Identification & classification of toxic comments & hate speech

Aishwarya Singh
aisingh@ucsd.edu
University of California - San Diego
San Diego, California, USA

Kamal Manchanda
kmanchanda@ucsd.edu
University of California - San Diego
San Diego, California, USA

Rohit Chandra
r2chandra@ucsd.edu
University of California - San Diego
San Diego, California, USA

Sourav Roy
s3roy@ucsd.edu
University of California - San Diego
San Diego, California, USA

Abstract

In this assignment, we want to build a multi-headed model that's capable of detecting different types of toxicity like threats, obscenity, insults, and identity-based hate that are assigned to each post on social media or on any online conversation platform/forum. Our work is inspired by the Kaggle contest "Toxic Comment Classification Challenge". The goal of this prediction is to correctly identify toxicity label of each posted conversation to help improve online conversation.

Introduction

The background for the problem originates from the multitude of online forums, where-in people participate and actively make comments. Discussing things you care about can be difficult. The threat of abuse and harassment online means that many people stop expressing themselves and seek different opinions. Platforms struggle to effectively facilitate conversations, leading many communities to limit or completely shut down user comments.

As the comments sometimes may be abusive, insulting or even hate-based, it becomes the responsibility of the hosting organizations to ensure that these conversations are correctly categorized as some platforms may be fine with profanity, but not with other types of toxic content. The goal of this project will be to use machine learning to identify toxicity in text, which could help deter users from posting potentially hurtful messages, craft more civil arguments when engaging in discourse with others, and gauge the toxicity of others users' comments.

This project aims to implement various machine learning models - specifically Logistic Regression, Long Short Term Memory Networks (LSTM), and Naive Bayers - to tackle the above task, assessing these models' performances on multi-label classification tasks.

1. Background and Related Literature

Sentiment classification regarding toxicity has been intensively researched in the past few years, mainly in the context of social media data where researchers have applied various machine learning systems to try and tackle the problem of toxicity and the related, more well-known, the task of sentiment analysis. Comment abuse classification research initially began with Yin et al's application of combining TF-IDF with sentiment/contextual features. They compared the performance of this model with a simple TF-IDF model and reported a 6% increase in Fl score of the classifier on chat-style datasets (Kongregate, MySpace)[1].

Nguyen and Nguyen[2] proposed a model for sentiment label distribution that involved a ensemble of using a Deep-CNN for character-level embeddings with a Bidirectional LSTM to produce sentence-level feature representations from word-level embeddings. This model attained a best prediction accuracy of 86.63% on the Stanford Twitter Sentiment Corpus. Their findings indicate the prospective advantages of utilizing LSTM and DeepCNN models on toxicity classification.

Chu and Jue[3] compared the performance of various deep learning approaches to this problem, explicitly using both word and character embeddings. They assessed the performance of RNNs with LSTM and word embeddings, a CNN with word embeddings, and a CNN with character embeddings. The best performance they achieved was a 93% accuracy using the character-level CNN model.

2. Data

2.1 Data Description

One of the primary challenges in classification cases is having appropriately labeled data, in which a representative training set could be extracted for modeling. This quandary is even more pronounced [4] for text-based or Natural Language Processing [NLP] problems. As the sentiment inference of written communication is subjective, there is little choice but to leverage human labeling instead of a proper analytical labeling model (hence the need for the classification algorithm). Although many large unlabeled text corpora are readily available, human-labeled data is much more infrequent [5].

