Foundations of NLP: Hate Speech Detection

Condrat Mihai (411) mihai.condrat@s.unibuc.ro
Guiță Bianca-Oana (411) bianca.guita@s.unibuc.ro
Meteșan Raul-Petru (406) raul.metesan@s.unibuc.ro

Abstract

This paper explores some of the available solutions for hate speech detection. By this time, a large amount of effort has been made to increase the efficiency of automated hate speech detection in online scenarios. Social media became the main environment for hateful expressions and we must succeed in identifying the non-hate and hate texts. Analyzing the available hate speech datasets shows that this subject is difficult to counter because hateful expressions have a variety of possibilities in which they occur. Statistical models such as Naive Bayes, Support Vector Machine, Random Forest, do not work with high efficiency, compared to Deep Neural Network models. The high efficiency of DNN structures, especially Convolutional Neural Network models, is based on features extraction, capturing the semantics of hate speech.

1 Introduction

To succeed building an artificial intelligence structure to detect hate speech, you first need to identify and track hate speech.

According to the Committee of Ministers, hate speech covers all forms of expressions that spread, incite, promote or justify, racial hatred, sexism, homophobia, religion, ethnicity, sexual orientation, xenophobia, anti-semitism or other forms of hatred based on intolerance.

Along with the development of new forms of media, online hate speech has been brought about. Hate speech in the online space requires further reflection and action on the regulation and new ways for combating it (Hat).

However, the term of "hate speech" is defined as any communication that disparages a person or a group on the basis of some characteristics such as race, color, ethnicity, gender, sexual orientation, nationality, religion or other characteristics (Zhang and Luo, 2018).
Social Media companies such as Facebook or Twitter, put hundreds of millions of euros in the game, every year, to detect hateful expressions on their platforms.
A large amount of papers were shown in recent years to research and develop automatic methods for online hate speech detection.

However, even if the result is impressive, current methods are mostly biased towards detecting the content which is not hateful, more than detecting hateful content and their types.

In this situation, "hate speech" is also presented to be offensive, bulling or abusive language. The interest in developing tools to stop propagation of hate speech is growing in recent years, especially on social media platforms.
So, first practical step was to conduct data analysis to get the characteristics from language in social media.

2 Related work

To predict if a textual content is considered hate speech, we established cases in which we can verify and identify hateful expressions. So, hate speech aims people or groups based on their features, usually contain offensive language and it's intention is to incite and harm people.

1. "They need to go back to their ******* countries!"

   Usual case of xenophobia targeting a group based on their races.

2. "You ******* stupid!"

   Offensive language targeting individual.

3. "You cannot come back to school, i will never let you here again!"

   Typical expressions used for bullying, without using offensive language.

Those hateful words can be identified using specified classifiers or models. Some related work on this topic is the following: **Surface features such as**

1. Bag of words, character n-grams have been used
2. The use of hashtags, URL mentions,capitalisation
3. Word generalisation, topic modeling, word embeddings
4. Lexical resources for negative expressions

Among those methods, SVM, Naive Bayes and Random Forest are the most often used supervised classifiers.

The best results have been obtained using Deep Neural Networks on large datasets.
The most popular models for network architectures are CNNs and RNNs, especially LSTM.

Because of his feature extractor capabilities, CNN is the most effective in literature.

Those three metrics are often used to measure performance and effectiveness:

1. Precision measures the percentage of true positives among the set of hate speech text
2. Recall measures the percentage of true positives among the set of real hate speech messages
3. F1 calculates the harmonic mean of the two above

3 Data

The used dataset is the Offensive Language Identification Dataset (OLID v1.0) (Zampieri et al., 2019). We used only the labeled data contained in the training file, using it in the machine learning models with a 66.6% test and 33.3% train split. The data contains a list of English tweets, each of them being labeled for one of the following tasks:

- A: Offensive Language Detection
- B: Categorization of Offensive Language
- C: Offensive Language Target Identification.

We are focusing on the first task (Offensive Language Detection) since it suits better the approach that we are going to present.

The data distinguishes two different tweet types:

- Not Offensive (NOT, Labeled '0'): clean tweets, without offenses/profanity
- Offensive (OFF, Labeled '1'): tweets containing profanities, targeted offenses, insults, etc.

NOT OFFENSIVE	8840
OFFENSIVE	4400

Distribution of labels

The dataset contains the message from the tweets along with it's label. Tweets have a 280 character limit, most of them containing tags and other information that is not relevant for the given task. This means that the used texts are on the lower end of the spectrum, when talking about the length.

For our task, there are no missing values and no data augmentation was done. For the preprocessing, some basic NLP steps were taken: lowercase the text, remove unwanted characters/mentions/tags/stopwords/emoticons, expand common contractions, etc.

