EE5180 : Introduction to Machine Learning

Project Report

Toxic Comment Classification

by
Group 15
Jaydev Joy(EE18B009), Anirudh Sagar Gollapalli(EE18B007)
Naga Deepthi Chimbili(EE18B043), Varshitha Vadapally(EE18B065)

1 Abstract

The anonymity that various social media services provide for their users, unfortunately brings out the worst in some people, which for most instances manifests itself as "toxic comments". This has attracted the attention of the creators of various social media platforms and websites. Moderating this manually is exceedingly impractical, given the tremendous pace at which the usage of social media services is increasing. Hence, a lot of social media giants are trying to automate this process of detecting toxic comments by implementing Machine Learning(ML) Algorithms. What we attempt to do in this project, is to come up with a classifier which can identify whether a comment is toxic, severely toxic, obscene, threat, insult, hate or non-toxic based on the level of toxicity of the comment under consideration.

The Conversation AI Team, a research initiative, founded by Jigsaw and Google put forth a data-set for this purpose by curating a large number of Wikipedia comments, which we are going to use for our classifier.

2 Introduction

Social media usage is growing at an exponential rate as one of the largest platforms for human conversations. Now, people are expressing themselves and their opinions more and also discuss things with others in these platforms. People have varied opinions and hence, conflicts might arise. If the conflicts take a wrong turn, it has a very high chance of turning into a verbal fight. The byproducts of these verbal fights are toxic comments. Toxic comments pose a risk of abuse and harassment. Due to the threat of toxic comments, many people think twice before expressing their opinion and do not do that at all most of the time, which leads to unhealthy discussion. In order to enforce healthy conversations, restriction of these negative interactions and behavior is necessary. Hence, there is a need for a solution that can be implemented on a large scale operation, which is where manual moderation fails. A solution for this problem is to implement a classification module for the various types of toxic comments. This can be used for automatic recognition of toxic comments which is very useful for moderators as well as for users who would want to filter unwanted contents. With all the progress and improvement in IT and data science, there is a requirement of a properly designed technique to find and isolate these kinds of comments that we call toxic, which is what motivated us to undertake this project.

First we would describe the Dataset. Then, we would touch upon some of the existing works on this task. After that we would explore the Dataset. Then, we would perform preprocessing and apply the clean Dataset on a few Machine Learning models. First, we would start with some simple models like Naive Bayes and Logistic Regression. Then, we would move on to more complex models like Random Forest Regression, Support Vector Machine(SVM). We also tried implementing a basic Long Short Term Memory(LSTM) but did not dwell deep into it and were not able to maximise the performance of the model. Finally, we would compare the results of all the models to determine which model might be the best to perform this classification.

3 The Dataset

The Conversation AI team is a research initiative by Google and Jigsaw are trying to work on tools that help to improve the fairness and health of online conversations. Studying negative online behaviour, like toxic comments(you know, comments that are disrespectful, rude and that make you feel to not be a part of that discussion) is a significant part of their vision. And they launched a Kaggle competition and titled it as Toxic Comment Classification Challenge. The main objective of this challenge was to develop a multi-label classifier that not only identifies the toxic comments but also detects the type of those toxic comments. Disclaimer: the dataset for this competition contains text that may be considered profane, vulgar, or offensive

The Data-set used for this task is sourced from that Kaggle competition and is split into training data and test data. It is composed of comments from Wikipedia's talk page edits. The training data-set consists of total 159571 instances with comments and corresponding multiple binomial labels: Severely Toxic, Toxic, Obscene, Threat, Insult, Identity Hate. Here are a few sample instances of the dataset :

	id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
0	0000997932d777bf	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0
2	000113f07ec002fd	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0
3	0001b41b1c6bb37e	"\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	0
4	0001d958c54c6e35	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0
5	00025465d4725e87	"\n\nCongratulations from me as well, use the ...	0	0	0	0	0	0
6	0002bcb3da6cb337	COCKSUCKER BEFORE YOU PISS AROUND ON MY WORK	1	1	1	0	1	0

