A comparison of Machine Learning Models in Hate Speech Classification

Sandesh Timilsina
Department of Computer Science
University of Maryland- Baltimore County
stimilsina@umbc.edu

***Abstract*—One of the important aspect of social media text analysis is hate speech detection. Due to the increase in the use of social media, the form and content of hate speech has been so diverse. This makes hate speech classification an interesting problem to solve. This paper describes my work to classify hate speech using machine learning models. The dataset used in this project contained comments extracted from Stormfront, a discuss forum, annotated at sentence level and labeled as containing hate speech or not. This paper implements a baseline model and tries to improve it using hyperparameter tuning. This paper also compares the results of different classification models on hate speech classification.**

I. Introduction

Over the last decade, social media has become a platform to not only meet new people but also to express one's opinion on different subject matters. Different people have different opinions about things and they express their opinion in different ways. Some people may express their opinion very harshly while some may use better language to express their disagreement or frustration. So, there is a need to monitor the platform and prevent users from adopting inappropriate languages which might demotivate or hurt other people.

Different social media platforms allow different mode of communication but they are all exposed to abusive behavior [2]. Abusive language can be a major risk to social media as it may lead to important members leaving the platform and may even lead to some legal issues in some counties[2].The manual verification of such activities is difficult primarily due to the ever growing usage of social media[2]. One possible solution is to automate the task of detecting offensive language.

The most popular and common mode of communication in social media is text. So, in this project, I worked to classify hate speech on text data. Although there is no standard definition for hate speech, most of the works seems to agree on "any communication that disparages a target group of people based on some characteristics such as race, gender, nationality, religion etc"[1].

The rest of the paper is structured as follows: Section 2 describes the related work in the field of hate speech detection. Section 3 discusses about the dataset used, its format and size. Section 4 presents the different data preprocessing methods used. Section 5 discuss the experiments that were carried out, the experimental settings and the machine learning algorithms that were implemented. Section 6 presents the results of both the baseline experiements and the experiments involving hyperparameter tuning and Section 7 summarises the conclusion.

II. Related Work

The growing concern of social media platforms to filter out the use of inappropriate languages has attracted more researchers in hate speech analysis and research. The research work in this field are diverse and are spread out. There is no official dataset for this task, so authors and researchers usually prepare and label their own dataset. Due to this, there is very limited publicly available resources for hate speech detection.

Hatebase[1] is an online repository of structured, multilingual, user-based hate speech. Its vocabulary is classified into eight categories[1]. Some of the popular studies make use of Hatebase to build a classifier for hate speech[1]. Kaggle's Toxic Comment Classification Challenge dataset consists of 150k Wikipedia comments annotated for toxic behavior. Google and Jigsaw developed a tool called Perspective that measures the 'toxicity' of comments[1].

Similarly, there are different variations in how researchers tackle the hate speech detection problem. Some study focus on subtypes of hate speech. Others focus on the annotation of hate speech as opposed to texts containing offensive language. Others focus on the strength of the hate speech if there is the presence of hate speech.

All in all, we can conclude that the annotation of hate speech is a difficult task, mainly because of the data annotation process[3,4]. Paper [3] conducted a study on the influence of annotator knowledge of hate speech on hate speech classification. In this project, I am using the dataset which is already annotated at sentence level.

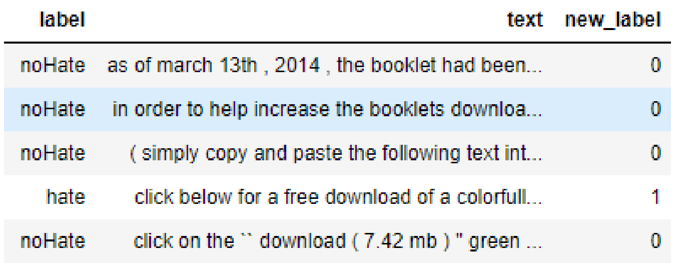
III. DataSet

The dataset used for this project is released by the paper[1] in their github repo[2]. Two datasets are merged to make a single dataset for this project. The first dataset consists annotation metadata which explains the file_id, user_id, subforum_id, num_contexts and label for the comment. The file_id column contains the file name in which the user's actual comment is stored in the second dataset, user_id contains the user's id in the discussion forum, subform_id column contains the

[1] https://www.hatebase.org/ [2] https://github.com/aitor-garcia-p/hate-speech-dataset

id of the subforum from which the comment is taken from, num_contexts column contains one of the contexts based on which the comments were scrapped from the discussion forum and the label column contains the class label on whether the comment was labled as hate speech or not. This dataset has 10944 rows.

