Fall 2018 ITCS 4111/5111

Introduction to Natural Language Processing

Final project Report

Hate Speech Detection

Lavina A Sabhnani

Venkataramana Hegde

Introduction

In United States, hate speech except for obscenity, fighting words, incitement is protected by the First Amendment of the U. S. Constitution assuring the right to free speech, and the internet exercises this right in many formats like blogs, social interactive sites like Facebook, Twitter, Yahoo. Hate speech can be defined as abusive speech targeting specific group characteristics, such as ethnic origin, religion, gender, or sexual orientation which disparages a person or a group. The anonymity and the amount of flexibility by the Internet have made it easy for users to communicate in an aggressive manner often. Due to the increasing of online hate speech, methods that automatically detect the hate speech are very much required.

In this project we have implemented a model which detects or identifies the hate speech in online text as words submissions are filtered for a fixed list of offensive words, an automatic classifier currently existing isn't publicly available. Hate speech detectors are used by most of interacting social media site and applications like Facebook, Instagram, Twitter. Terms of services prohibiting use of hate speech are stated by these sites like for example; Facebook terms are "Content that is unlawful, harmful, threatening, abusive, harassing, tortuous, defamatory, vulgar, obscene, libelous, invasive of other's privacy, hateful, or racially, ethnically or otherwise objectionable."

The paper describes the model we've created by implementing Logistic regression along with Naïve Bayes function as classifier to implement features like NGrams, TFIDF and Word Embeddings on Kaggel toxic comment classification dataset for Hate Speech Detection. The project is implemented in four steps: Loading the data, Cleaning the data, Feature Implementation, Build and Train Model, and lastly Validating the model. We compare our model with Naïve Bayes classifier for the features implemented to compare the models for better accuracy and also tested the model on a different Twitter dataset.

Background

Most of the prior work in the area of Hate speech detection has spread across many fields. It might include different work carried on detecting the abusive language in several areas which include social media, blogs, news and information sites, and video streaming applications. These texts make the task difficult since most of these are generated by user either by commenting or posting the articles, text samples etc. which contain derived words, irregular sentence structures. One of the earliest efforts to find the abusive language was **'Detection of harassment on web 2.0'** [1] which used n-gram, patterns of manually developed regular expression, contextual features to understand the context by looking into previous sentences along with supervised classification technique. The model partitions the sentences into subsamples thus affecting the results, whereas in our model we have considered whole sentence as one thus increasing the accuracy of results.

One more work which got extensive research attraction was **'Blocking blog spam with language model disagreement'** [2] which was more specifically to find the spam on social media and blog comment. This work used the idea that spam comments are off topic texts, so they have different language model compared to normal comments. However, it is important to note that most of the methods used in spam detection can't be applied directly into hate speech detection, but the making use of language model would help in hate speech detection in many of the use cases.

In **'A Survey on Hate Speech Detection using Natural Language Processing'** [3] uses the Natural Language Processing approaches such as Simple Surface n-gram features, Word Generalization, Lexical Resources, and Linguistic Features to collectively detect the abusive speech and the hate speech. The model fails to detect sarcastic comments and is not able to identify hate sentiment in them.

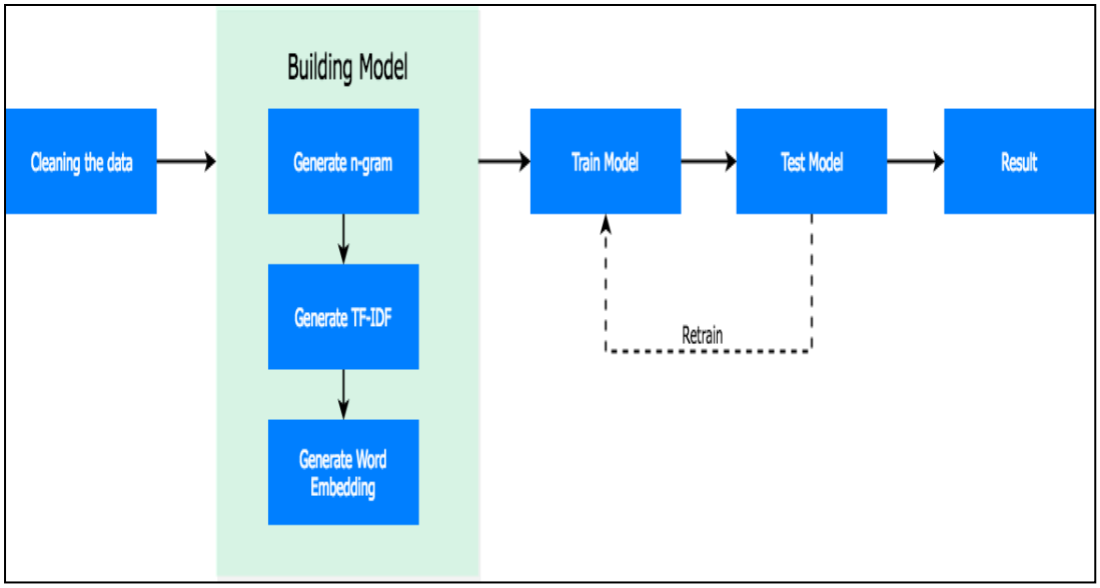
In **'Baselines and Bigrams: Simple, Good Sentiment and Topic Classification'** [4] by Sida Wang and Christopher D. Manning shows the use of Naive Bayes (NB) and Support Vector Machine (SVM) models as baselines for other methods in text categorization and sentiment analysis research. We've

implemented our model based on this research but instead of SVM we've used Logistic Regression with Naïve Bayes (NB-LogReg) feature as baseline resulting in similar accuracy as that of the model defined in the Baseline and Bigrams paper.

'Semantic Structure and Interpretability of Word Embeddings' [5] proposes a framework that enables quantitative assessments on the intrinsic semantic structure and interpretability of word embeddings based on calculation of category weights using a Bhattacharya distance metric. The implementation of the feature word embeddings in our project is based on our understanding of the learning's from the Semantic Structure and Interpretability of Word Embeddings.

