Toxic Comments Classifier

Ayan Gautam
200101022
g.ayan@iitg.ac.in

Naman Anand
200101070
a.naman@iitg.ac.in

Lokesh Nahar
200101060
n.lokesh@iitg.ac.in

Yash Garg
200101113
g.yash@iitg.ac.in

1 ABSTRACT

In this paper, we aim to build a toxic comment classification project capable of detecting various toxic comments in social media that are abusive and insulting. We will try to do analysis on various models and embeddings and try to find the best model. Also we will try to see impact of oversampling/ undersampling on our model.

2 INTRODUCTION

- **Problem Definition** - The main idea of the model is to identify the toxicity in online conversations where toxicity is defined as anything rude, disrespectful or otherwise likely to make someone leave a discussion. Our main task is build a binary classifier model that recognizes toxicity.

- **Motivation** -
  - We got the idea of toxic comments classification by surfing through the internet we got this site https://neptune.ai/blog/10-nlp-projects and out of the given topics we liked this the most.
  - The impact of toxic comments affects the engagement of users in conversations. Many people stop expressing themselves and give up in presence of a threat of abuse and harassment online.
  - Also due to toxic comments it is not advisable for children to use or becoming active online else they would also learn from them and may continue in future conversations. So if there is a model that would predict as accurate then it would be very helpful to use that models in social media account or as an extension that would remove those comments.
  - Identify toxic comments can lead communities to facilitate and improve online conversations and exchange of opinions. Cyberbullying is one of the major societal problem.

- **Direction of work** -
  - Cleaning of text and preprocessing of test.
  - Splitting into training and testing dataset
  - Tokenization and stemming to transform into a list of tokens.
  - Different embeddings like tf-idf etc .
  - Now we will use diff test models:
    - Logistic Regression
    - Naïve Bayes
  - Algorithm Selection and Hyperparameter tuning
  - Performance of the model

3 Methods

We had uses tf-idf and word2vec embedding in our model with various types of model such as RandomForest, NaiveBayes , LinearSVC and LogisticRegression. We had done various types of samplings on dataset (around 5 types) and then we tried to find the highest accuracy (for around 5*4*2 = 40 combinations) .After finding highest accuracy , we tried to optimize our model by hyperparameter tuning on our model.

4 Experiment and Dataset

For this experiment there were various Datasets available and we choose from Kaggle. We founded one of the in the Kaggle **Jigsaw Unintended Bias in Toxicity Classification** challenge and other one while surfing through **Kaggle Twitter hate speech Dataset** . The link are as follows :

- Training and Testing Dataset
  https://www.kaggle.com/competitions/jigsaw-toxic-comment-classification-challenge/data

- Testing Dataset
  https://www.kaggle.com/datasets/vkrahul/twitter-hate-speech?select=train$_E6oV3lV.csv$

5 Analysis Of Datasets

- We are using the train set provided by Kaggle and will proceed with the usual splitting into train and test sets. The dataset consists of **143346 non-toxic comments and 16225 toxic comments**.Our other Dataset that is just used for testing is around of 29350 comments.