The second dataset consists of file for each row in the first dataset and each file contained the actual comment made by the user using sentence-wise split. This means if a comment made by user contains multiple lines, then each line is kept in a separate file and each sentence of the comment has a different entry in the first dataset. The file_id column of first dataset is used to merge the two datasets. The final dataset obtained after merging the two datasets is used for this project.



label	text	new_label
noHate	as of march 13th , 2014 , the booklet had been...	0
noHate	in order to help increase the booklets downloa...	0
noHate	(simply copy and paste the following text int...	0
hate	click below for a free download of a colorfull...	1
noHate	click on the `` download (7.42 mb) " green ...	0

Fig. 1. Final dataset after merging of two datasets

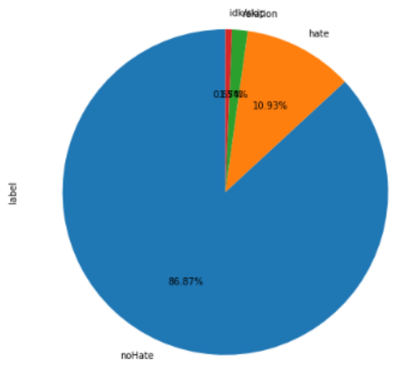


Fig. 2. Class label distribution in the dataset

A. Class Labels

The dataset consists of 4 different class labels - hate, noHate, relation, idk/skip. Comments with hate speech are labeled as hate, comments with no hate speech are categorized as noHate, comments which can only be interpreted as hate as noHate based on the previous sentences/preface are categorized as relation and ambiguous comments which can be interpreted both as hate and nohate are categorized as idk/skip. For this project, I focused on the hate and noHate label instances for convenience of classification. The distribution of each labels in the dataset is shown in the figure 2 above.

IV. Data Preprocessing

The data preprocessing methods used to prepare the dataset are as follows:

1) Filtering out relation and idk/skip labels: The comments which were labeled as *relation* and *idk/skip* were removed for the ease of classification work. Also, these comments were less in number due to which the size of the balanced dataset would be small if these labels were to be included.

2) Preparing a balanced dataset: In order to avoid any bias due to having larger number of comments for one label, I chose to balance the dataset. Further details is discussed in *Experimental Settings* of *Section 5*.

3) Removing URLs: Several comments in the dataset contained URL links to other websites,web-pages etc. Since there URLs in the text does not add much to the sentiment of the sentence, I decided to remove the URLs in the text and replace with a string '<url>'.

4) Removing Non-English characters and words : The dataset was pre-processed to remove the occurrence of any non-English characters/words. This decision was mainly influenced by my familiarity with the English language. Here, I replaced each occurrences of non-English letter/word with an empty string.

5) Removing extra white-space: Any extra white-space between the words or in the beginning or at the end of the text were removed in this step. This was done to make the text uniform and remove white-space as a character.

V. Experiments

Different experiments were carried out to understand if the annotated classes are separable solely based on the text of the labelled instances [1]. These experiments differed on the machine learning model used for classification.

A. Experimental Settings

The experiments were carried out on a balanced subset of dataset with only sentences labels as *hate* and *noHate*. All the sentences labelled as hate were collected and an equivalent number of noHate sentences were randomly sampled to collect a total of 2392 labelled sentences.

The experiments can be broadly classified into 2 categories.

*1) **Baseline Experiments***

The experiments in this category were a simple implementation of different machine learning models in hate speech classification. These experiments used default parameters for all the models and with no hyper-parameter tuning.

*2) **Advanced Experiments***

In these experiments, the baseline models were optimized using hyper-parameter tuning. Scikit-Learn's GridSearchCV package was used to combine an estimate with a grid search preamble to tune hyper-parameters. GridSearchCV picks the optimal parameter from the grid search and uses it with the estimator selected by the user. For neural network model, hyper-parameters were tuned manually to come with better accuracy.

