Case-study of Offensive Language Classification from Online Tweets

Jyotsna Singh
Saarland University
s8jysing

Soumya Ranjan Sahoo
Saarland University
s8sosaho

Sourav Dutta
Saarland University
s8sodutt

{@stud.uni-saarland.de}

Abstract

Classifying online social media text is more challenging than dealing with normal corpus data due to the use of informal language and non-textual characters. Here we try to analyze different approaches of online tweet classification into offensive and non-offensive categories. We try to enhance the tokenization process to capture better sparsity in our non-textual data. We first attempt to classify tweets using a simple Naive Bayes text classifier. After this, we also try out other advanced text classification approaches on the same data with the aim to get better results. We here try to present a case-study comparing some of such methodologies by analyzing their predictions and results.

1 Introduction

Text classification is one of the classical problems in Natural Language Processing (NLP). Classifying text into specific categories is not a new challenge. Researchers have used different approaches to detect patterns in data and assign text data to different classes based on such features. The most common classification algorithm used for such tasks is the Naive Bayes classifier. In this experiment, we have tried to identify tweets from online social media which contain strong language and are possibly tagged as offensive. Gimpel et. al., 2011 have previously developed a toolkit titled *CMU Ark NLP Tagger* where they tokenize input tweets and then predict part of speech tags for each of the tokens. This approach works well for online social media text from Twitter. However, it does not take into account instances where multiple non-textual characters (*emojis*) are used together. We here try to extend their work with an improved tokenization strategy where we divide the multiple emojis used together into separate individual emoji representations so that it increases the sparsity of features and therefore improves the overall efficiency of our classifier model.

This report is divided into multiple sections for easy readability. Section 2 contains information about some related work done in the field of text classification, specifically sentiment analysis. In section 3, we discuss the online text data used, the preprocessing step, and the detailed walk through our experiment and advanced analysis. We discuss and analyze our results in section 4. We conclude the report and propose some possible future work in section 5.

2 Related work

Trying to classify text from social media is not a new challenge. Researchers have attempted to classify such online data and calculate the sentiment scores of sentences from the internet to predict public opinions before. There are literature surveys which tell us about different methods for sentiment analysis and offensive language mining (Schmidt and Wiegand, 2017).

2.1 Vader Sentiment Classifier

The NLTK[1] toolkit introduces its Vader classifier in order to classify sentiment present in a given input text (Loper and Bird, 2002). In most cases, it is able to accurately distinguish between texts that carry

[1] http://www.nltk.org/howto/sentiment.html

positive opinions from texts which convey negative meaning. It able to statistically calculate sentiment scores for positive, negative , and compound labels. However, it is not designed with the aim to identify offensive language. Therefore, it is often unable to classify strong offensive text data as desired. This is the reason why we do not use the Vader classifier from NLTK toolkit for our experiments.

2.2 SVMs are better for *Emojis*

Sequential neural networks are widely popular for tasks related to natural language. As a result, *Recurrent Neural Networks (RNNs)* have been predominantly the state-of-the-art for such NLP tasks. It has been proved that *Support Vector Machines (SVMs)* give better results than RNNs when they are trained on character n-grams as features for emoji prediction tasks (Çöltekin and Rama, 2018). Their work deals with tweets from Twitter, and so we can assume that maximum marginalization process in SVMs should also be able to classify offensive language among these tweets.

2.3 Feature engineering and lexical resources

Research work has proved that incorporating certain kinds of handcrafted features and lexical resources while training the classification model may result in a creating an overall better classifier for detecting different opinions in Twitter sentiment analysis tasks (Mohammad et al., 2013). We assume that following a similar methodology for classifying offensive tweets also may produce effective results.

2.4 FastText: Bag of tricks

Facebook Research (FAIR) has developed FastText, a simple and light-weight text classification library that is really efficient. It divides each word into a list of character n-grams (n = 3, 4, or 5) and prepares a bag of such n-gram characters as its vocabulary. It also adds $<$ and $>$ to the start and end of the words before tokenizing it so that it is able to differentiate such tokens from 3-character words present in the corpus. For example, with n = 3, the word *beautiful* is treated as -

```
<be, bea, eau, aut, uti, tif, ifu, ful, ul>
```

This enables the classifier to be trained on character level instead of just words and increases the sparsity and efficiency of the model such that it is able to tackle new unseen words in the test data set. In this way, this approach should be an efficient method to classify online offensive tweets from Twitter. Here in this experiment we have used character bigrams instead of trigrams.

