Hate Speech Detection on Social Media

Gökhan Özeloğlu [1] **Ege Çınar** [2] **Yiğit Barkın Ünal** [3]

Abstract

In this paper, we introduce our first model's results and Naive Bayes model. We used a dataset which has 24,783 tweets and it has 3 different labels [5]. Our experiments have resulted in unigram, bigram, trigram, and 4-gram. We obtained 80% accuracy at maximum with unigram model. Also, we obtained 76% accuracy at minimum with 4-gram.

1. Introduction

Hate speech is a growing problem in relation with the growing user-base of the social media platforms such as Twitter and Reddit. But first of all, what is hate speech? Hate speech is defined by Cambridge Dictionary as "public speech that expresses hate or encourages violence towards a person or group based on something such as race, religion, sex, or sexual orientation". It effects millions of people every day and just because of this, people are getting depressed and tend to be suicidal. Hate speech occurs in everywhere such as social life and social media. Hate speech can scare away quality discussions which is something most social media platforms do not want. Hate speech can also increase polarization of different groups in society and lead to inter-group violence between different racial or political groups. The only efficient way to detect hate speech is using machine learning techniques which are already used by some social media platforms to some degree.

In this project we aim to detect content which include hate speech. We have focused on Twitter because millions of tweets are shared every day. We used datasets which have multi-class labels. So far, we used one dataset which has three different classes [5]. We implemented Naive Bayes model. We trained and tested our model. We are going to explain details in the following sections. Also we discussed experimental results and stated what we are going to do in the future.

2. Related Work

There is a lot of research done in recent years in detecting and identifying types of hate speech. The articles generally concentrate on either binary classification of hate speech or multi-class classification. Dictionary based approaches fall short on detecting hate speech that is expressed in implicit terms. So corpus based approach is more suited to our project.

Shervin et. al. [1] they were applied supervised classification methods. Their data-set is Hate Speech Identification [9]. Their system uses *n-gram, word n-gram* and *word-skip grams*. They obtained 78% accuracy. Their data-set has three different classes, which are *hate, offensive*, and *no offensive content*. This detest is our data-set's older version which has 15K tweets.

Tulkens et. al. [4] used a dictionary based approach. They created their dictionary from the comments sections of the Facebook pages then they extended their dictionary by taking similar words to the ones already in their dictionary. They used word2vec to get the most similar words. After expansion they manually discarded obviously incorrect terms. The researchers did not get the accuracy they expected.

Salminen et. al. [2] created a multi-class classification which identified hate targets and theme of the hate speech. They first detect if a comments is hateful or not using a binary classifier. Then they use a multi-class classifier to detect hate targets and the theme of the hate speech. They used different models in both learning algorithms and feature categories. The classifiers are logistic regression, decision tree, random forest, adaboost and support vector machine. The features are extracted using TF, TF-IDF, semantic features, and word2vec. The average precision of the best model, SVM, was 0.90.

Xu, et. al. [3] focused on bullying and they tried to identify actors in a bullying session such as victim, bully, accuser and reporter. They got their data from twitter. The data they work on is a collection of tweets which have indication of bullying. They used part of speech tagging and named entity recognition. They detected the author of the tweets' roles using linear support vector machine with uni-gram plus bigram feature representation. The actors can have interchangeable roles so that is a challenging part of detecting the actors involved. They had a cross validation accuracy of 61 percent.

We have studied these research papers and came to the conclusion that we are going have a corpus based approach and we will implement different multi-class classifiers and incorporate feature extraction tools such as TF-IDF and word2vec.

3. The Approach

We used Naive Bayes model to make prediction as a baseline model. Naive Bayes is a simple text classification model. It simply calculates probability of each words in given text and assigns the text in one of the class. Also, Naive Bayes uses class priors. One of the most used methods is bag-of-words, or BoW shortly. It is used to extract features from the text. A bag-of-words is a representation of text which describes the occurrence of words. It is called *"bag"*, because the word order is not important in this model. We just simply calculates the each word's probabilities and make predictions.

Another important feature extracting is stop words. Stop words can be considered as meaningless words for training and prediction. For instance, *and, or, of* are meaningless words in English for prediction. These words can be used in anywhere. Extracting the stop words may satisfy less computation time and more accurate results.

Moreover, *n-gram* is another feature extraction. Basically, it counts the *n* adjacent words in given text. If *n* is 1, it just takes the words and counts the occurrences. If *n* is 2, it takes two words which are adjacent, and counts the occurrence.

We experimented with both including stop words and excluding stop words. Also, we used *n-gram* to build accurate model. In some experiments, we included stop words and in other ones, we excluded them. Also, we used *unigram, bigram, trigram, 4-gram, 5-gram* and *6-gram*. While we have used all text document for *unigram*, we could not use all text data for other models. We had problem with memory, and so, we ignored some words which has low frequency. We defined a threshold values for this ignoring operation.

TF-IDF is an acronym meaning Term Frequency — Inverse Document Frequency. It is a technique used to understand how important a word is in a document. It is widely used in NLP field. TF-IDF counts how many time a specific word is used in the document. It is reversely proportional to the number of documents in the corpus that word appears.