B. Train-Test split

In all the experiments, 80% of the total sentences were used for training and the remaining 20% for testing. In the cases where validation was used, the validation set contained 50% of the sentences in the test set. 5-fold cross validation was used.

C. Evaluation Models

The evaluation models implemented are as following:

*1) **Naive Bayes(NB) Classifier**:* Word-count based vector were computed and fed into Python Scikit-Learn MultinomialNB classifier to separate instances of different labels.

*2) **Logistic Regression(LR)**:* Two different techniques were used to convert text into feature vector - *Word-count* based and *tf-idf* based. Vectors computed using word-count based approach and tf-idf approach were fed into Python Scikit-Learn Logistic Regression to differentiate different class label instances.

*3) **Support Vector Machines (SVM)**:* Word-count based vectors were computed and fed into Python Scikit-Learn SVC classifier to separate the instances of different labels. SVMs with linear, poly and rbf kernels were implemented.

*4) **Neural Network (NN)**:* The implementation is a simplified version using 2 hidden layers. The batch size used was 150, learning rate was 0.01 and the number of epochs was 0.01.

D. Performance Metrics

To measure the performance of different models, I used *classification accuracy* and *F1 score* as metrics. Classification accuracy measures how many correct predictions the model made whereas the F1 score measures the balance between precision and recall.

VI. Results

TABLE I
Results of baseline Experiments

ML Model	***Accuracy***	***F1 Score***
Naive Bayes Classifier	0.7086	0.691
Logistic Reg (countvectorizer)	0.7306	0.7351
Logistic Reg (tfidf vectorizer)	0.7369	0.7567
SVM (linear kernel)	0.7306	0.7383
SVM (poly kernel)	0.4613	0.000
SVM (rbf kernel)	0.4634	0.015
Neural Nets	0.67	0.665

TABLE II
Results of Advanced Experiements (tuning hyper-parameters)

ML Model	***Accuracy***	***F1 Score***
Logistic Reg (countvectorizer)	0.7311	0.7351
Logistic Reg (tfidf vectorizer)	0.7432	0.7648
SVM (linear kernel)	0.7202	0.7172
SVM (poly kernel)	0.6221	0.5895
SVM (rbf kernel)	0.7202	0.7219
Neural Nets	0.7245	0.7146

From table1 and table2, we can clearly observe that tuning the hyper-parameters helped to improve the classification accuracy and F1 score in most of the models. In some models, even though there was no change in classification accuracy, the F1-score increased. The best baseline model for this classification task was found to be Logistic Regression using tf-idf vectorizer with an accuracy score of 73.69% and f1-score of 75.67%. Similarly, the best model after hyper parameter tuning was found to be again Logistic Regression using tf-idf vectorizer with an accuracy score of 74.32% and 76.48%. The maximum change in accuracy and f1 score due to hyper-parameter tuning was observed in SVM using poly and rbf kernel.

VII. Conclusion

This paper is a comparative study of the baseline model and the improvement that can be brought by hyper-parameter tuning for different machine learning models. The models were implemented on balanced subset of the dataset prepared by merging two datasets. To make the classification task easier and more accurate, only the comments labeled as hate and noHate were used. The comparison of the results from table 1 and table 2 shows that the models can be optimized by tuning hyper-parameter tuning and classification accuracy and f1 score combined is a good performance metrics.

References

[1] O. Gibert, N. Perez,A. Garcia-Pablos and M.Cuadros, “Hate Speech Dataset from a White Supremacy Forum,” Proceedings of the 2nd Workshop on Abusive Language Online. Brussels, vol. ALW2, pp. 11–20, October 2018.

[2] N. Cecillon, V. Labatut, R Dufour and G. Linares, “Abusive Language Detection in Online Conversations by Combining Content and Graph Based Features” Front Big Data. Avignon, France, June 2019.

[3] Z. Waseem and D. Hovy, “Hateful Symbols or Hateful People? Predictive Features for Hate Speech Detection on Twitter” NAACL Student Research Workshop, San Diego, California, pp. 88-93, June 2016.

[4] A. Schmidt and M. Wiegand, “A Survey on Hate Speech Detection using Natural Language Processing”, Valencia, Spain, W17-1101, pp. 1-10, April 2017

[5] S. Malmasi and M. Zampieri, “Challenges in Discriminating Profanity from Hate Speech”, cs.CL, March 2018.