3 Experiment

In this experiment, we follow similar footsteps as the research work carried out in the CMU Ark NLP Part of Speech Tagger[2] where researchers have created a toolkit for tokenizing tweets and predicting the parts of speech for each token (Gimpel et. al., 2011). We have first tokenized each of the raw tweets and separated multiple joint emojis into individual characters to increase the sparsity of our model. We have not removed stop words from the corpus. Once we have preprocessed the tweets into a usable form, we proceed to train our model based on multinomial Naive Bayes classification algorithm. Apart from this basic classifier model, we have also tried to experiment with Term Frequency - Inverse Document Frequency (*tf-idf*) methods and FastText. Here we have experimented FastText with 20 epochs, character bigrams, and a learning rate of 0.1. We try to analyze these approaches by comparing their results. Each of the steps of the experiment are discussed in detail below. The code will be made open-source[3].

3.1 Data

This experiment deals with text data from online social media. Here we use a collection of tweets from Twitter[4]. Most of the tweets are mainly based on the topics of *Presidency Election and Polls in the United States of America (USA)*, *Donald Trump*, and *Make America Great Again*, and the *#MAGA* hashtag. A large number of the tweets contain non-textual characters called *Emojis* which are nowadays used widely

[2]http://www.cs.cmu.edu/ ark/TweetNLP/

[3]https://github.com/SouravDutta91/offensive-tweet-classification

[4]https://twitter.com/

to express strong emotions and feeling for or against a particular topic. The training and the development data sets contain tweets labeled against two classes, Offensive (OFF) and Not Offensive (NOT). The Test data set contains equal number of tweets as the Dev data but unlabeled. Table 1 contains a quantitative summary of the tweets used in this experiment.

Data set	# Tweets
Train	11240
Dev	1000
Test	1000

Table 1: Number of tweets in dataset.

3.2 Preprocessing

The tweets are online text data used by people to express their opinions and feelings on topics. As expected, these kinds of data are not composed of just normal text characters. They contain various non-textual representations including *emojis* which are graphical characters that are used to express strong emotions or opinions on any topic. It is not such an easy task for machines to calculate and predict the user's sentiment when he/she uses these kind of graphical expressions in tweets. Often emojis contain lot of information that could affect the overall sentiment of a sentence. This the reason why we need to take these emojis into account when dealing with this experiment.

Tokenization: The tweets cannot be used for any mathematical calculations in their raw form. We need to perform certain preprocessing steps before we start to train our classifier with them. We apply basic tokenization on each of the tweets based on their whitespaces, that is, tabs and spaces. This includes emojis as well. However, we also separate out individual emoji expressions whenever multiple emojis are used together in any instance of tweets. This makes sure that we make each emoji character a separate token, allowing our classifier to treat each of the textual and non-textual character representations separately as features for its model, thus increasing the overall sparsity of our classifier. In this way tokenization is an important step in our experiment.

Stopwords: One important step during the preprocessing of raw text data is to remove a predefined set of common words from the input corpus. This set of widely used words are known as stopwords and they contain words which are used more widely but contribute less to the overall meaning of the corpus. These generally include words like articles, pronouns, modal verbs, negation words, and other such words which are used more in any language. We do not remove stopwords for our experiment. Although these kind of words do not carry much weight for calculations in natural language, they are important when it comes to the overall sentiment of the sentence. While stemming and lemmatization helps for opinion mining, stopwords filtering is not as straightforward. The goal of stop words is to remove unnecessary words, but if we look at available lists of stop words, the one from NLTK library for instance, we find words that potentially convey negative sentiments such as: *not*, *don't*, *hasn't* but for sentiment analysis problem we want to keep negative words. It is evident that *"It is a good game"* and *"It is not a good game"*, provide opposite sentiment. Hence one needs to either edit the stop words list to exclude words that convey negative meaning or not use stop words at all. Here in our experiment, we choose not to exclude stop words at all.