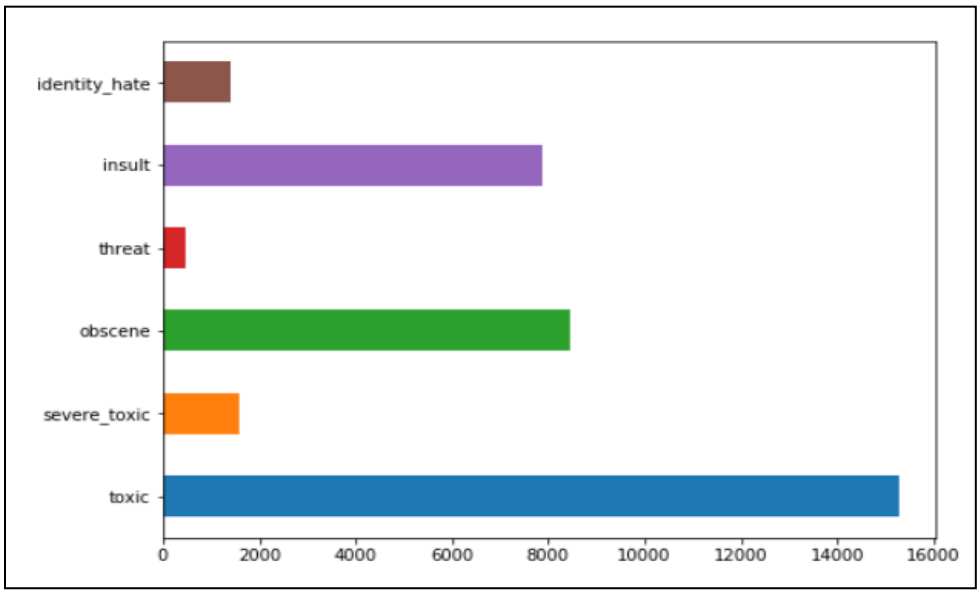
Project

In this project we have implemented a model which detects or identify the hate speech in text under six labels of that of the dataset- Toxic, Sever Toxic, Obscene, Threat, Insult, and Identity Hate. We train the model with training dataset and their corresponding features- NGram, TFIDF, and Word Embeddings. Character level **n-gram** features provide a way to attenuate the spelling variation problem often faced when working with user generated comment text. Local Feature is the most basic feature where each distinct term as one features and calculates a **Term Frequency/Inverse Document Frequency (TFIDF)** value for each feature. Term frequency provides a measure of how important a particular term is in a given comment or text. **Word Embedding vectors** is another technique that we have used to quantify and categorize semantic similarities between the hate words based on its distribution in our data set. We have used Naive Bayes (NB) with Logistic Regression as baselines for building our models. Naive Bayes-Logistic Regression (NB-LogReg) will be the classifier. Our dataset is a Kaggel toxic comment classification dataset with train dataset with 159571 comments and test dataset with 153164 comments. The train.csv constitutes of following attributes: id, comment_text, toxic, sever_toxic, obscene, threat, insult, and identity_hate. The test.csv comprises of the same attributes as that of the train dataset. We are using the following packages: numpy, pandas, nltk, sklearn, matplotlib, scipy, gensim. The following diagram describes the project implementation cycle:

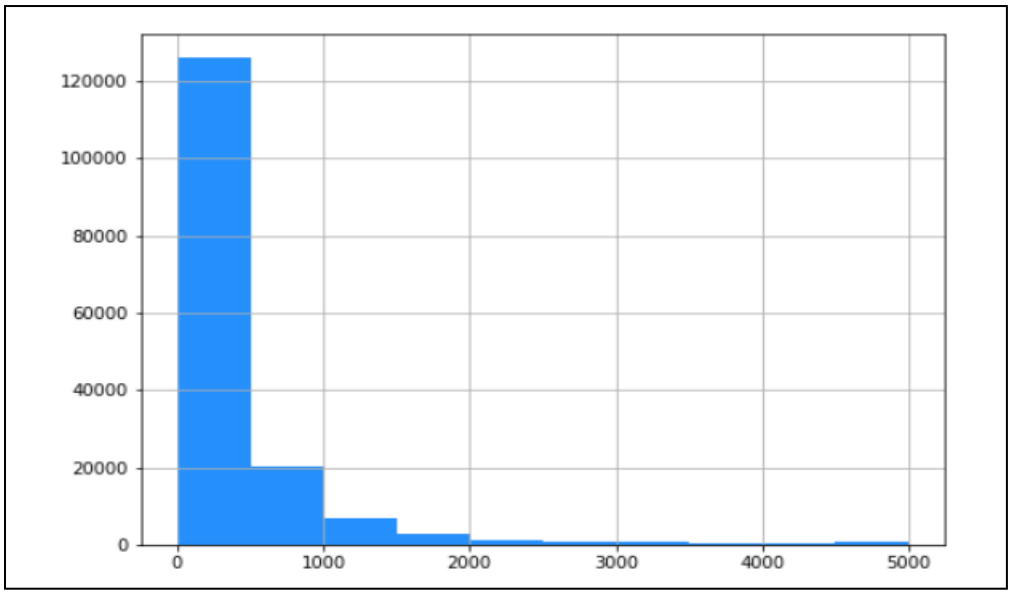


Step1: Loading the train data and test data from csv files:

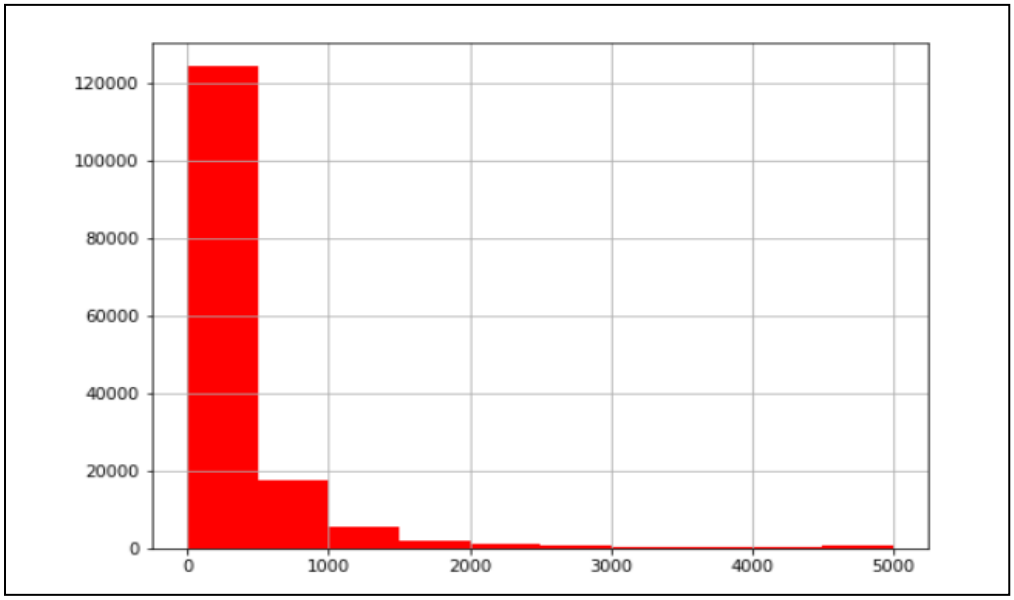
Class PrepareData is made to set up the data for training and testing by providing instance and static methods. The data is then split into train data as 70% and test data as 30%. Other labels than the 6 mentioned are removed while classifying all the comments under the label they categorize as the training dataset is multilabeled. Comments are categorized as: 15294 toxic comments, 1595 severe toxic comments, 8449 obscene comments, 478 threat comments, 7887 insult comments, and 1405 identity hate comments.