4 Existing Works

Many researchers have attempted to find a good and efficient solution for this problem. Methods to solve this problem in the recent past were limited to Naive Bayes, Random Forest, Decision Tree. But, the recent advancements made in Machine Learning gave rise to much more efficient and complex methods. And so, new algorithms such as Convolution Neural Networks (CNN), Long Short-Term Memory (LSTM), etc. are being used now. Here is a list of papers and the approaches they used :

1. *Multilabel Toxic Comment Classification Using Supervised Machine Learning Algorithms.*
   Darshin Kalpesh Shah, Meet Ashok Sanghvi, Raj Paresh Mehta, Prasham Sanjay Shah, Artika Singh

   - 'The aim of this paper is to perform multi-label text categorization, where each comment could belong to multiple toxic labels at the same time. We tested two models: RNN and LSTM. Their performance is significantly better than that of Logistic Regression and ExtraTrees, which are baseline models.'

2. *Detecting Abusive Comments Using Ensemble Deep Learning Algorithms.*
Ravinder Ahuja, Alisha Banga, S C Sharma

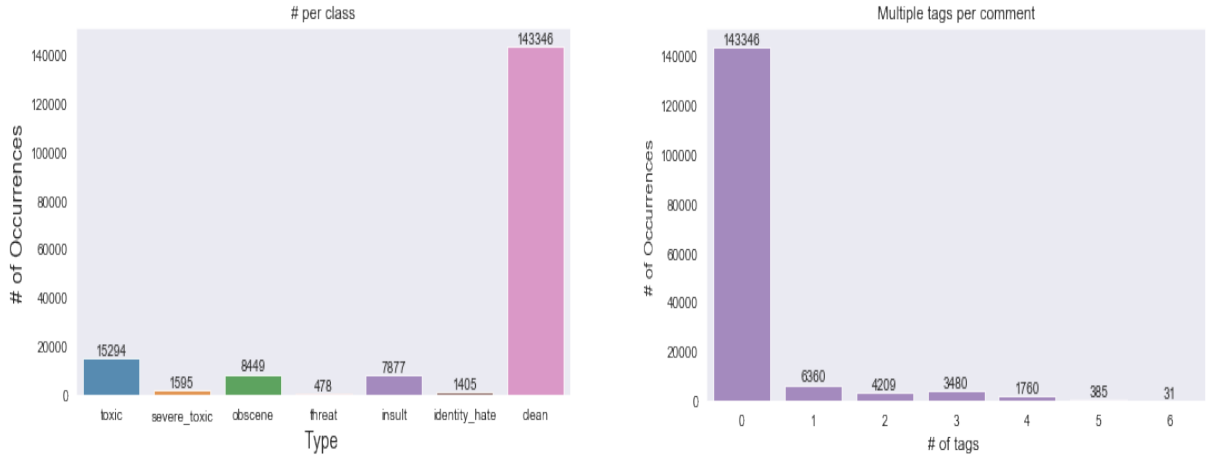
   - 'We have applied four classification algorithms: Naive Bayes, Random For-est, Decision Tree, and Support Vector Machine, with Bag of Words fea-tures. Deep learning algorithms: Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), and an ensemble of LSTM and CNN are applied using GloVe and fastText word embedding'

3. *Toxic Comment Classification.*
Sara Zaheri, Jeff Leath, David Stroud

   - 'Accordingly, aiming to find and develop an efficient algorithm to identifytoxic comments, the current study implemented several algorithms, includ-ing Naıve Bayes (as a benchmark), Recurrent Neural Network (RNN), and Long Short Term Memory (LSTM).'

5 Exploring the Data

Let's find the number of comments under each category and also see the number of categories per comment :



(a) Number of Comments in each category

(b) Number of categories per each comment

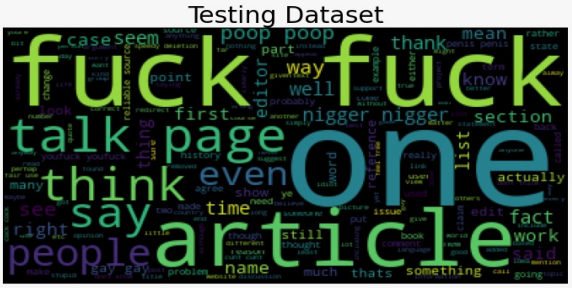
Figure 1

As we can see, most of the comments(143346) are clean and the category that has the least comments(478) is Threat. 15294 comments are Toxic, 8499 comments are Obscene, 7877 comments are Insult, 1595 comments are Severe Toxic and 1405 comments are Identity Hate. Since clean comments are not toxic, they don't have any tags. So, we see that most of the comments(143346) have zero tags. We can also see that 6360 comments have a single tag, 4209 comments have two tags, 3480 comments have three tags, 1760 comments have four tags, 385 comments have five tags and 31 comments have all the six tags.

