Toxic Comment Classification

NaN-Prediction Pending

Aneri Dalwadi AU1940153

Mananshi Vyas AU1940289

Nandini Bhatt AU1940283

Kairavi Shah AU1940177

Overview

- Problem Statement
- Introduction
- Literature Review
- Flowchart
- Our Approach
- Model Used
- Results & Conclusion
- Future Work
- References

Problem Statement

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.
We aim to filter these comments via multilabel classification according to their toxicity level.

Introduction

- We try to classify toxicity and calculate its severity 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, severe toxic, obscene, identity hate, severe, insult, etc.
- For the complete process of choosing the best model, we implement text vectorization, tf_idf calculations, and primary use of linear algebra to get the reduced dimension of the vector form of our input. We then apply ML models like LDA, QDA, LR, RFC and KNN.

Literary Review

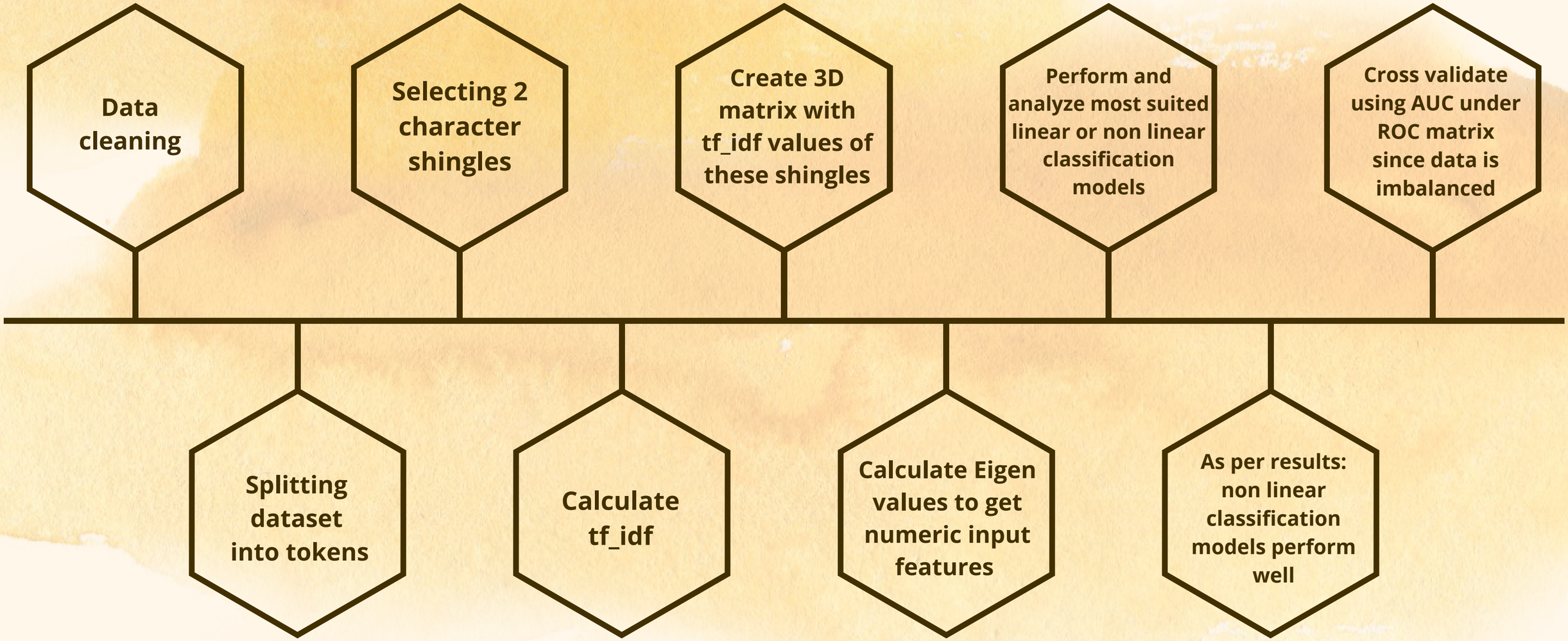
Toxic Comment Classification on SocialMedia Using SVM and ChiSquare Feature Selection

In this paper, 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%.