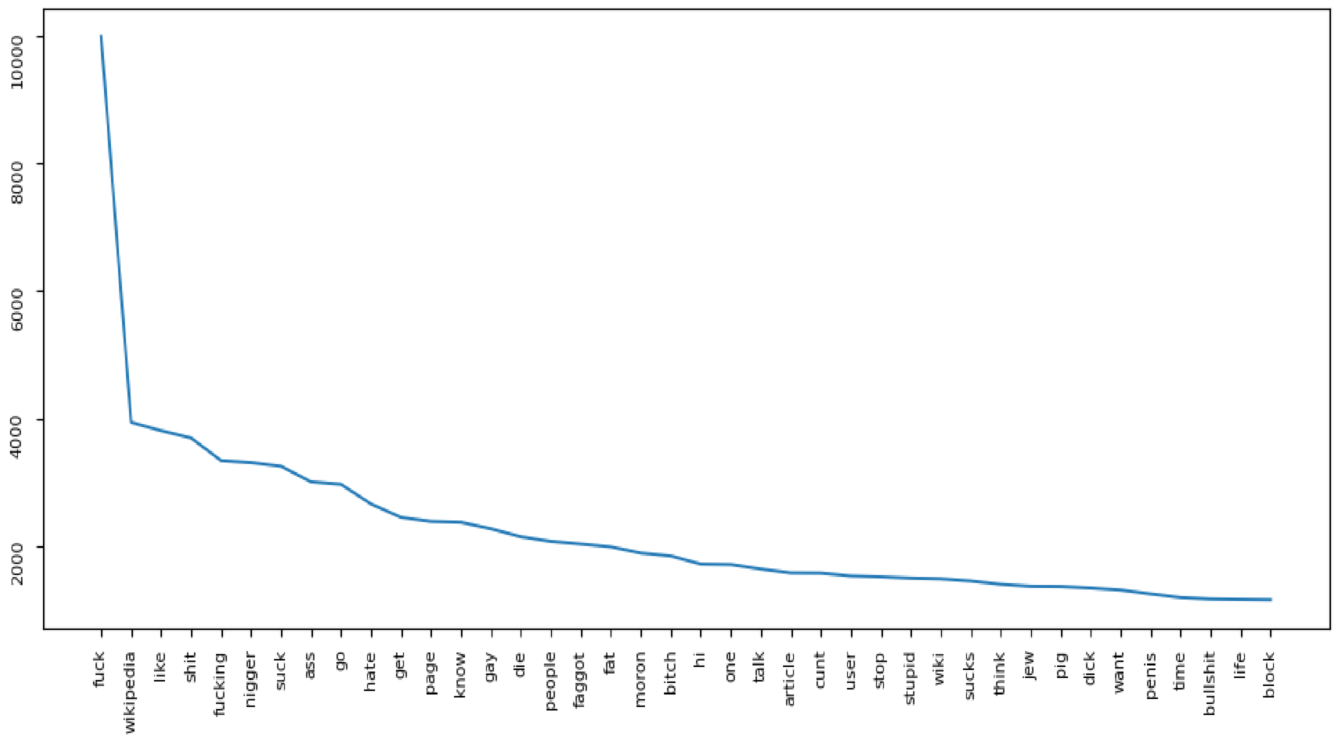


Figure 1. Most Frequent words used comments labelled under any toxic category

Fortunately, for various altruistic reasons, several organizations are providing open-sourced labeled data sets1. One such source is the WikiMedia Foundation, which offers text comment snippets from Wikipedia talk pages. Crowd-sourcing was used to manually label 159,571 text comments, flagging each as one of seven following options: 1) Non-Toxic, 2) Toxic, 3) Severe Toxic, 4) Obscene, 5) Threat, 6) Insult, and 7) Identity Hate. The data set has a large number of Wikipedia comments which human raters have labeled for toxic behavior. The following information on the data set is available:

1. id: unique identification for a comment
2. text: text for a comment
3. toxicity labels: in which category of toxicity label does the comment fall in.

2.2 Exploratory Data Analysis

In the data analysis part we first cleaned the comments by removing the Punctuation,stop words and then converting every comment into lower case by doing this we get the cleaned data to run a predictive task or do the further analysis. We looked into the actual most frequent words in the comments which were labeled under any one of the 6 different labels. Not to any surprise we found some vulgar and derogatory words in the top 30 most frequent words under non-toxic comments(Figure 1).

