Toxic Comment Classification

Chakshu Anup Dhannawat, Ronak Mohata, Karan Jain

Abstract – This paper reports our experience with building a Toxic comment classifier. We have a dataset provided by Jigsaw. At the end of 2017 the Civil Comments platform shut down and chose to make their ~2m public comments from their platform available in a lasting open archive so that researchers could understand and improve civility in online conversations for years to come. Jigsaw sponsored this effort and extended annotation of this data by human raters for various toxic conversational attributes.

I. Introduction

Platforms that aggregate user content are the foundation of knowledge sharing on the Internet. But the catch is that not all people on the Internet are interested in participating nicely, and some see it as an avenue to vent their rage, insecurity, and prejudices. The problem with this is that people will frequently write things they shouldn't, and to maintain a positive community this toxic content and the users posting it need to be removed quickly. Our project aims to solve this problem by classifying comments as toxic and non toxic. Along with that we also aim to classify the comment into a toxic subtype.

Datasets

The file all_data.csv is used as the training dataset.
The train dataset contains 1999516 rows with 46 columns.In the data, the text of the individual comment is found in the comment_text column. Each comment in Train has a toxicity label (target), and models should predict the target toxicity for the Test data. This attribute (and all others) are fractional values which represent the fraction of human raters who believed the attribute applied to the given comment. For evaluation, test set examples with target >= 0.5 will be considered to be in the positive class (toxic).The data also has several additional toxicity subtype attributes. Subtype attributes are:

- severe_toxicity
- obscene
- threat
- insult
- identity_attack
- sexual_explicit

II. Methodology

Overview

There are various classification algorithms present out of which we shall implement the following

- *Random Forest Classification*
- *KNN*
- *Logistic Regression*
- *SVM*
- *MLP*
- *Gaussian Naive Bayes*
- *Logistic Regression*

We also make use of PCA and LDA for dimensionality reduction, and SFS for feature selection

Exploring the dataset and pre-processing

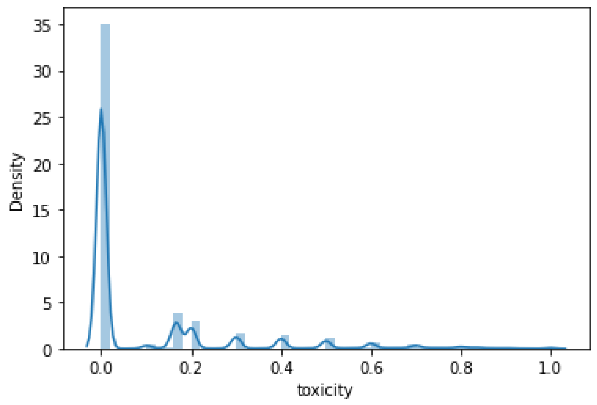


Fig 1.1: Distribution of label 'toxicity'

As the features of the dataset had continuous values(floating values between 0-1), we converted them into binary variables after using a specific threshold for each individual category after looking at the distributions of the classes. As seen from Fig 1.1, the density of the values is maximum at 0, so for this classification problem, we have chosen the threshold to be 0.2, to reduce the imbalance in our dataset.We have performed similar thresholding method for other labels too.

The methods used for preprocessing of text are as follows:

- **PorterStemmer and removing stopwords**: The Porter stemming algorithm (or 'Porter stemmer') is a process for removing the commoner morphological and inflexional endings from words in English. Its main use is as part of a term normalisation process that is usually done when setting up Information Retrieval systems. Stopwords are the commonly used english words which can be ignored while classifying text comments.
- **TfidVectorizer:** Transforms text to feature vectors that can be used as input to estimator. This transforms the dataset into sparse matrix.
- We also used another Vectorizer which coveted the dataset into dense matrix.(Only Count Vectorizer)

  Some basic text cleaning methods as removing punctuation, etc are also used.

Implementation of classification algorithms

After the dataset is pre-processed, we have used the above mentioned machine learning models to classify the comment text into a binary class: 'toxic' or 'non-toxic'. If the comment is toxic, then we further classify the comment text into the subcategories of a toxic comment.