We, firstly, preprocessed our data and removed noisy texts from tweets. After that we removed 3 unnecessary columns in our data frame which are CrowdFlower user's votes for each tweet. Then, we applied TF-IDF into tweets and were created TF-IDF vectorizer. Also, we split data-set as training/test. 70% is for training, 30% is for test data-set. Finally, we converted to array each text document. We made all operations by using arrays, and this would satisfy more faster computations in training process. We used scikit-learn library for training and testing our model.

4. Experimental Results

In this study, we used Thomas Davidson dataset [5]. The dataset is produced from Twitter. Each tweet was coded by CrowdFlower users. At the end, each tweet was assigned to a label which are HATE SPEECH, OFFENSIVE LANGUAGE and NEITHER. Each label counts are shown in Table 1.

Class Name	Class Count
HATE	1430
OFFENSIVE	19190
NEITHER	4163
Total	24783

Table 1. Class Distribution of classes and tweets in dataset

Before we started building model, we, firstly, preprocessed our dataset to make it more clear. There were some noisy data like hashtags, user names, web site links, and so on.

We experimented Multinomial Naive Bayes model. Our dataset is a multi-label dataset which have 3 different classes [5]. So, we choose Multinomial Naive Bayes model as baseline. We produced results by using different feature extraction techniques. These are n-gram, stop words and TF-IDF. Also, we've narrowed dataset for 2-gram, 3-gram, 4-gram, 5-gram, and 6-gram to fit our data into arrays.

We had troubles with memory because RAM was full while storing text features in array. To solve this problem, we ignored words which are documentation frequencies are lower than 0.0005 for *2-gram, 3-gram, 4-gram, 5-gram*, and *6-gram*. Moreover, we excluded stop words from text documentation in some experiments. We measured our model by using accuracy metric. We also split the dataset into training and test sets. 70% for training set, and 30% for test set. There are 24,783 tweets in dataset and 17348 of them used for training, 7435 of them used for testing our model. We got similar accuracy values in all experiments. All results are shown in Table 2.

We got best accuracy in unigram with stop words. Without stop words, we got the best accuracy in unigram model. The accuracy values are very close to each other. When we increased the *n* in n-gram, accuracy is decreased. Also, generally, *unigram* accuracy values are better than *bigram, 3-gram, 4-gram, 5-gram, 6-gram*. The reason is may be narrowed data. As we stated above, we ignored the words which has document frequency lower than 0.0005. It means that we did not use all of the words for training process. If we possibly used all data, instead of narrowing text document, we may got different results.

Model Name	Stop word	Accuracy
1-gram	**Included**	**0.8078**
1-gram	**Not included**	**0.7987**
2-gram	Included	0.7861
2-gram	Not included	0.7897
3-gram	Included	0.7732
3-gram	Not included	0.7786
4-gram	Included	0.7748
4-gram	Not included	0.7674
5-gram	Included	0.7804
5-gram	Not included	0.7739
6-gram	Included	0.7736
6-gram	Not included	0.7729

Table 2. Accuracy results

5. Future Work

So far we build a simple, accurate Naive Bayes model as a baseline. We got some accurate results for our dataset. We may extend Naive Bayes model by adding more n-grams likes 7-gram or 8-gram. Nevertheless, this may not be affect positively for model's accuracy. We are going to use other three datasets to evaluate our models [6, 7, 8]. These datasets have different labels and sizes. Also, we are going to build other models like Linear Support Vector Machines, Logistic Regression, and word2vec. To not waste datasets, we are going to make cross-validation. At the end, we are going to compare each model and datasets with respect to their accuracy values.

References

[1] Shervin Malmasi and Marcos Zampieri *Detecting Hate Speech in Social Media*. arXiv:1712.06427 [cs.CL]

[2] Joni Salminen, Hind Almerekhi, Milica Milenković, Soon-gyo Jung, Jisun An, Haewoon Kwak, and Bernard J. Jansen *Anatomy of Online Hate: Developing a Taxonomy and Machine Learning Models for Identifying and Classifying Hate in Online News Media*

[3] Jun-Ming Xu, Kwang-Sung Jun, Xiaojin Zhu, and Amy Bellmore *Learning from Bullying Traces in Social Media* In HLT-NAACL, ACL (2012), 656–666.

[4] Stéphan Tulkens, Lisa Hilte, Elise Lodewyckx, Ben Verhoeven, Walter Daelemans *A Dictionary-based Approach to Racism Detection in Dutch Social Media* arXiv:1608.08738v1 [cs.CL]

[5] Dataset which we used to evaluate our model. [Online]. Avaliable: https://github.com/t-davidson/hate-speech-and-offensive-language.

[6] Dataset which we are goint to use to evaluate our model. [Online]. Avaliable: https://github.com/wvs2/data-hate/tree/master/wassen.

[7] Dataset which we are going to use to evaluate our model. [Online]. Avaliable: https://github.com/ENCASEH2020/hatespeech-twitter.

[8] Dataset which we are going to use to evaluate our model. [Online]. Avaliable: https://github.com/aitor-garcia-p/hate-speech-dataset.

[9] Hate Speech Identification. [Online]. Available: https://data.world/crowdflower/hate-speech-identification