We further found out the number of comments in each toxic category and the number of comments having one or more label of toxicity

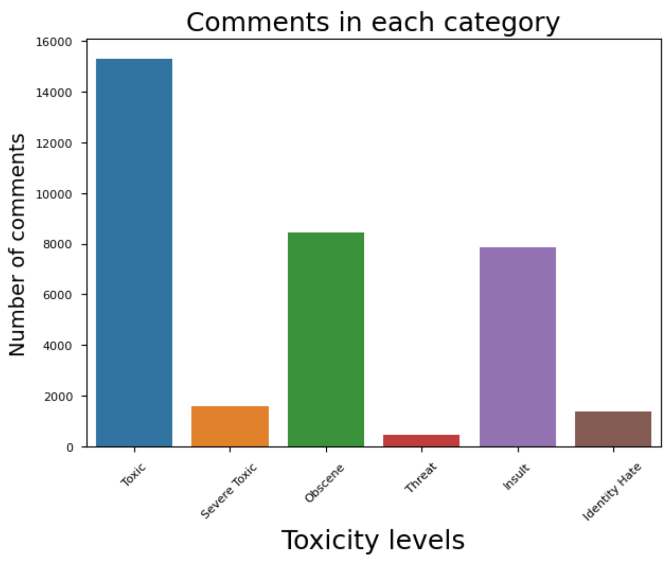


Figure 2. Distribution of toxicity labels in the dataset

It should be noted that 143,346 were not labeled in any category. Out of 159,571 comments only 16,225 comments are labeled under toxic comment categories. 8,449 comments tagged to obscene category following that 7,877 comments are tagged to insult making it third most popular tag in toxic comment calcification

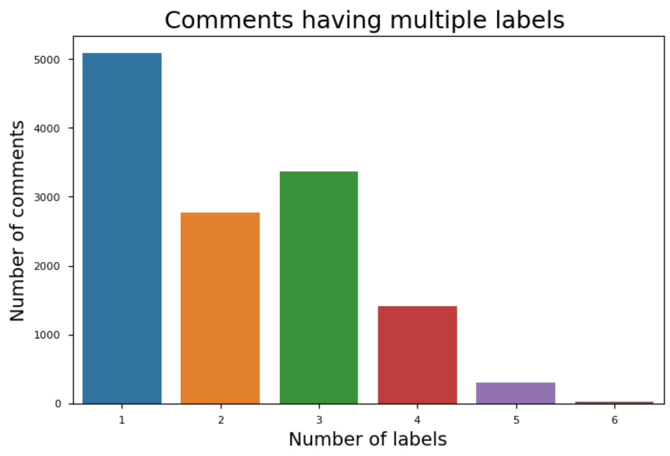


Figure 3. Comments can be categorized into multiple labels

Going further we visualised how many comments have multiple labels.It was interesting to see that most of the comments are tagged either to 1 label or 3 labels .

3. Approach

This project focuses on studying the effects of using different word embeddings and classification models for the multi-class classification of toxic comments. As part of this, we studied the word embeeddings generated by tf-idf, Word2Vec GloVe model. The word embeddings generated for individual tokens were averaged in order to generate the feature vectors for the processed comments. The feature vectors generated from each of the word embedding model was then used to fit several multi-class and multi-output classification algorithms, as part of this project we tested multinomial naive bayes, logistic regression, support vector machine, LSTM and decision tree based stochastic gradient descent and random forest classifier. The naive bayes classifier with tf-idf embedding was chosen to serve as a baseline for benchmarking the rest of the candidate models.

3.1 Word Vector Representation

A basic word vector representation consists of building a sparse representation of the words in a document either in terms of their frequency or binary representation indicating the presence of a word in a document. A major drawback of this approach is its failure to capture any kind of semantic or contextual information from the documents. Word embeddings refer to dense representations of words in a low dimensional vector space where each word in the vocabulary gets an n-dimensional vector which can then be used to generate embeddings for the whole document through various statistical methods like simple mean, weighted averages etc. These models have been shown to be successful in capturing semantic context in the documents <insert ref> by being able infer the different meanings of the same word used in different contexts. This proves to be specifically powerful for our use case as the documents are composed of social media comments by users where similar words can be used to convey differential messages. Word embedding feature representation results in better classification accuracy for machine learning models as compared to binary of frequency based vectorizers where the models can get trained to look for dominant words to allot classes. Below, we a provide an overview and outcomes of word vector representation we used for the problem.