3.3 Classifier

We use the *Naive Bayes* algorithm for classifying the offensive tweets from the collection of all samples. We use the open-source implementation of multinomial naive Bayes classifier offered by the *scikit-learn* python library. The **MultinomialNB** package in scikit-learn toolkit implements the naive Bayes algorithm for multinomially distributed data, and is one of the two classic naive Bayes variants used in text classification experiments. We tried the second option (GaussianNB) but MultinomialNB worked better. The distribution is parameterized by vectors $\theta_y = (\theta_{y_1}, ..., \theta_{y_n})$ for each class y, where n is the

number of features (size of vocabulary in text classification) and θ_{y_i} is the probability $P(x_i|y)$ of the feature i appearing in a sample belonging to class y.

The parameters θ_y are estimated by a smoothed version of maximum likelihood, i.e. relative frequency counting:

$$\theta_{y_i} = \frac{N_{y_i}+\alpha}{N_y+\alpha n}$$

where $N_{y_i} = \Sigma_{x \in T} x_i$ is the number of times feature i appears in a sample of class y in the training set T, and $N_y = \Sigma_{i=1}^{n} N_{y_i}$ is the total count of all features for class y. The smoothing prior $\alpha \geq 0$ accounts for features not present in the learning samples and prevents zero probabilities in further computations. Setting $\alpha = 1$ is called *Laplace smoothing*, while $\alpha < 1$ is called *Lidstone smoothing*. In our experiment, we use a list of values for the smoothing hyperparameter $\alpha = \{0.01, 0.1, 1\}$ and later use *GridSearch* algorithm to find the most optimal combination of hyperparameters for the classifier.

4 Results and Discussion

As discussed before, we here try different approaches to classify offensive tweets from a collection of online text data. As there are two possible class labels (*Offensive* and *Not Offensive*), the chances of randomly classifying an input text as Offensive has a probability of 50%. We treat this accuracy percentage as the **Baseline** for comparing the results of our methods from the experiment. **MultinomialNB** refers to our base method where we use the multinomial naive Bayes classifier with improved tokenization. **MultinomialNB + *tf-idf*** refers to our second methods where we try to include tf-idf approach along with the first method. **FastText (minimal)** shows the accuracy scores that we obtain with the advanced method of FastText using only minimal values for its hyperparameters. Table 2 gives us a quantitative summary of the accuracy scores of the different classifiers we used in our experiment.

Method	Train Accuracy	Validation Accuracy
Baseline	50%	50%
MultinomialNB	66.50%	**67.05%**
MultinomialNB + *tf-idf*	62.30%	56.87%
FastText (minimal)	**74.01%**	55.50%

Table 2: F1 accuracy scores of offensive tweet classifiers.

First of all, we see that all the methods perform better than the baseline approach. The *tf-idf* method does not live up to the expectation. We expected FastText to give the best results out of all the approaches as it involves the application of neural networks on a character n-gram vocabulary. We see that it has achieved the highest accuracy (**74.01%**) on the training data set. However, it is unable to show similar results for the development data set and only beats the baseline by a narrow margin of 5%. Although we expected it to perform well with unseen words, it is not capable of doing so for online tweets. The most likely reason for this failure is the unpredictability of online text data and the ever-increasing random non-textual characters like emojis which are invented each and every day in social media. The FastText algorithm is expected to work much better if we use more epochs, and higher character ngrams during the training phase.

