	67.26	71.98	63.13
BiLSTM	96.32	65.18	69.34	61.50	97.19	67.05	**70.74**	63.74	**97.19**	**68.16**	70.66	**63.99**

Table 1: Comparison of AUC ROC, F1-measure, recall and precision with respect to three different pre-processing techniques. Method based on LSTM outperforms all other methods. Best results of each method with respect to pre-processing are bolded.

be outperform Bi-directional LSTM network with standard pre-processing [9] for 1.5%, 2% and 1% at F1, recall and precision, respectively.

The overall best method is based on LSTM with fastText embedding. It outperforms all other methods at all three stages of pre-processing with respect to all used measures. LSTM method performs best with proposed pre-processing outperforming the LSTM with standard processing by 0.6% and 0.9% in F1-score and precision, respectively. Recall however decreases. This interprets as classifying less clean comments as any class of toxic and classifying more toxic comments into clean category. Even though the recall drops, F1 raises, resulting in better overall performance. At the classification of hate speech it is important to have high recall and also precision, thus F1 score is most viable.

6 Conclusion

We presented a novel method for text pre-processing for the task of classification of hate speech on comments taken from Wikipedia site. The novelty of our method is removing all the punctuation except exclamation marks, as we found in the dataset analysis that they are more commonly present in toxic comments. Furthermore we detected a correlation between number of upper-cased words and end class. The presented method was tested in comparison with no pre-processing and pre-processing proposed in [9]. Based on detailed experiments, we have identified improvements of the overall performance of the methods based on LSTM and BiLSTM. The precision, mean AUC and F1 are higher with proposed pre-processing, but the recall drops, resulting in less falsely clean comments labeled as toxic but also more toxic comments labeled as clean.

For the toxic comments classification to be successful it should report as little comments that are not toxic in reality, but it should also not overlook the toxic comments. At the end it is up to different use cases, e.g. in some scenario it would be completely inappropriate to have toxic comments, thus the method should be trained to achieve high recall. But in most cases, a person has to look through all these comments that are reported to be toxic. At the end it would be most appropriate to have at leas two thresholds. The first one being a partition between clean and probably toxic and second one marking toxic comments. Then, probably toxic comments are manually checked.

In the future, we would like to investigate manual creation of features that are concatenated with the input to the model. Instead of not removing exclamation marks, and words that are written upper-cased, we could count the occurrences and add them as additional features. That could be helpful as we would still remove the noise from text, but also use important features that should not be castaway. Furthermore, we would like to investigate training two classifiers, a binary one for predicting if comment is toxic, and multi-class one for classifying the toxicity – referring to the imbalance problem of the dataset. This will be the topic of our future work.

References

1. Greevy, E. & Smeaton, A. Classifying racist texts using a support vector machine. *The 27th ACM SIGIR Conference 2004, Sheffield, UK.* (Jan. 2004).
2. Kwok, I. & Wang, Y. *Locate the Hate: Detecting Tweets against Blacks* in *AAAI* (2013).
3. Pennington, J., Socher, R. & Manning, C. D. *Glove: Global vectors for word representation* in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (2014), 1532–1543.
4. Joulin, A., Grave, E., Bojanowski, P. & Mikolov, T. *Bag of Tricks for Efficient Text Classification* 2016. arXiv: 1607.01759 [cs.CL].
5. Sigurbergsson, G. I. & Derczynski, L. 2019. arXiv: 1908.04531 [cs.CL].
6. Wulczyn, E., Thain, N. & Dixon, L. Ex Machina: Personal Attacks Seen at Scale. *CoRR* **abs/1610.08914.** arXiv: 1610.08914 (2016).
7. Chakrabarty, T., Gupta, K. & Muresan, S. Pay "Attention" to your Context when Classifying Abusive Language. *Proceedings of the Third Workshop on Abusive Language Online* (2019).
8. Van Aken, B., Risch, J., Krestel, R. & Löser, A. Challenges for Toxic Comment Classification: An In-Depth Error Analysis. *CoRR* **abs/1809.07572.** arXiv: 1809.07572 (2018).
9. Naseem, U., Razzak, I. & Eklund, P. A survey of preprocessing techniques to improve short-text quality: a case study on hate speech detection on twitter. *Multimedia Tools and Applications* **80,** 1–28 (Nov. 2021).
10. Mikolov, T., Grave, E., Bojanowski, P., Puhrsch, C. & Joulin, A. *Advances in Pre-Training Distributed Word Representations* 2017. arXiv: 1712.09405 [cs.CL].
11. *Toxic comment classification challenge* https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/.