In train dataset, total number of characters is 62882658, longest comment length is 5000, and shortest comment length is 6. The following graph represents the number of characters for each comment lying between the scales of longest and shortest comment.



In the test dataset, total number of characters is 55885733, longest comment length is 5000, and shortest comment length is 1. The following graph represents the number of characters for each comment lying between the scales of longest and shortest comment.



Step2: Cleaning the Train and Test data:

Utility class is used to provide instance and static method to clean the train and test data. The null data is filled used fill_null method. We removed special characters like for example; can't will be written as cannot, what's is written as what is, 're is written as are, 'll is written as will, and many more. Stop words are also filtered using nltk stop word method.

Step3: Feature Implementation, Build and Train Model:

We have used sklearn's Logistic Regression method along with naïve bayes function to build our classifier. Word Vectorization class provides method to perform vectorization of words. It generates TFIDF method for the class which provides NGram value as parameter. The **TFIDF function** is given as:

```
word_vec = TfidfVectorizer ( sublinear_tf=True, strip_accents='unicode',
                              analyzer='word', token_pattern=r'\w{1,}',
                              stop_words='english',
                              ngram_range=(1, ngram_value),
                              max_features=10000
                             )
```

Above class provides method to get the term documents for words. Also, NGram feature is implemented before TFIDF vectorization. We can dynamically specify the NGram feature to its method. CharVectorizer class provides methods to perform character level NGram and TFIDF. char_vector method generates the TFIDF vector for the data set. The **TFIDF function** is given as:

```
char_vec = TfidfVectorizer( sublinear_tf=True, strip_accents='unicode',
                            analyzer='char', stop_words='english',
                            ngram_range=(4, 6), max_features=50000
                           )
```

The above class is to implement the character level NGram and then generate the term frequency vector. This helps to find out the hidden toxic characters or group of characters which are usually found within a word.

We combine both the word features and character level features for both train and test data. The classifier inherits from BaseEstimator and ClassifierMixin sklearn's base package. The classifier Logistic Regression with Naive Bayes features is defined with the following arguments:

C : Inverse of regularization strength; must be a positive float.
Like in support vector machines, smaller values specify stronger regularization.
default: 1.0
solver : Algorithm to use in the optimization problem.
default: 'sag'
Preferred for multiclass problem - 'newton-cg', 'sag', 'saga' or 'lbfgs'
n_jobs : Number of CPU cores used when parallelizing over classes if multi_class='ovr'".
default: -1 (using all processors)
max_iter : Maximum number of iterations taken for the solvers to converge.
default: 4000
dual: Dual or primal formulation. Dual formulation is only implemented for l2 penalty with liblinear solver.
Prefer dual=False when n_samples > n_features.
default: False

For **Word Embeddings**, since our dataset is multilabeled, we have used one-vs-all strategy is implemented using OnevsRestClassifier; for each classifier, the class is fitted against all the other classes. As for embeddings, these classifiers cannot work on the texts, so convert the texts into numerical equivalents using Word2Vec method.

Fit the model according to the given training data.

Args:
X : {array-like, sparse matrix}, shape (n_samples, n_features)

Training vector, where n_samples is the number of samples and n_features is the number of features.

y : array-like, shape (n_samples,) or (n_samples, n_targets)

Target vector relative to X.

sample_weight : array-like, shape (n_samples,) optional

Array of weights that are assigned to individual samples.

If not provided, then each sample is given unit weight.

Method to transform the word embeddings to sentence embedding

Args:

X : {array-like, sparse matrix}, shape (n_samples, n_features)

Training vector, where n_samples is the number of samples and n_features is the number of features.

Step4: Validating the Model:

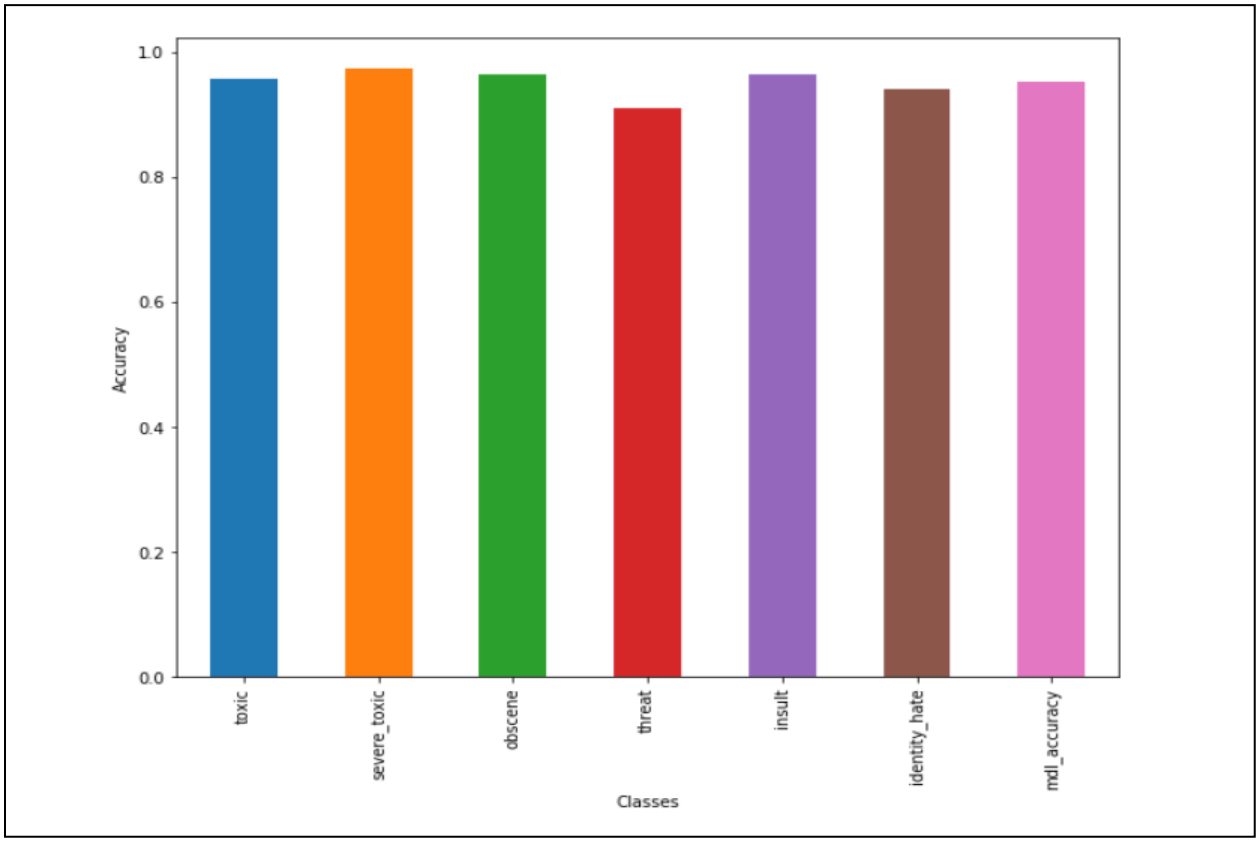
We validate the model we created using Cross Validation score method to check its accuracy as it's the most efficient because it optimizes the model parameters to make the model fit the training dataset as well as possible. The final result is compared with model that comprises of only Naïve Bayes as its classifier.

The CV score function can be given as:

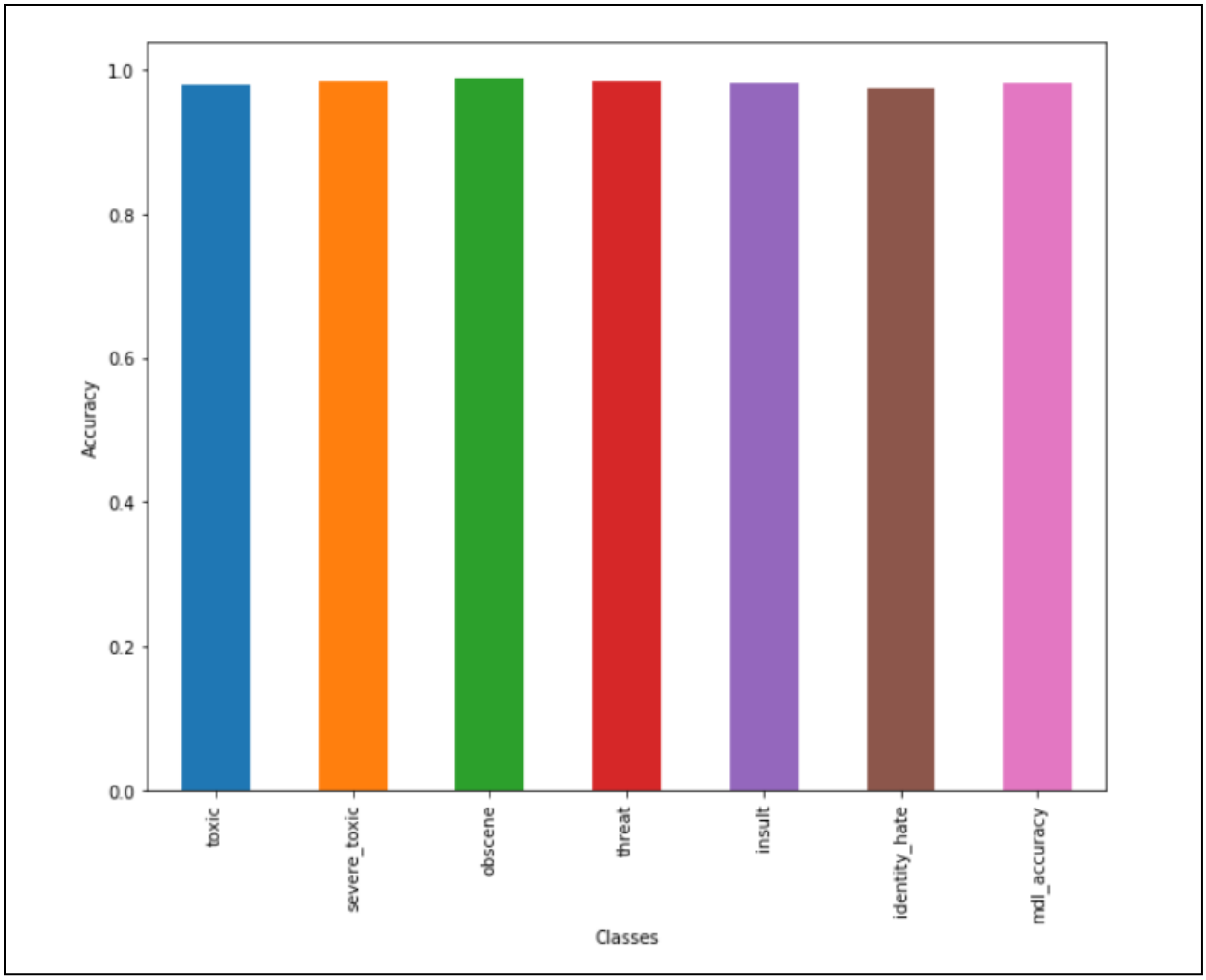
```
cv_score = np.mean( cross_val_score
                                ( nb, train_features,
                                 train_target, cv=3,
                                  scoring='roc_auc'
                                )
                            )
```

RESULTS:

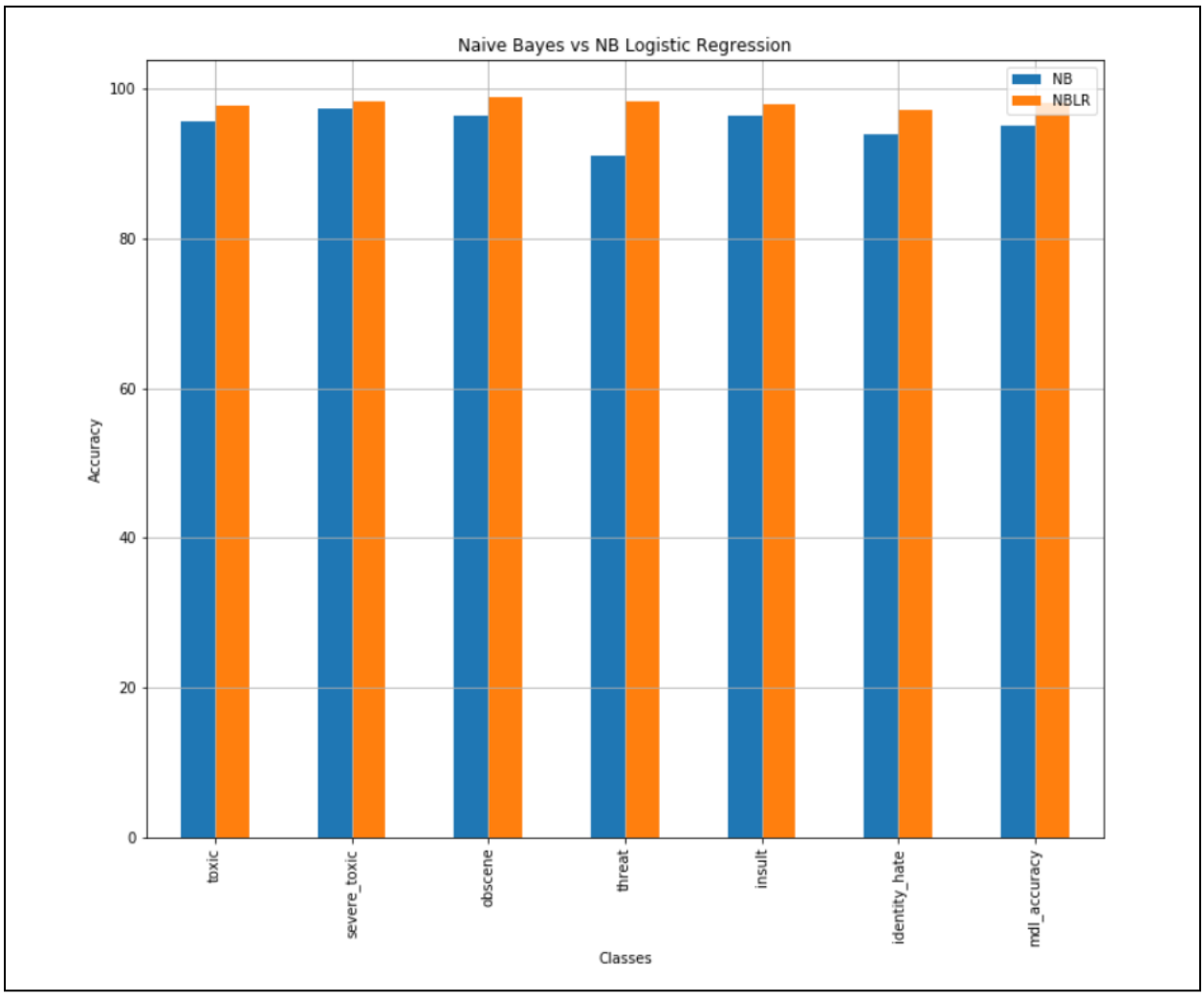
For TFIDF with NGram given as parameter the CV score using Naive Bayes classifier obtained is 95.150%



For TFIDF with NGram as parameter, the model that we built- Logistic Regression with Naïve Bayes (NB-LogReg), the CV score obtained is 98.13%.


1.0
0.8
0.6
0.4
0.2
0.0
Accuracy
toxic
severe_toxic
obscene
threat
insult
identity_hate
mdl_accuracy
Classes

Comparing both the models CV scores:



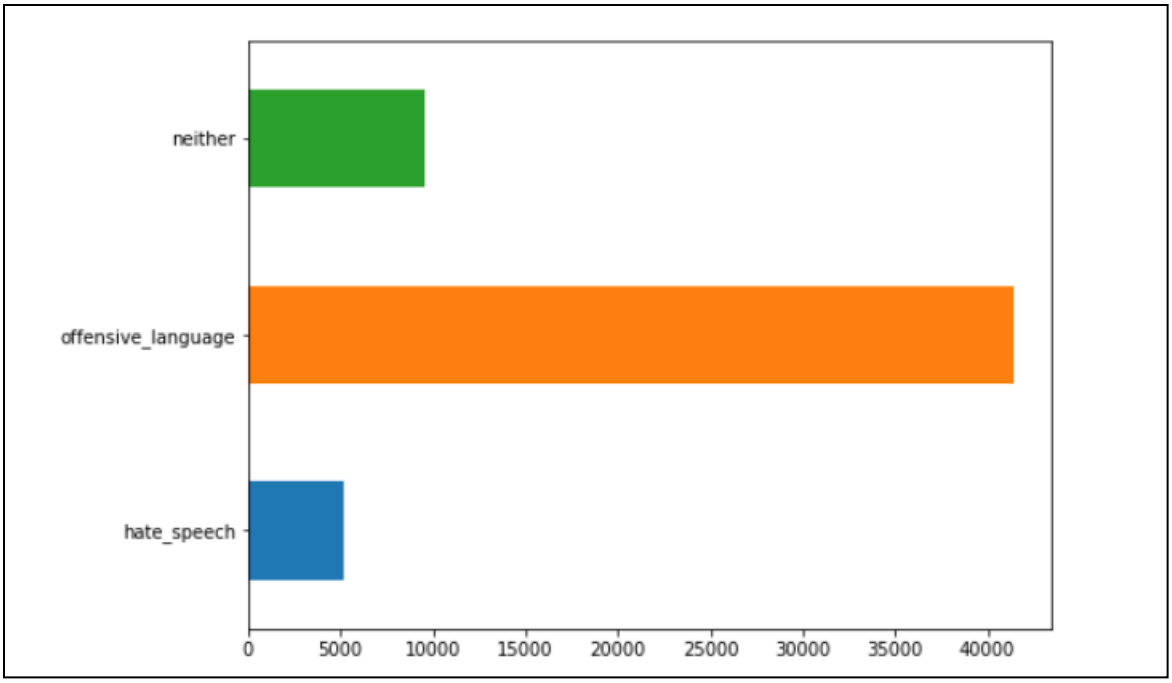
The implementation of Support Vector Machine (SVM) as classifier along with Naïve Bayes function as we were trying to implement earlier in the project didn't work as the model needed way longer to run along with efficiently normalized dataset was creating problems. Thus we switched to Logistic Regression with Naïve Bayes function as the accuracy obtained by both the models is very similar.

