Categorizing Inappropriate Texts

Aditi Gupta
aditi19292@iiitd.ac.in

Mansi Singhal
mansi19370@iiitd.ac.in

Nimisha Gupta
nimisha19315@iiitd.ac.in

1. Introduction

Conversations that take place over online platforms can become inappropriate and discourteous, degrading the quality of online conversations and having serious consequences like trolling, cyberbullying, and accessing adult explicit content. To avoid such situations, we want to build a **multi-label classification model that will detect inappropriate texts** and help us to further categorize them.

2. Related Work

We are dealing with text classification and currently Linear Support Vector Machine and Deep Learning algorithms are among the famous best algorithms explored for these problems. We evaluated some of the best performing inappropriate text classification models on our dataset using our evaluation metric. The highest score among these models is coming around *0.71*. Most of these approaches have used neural networks like WordBatch and FM_FTRL [1, 3] to solve the classification problem. The best performing model [0] (state-of-the-art) has final score *0.8072* and used Support Vector Machine with Naive Bayes Features.

3. Dataset Evaluation

We are using a Kaggle dataset [2] containing Wikipedia comments labeled by human raters for inappropriate behaviour. The dataset [1] has 6 labels: 'toxic', 'severe toxic', 'obscene', 'threat', 'insult' and 'identity hate'. We have a total of 2,23,286 data points in our dataset and we have divided it into: 80% training, 10% validation and 10% testing set. Thus the training set has 1,78,626 samples and the validation and testing set have 22,329 samples each.

3.1. Feature Extraction

Before applying any models on our dataset, we have done text cleaning which involves: converting to lowercase, removing special characters, removing numbers, removing stop words, replacing contractions with their full forms, Lemmitization.Till now we have used 3 techniques for feature extraction: Bag of Word, TF IDF, Word2vec. Word2Vec gave the best results with all the models as shown in figure 1.

3.2. Evaluation Metric

For our dataset we have chosen the evalutaion metric to be sum of 60% recall value and 40% precision. We have chosen this evaluation metric since the cost of the model classifying a decent text as obscene (false positive) is far less than the cost of it missing an obscene text (false negative).

4. Analysis and Progress

In our dataset, 22,468 out of 2,23,549 comments are classified as toxic i.e. 10.05%. These inappropriate comments contain 21,384 toxic comments, 1,962 severe_toxic comments, 12,140 obscene comments, 689 threat comments, 11,304 insult comments and 2,117 identity_hate comments as shown in figure 2. To understand the correlation and overlapping between these labels we plotted a correlation matrix and graph showing number of comments with multiple labels.

The correlation matrix shown in figure 3 shows that there is a high overlap between 'toxic', 'obscene', and 'insult' comments. Since there are not a lot of data points for 'Severe_toxic', identity_hate' and 'threat', they don't really overlap much with any other labels. The graph in figure 4 shows that most of the comments present in the dataset are clean and that comments with more than one toxicity labels decreases with the increase in the number of multiple labels. To understand what comments of various categories look like we used a tokenizer with standard stopwords to count the overall frequency of various words and plotted the ones with highest counts. figure 5 shows word frequencies of clean comments and figure 6 shows the word frequencies of comments under the threat category.

[1]Dataset link: https://drive.google.com/drive/u/1/folders/1yY_GmS89Rvm_YMB6TqNr3hZwrqVFmNvh

4.1. Algorithms and Techniques

After doing EDA, text cleaning, and feature extraction, we implemented the following algorithms with default parameters and various multi-label classification techniques like Binary Relevance, Classifier Chains, Label Powerset and Multi Output Classifier: *Baseline models: Naive Bayes and Logistic Regression Advanced Models: Random Forest and SVM* Classifier Chain showed the best results for both Naive bayes and Logistic Regression. The final score of Naive Bayes was 0.55 and Logistic Regression was 0.59. Binary Relevance showed the best results with Random Forest. The final score was 0.616. For SVM, both classifier chains and binary relevance showed similar results around 0.639. We have also implemented a basic Neural Network algorithm with one hidden layer to solve the problem. It is a fully connected neural network which uses ReLU activation for the input and hidden layers, each of which have one node and Sigmoid activation of the output layer, which has 6 nodes. The final score for this model was 0.6514.

4.2. Optimisation and tuning

To find the best parameters of all models, we plotted graphs of the important parameters of all models and saw how different parameters impact our models in term of overfitting and underfitting. In case of Naive Bayes the best value for var smoothing is 0.1, As shown in figure 7. We got the same result through Grid Search too. For Logistic Regression, the best values are solver= newton-cg, C=0 and max_iter =45 as shown in figure 8. For Random Forest, the best parameters are criterion = 'entropy', n_estimators = 25, max_features = 'auto', max_depth = 12 as shown in figure 9. The performance of evaluation metric remains constant after n_estimators becomes 25. For max_depth, the test set stops improving performance after 12 so that is the best value. The evaluation metric of Tuned Random Forest did not increase much but the variance decreased drastically from 0.42 to 0.14.