MultinomialNB gives the best accuracy score (**67.05%**) on the validation data set among all the methods here. Getting a higher accuracy score in the development data than that of the training data is an achievement. From this, we can say that our statistical algorithm of text classification performs much better than the neural network of FastText (with minimal values of hyperparameters) on unseen data. Even though we expected FastText to perform much better while dealing with unseen data, we see that the statistical maximum likelihood approach is able to classify unseen offensive tweets in a better way. Let us look at some examples of tweets which were wrongly classified.

```
@USER @USER There ain't no MAGA hats sold here but that don't stop
me from wanting to MAGA.
```

This is one of the non-offensive tweets which was classified as offensive. The *MAGA* token is generally seen with a hashtag (*#MAGA*) in the training data. However, in this case, our model failed to mark the above tweet correctly due to the inadequate occurrences of the single token *MAGA* in the data.

```
@USER But I thought Antifa were actually fascists but this shows
Rethuglicans are the actual fascists in the room.
```

The above offensive tweet was classified as non-offensive. The most likely reason for this wrong classification is due to the introduction of new words like *Antifa* and *Rethuglicans* which are custom words invented by the public to criticize and ridicule the political entities in the USA. Our model does not have enough political knowledge to identify possible sarcastic comments in the data.

```
@USER @USER Right?! I wanna know why she on the phone calling the
police anyway? what did he do"? Fuckin "Run Me Over Rhonda" lol"
```

Here is another example where the above offensive tweet was classified as non-offensive by our model. Online text data often contains words that are shortened, abbreviated or trimmed by the users in order to make them more informal and easy to use, and/or sound *"cooler"*. Statistical methods treat each single token as a separate token in the corpus, and this includes misspelled words as well. In this case, our model treats *F*ckin* and *F*cking* as two separate entities and so the probability assigned to the entity in the above tweet is much lower when seen as a feature for classifying it as an offensive tweet.

5 Conclusion

Here we have seen different methods to calculate sentiments of online text from Twitter social media. We have tried our hands on different approaches to detect offensive tweets. While analyzing a handful number of approaches for the task, we find our statistical method of simple Naive Bayes classification trained on an improved tokenization step outperforms the neural FastText method with minimal values of hyperparameters on the unseen validation data set, which was supposedly assumed to produce better results.

6 Future Work

Although we see that our suggested method outperforms the baseline as well as the minimal implementation of FastText, there were instances where our model was not able to classify tweets correctly due to several possible reasons stated above. As a continuation of this work, we can try to figure out the reasons why our model failed in certain scenarios and come up with methods to improve it. We can also look into a better preprocessing step so that FastText is able to produce much improved results on unseen data as expected. Along with this, we can also try to experiment more with multiple values of hyperparameters in FastText like its learning rate, character ngrams, number of epochs during training phase, and dimensionality. We can also try the other advanced methods of classifying online tweets mentioned in this report before.

Acknowledgements

This experiment was conducted as a part of the final project for the lecture *Statistical Natural Language Processing (SNLP)* offered in Summer 2019 in Saarland University, Germany. We would like to thank **Prof. Dr. Dietrich Klakow** for offering this lecture. We are also grateful to our tutors for guiding us whenever we needed their help.

References

Çöltekin, Ç. and Rama, T. 2018. *Tübingen-Oslo at semeval-2018 task 2: SVMs perform better than RNNs in emoji prediction*. In Proceedings of The 12th International Workshop on Semantic Evaluation, pages 34–38.

Joulin, A., Grave, E., Bojanowski, P., and Mikolov, T. 2016. *Bag of tricks for efficient text classification*. arXiv preprint arXiv:1607.01759.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith 2011. *Part-of-Speech Tagging for Twitter: Annotation, Features, and Experiments*. In Proceedings of the Annual Meeting of the Association for Computational Linguistics, companion volume, Portland, OR, June 2011. http://www.ark.cs.cmu.edu/TweetNLP/gimpel+etal.acl11.pdf

Loper, Edward and Bird, Steven. 2002. *NLTK: the natural language toolkit*. arXiv preprint cs/0205028.

Mohammad, S., Kiritchenko, S., and Zhu, X. 2013. *Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets*. In Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013), pages 321–327.

Owoputi, O., O'Connor, B., Dyer, C., Gimpel, K., Schneider, N., and Smith, N. A. 2013. *Improved part-of-speech tagging for online conversational text with word clusters*. In Proceedings of the 2013 conference of the North American chapter of the association for computational linguistics: human language technologies, pages 380–390.

Schmidt, A. and Wiegand, M. 2017. *A survey on hate speech detection using natural language processing*. In Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media, pages 1–10.