Toxic Comment Classification

Aneri Dalwadi AU1940153, Kairavi Shah AU1940177,
Mananshi Vyas AU1940289, Nandini Bhatt AU1940283

***Abstract*—Considering to moderate countless inappropriate or toxic comments on social media manually is a serious impractical approach. Thus, in our project, we implement this using and comparing various ML models. The data is downloaded from Kaggle, containing 1.5 million comments from wikipedia's talk page and has to be implemented with multiclass-classification - toxic, obscene, identity hate, severe, etc. For the complete process till choosing the best model, we implement text vectorization, tf-idf calculations and basic use of linear algebra to get the reduced dimension of the vector form of out input. We then apply ML models like SVM, Naive Bayes Algorithm, Random Forest, etc. We compare the models using... To get more accurate predictions, we tune the parameters.**

.

***Index Terms*—Dimension reduction, Vectorization, tf-idf, Eigen Values, Accuracy, Multilabel Classification, Machine Learning Algorithms, Toxic Comments**

I. Introduction

SOcial media sites are one of the most popular websites on the internet today flooding with comments. It is vital to manage the user-generated offensive content on many of these sites that can make a user's online experience unpleasant, and we may want to filter these(important for parents as to what their child reads online). Most researchers tried solving this problem using NLP using a neural network but we as a group avoid NLP and implement this using the crux of linear algebra(feature selection), statistics(model selection), ML models(classification), Data Science concepts(data collection, preprocessing) and mathematics(tfidf value calculations).

Using different Machine learning classifiers we try to detect it into various types of toxicity categories.

Since ML models work with numbers, we need to convert our data into vector format after cleaning it. Starting briefly, we then split them into tokens: single word tokens, 2 gram, single character shingles, 2 character shingles. We calculate the tf-idf value for all these combinations and analyze them using ggplot. We finally use the combinations of 2 character shingles to make a 3D matrix and fill in the previously calculated tf-idf values of the same in the matrix. We then fuse the concepts of linear algebra like eigenvalues and eigenvectors for dimensionality reduction on this square matrix. We store the top 5 eigenvalues and we use them as our 5 features. We then use models for classification to train and test our model. The end goal of this project is to create a better, safer and appropriate environment online. In the long term, this would allow people to better connect with each other in this increasingly digital world.

II. Literature Survey

In past years the classification of toxic comments has been done by various researchers extensively. These researchers have applied different machine learning algorithms to classify the comments. Following are some the papers:

- In this paper[2], authors have used SVM with TF-IDF as the feature extraction and Chi Square as the feature selection. The best performance obtained using the SVM model with a linear kernel, without implementing Chi Square, and using stemming and stopwords removal with the F 1 Score equal to 76.57%.
- In another paper[3], authors have used a semi-supervised approach to detect profanity-related offensive content on Twitter. They achieved a 75.1 % TP rate with Logistic Regression and a 69.7 % TP rate with popular keyword matching baseline. The false-positive rate was identical for both at about 3.77%.

III. Implementations

Here we present our thinking and complete implementation along with explaining several concepts in brief. We begin with gathering the dataset and problem statement (https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge) from Kaggle.

We undertake data cleaning using Apache Spark(pySpark library) and regex and then we split the dataset into documents where each document is a comment. Each comment is split into tokens of single word tokens, 2 gram, single character shingles, 2 character shingles using language R and its library(tidyverse). We focus on observing the patterns for the tf-idf values for these using ggplot. After understanding the importance of each token type, we choose 2 character shingles. Wanting numeric features for our model to train we have 2 approaches:

- Approach 1 - convert each character to unicodes and perform dimensionality reduction. However, there is a problem with using pca, that for m*n matrix, it will give output as min(m,n). Thus for our case it will return a single integer, where we need some d dimensional features.
- Hence, Approach 2 - take all 2 character shingles and create a 3D matrix - Following is the description of the dimensions(d) D1, d2 - characters from a-z D3 - above 2D matrix for each comment.

Hereby we implement approach 2: Now since we have already calculated the tf-idf for these 2 character shingles, we fill in the matrix with the tf-idf values. We use eigen values linear algebra for dimensionality reduction. Eigen values help us to determine the factor by which myvector will change upon change in any input(2-character shingles). With motive of base check of our method, we begin with getting the 5 most significant eigen values for the phase-2 of our project,

we further add the number of eigen value as parameter which will be tuning. Further, we use these 5 values as from each comment as our integral input features. Now, we train and test our models using various ML models: (Base models: Linear Discriminant Analysis, Quadratic Discriminant Analysis and Logistic Regression Other models: Random forest classifier, KNN).

IV. Modelling

We specifically train 5 models:

- Logitic Regression
- K-Nearest Neighbors
- Random Forest Classifier
- Linear Discriminant Analysis
- Quadratic Discriminant Analysis

The first task was to establish a baseline model. We select 3 baseline models: Logistic Regression, LDA and QDA for this purpose. The justification for choosing these as base models is given below.