4.3. Challenges and Design choices

Some of the challenges that we are facing are:

1. SVM hyperparameter tuning is not working currently. To counter this we will try and make our set up more powerful by running our models on our local machine with High RAM access.

2. There is a class imbalance for which we will try oversampling. We will prefer oversampling over undersampling because the total inappropriate data points are less.

To check whether the training of our models is being done correctly or not and to make sure the model is not overfitting or underfitting we have calculated the variance for all our models and generated learning curves for our baseline models. The variance for baseline models is negligible, for SVM is very less 0.07 , the variance for Tuned Random forest is 0.14. The learning curves for Naive Bayes and Logistic Regression shown in figure 10 show that for a small dataset the model was overfitting but as the dataset size increased the model was fitting well to the data. This helps us conclude that the size of our dataset is big enough to provide stable training.

5. Result

The model which has given the best result so far has been Neural Network model, with a score of 0.6514. We also found that out of six labels, the individual scores for labels 'severe_toxic', 'threat', 'identity_hate' are coming out to be low, as we can see from figure 11. This is probably because there are fewer samples for these labels. This gives us a starting point in our error analysis. There is a significant gap between our results and the results of the state of the art. One possible reason for this can be that while we are using SVM the state of the art approach is using SVM with Naive Bayes features. Another difference is that while the state of the art appraoch has used TF-IDF method for feature extraction, we have used word2vec since that was giving us better results.

6. Future Work

We are planning to do error analysis for our final model. For that we will first identify which labels are being misclassified most and then try methods to prevent that. We will also try clustering algorithms to identify subsets of data which might be worth exploring. Apart from that we will explore more architectures of Neural Networks and try and tune them. We will also explore various more strategies with SVM like SVM with Naive Bayes features and LightGBM. We will also try extracting more features from our data like, length of the comment, average word length, amount of capitalization's, and presence of punctuation's like exclamation points and question marks, and analyse how it changes the results.

Originally we were using micro-averaging recall as our optimizing metric and micro-averaging precision as our satisfying metric. But this matrix was prioritizing recall too much also it was getting difficult to fix the satisfying value for precision. Thus we felty the need to change metrics.

Individual Contribution:

1. Aditi: Neural network, error analysis

2. Mansi: SVM, error analysis

3. Nimisha: Further tuning of Random Forest, error analysis

References

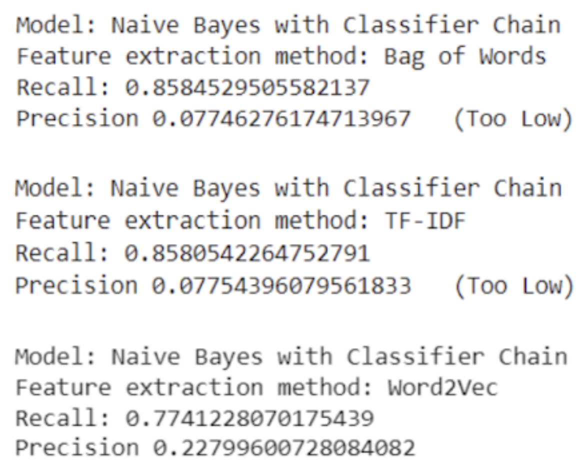
[1] Fork of blend it al + ensamble. https://www.kaggle.com/code/maksimovka/fork-of-blend-it-al-ensamble/script. 1

[2] Toxic comment classification challenge. https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/data. 1

[3] Wordbatch fm$_{f}trlusingmselb0.9804$.. 1

Jay Speidell. Toxic comment classification. https://github.com/jayspeidell/ToxicCommentClassification-, 2018. 1

Figure 1. Results for Naive Bayes with all 3 Feature Extraction techniques



```
Model: Naive Bayes with Classifier Chain
Feature extraction method: Bag of Words
Recall: 0.8584529505582137
Precision 0.07746276174713967   (Too Low)

Model: Naive Bayes with Classifier Chain
Feature extraction method: TF-IDF
Recall: 0.8580542264752791
Precision 0.07754396079561833   (Too Low)

Model: Naive Bayes with Classifier Chain
Feature extraction method: Word2Vec
Recall: 0.7741228070175439
Precision 0.22799600728084082
```