The contributions made by us are given concisely as follows:

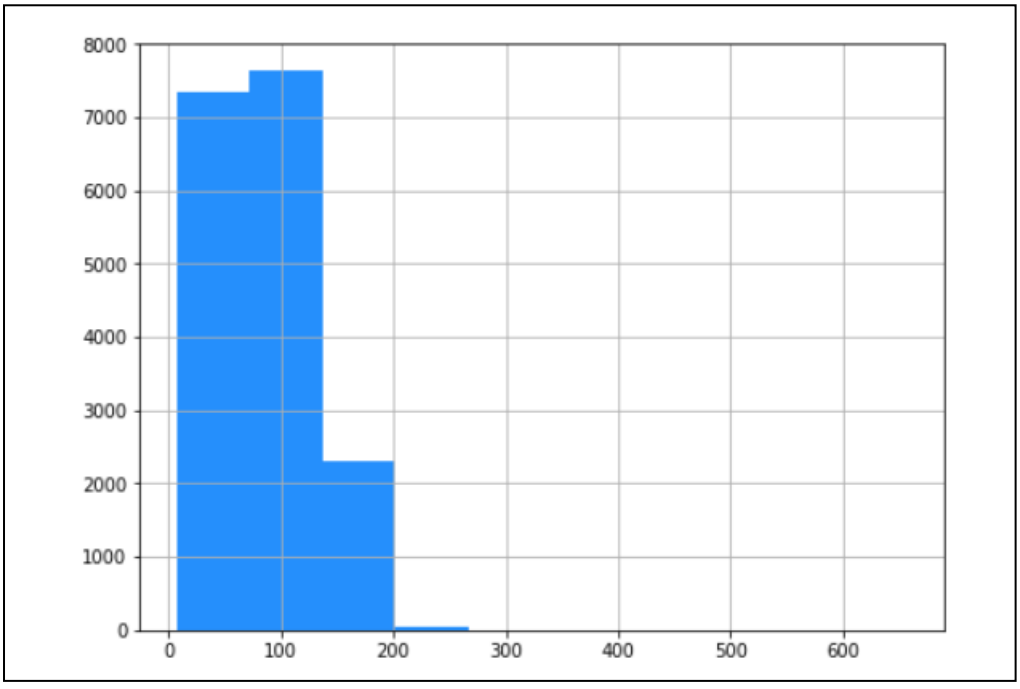
Team-Member ID	**Team-Member Name**	**Responsibilities Completed**
801036525	Lavina Sabhnani	Preprocessing the data, generation of NGrams, Tuning of models, Validation of Model, Testing on other dataset, Comparing the results, Mid-Progress Report and PPT, Final Project Report and PPT.
801053064	Venkataramana Hegde	Project Proposal report, Cleaning the data, generation of TF-IDF, Word Embeddings, Training the dataset, Testing the dataset, Comparing the results generated, Final Project Report and PPT.

Testing on Different Twitter Dataset:

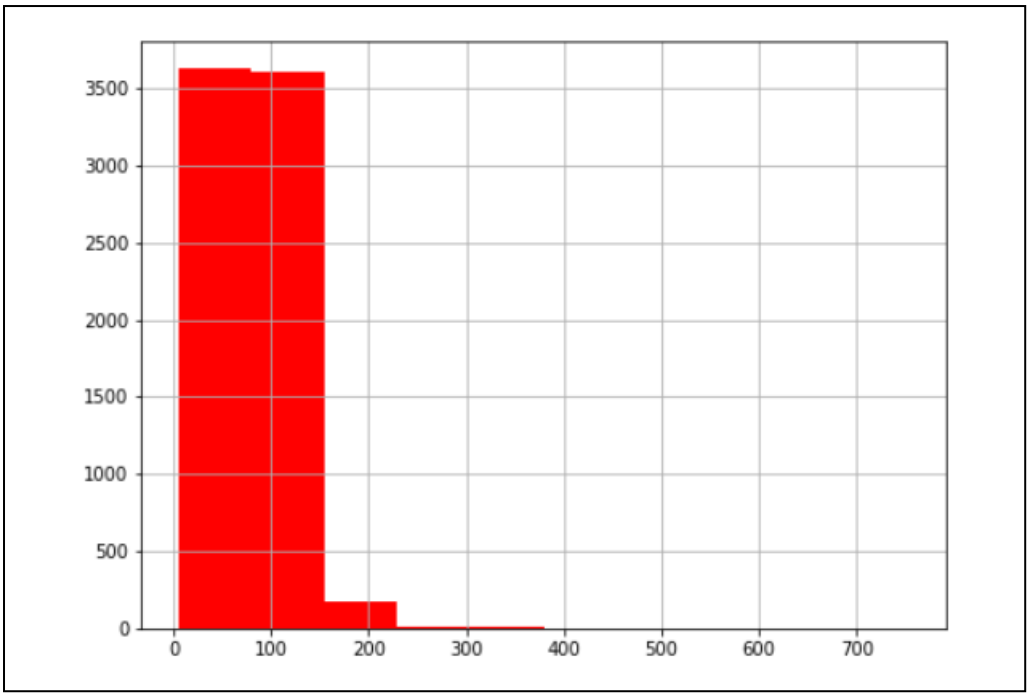
The Twitter Train dataset comprises of 17348 comments and Test dataset comprises of 7434 comments. The train.csv and test.csv constitutes of following attributes: id, comment_text, hate speech, offensive language and neither.