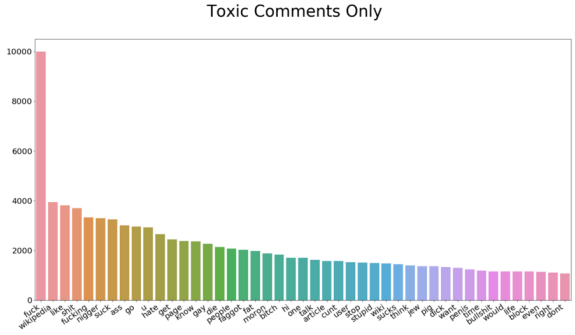


Figure 1: Toxic Words

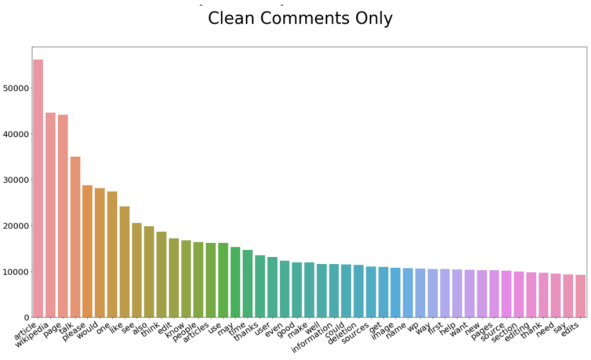


Figure 2: Non-Toxic Words

- The original toxicity of both the datasets labels have been assigned in a manual way by humans, according to the description of the dataset available on Kaggle.

6 Feature Engineering

- **CLEANING THE COMMENTS AND STEMMING**
  - We had removed the stop-words because stop-words are not toxic words so it will not have much impact on toxicity but it will help us to train and work with our model more efficiently.
  - We used Snowball Stemmer as it is the best Stemmer for datasets according to various sources. This algorithm is also known as the Porter2 stemming algorithm. It is almost universally accepted as better than the Porter stemmer, even being acknowledged as such by the individual who created the Porter Stemmer.

- **VECTORIZATION**
  We had used both TF-IDF and Word2Vec vectorization and we had compared between them. We finded out word2vec is having far more accuracy for all the models.

- **BIASING IN DATASETS**
  We can see our data is biased as it contains a large no of non toxic comments

```
Counter({0: 143346, 1: 16225})
```

Figure 3: Count

So to reduce variances of key statistics of a target sub- population, we can used a no of sampling techniques :

- RandomOverSampling
- RandomUnderSampling
- SMOTE
- ENN
- SMOTEENN

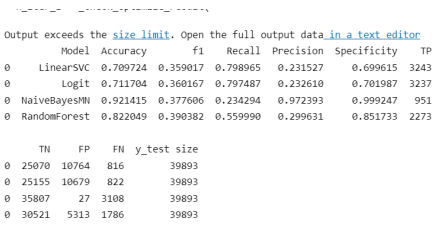
7 Training and Analysis On Training

We trained various combinations and tried to find the best combination and also tried to find conclusions for various results.

- **Avoid sampling on test Data**
  We initially did sampling on test data and got a much better model but later on we understood sampling on test data is not good as it will not help in finding the real accuracy.
  As our future data can also comes with imbalance(after deploying), so hence we will do only oversample/undersample in the train set only. If we balance

the Validation set, our model may work well(may get better score in Val) but in the future after deploying, it may not work better so while training, we should validate with imbalance data only.

- **TF-IDF embedding**
  For TF-IDF embedding we are getting best Results in **NaiveBayes Model using ENN undersampling**



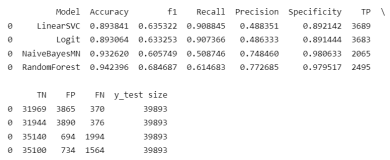
Output exceeds the size limit. Open the full output data in a text editor

	Model	Accuracy	f1	Recall	Precision	Specificity	TP	\
0	LinearSVC	0.709724	0.359017	0.798965	0.231527	0.699615	3243	
0	Logit	0.711704	0.360167	0.797487	0.232610	0.701987	3237	
0	NaiveBayesMN	0.921415	0.377606	0.234294	0.972393	0.999247	951	
0	RandomForest	0.822049	0.390382	0.559990	0.299631	0.851733	2273	

	TN	FP	FN	y_test size
0	25070	10764	816	39893
0	25155	10679	822	39893
0	35807	27	3108	39893
0	30521	5313	1786	39893

Figure 4: Result

- **Word2Vec embedding**
  For Word2Vec embedding we are getting best Results in **RandomForest Model using both ENN undersampling**



	Model	Accuracy	f1	Recall	Precision	Specificity	TP	\
0	LinearSVC	0.893841	0.635322	0.908845	0.488351	0.892142	3689	
0	Logit	0.893064	0.633253	0.907366	0.486333	0.891444	3683	
0	NaiveBayesMN	0.932620	0.605749	0.508746	0.748460	0.980633	2065	
0	RandomForest	0.942396	0.684687	0.614683	0.772685	0.979517	2495	