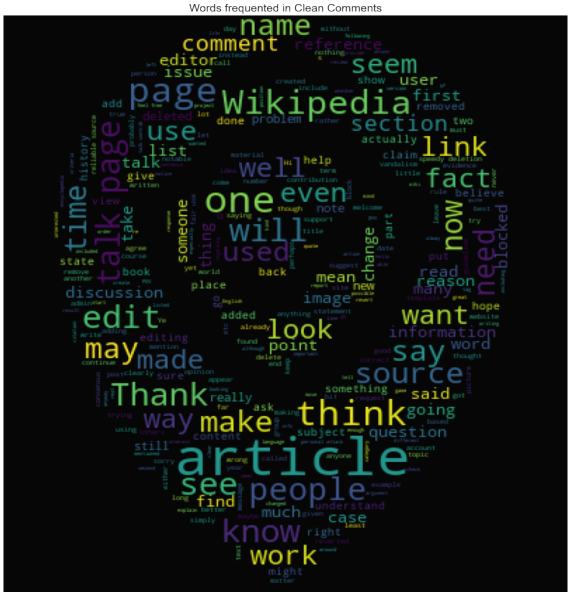
Now let's see the most frequent words occurring in the train and test dataset. We are dealing with toxic comments here, so it is inevitable that we would probably bump into some *bad* words. We have plotted these words using wordclouds, where the more frequent the word appears, the bigger it appears.

And also let's dig deep and see the most frequent words that occur in each category. We are specifically diving into toxic comments, so expect a lot of vulgar, cuss words and racial slurs.

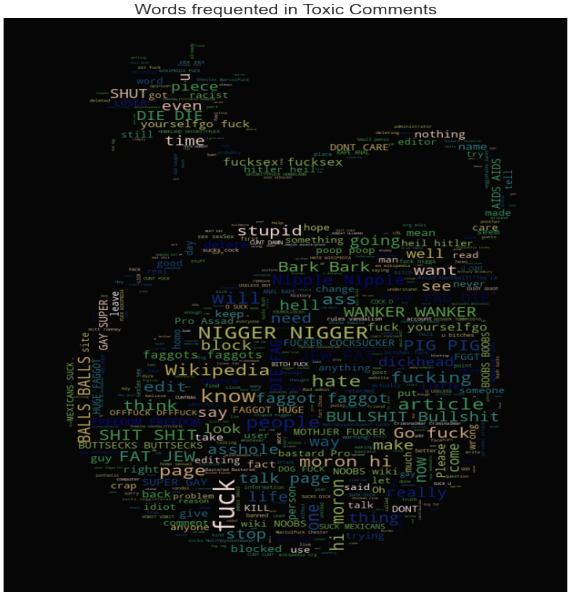
(a) Most frequent words in the Train Dataset



(b) Most frequent words in the Test Dataset



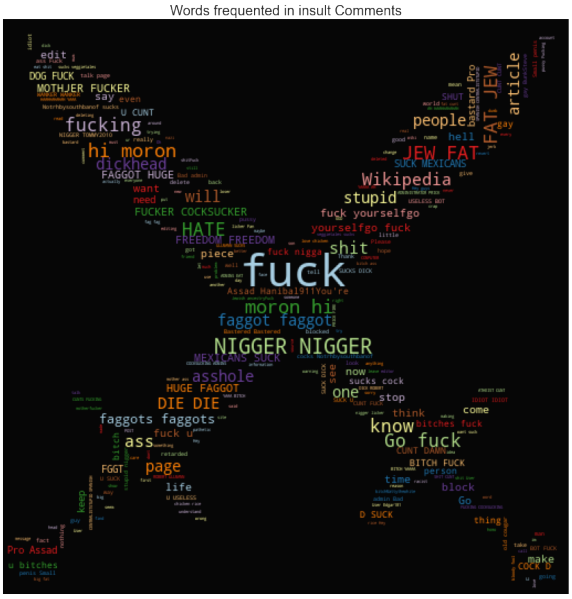
(a) Most frequent words in the Clean category