Each model serves its own purpose and the output that it gives can be interpreted in various ways that will be useful to us which will in turn help us choose our modelling strategy.

Logistic Regression

- Reasoning:
  - This is a highly popular choice due to its simplicity.
  - Getting a high AUC under ROC score would mean that the predictors are associated
  - linearly with the probability of outcome.

Linear Discriminant Analysis

- Reasoning:
  - LDA a popular choice when it is assumed that the predictors are normally distributed
  - The decision boundary is linear if it yields a high AUC score.
  - If the data is truly linear, the metrics of Logistic Regression and Linear Discriminant Analysis should be in agreement with one another.

Quadratic Discriminant Analysis

- Reasoning:
  - QDA is an extension of Linear Discriminant Analysis.
  - QDA models non-linear decision boundaries between outcomes and predictors.
  - It also assumes that predictors are normally distributed. Both LDA and QDA fall in category of discriminative classifiers.
  - If the AUC under ROC score for QDA is high than LDA and LR, we will be able to interpret that non-linear classifiers will perform well.

Random Forest Classifier

- Reasoning:
  - RFC is a method which uses bagging to build an ensemble of decision trees.
  - Decision trees are highly prone to over-fitting, hence, we apply Random forests which reduces that variance which in turn provides stability of model predictions.
  - Random Forest Classifier provides good results when the data is truly non-linear and the posterior distribution of predictors is very hard or impossible to be modelled simply.

K-Nearest Neighbors

- Reasoning:
  - KNN the technique which uses distance metrics and knowledge of neighborhood to evaluate the outcomes and predictors.
  - It is used to treat completely non-linear data as well but end up differing from RFC when we do not want a tree or ensemble.
  - There is no explicit modelling involved but rather the data is treated directly at how it is distributed in values and not as probability distributions.

V. Results

Model	AUC	F1 Score
RFC	0.5717	0.1674
QDA	0.6071	0.2031
KNN	0.5455	0.1784
LDA	0.5731	0.1777
LR	0.5271	0.1776

Models and Metrics

It is quiet evident form the AUC under ROC score that the models do not have high accuracy. The prime reasoning for this stated as follows:

- The main motive for the phase-1 of our project is to understand the core applications and interpretations of our base models, learn different approaches to vectorize the given text data without entering the domain of NLP and learn to pre-process real-time big-data.
- Hyper parameter fine tuning of all the linear and non-linear classifiers, including varying and cross validating different number of eigen values will be the phase-2 of this project.
- The results from phase-2(Final end-sem presentation) will contain the finely tuned models with comparison including testing on more non-linear classifying models