Initially, we lematized the data. However Camacho-Collados and Pilehvar, 2017 finds no significant impact on accuracy for text classification using neural architectures, so we can use lemmatization for classical methods and stemming for Neural Networks. To better compare the models for the same data and encoding of the data, we used the English Snowball Stemmed for our data.

4 Methods and Results

We want to compare a Convolutional Neural Network (CNN) architecture similar to the one proposed in Zhang and Luo, 2018 with classical approaches.

We used a Word2Vec encoding for our dataset. The sequence of words provided after preprocessing was passed to word embeddings with Word2Vec. For that we used two Word2Vec models. One has pre-trained vectors trained on part of Google News dataset that has about 100 billion words. The model contains vectors having

300-dimensions for 3 million words and phrases. However, we decided to create and train our own Word2Vec model having all the tweets in our dataset so if there was any word that couldn't be found in the pretrained model, we would find and process it with this one. We kept the vector size of 300 so that there wouldn't be any mismatches, the maximum distance between the current and predicted word within a sentence has the value set to 5 and the training algorithm is based on Continuous Bag of Words (CBOW). These embeddings were used for all the models that we approached.

For the classical methods, the approach is pretty standard. The processed data was used to train multiple machine learning models. We used a grid search on the available parameter space for some methods, trying to find a best fit. From our trials, these were some of the better performing ones:

Model	Accuracy
SVM	0.75
SGDC	0.72
Gradient Boosting	0.72

The best model used was a SVM classifier with radial kernel and a C factor of 5. It had an accuracy of 0.75 and a F1 score of 0.69.

For the SGD Classifier we used the convex function hinge loss, which is used for "maximum-margin" classification. For the regularization term (penalty), we chose 'l1' because it might bring sparsity to the model (feature selection) not achievable with 'l2'. We went for a number of maximum passes over the training data of 10 (number of epochs).

Regarding Gradient Boosting Classifier, we went for a total number of boosting stages to perform equal to 100. Gradient boosting is reasonably robust to overfitting so if we use a bigger number for the estimators, the performance would be better. The learning rate of 1 shrinks the contribution of each tree. In our case, for binary classification, there is a single tree included. For the loss function we opted for 'exponential', where gradient boosting recovers the algorithm named AdaBoost. The function to measure the quality of a split (criterion) is based on mean squared error.

Another models that we tried are the Decision Tree Classifier having a maximum depth of the tree of 5 and scoring 0.69 accuracy, Random Forest Classifier that after applying Grid Search we obtained a score of 0.70 accuracy with a number of 1500 estimators and 'sqrt' as the total number of features to consider when looking for the best split.

The base CNN model contains three 1D convolutional layers, each one of them having 100 filters and a stride of 1, but different kernel sizes of 2, 3 and 4. Every CNN layer is considered to extract bi-gram, tri-gram and quad-gram features. For the activation function, we used Rectified Linear Unit (ReLU). The output of each layer of convolutional neural network is down-sampled by a 1D max pooling layer, having a kernel size and a stride of 4 for further feature selection. After that, we flatten the output and pass it into a Linear layer which applies a linear transformation to the incoming data. Lastly, we apply the element-wise function, Sigmoid, which maps the entire number line into a range between 0 and 1. The CNN model is trained on 15 epochs, having the Binary Cross Entropy (BCE) loss function, Adam optimiser and a learning rate of 1e-4.

$$\begin{bmatrix} 2359 & 504 \\ 675 & 807 \end{bmatrix}$$

CNN's Confusion Matrix

Statistic	Value
Accuracy	0.7454
Precision	0.6965
Recall	0.6842
F1 score	0.6889

CNN's Statistics

The Zhang and Luo, 2018 paper used different datasets, so their results are bound to be different. Their SVM approaches had F1's of around 0.60 and their CNN's ranging from 0.68 to 0.83, depending on the used datasets.

The Zampieri et al., 2019 paper had a SVM F1 of 0.69 and CNN F1 of 0.80.

5 Conclusions and Future Work

Convolutional Neural Networks offer us a good way of classifying hate speech discourse, given a large enough dataset.

As presented by Zhang and Luo 2018, CNN's can be used in Deep Learning configurations to obtain the best results available.

So for further development, we can try to use the proposed CNN architecture in a Deep Learning approach for better results on hate speech classification. Also, we can try other datasets (such as the ones used in (Zhang and Luo, 2018)) to check the performance of this approach.

The data provided by Zampieri et al., 2019 for the second and third task is kinda sparce, but still useful enough for obtaining further results.

References

Hate speech.

José Camacho-Collados and Mohammad Taher Pilehvar. 2017. On the role of text preprocessing in neural network architectures: An evaluation study on text categorization and sentiment analysis. *CoRR*, abs/1707.01780.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of NAACL*.

Ziqi Zhang and Lei Luo. 2018. Hate speech detection: A solved problem? the challenging case of long tail on twitter.