Figure 2. Number of comments under each label

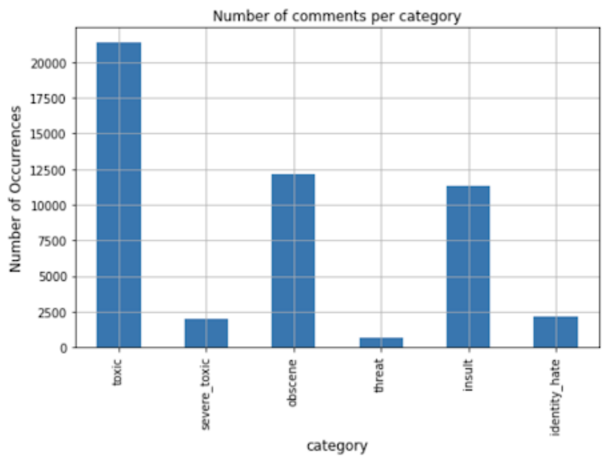


Figure 3. Correlation Matrix of all labels

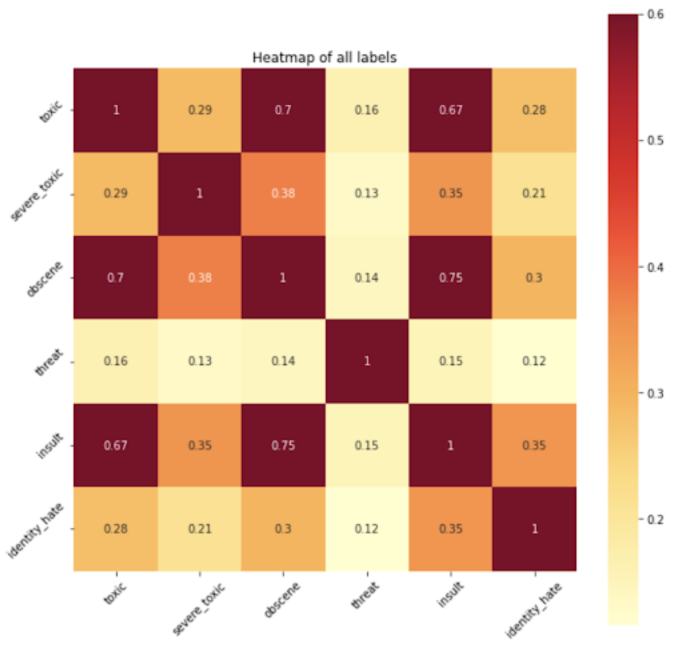


Figure 4. Number of comments with multiple labels

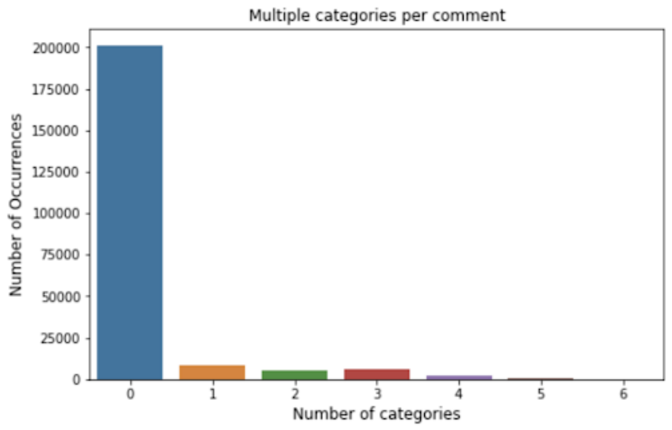


Figure 5. Word Frequency of Clean comments

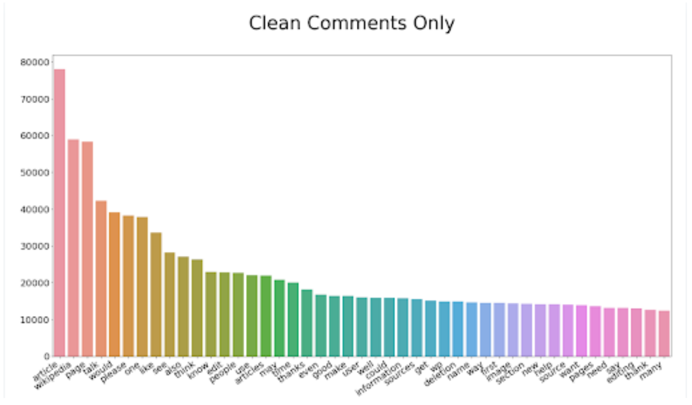


Figure 6. Word frequency of comments under the 'threat' category

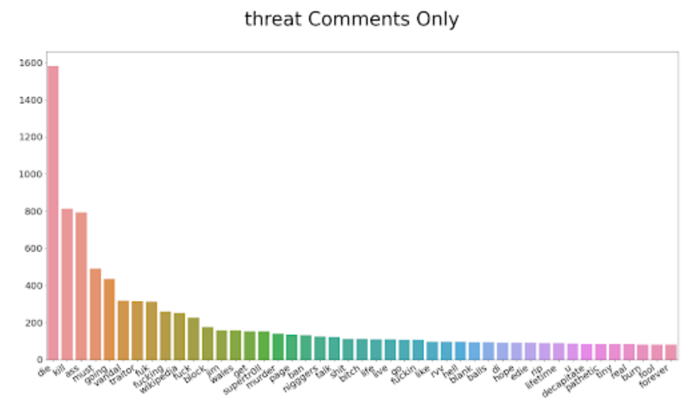


Figure 7. Hyperparameter tuning for Naive Bayes