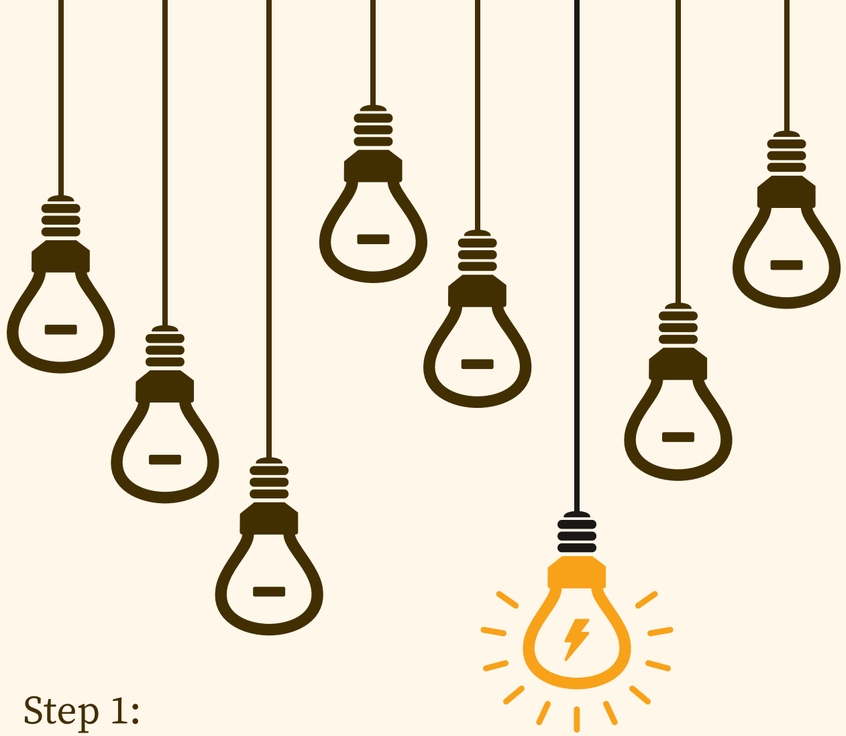
Detecting Offensive Tweets via Topical Feature Discovery over a Large Scale Twitter Corpus

In this paper, 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%.

Flowchart


Data cleaning
Selecting 2 character shingles
Create 3D matrix with tf_idf values of these shingles
Perform and analyze most suited linear or non linear classification models
Cross validate using AUC under ROC matrix since data is imbalanced
Splitting dataset into tokens
Calculate tf_idf
Calculate Eigen values to get numeric input features
As per results: non linear classification models perform well

Approach



Step 1:
Data and problem from kaggle

Step 2:
Data cleaning using Apache Spark(pySpark library) and regex.

Step 3:
Split the dataset into **documents** where each document is a comment. Each comment is split into tokens - **single word tokens, 2 gram, single character shingles, 2 character shingles**. Use of R and its library(tidyverse)

Approach 1

Step 3.1.1: **character to Unicode** 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.

Approach 2

Step 3.2.1:
Take all 2 character shingles and create a 3D matrix - The following is the description of the dimensions(d).

d1, d2 - characters from a-z
d3 - above 2D matrix for each comment.

Step 3.2.3:
We fill in the matrix with the tf-idf values. Now we use linear algebra for dimensionality reduction very well.

Step 4:
Get eigen values for each comment

Step 5:
Select 5 most significant EVs and use these as feature inputs to train models

Why 2-character shingles?

- Comments are unpredictable. Somebody can write "heck" or "heckheck" or even "heckheckheck".
- This affects the TF-IDF values.
- Hence, we avoid using any n-gram.

What if these 2 character are present in other non-toxic word?

- The major advantage is in using the 3D matrix of TF-IDF values
- For eg: Back | Heck.
  - Clearly, the vector formation of these 2 words will be different creating no further problems.
  - And since we are looking at the TF-IDF value matrix, it will give us the relative change in vector as per its occurrence.

Problem Faced

- On obtaining 2-character shingles of 1.5 million comments, 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 modeling techniques along with other preprocessing as well.

Models Used

Base Models

- **Logistic Regression -**
  - Linear classifier
- **Linear Discriminant Analysis -**
  - Linear Classifier. Assumes gaussian distribution of input features
- **Quadratic Discriminant analysis -**
  - Non-linear classifier. Assumes gaussian distribution of input features

Further application of non-linear classifiers from results

- **Random Forest Classifier-**
  - For non-linear classification
  - Better and less prone to overfitting than decision Tree
- **KNN -**
  - Uses distance metrics and knowledge of neighborhoods to evaluate the outcomes and predictors.