(b) Most frequent words in the Toxic category

(a) Most frequent words in the Severe Toxic category

(b) Most frequent words in the Obscene category



(a) Most frequent words in the Insult category

(b) Most frequent words in the Threat category

Let's plot the correlation matrix for all categories and see how related a category is to the other. If the correlation coefficient is 1, it means that one category can be calculated if we know the other. If the correlation coefficient is 0, it means that one category is independent of other. We saw that most of the non clean comments are toxic, so let's also plot confusion matrix of toxic with respect to other categories and see if we can gain some insight.

(a) Correlation Matrix of all categories

	severe_toxic		obscene		threat		insult		identity_hate	
severe_toxic	0	1	0	1	0	1	0	1	0	1
toxic										
0	144277	0	143754	523	144248	29	143744	533	144174	103
1	13699	1595	7368	7926	14845	449	7950	7344	13992	1302

(b) Confusion Matrix of Toxic wrt other categories

Figure 2

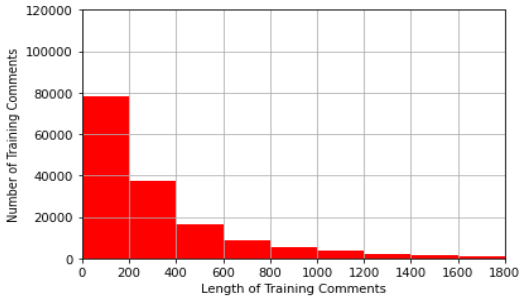
As we can see, categories are neither highly correlated(corr coeff>0.9) nor are they completely independent of each other(corr coeff<0.1). Now, coming to the confusion matrix, if we see the element where severetoxic=1 and toxic=0, we observe that it is zero. Which means that there are no instances(or comments) where that particular comment is severetoxic but is not toxic. So, if the comment is severe toxic, it has to be toxic. Whereas, for other categories we don't see observe any special cases like this.

6 Preprocessing

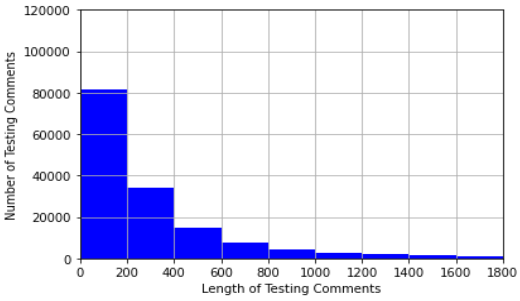
Since this is a text based dataset, there are a few specific set of preprocessing tasks that we should do so that we can refine the dataset and help the model perform better.

1. *Punctuation Removal*
    - We would have to remove white spaces, punctuations, etc. Because, they will be of no use to what the text is implying.
2. *Stopwords*
    - We should also remove stop words. Because, stopwords such as there, over, etc do not contribute to the context or meaning of the text.
3. *Lemmatization*
    - We should lemmatize the text. Because, words like changing, changed , changes etc. more or less mean the same thing, i.e change.

Let's plot the length of comments in both the train and test datasets before and after preprocessing to see how much preprocessing helps.