	TN	FP	FN	y_test size
0	31969	3865	370	39893
0	31944	3890	376	39893
0	35140	694	1994	39893
0	35100	734	1564	39893

Figure 5: Result

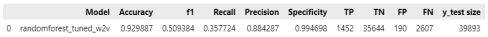
- **ENN works better But... Again a problem**
  This method will clean the database by removing samples close to the decision boundary. Undersampling should mostly not be preferred because it causes a **huge amount of data loss**. In the end, we are giving so much effort to collect data and it basically does not make sense when we throw them away. The issue is here is that we lose samples where our model could learn new things.

- **Doc2Vec ... A very Heavy model** Doc2Vec models can be very large.Our system was not supporting that model so we finally decided it to remove from our project.

8 RESULTS FOR VARIOUS MODELS AND HYPERPARAMETER TUNING

– RandomForest

After Applying HyperParameter tuning on RandomForest, we finded the accuracy of our model when we used ENN undersampling and word2vec embedding as it gived the best result while comparision in combinations. Result :



	Model	Accuracy	f1	Recall	Precision	Specificity	TP	TN	FP	FN	y_test size
0	randomforest_tuned_w2v	0.929887	0.509384	0.357724	0.884287	0.994698	1452	35644	190	2607	39893

Figure 6: Result

– LogisticRegression

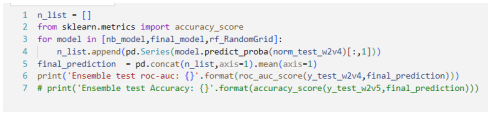
After Applying HyperParameter tuning on Logistic Regression, we finded the accuracy of our model when we used Random oversampling and word2vec embedding as it gived the best result while comparision in combinations. Result :

	Model	Accuracy	f1	Recall	Precision	Specificity	TP	TN	FP	FN	y_test size
0	logit_tuned_w2v	0.917128	0.684722	0.884454	0.558581	0.920829	3590	32997	2837	469	39893

Figure 7: Result

– Finding Optimal Threshold For Binary Classification

After Ensemble Learning Result :



```
n_list = []
from sklearn.metrics import accuracy_score
for model in [nb_model,final_model,rf_RandomGrid]:
    n_list.append(pd.Series(model.predict_proba(norm_test_w2v4)[:,1]))
final_prediction  = pd.concat(n_list,axis=1).mean(axis=1)
print('Ensemble test roc-auc: {}'.format(roc_auc_score(y_test_w2v4,final_prediction)))
# print('Ensemble test Accuracy: {}'.format(accuracy_score(y_test_w2v5,final_prediction)))
```

```
Ensemble test roc-auc: 0.9619029277964721
```

Figure 8: Result

9 Predictions on Other Datasets

We got an accuracy of around 0.76 on other dataset whom we haven't trained and just used to predict is our model biased towards only a few data and maybe it can we said it is little biased towards a particular data.

```
dataset2 = pd.read_csv("train_E6oV3lV.csv")
Counter(dataset2["label"])
dataset2["tweet"] = apply_regex(dataset2["tweet"])
dataset2["tokenized"] = dataset2["tweet"].apply(tokenize)
dataset2['stemmed'] = dataset2['tokenized'].apply(applyStemming)
other_test = dataset2["stemmed"]
ans_test = dataset2["label"]
other_test_w2v = embedding_feats(other_test, wordVec)

```

```
p_1 = rf_RandomGrid.predict(other_test_w2v)
make_classification_score(ans_test,p_1,"randomforest_tuned_w2v")

```

	Model	Accuracy	f1	Recall	Precision	Specificity	TP	TN	FP	FN	y_test size
0	randomforest_tuned_w2v	0.756805	0.136621	0.274309	0.090963	0.793203	615	23574	6146	1627	31962