Results

V. Results

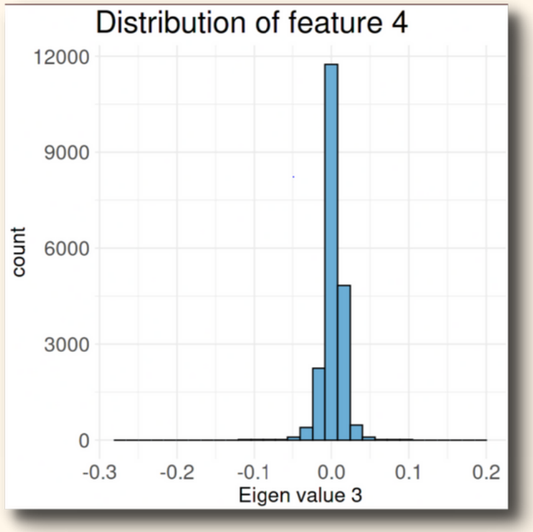
Model	AUC	F1 Score
RFC	0.5717	0.1674
QDA	0.6071	0.2031
KNN	0.5455	0.1784
LDA	0.5731	0.1777
LR	0.5271	0.1776

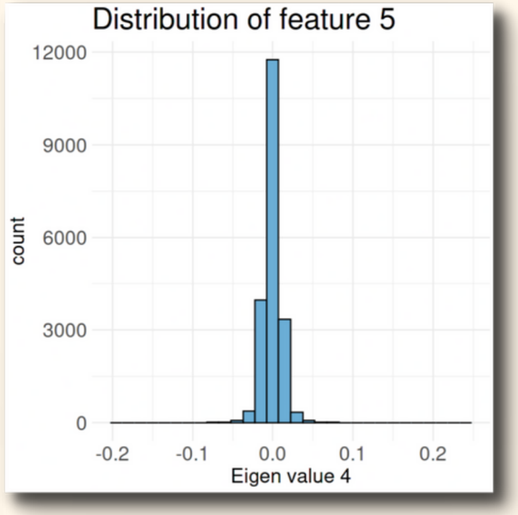
Models and Metrics

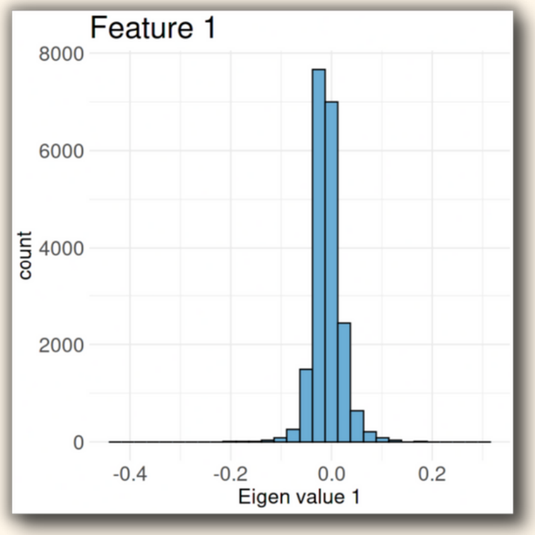
- As inferred from the table, the AUC score for LDA and LR are close to each other, yet less than the score of QDA.
- Thus non-linear classification models will perform better on our given data set. Also, the AUC score of QDA and LDA are high then all of the 5 models implemented.
- This shows that the distribution of the input features must be nearly Gaussian(as justified in next slide)