3.1.1 TF-IDF

TF-IDF refers to term frequency and inverse document frequency which is a statistical feature vector generation method intended to indicate how important a word is in a document or a corpus. The tf–idf value increases proportionally to the number of times a word appears in the document and is offset by the number of documents in the corpus that contain the word, which helps to adjust for the fact that some words appear more frequently in general, in this way the TF-IDF approach captures the relevancy among words, text documents and particular categories. Although tf-idf tries to overcome the problem of common terms in the document, it still suffers from some limitations such as it cannot account for the similarity between the words in the document since each word is independently presented as an index.This approach also fails for unseen words in the training corpus. For our use case, we decided to use tf-idf word vector representation to serve as our baseline as it is the simplest sparse representation that captures word importance in the corpus.

$$w_{ij} = tf_{ij} * log\frac{N}{df_i}$$

where,
N = Total number of documents

$$tf_{ij} = number\ of\ occurrences\ of\ word\ i\ in\ jth\ document$$

$$df_i = number\ of\ documents\ in\ which\ word\ i\ appears$$

3.1.2 Word2Vec

Word2Vec is one of the most popular techniques of unsupervised language modeling based on neural networks. Word2Vec creates vectors of the words that are distributed numerical representations of word features – these word features could comprise of words that represent the context of the individual words present in our vocabulary. The model gives two options to learn the features, namely continuous bag of words (CBOW) and skip-grams (sg). In the skip-gram approach, the aim is to predict context words from a given word in a sliding window of size s which is specified as a parameter for the model. For our use case, the best performance on the baseline classifier was achieved by using skip-grams with a window size of 10 and a down-sampling factor of 0.001 for infrequent words in the vocabulary to generate feature vector size of 250 for each word.

3.1.3 GloVe

GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, the general equation for GloVe feature vector is given below.

$$f(w_i - w_j, \hat{w}_k) = \frac{P_{ik}}{P_{jk}}$$

where,

$$w_i = \textit{word vector of word } i$$

$$P_{ik} = \textit{probability of seeing words i and k together in a corpus}$$

The word embeddings generated from the models were averaged using arithmetic mean to obtain vector representations for individual comments.

3.2 Tasks

With the document vector representation in place, the goal is to use these as features for training models to generate toxicity labels for the input comments.

3.2.1 Multi-Label Classification. The goal of multi-label classification task was to determine whether a comment is toxic or non-toxic and, if toxic, to determine what the given subtypes of toxic comments viz. identity hate, obscenity, threat and insult.

3.3 Evaluation Metrics

Looking into the task in hand we have to be careful and we want to make predictions with high recall but we have to keep in mind the precision as well. We have used macro averaged Precision, Recall and F1 score to evaluate all of our models.

$$\textit{Precision for a class}(Pi) = \frac{TP_s(i)}{TP_s(i) + FP_s(i)}$$

$$\textit{Recall for a class}(Ri) = \frac{TP_s(i)}{TP_s(i) + FN_s(i)}$$

$$\textit{Macro-Averaged Precision}(P_{M_A}) = \frac{\sum_i P(i)}{N}$$

$$\textit{Macro-Averaged Recall}(R_{M_A}) = \frac{\sum_i R(i)}{N}$$

$$\textit{Macro-Averaged F1 score}(F1_{M_A}) = N * \frac{P_{M_A} * R_{M_A}}{P_{M_A} + R_{M_A}}$$