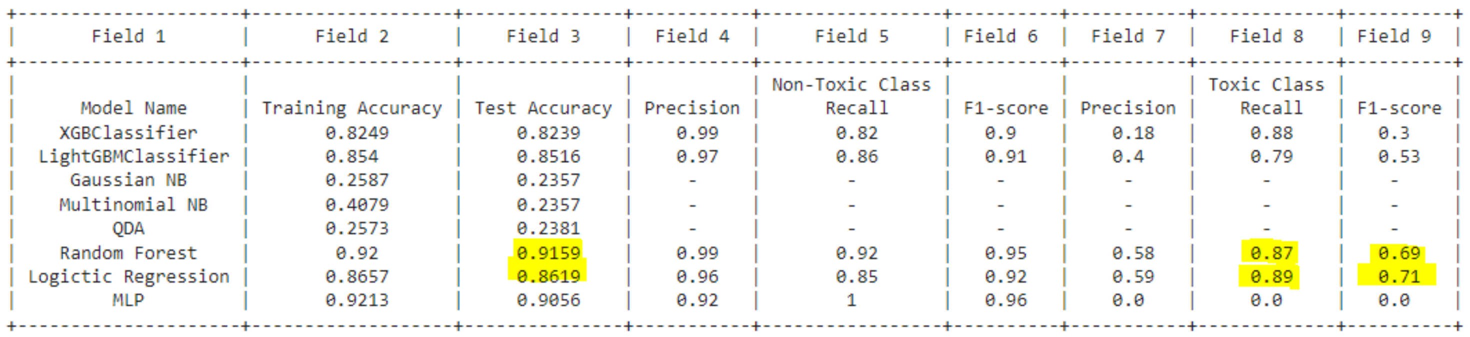
As TfidVectorizer converets the data into sparse matrices, the vectorised data has 100000 columns thus using dimensionality reduction techniques are of importance. But when dimensionality reductionality techniques were used, we observed that the ML models were performing poorly as compared to when LDA and PCA were not applied. This maybe due to the fact that many values in the sparse matrix are 0, and when dimensionality reduction techniques are used, the important values are getting diminished when multiplied by zeroes. Also, the vectorised dataset was quite big, which our PC was not able to support, even when the number of features were 1000. Using features lesser than these would have degraded the quality of dataset, thus we did not use PCA and LDA in our further models.

III.CLASSIFICATION ON DATASET AND EVALUATION OF MODELS

The models implemented were evaluated using techniques like - Classification report : precision , recall , f1 score and support , Confusion matrix, accuracy score and cross validation scores

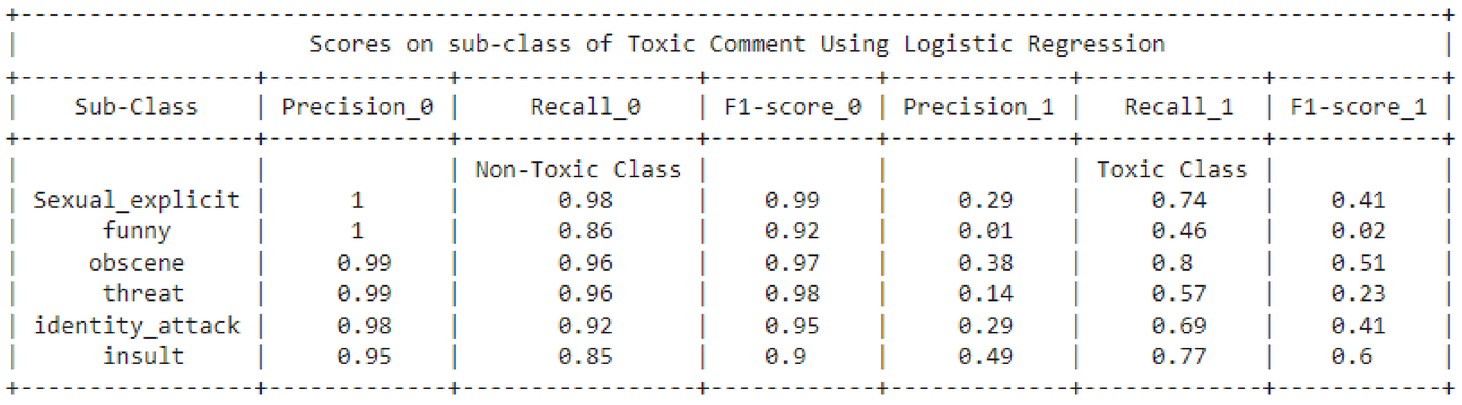
We observed that since the dataset is highly skewed wrt to ‘non-toxic’ class, so using **‘test_accuracy**’ as a metric will not be beneficial, thus,our prime evaluation estimator was **f1-score, recall** of ‘toxic’ comment class.