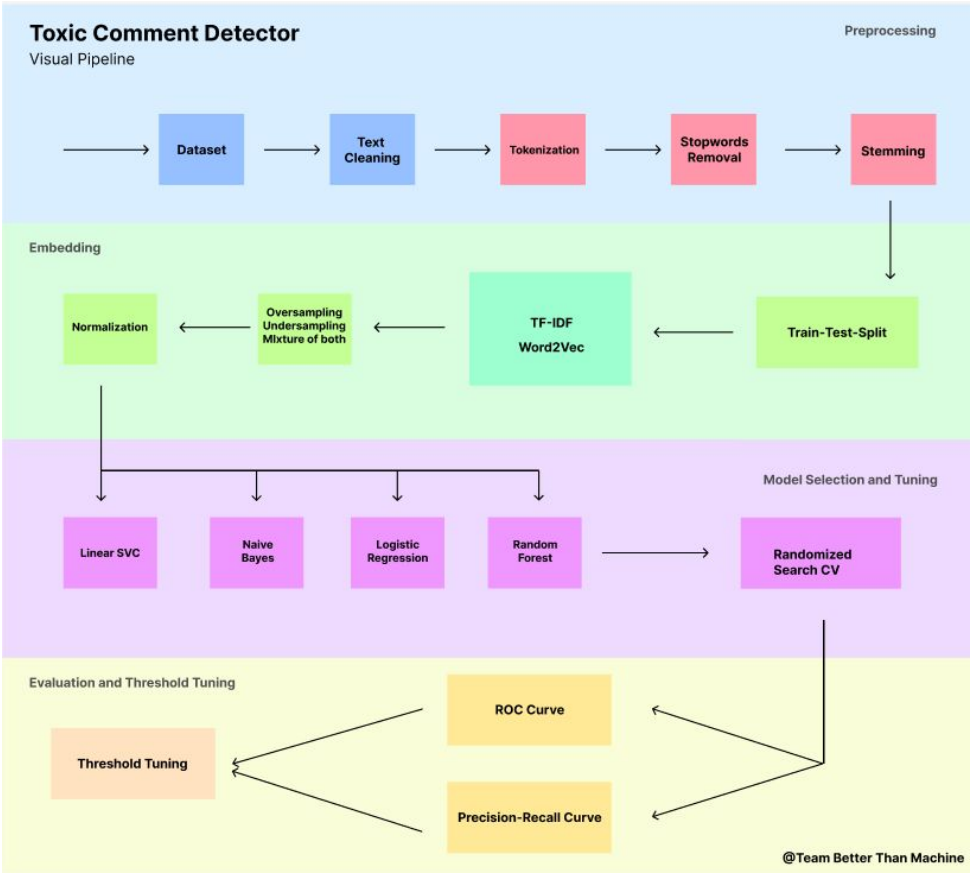
Figure 9: Result

Our new dataset contains only comments of hatespeech so we can say our model is not much trained on this type of category toxicity.

10 References

- Reference 1 : Classification of Toxic Comments on Social Media using Machine Learning Techniques
- Reference 2 : Toxic Comment Classification by Pallam Ravi, Hari Narayana Batta, Greeshma S, Shaik Yaseen
- Reference 3 : Predicting the Toxicity of Comments Using Text Classification by Sanket Barhate

11 METHODOLOGY REPRESENTATION



12 Various Graphs and curves

Various Graphs and curves of result can be finded in following link : Link for Code

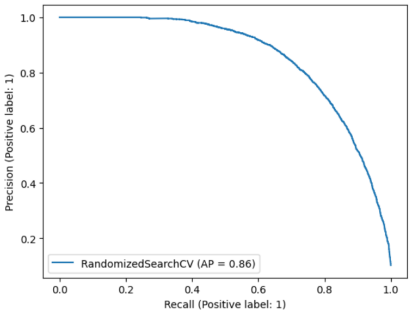


Figure 10: Result

13 Future Work

- Currently we worked only with English Datasets but we will try to extend to Multilingual Datasets like Hinglish to fit it in the Indian Context.
- Also we are planning to build a browser extension that could filter out the toxic comments based on the levels set in the filter