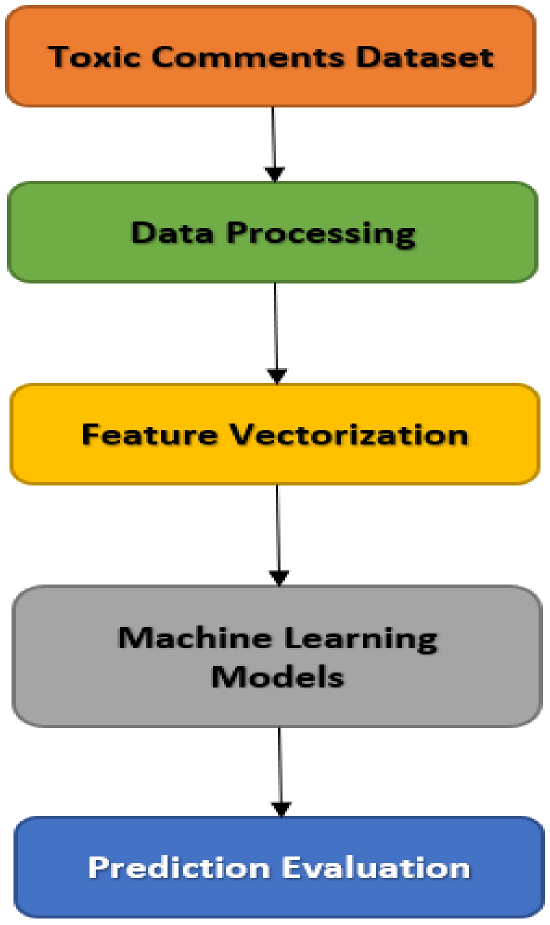


Figure 4. Process Flow Diagram

where,
TP = True Positives
FP = False Positives
FN = False Negatives
i = Tag class
N = Number of classes

Precision is the measure of quality, recall is the measure of quantity, F1 score is the harmonic mean of precision and recall, and Hamming loss tells us the fraction of wrong labels to the total number of labels.

4. Model

Given a comment, we have to predict the tags that can be assigned to it. Although it looks like a simple classification problem but it isn't one. Binary and multi-class classification has output values as (0,1) and (0,1,2,...,N) respectively. Unlike them, here each comment can have set of tags assigned to it, i.e. nth comment can have k number of tags assigned to it.

$$C_n = t_1, t_2, ..., t_k$$

To make such prediction we are building a multi-label classification model.

4.1 Baseline: Naive Bayes

For baseline, we have used Naive Bayes Classifier with One-vs-the-rest(OVR) strategy. OVR uses the binary relevance method to perform multi-label classification, which involves training one binary classifier independently for each label. The baseline model has a F1 score of 0.42, recall of 0.39 and a precision of 0.45.

4.2 Logistic Regression

To better our prediction, we have used Linear Classifiers with Stochastic Gradient Descent learning[passing alpha(constant that multiplies regularization), loss(log loss function to get probabilities) and penalty as parameters] and Logistic Regression separately[passing solver(liblinear), C(Inverse of regularization strength) and penalty as parameters] and compared the model performance.

Stochastic Gradient Descent is computationally fast, it allows minibatch learning, and for large datasets, it can converge faster as it updates its parameters frequently. Since the updates are frequent, they are computationally expensive. Logistic Regression is easy to implement can be easily extended to multiple label classification(that is what we need). Training a logistic regression on such high dimensional data is cheap and less inclined to over-fitting. However, a big setback of logistic regression is that it assumes a linear relationship between independent and dependent variable, also it requires independent variables to have average or no multicollinearity. Due to the complexity of modeling with OVR, using Random Forest and Support Vector Machine was not feasible. We have used L1 penalty(as the data is huge and sparse) for all models and validation performance on the test to prevent over-fitting.

4.3 Neural Network Model - Long Short Term Memory

Our usage of a Long Short Term Memory network was meant to combat that weakness by providing a model that could be unrolled across non-arbitrary comment lengths, analyzing comments word by word. For our LSTM model, we padded/clipped our input to standardize each input length. We experimented with various LSTM layerings and output dimensions. We use ReLU activations between the input and output layers, and our final output layers use sigmoid activation. We use an Adam optimizer for optimizing cross-entropy loss.

5 Model Performance

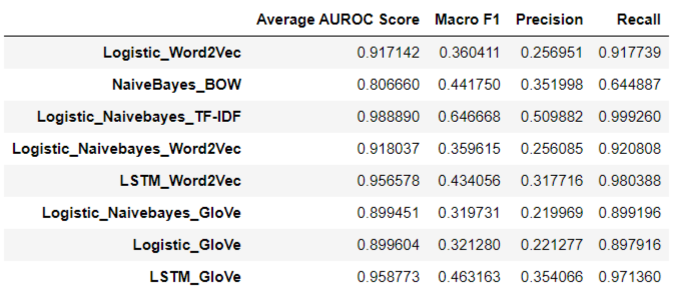
5.1 Train Test Split

We randomly split the 159K cleaned comments into train and test sets at 80:20 ratio. There were a total of data points in training and in test.