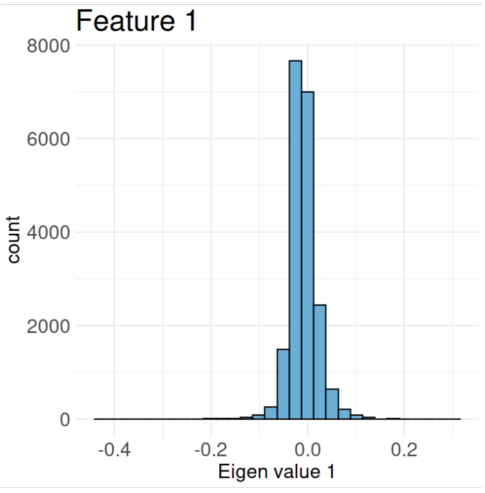

Figure 1

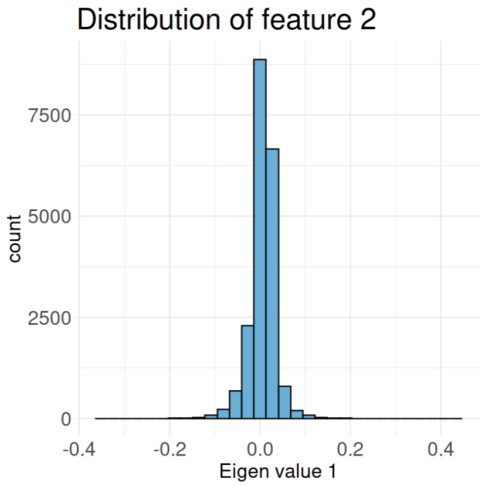

Figure 2

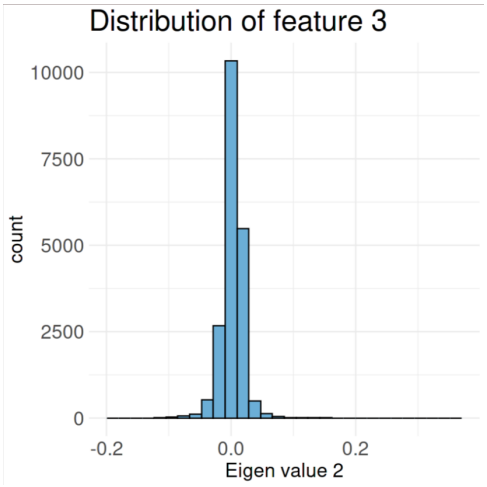

Figure 3

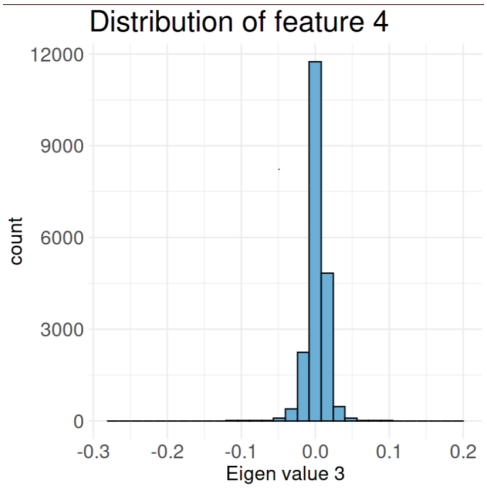

Figure 4

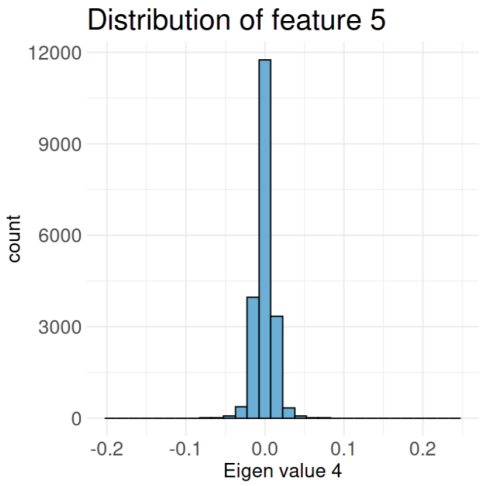

Figure 4

VI. Interpretation

As inferred from the table, the AUC score for LDA and LR are close to each other, yet less than the score of QDA. With this, we can say that non-linear classification models will perform better on our given data set. Also, the AUC score of QDA and LDA are high than all of the 5 models implemented. This shows that the distribution of the input features must be nearly Gaussian. The justification for this inference is visible in the graphs plotted representing the Gaussian distribution if the feature. It is interesting to note that all the features are similarly distributed.

VII. Conclusions

Our initial training dataset is very big and as a result when we do some preprocessing like creating n-grams or character-shingles, we end up with a massive dataset that is impossible to be handled on our local systems. So to justify a proof of concept, we use a random sample of a size that provides enough playground to experiment. This tradeoff helps us explore various modelling techniques along with other preprocessing as well.

In our data, we see that AUC under ROC curve of QDA and RFC is high but that of KNN is low which means that our data shows non-linearity between predictors and response but having some information about the posterior distribution might boost our results and so in such situations using models like Gradient Boosting Trees which handle interaction between predictors well can yield high results.

VIII. Languages and libraries used

1) Apache Spark (pySpark) - Data pre-processing implementing Regex in SQL along with parallelizing the process in Spark
2) R (Tidyverse) - Data manipulation, tokenization, visualizations
3) ggplot - Used in R for visualizations
4) Python - Extracting features, model training and testing

REFERENCES

[1] "Toxic comment classification challenge," Kaggle. [Online]. Available: https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/data. [Accessed: 20-Mar-2022].

[2] K. M. Lhaksmana, D. T. Murdiansyah, and N. S. Azzahra, "Toxic Comment Classification on SocialMedia Using Support Vector Machine and Chi Square Feature Selection," View of toxic comment classification on social media using support Vector Machine and Chi Square feature selection. [Online]. Available: http://socj.telkomuniversity.ac.id/ojs/index.php/ijoict/article/view/552/316. [Accessed: 20-Mar-2022].

[3] G. Xiang, B. Fan, L. Wang, J. I. Hong, and C. P. Rose, Detecting Offensive Tweets via Topical Feature Discovery over a Large Scale Twitter Corpus. [Online]. Available: https://www.cs.cmu.edu/ lingwang/papers/sp250-xiang.pdf. [Accessed: 20-Mar-2022].

[4] Van Rossum G, Drake Jr FL. Python tutorial. Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands; 1995. Available: https://www.python.org/

[5] Wickham H. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York. ISBN 978-3-319-24277-4, 2016. Available: https://ggplot2.tidyverse.org

[6] R Core Team (2019). R: A language and environment for statistical¿ computing. R Foundation for Statistical Computing, Vienna, Austria.¿ Available: https://www.R-project.org/.