The results obtained from the models are as follows:



```
+--------------------+------------------+---------------+-----------+------------------+----------+-----------+------------+----------+
|      Field 1       |     Field 2      |    Field 3    |  Field 4  |     Field 5      | Field 6  |  Field 7  |  Field 8   | Field 9  |
+--------------------+------------------+---------------+-----------+------------------+----------+-----------+------------+----------+
|                    |                  |               |           | Non-Toxic Class  |          |           | Toxic Class |          |
|     Model Name     | Training Accuracy | Test Accuracy | Precision |      Recall      | F1-score | Precision |   Recall   | F1-score |
|   XGBClassifier    |      0.8249      |    0.8239     |   0.99    |       0.82       |   0.9    |   0.18    |    0.88    |   0.3    |
| LightGBMClassifier |      0.854       |    0.8516     |   0.97    |       0.86       |   0.91   |    0.4    |    0.79    |   0.53   |
|    Gaussian NB     |      0.2587      |    0.2357     |     -     |        -         |    -     |     -     |     -      |    -     |
|   Multinomial NB   |      0.4079      |    0.2357     |     -     |        -         |    -     |     -     |     -      |    -     |
|        QDA         |      0.2573      |    0.2381     |     -     |        -         |    -     |     -     |     -      |    -     |
|   Random Forest    |       0.92       |    0.9159     |   0.99    |       0.92       |   0.95   |   0.58    |    0.87    |   0.69   |
| Logictic Regression |      0.8657      |    0.8619     |   0.96    |       0.85       |   0.92   |   0.59    |    0.89    |   0.71   |
|        MLP         |      0.9213      |    0.9056     |   0.92    |        1         |   0.96   |    0.0    |    0.0     |   0.0    |
+--------------------+------------------+---------------+-----------+------------------+----------+-----------+------------+----------+
```