5.2 Model Prediction

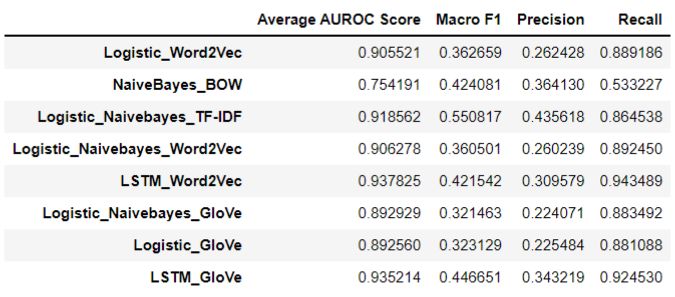
For a given comment, the model return a set of probabilities for each of the six different toxicity levels. Based on the proportion of each toxicity level in the train set we have come up with the breakeven point for each one of them. A predicted probability greater than the breakeven point indicates the comments is associated with that particular tag. The predicted probability of a tag is independent of the other except for severe_toxic. A comment can only fall under a severe_toxic category when it was found to be toxic. It is of utmost importance that we correctly identify the comments which have toxicity involved at the cost of precision as wrongly identifying non toxic comments under a toxicity label will not cause any harm to others but the opposite situation will.

Although recall was our main focus here but we needed to find a . The cutoff to assign one of the six toxicity labels to each comment was based on the proportion of each label in the training dataset. When looking into recall more each model, we saw a ensemble of naive bayes and logistic regression with the TF-IDF learned most on training set(Figure 5) but LSTM with GloVe performend best on the test set, Word2Vec performance was also very close. The precision for the ensemble of naive bayes and logistic regression with the TF-IDF was better amongst all model in both training and testing.



	Average AUROC Score	Macro F1	Precision	Recall
Logistic_Word2Vec	0.917142	0.360411	0.256951	0.917739
NaiveBayes_BOW	0.806660	0.441750	0.351998	0.644887
Logistic_Naivebayes_TF-IDF	0.988890	0.646668	0.509882	0.999260
Logistic_Naivebayes_Word2Vec	0.918037	0.359615	0.256085	0.920808
LSTM_Word2Vec	0.956578	0.434056	0.317716	0.980388
Logistic_Naivebayes_GloVe	0.899451	0.319731	0.219969	0.899196
Logistic_GloVe	0.899604	0.321280	0.221277	0.897916
LSTM_GloVe	0.958773	0.463163	0.354066	0.971360

Figure 5. Performance on train data



	Average AUROC Score	Macro F1	Precision	Recall
Logistic_Word2Vec	0.905521	0.362659	0.262428	0.889186
NaiveBayes_BOW	0.754191	0.424081	0.364130	0.533227
Logistic_Naivebayes_TF-IDF	0.918562	0.550817	0.435618	0.864538
Logistic_Naivebayes_Word2Vec	0.906278	0.360501	0.260239	0.892450
LSTM_Word2Vec	0.937825	0.421542	0.309579	0.943489
Logistic_Naivebayes_GloVe	0.892929	0.321463	0.224071	0.883492
Logistic_GloVe	0.892560	0.323129	0.225484	0.881088
LSTM_GloVe	0.935214	0.446651	0.343219	0.924530