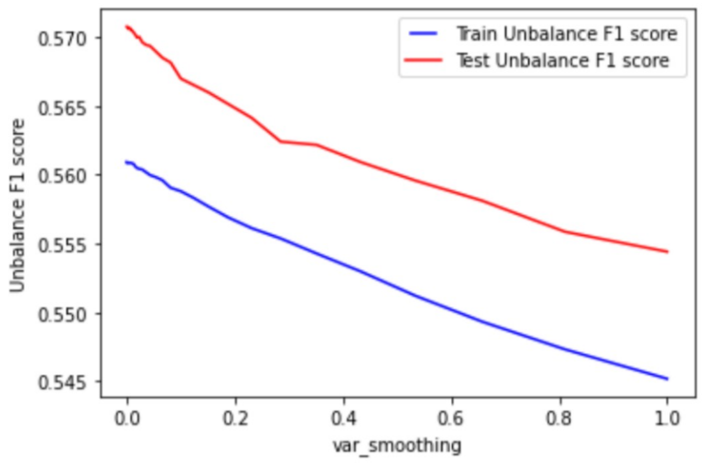


Figure 8. Hyperparameter tuning for Logistic Regression

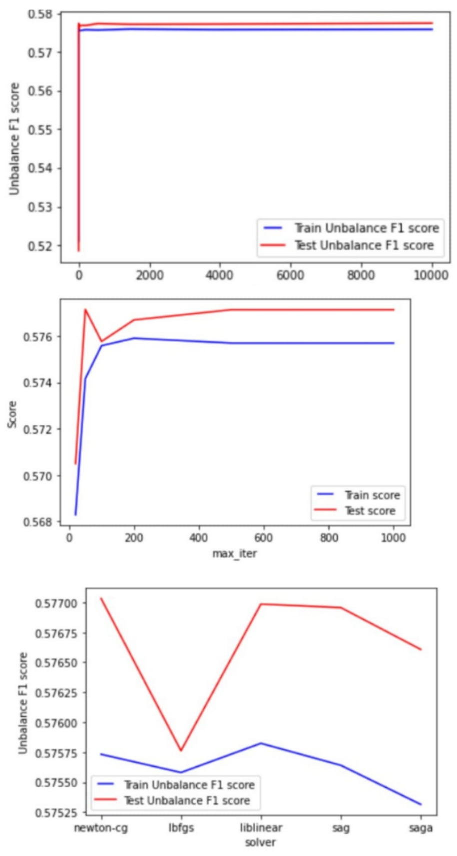


Figure 9. Hyperparameter tuning for Random Forest

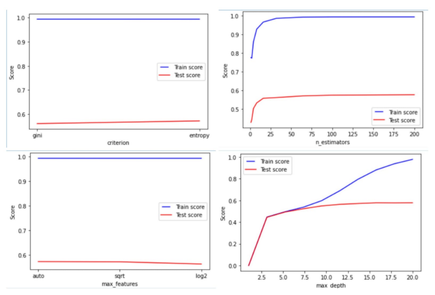


Figure 10. Learning curves

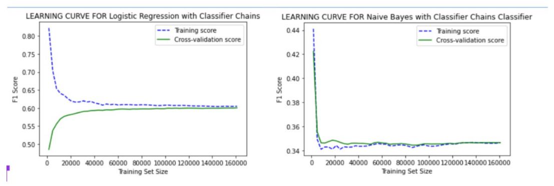
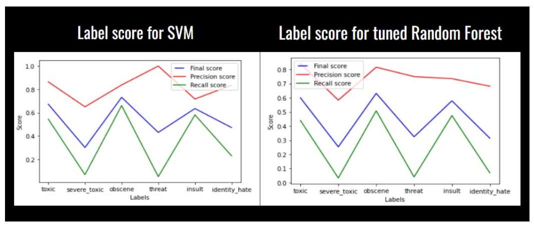


Figure 11. Label Score for models


Label score for SVM
Label score for tuned Random Forest
Final score
Precision score
Recall score
Score
Labels
toxic
severe_toxic
obscene
threat
insult
identity_hate