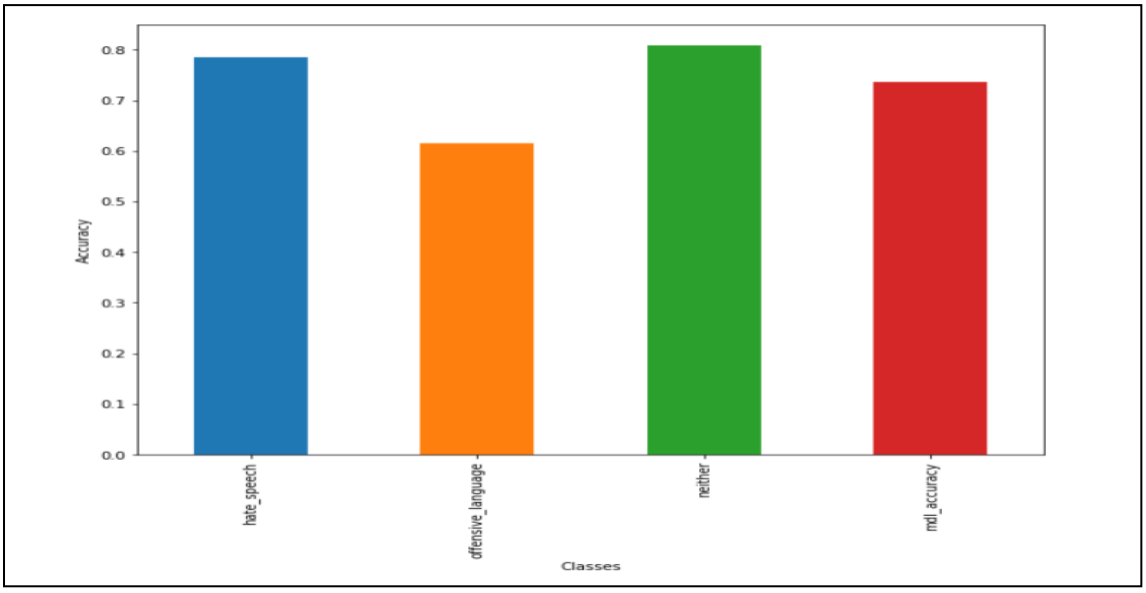
In train dataset, total number of characters is 1479276, longest comment length is 657, and shortest comment length is 7. The following graph represents the number of characters for each comment lying between the scales of longest and shortest comment.



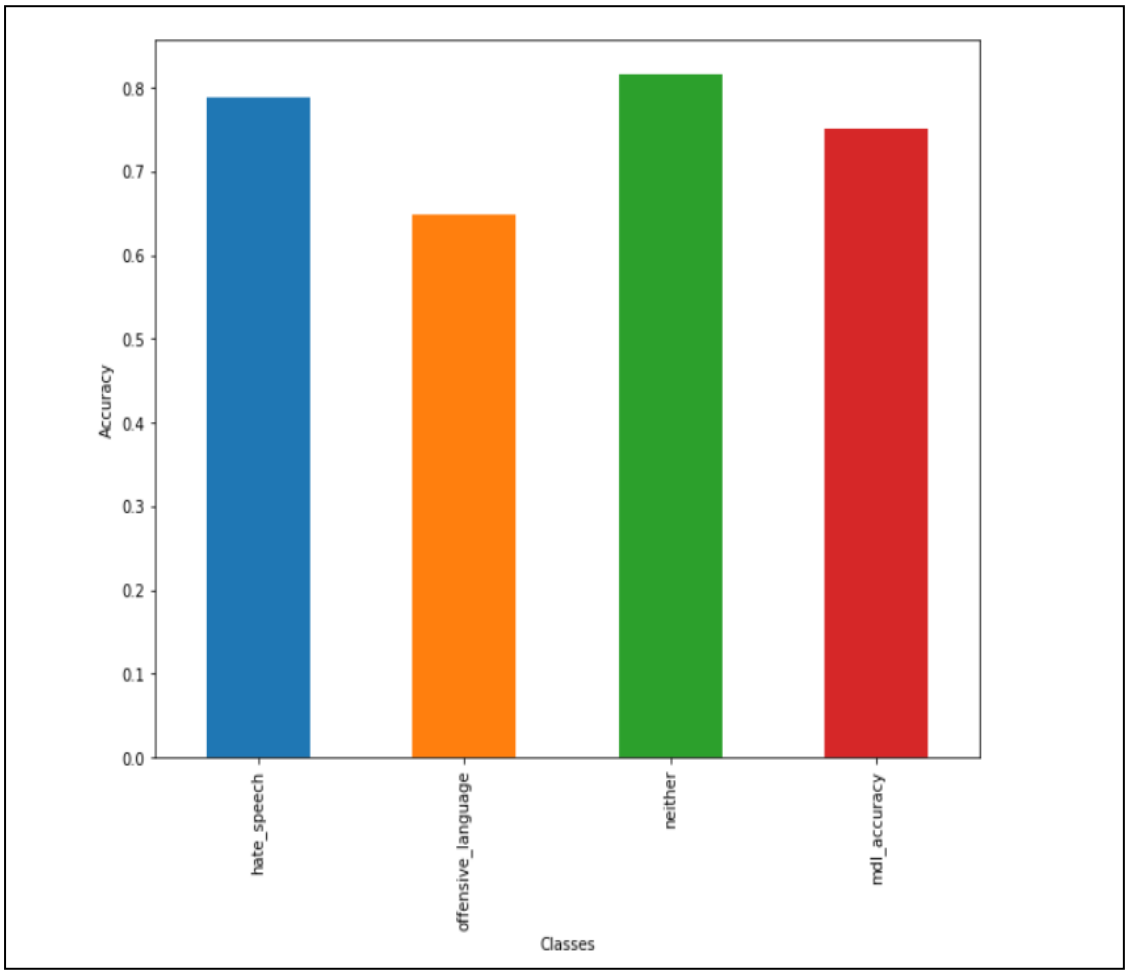
In test dataset, total number of characters is 637959, longest comment length is 754, and shortest comment length is 5. The following graph represents the number of characters for each comment lying between the scales of longest and shortest comment.