Figure 6. Performance on test data

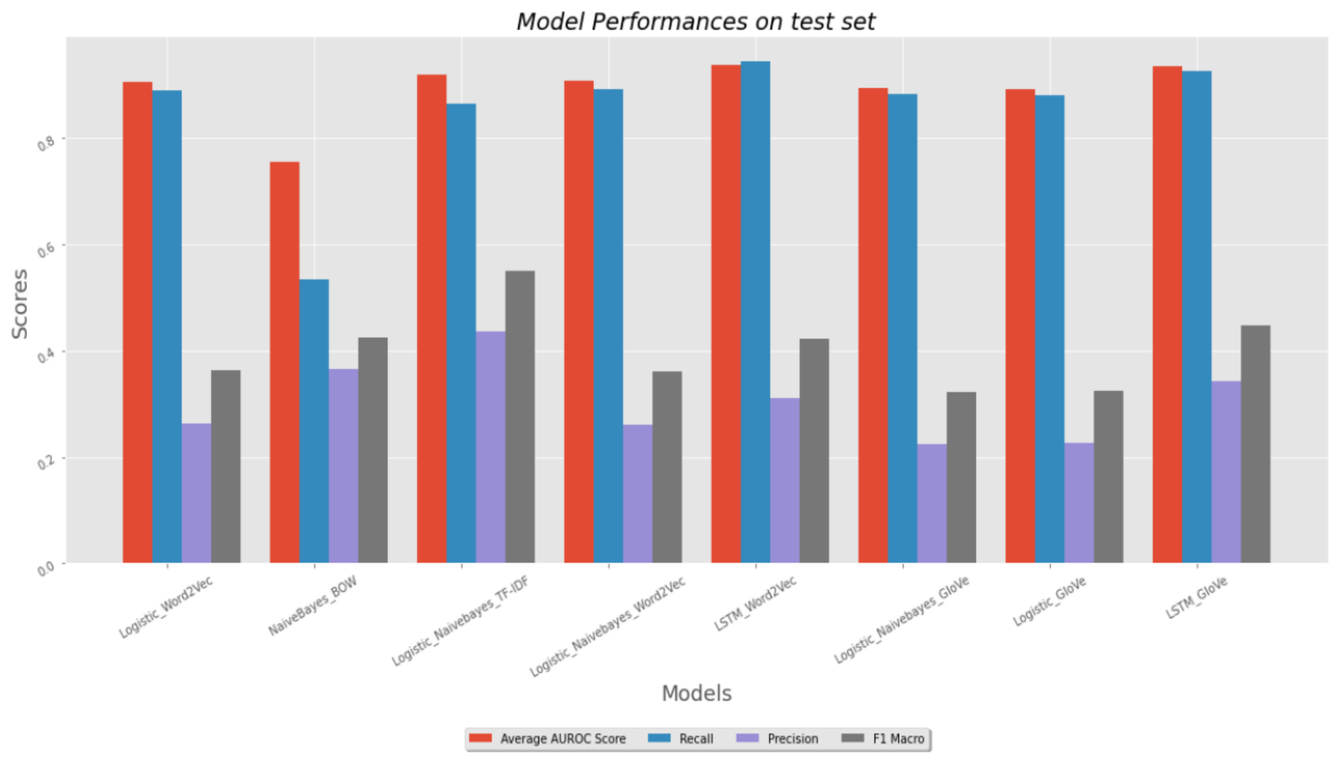


Figure 7. Performance of various models

5.3 Hyper Parameter Tuning

After drawing down on models to use, we used hyper parameter tuning to find the best fit model for the data we processed. Initially, we tweaked with the feature preparation process and prepared combination of uni-gram and bi-gram for TF-IDF. Similarly, we just involved the uni-gram feature in our model and checked the performance. We had even adjusted the vector size, minimum count and window within Word2Vec to extract features.

The feature space for TF-IDF, Word2Vec and GloVe was 357,244, 250 and 300 respectively. It was very costly to increase the feature space for GloVe and Word2Vec, while the performance enhancement was minimal. There was a similar scenario while tuning the models, tuning logistic regression by iterating over inverse of regularization strength was easy but tuning of LSTM by changing the number of hidden layers was very costly. Finally, we used logistic regression with a C value of 3 when combined with Naive Bayes and normally a C value of 23. Bidirectional LSTM was used with 50 hidden layers with an additional dense layer of 50 and a relu activation with the final layer of 6 output and sigmoid activation
.

6. Conclusion

In this paper, we described about how we solved a multi-label classification task to predicts tags to toxicity labels in comments. We made use of Gaussian Naive Bayes, Logistic Regression and Long Short Term Memory to solve the problem in hand.