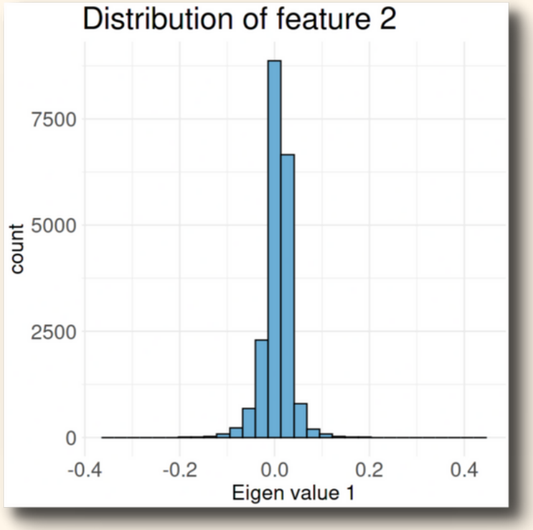
Graphs

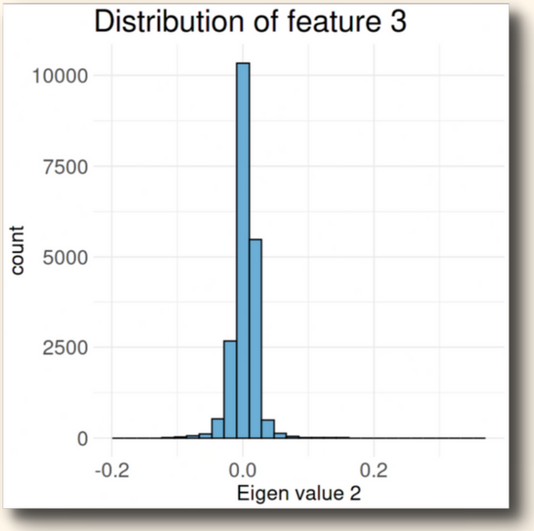
```
ggplot(df)
+ geom_histogram(aes(f0), color = "black", fill = "#6baed6")
+ theme_minimal()
+  ggtitle("Distribution of feature 1")
+ scale_x_continuous("Eigen value 1")
+  theme(axis.title = element_text(size = 20),
        axis.text = element_text(size = 20),
        plot.title = element_text(size = 30))
```





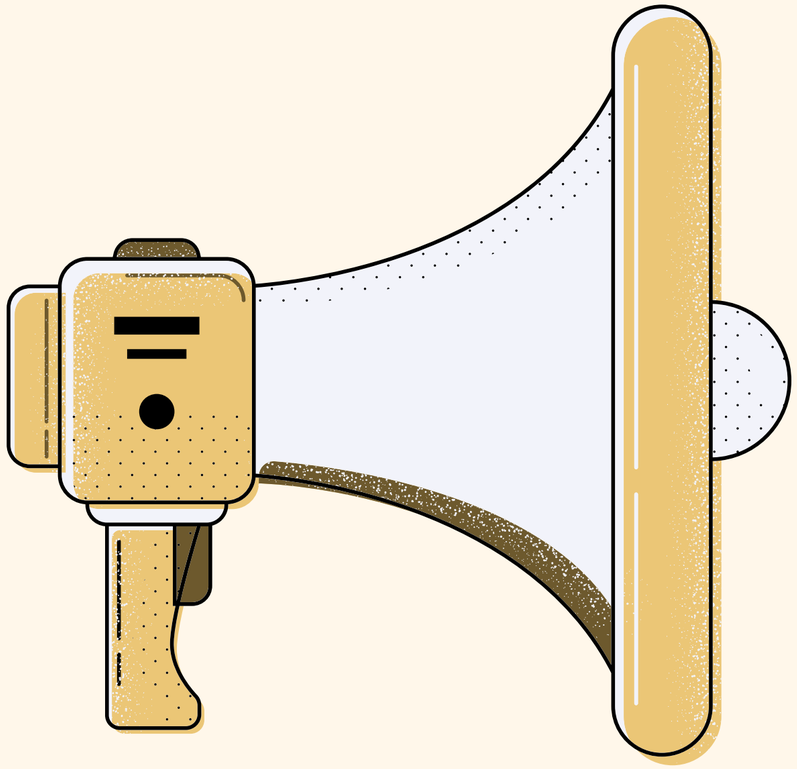






Conclusion

- Hyperparameter fine-tuning of all the linear and non-linear classifiers, including varying and cross-validating different numbers of eigenvalues will be 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
- In our data, we see that AUC under the 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 the interaction between predictors well can yield high results.



Roles

Aneri: Implemented codes in R and ggplot for various inferences. Implementation and interpretation of models. Interpretation of use of Eigen values.

Mananshi: Implemented codes in Apache Spark for data cleaning and pre-processing. Understanding and interpreting models. Finding appropriate approach to solve the problem.

Nandini: Helped in implementing pre-processing with use of regex and apache spark. Interpretation of use of eigen values.

Kairavi: Helped with implementation of models, data collection and conversion to csv files and reducing the data size. Understanding and interpreting models.

Future Work

Observe trends and accuracy by doing the following:

1. Hyperparameter fine tuning
2. Change the dimension of input data
3. Use different matrices for dimension reduction
4. Select randomized train data and see how the accuracy varies due to variation in the distribution of input data
5. Use more non-linear classification models

6. Internal parameter tuning in models
7. Explore the distance matrix from the calculated Unicode vectors of the comments and see the varying trends

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/.

Thank You!