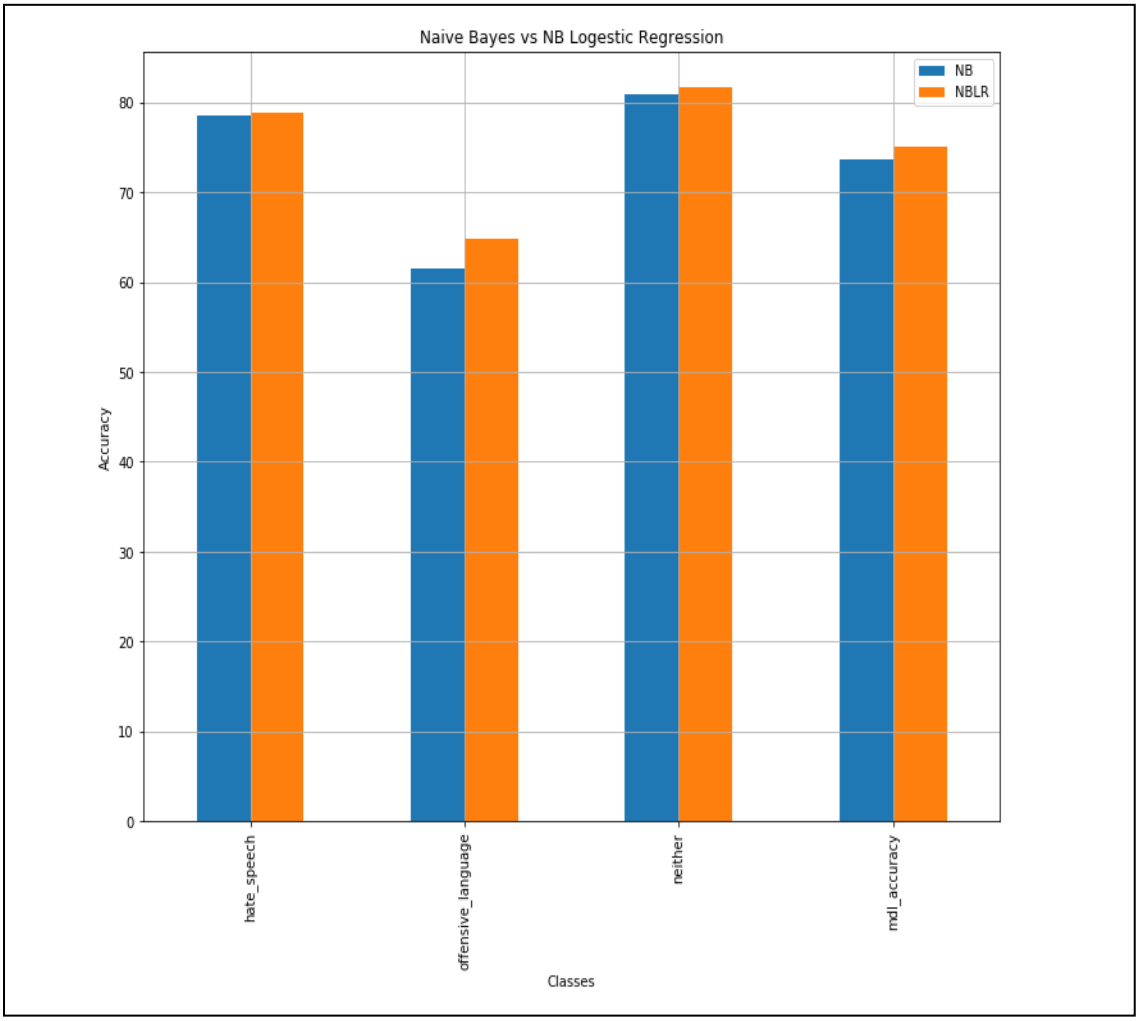
Upon validating the dataset with Naïve Bayes model we get the accuracy of 73.60%



When the model is validated with our classifier: Logistic regression with naïve bayes function we get an accuracy of 75.11%



Comparing both the models CV scores:



Summary

Hate Speech has propagated on social media increasing significantly in these recent years due to ambiguity as well as mobility of such platforms. Despite significant efforts from law enforcement departments, legislative bodies along with millions of investments from social media companies, automated semantic analysis of the content is the technique that have been relied on. Hate Speech Detection is the main driving force behind classification of hate speech depending on its targeting characteristics.

We reviewed different research works on hate speech detection implementing various models to detect hate speech like Detection of harassment on web 2.0 which evaluated harassment using supervised learning, Semantic Structure and Interpretability of Word Embeddings evaluated hate speech by quantifying the interpretability of word embeddings.

In this paper we proposed a model comprising Logistic Regression with Naïve Bayes function to obtain better accuracy for detection of hate speech than regular Naïve Bayes classifier. We used Kaggle toxic comment classification dataset for Hate Speech Detection and trained this dataset around the model we built after loading, cleaning and implementing feature like NGrams, TFIDF and Word Embeddings. We found the accuracy increase by almost 3% than that of normal Naïve Bayes classifier.

Conclusion

Throughout the project we learnt various facts like depending on the structure of the datasets, the length of comments, noise present in it, impacts the performance of the model by significant levels. Increasing the NGram level gives a refined result but it isn't true in all cases, for some datasets when trigram is implemented, accuracy may drop. Also, when we only implemented Word level NGram, we found a drop difference of accuracy of the model around 2-3% than when we implemented Character level NGram along with Word level NGram. Different classifiers and their combinations create different models for datasets, and selection of feature extractions affects its accuracy and performance for hate speech detection.

Implementation of the model with Support Vector Machine (SVM) along with Naïve Bayes function is the one aspect we could have performed differently along with normalization of dataset for better performance and increased accuracy.

Future work includes creation of model that operates beyond specific domains for datasets so that it can be tested with re-training the model on the basis of the new dataset. Building of an interactive application based on the model such that it'll take online comment text and returns one class indicating the type of label it categorizes under like toxic or insult.

References

[1] D. Yin, Z. Xue, L. Hong, B. D. Davison, A. Kontostathis, and L. Edwards. Detection of harassment on web 2.0. Proceedings of the Content Analysis in the WEB, 2:1–7, 2009.

[2] G. Mishne, D. Carmel, and R. Lempel. Blocking blog spam with language model disagreement. In Proceedings of theFirst International Workshop on Adversarial Information Retrieval on the Web (AIRWeb), May 2005.

[3] Anna Schmidt, Michael Wiegand. A Survey on Hate Speech Detection using Natural Language Processing.

[4] Sida Wang and Christopher D. Manning. Baselines and Bigrams: Simple, Good Sentiment and Topic Classification .

[5] Lutfi Kerem Senel, Ihsan Utlu, Veysel Yucesoy, Aykut Koc, and Tolga Cukur. Semantic Structure and Interpretability of Word Embeddings

[6] https://www.noswearing.com/dictionary

[7] William Warner, Julia Hirschberg. Detecting Hate Speech on the World Wide Web.

[8] Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, Yi Chang. Abusive Language Detection in Online User Content