If we consider only recall, we would choose LSTM with GloVe but even blindly predicting all comments to be toxic would give us 100% recall. As earlier mentioned we need to find better balance between precision and recall, so we choose to go with ensemble of Logistic Regression and Naive Bayes with TF-IDF. Although the recall here is 8% less than the LSTM one, the precision is better by over 12%. This might not seem very different but when we look in terms of actual number of comments they are vastly different because majority(over 80%) of the comments are non-toxic. Forget about different toxic labels, if we just look at whether or not the models detect any toxicity, we will get a better picture on the misclassification by both models and reason behind trying to find a balance between recall and precision.

Further improvements can be made to the models by using feature vectorization techniques such as Bert and using other deeper neural networks.

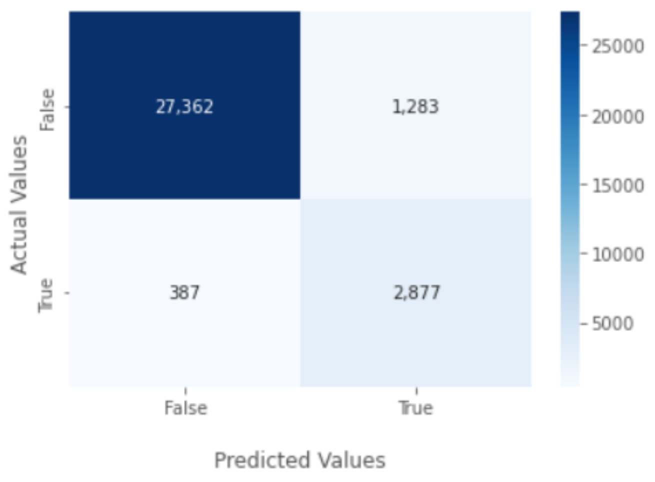


Figure 8. Confusion Matrix for ensemble of Logistic and Naive Bayes with TF-IDF

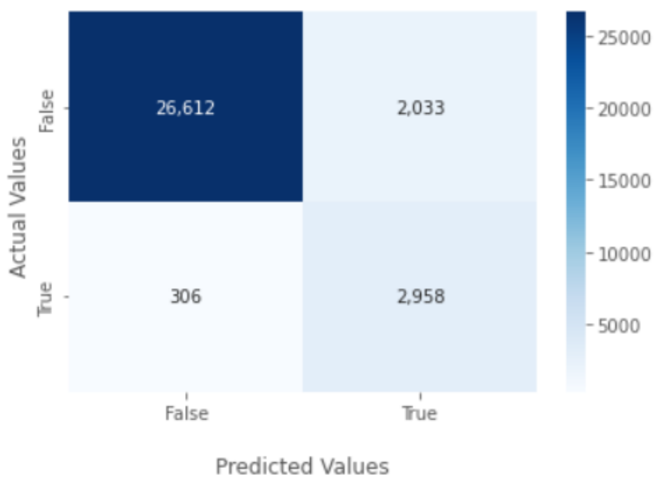


Figure 9. Confusion Matrix for LSTM with GloVe

References

[1] Yin, Dawei, et al. Detection of harassment on web 2.0. Proceedings of the Content Analysis in the WEB 2 (2009)
[2] Huy Nguyen Minh-Le.N. A Deep Neural Architecture for Sentence-level Sentiment Classification in Twitter Social Networking(2017)
[3] Chu,T Jue,K. Comment Abuse Classification with Deep Learning
[4] Nemanja Djuric, Jing Zhou, Robin Morris, Mihajlo Grbovic, Vladan Radosavljevic, and Narayan Bhamidipati. Hate speech detection with comment embeddings. In Proceedings of the 24th international conference on world wide web, pages 29–30, 2015.
[5] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. Amazonaws, 2018.
[6] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation.
[7] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed Representations of Words and Phrases and their Compositionality. In Proceedings of NIPS, 2013.
[8] Ramos, Juan. "Using tf-idf to determine word relevance in document queries." Proceedings of the first instructional conference on machine learning. Vol. 242. No. 1. 2003.
[9] J. B. Brown. "Classifiers and their Metrics Quantified". In: Molecular Informatics 37.1-2 (2018), p. 1700127.
[10] Juri Opitz and Sebastian Burst. Macro F1 and Micro F1. 2019. arXiv: 1911.03347