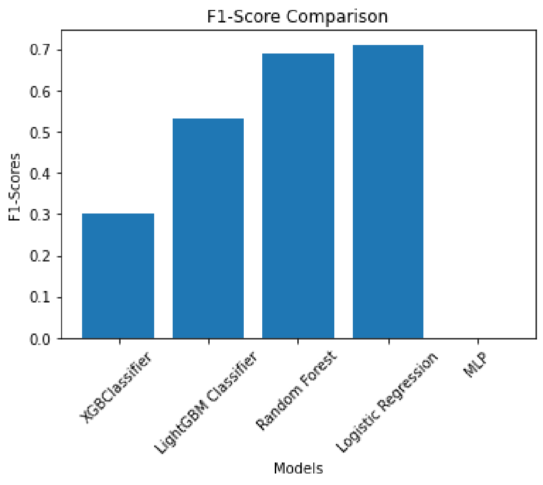
Fig 1.1: Analysis of Vrious Classifiers

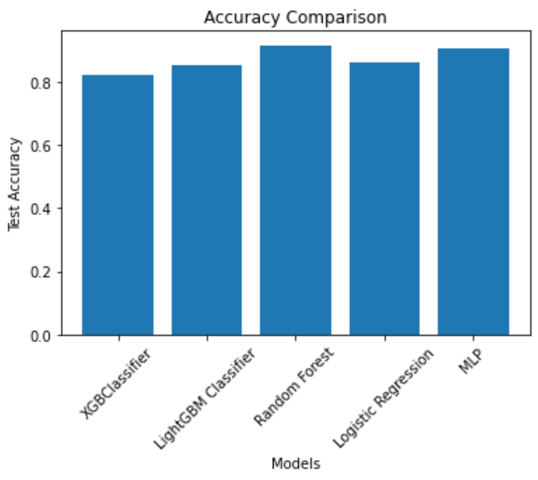


```
+--------------------------------------------------------------------------------------------+
|                   Scores on sub-class of Toxic Comment Using Logistic Regression           |
+----------------+-------------+----------------+------------+-------------+-------------+------------+
|   Sub-Class    | Precision_0 |    Recall_0    | F1-score_0 | Precision_1 |  Recall_1   | F1-score_1 |
+----------------+-------------+----------------+------------+-------------+-------------+------------+
|                |             | Non-Toxic Class |            |             | Toxic Class |            |
| Sexual_explicit |      1      |      0.98      |    0.99    |    0.29     |    0.74     |    0.41    |
|     funny      |      1      |      0.86      |    0.92    |    0.01     |    0.46     |    0.02    |
|    obscene     |    0.99     |      0.96      |    0.97    |    0.38     |     0.8     |    0.51    |
|     threat     |    0.99     |      0.96      |    0.98    |    0.14     |    0.57     |    0.23    |
| identity_attack |    0.98     |      0.92      |    0.95    |    0.29     |    0.69     |    0.41    |
|     insult     |    0.95     |      0.85      |    0.9     |    0.49     |    0.77     |    0.6     |
+----------------+-------------+----------------+------------+-------------+-------------+------------+
```

Fig 1.2: Results of Logistic Regression Models on Classifying the subcategories of Toxic Comments.

IV. RESULTS AND ANALYSIS





As we can see, the models which give a good result are ‘Random Forest’ and ‘Logistic Regression’. Although the test accuracy of Random Forest Model is higher than the test accuracy of Logistic Regression Model, we find that the recall and F1-Score of the ‘Toxic Class’ is better in the case of Logistic Regression Model. Since our primary evaluation technique is F1-Score, so we conclude that Logistic Regression will be best for Deployment in our website.

Other Classifiers like ‘Gaussian Naive Bayes’, ‘Multinomial Naive Bayes’, ‘QDA’ were not performing well, as according to us, they were not able to work properly with the sparse dataset.

Our project ends here, but it brings us to a lot of paths and ways to further improve the Toxic Comment Classification.
We can try new models of Classification and check if any of them works better(Those which we have not learnt in syllabus)
We can also try complex Neural Networks as there exists many varieties of Neural Networks to accomplish different purposes.
We can also try many advanced methods of Natural Language Processing, so as to improve the information gained from the comment texts, and improve the overall performance of our model.

V. LINK TO THE ONLINE DEPLOYMENT:

Link to the website developed: **Web Deployment**
Link to the github code for backend: Github Backend

Contributions

The learning and planning was done as a team.The individual contributions are as given

- Chakshu Anup Dhannawat(B20AI006): Data preprocessing, EDA,NLP(Stemming and Vectorization),Logistic Regression, Decision Trees, Report.
- Ronak Mohata(B20EE053): NaiveBayes Model, XGBclassifier, LightGBM Classifier , Model Deployment on the web, implementing end-to-end pipeline, Report
- Karan Jain(B20AI016): EDA, Random Forest , KNN, Multi-LayerPerceptron ,Classification of subtypes of toxic comments, worked on comparison between both the vectorizer models, Report.

References

[1] Pattern Classification -Book by David G. Stork, Peter E. Hart, and Richard O. Duda
[2] Machine learning methods for toxic comment classification: a systematic review
[3] Kaggle Dataset