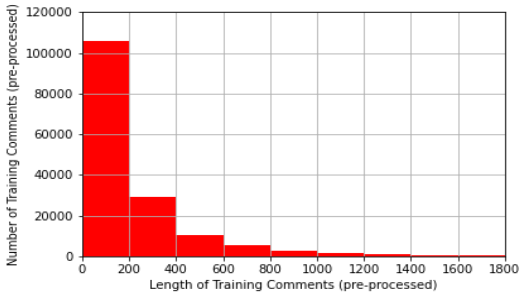


(a) Length of comments in the train dataset

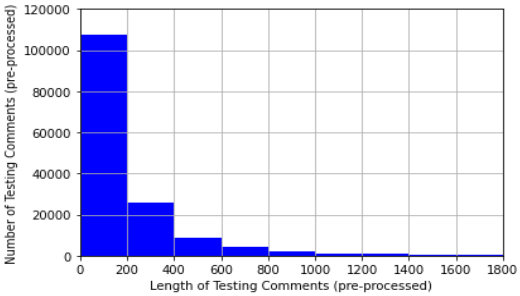


(b) Length of comments in the test dataset

Figure 3: Before Preprocessing



(a) Length of comments in the train dataset



(b) Length of comments in the test dataset

Figure 4: After Preprocessing

Since preprocessing filters, removes unwanted stuff, we can see that short length comments have increased and long length comments decreased.

7 Word Embedding

Machine learning models are designed to work with numbers. So, in order to solve text based problems, we need to represent the text numerically. Word Embedding is a technique where we transform our text data in to a sort of numerical vectors. There are numerous methods for performing Word Embedding. The most simple way is the Bag-of-Words approach.

Bag-of-Words :

Bag-of-Words is a technique where a bag is maintained which contains all the different words present in the corpus. For each and every word present in that corpus, counts of these words become the features for all the comments present in the corpus. We used this for the multinomial naive bayes. Let's see an example :

T1 = That is a good thing. That is also good.

T2 = And also that.

	that	is	a	good	thing	also	and
T1	2	2	1	2	1	1	0
T2	1	0	0	0	0	1	1

One-hot encoding :

For a few models, we need to vectorize our text in order to use that model. The most simplest way is by One-hot encoding. In One-Hot encoding, we represent each element in the transformed vector as 1 when the word is present and 0 when the word is absent. Let's take the previous example, the One-hot encoding of the word 'that' in T1 would be : [1,0,0,0,0,1,0,0,0,0]. For word 'and' in T2, it would be : [1,0,0].

TF-IDF Vectoriser:

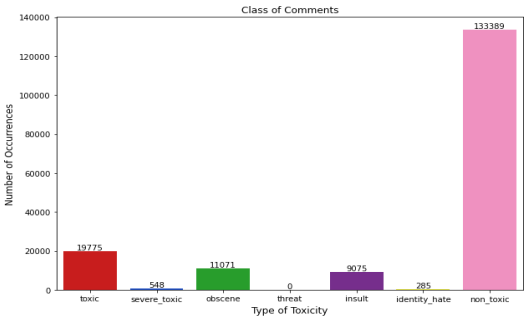
One-hot encoding might be the simplest way to vectorize text data, but it is not efficient. It treats all the words with same weightage, when we know article words like 'the','a',etc. do not give any sort of information. So, we need to also weight our words. We used TF-IDF vectoriser for all our remaining models(Logistic Regression, SVM, Random Forest and LSTM). The featured Bag-of-Words Model or Matrix for the whole corpus is transformed into a matrix whose every element is a product of Term Frequency (TF) and Inverse Document Frequency (IDF), combined together as tf-idf.

Let's take an example, TF : Term frequency for a word is determined by the number of times it occurs on the whole corpus. So, words like 'the' have a very high TF. Now, let's say a few documents in the corpus contain the word 'Federer'. So, we can say that it is highly likely that the corpus is about Tennis. Let's say we find a very rare word in a document in the corpus: 'discombobulated' which does not give much info. Eg : TF('the') = 100, TF('Federer') = 5, TF('discombobulated') = 1. So, we need to assign high weightage to the 'Federer' and less weightage to 'discombobulated' and even less to 'the'. This is where IDF term helps. IDF for a word is calculated as log of number of docs in the corpus divided by the number of docs in which the 'word' appears. Eg: IDF('the') = log(10/10) = 0 , IDF('Federer') = log(10/3) = 0.52, IDF('discombobulated') = log(10/1) = 1. The final weights are calculated by multiplying the

TF and IDF terms. Ie. TFIDF = TF x IDF. The final weights would be : ‘the’ = 100 x 0 = 0, ‘Federer’ = 5 x 0.52 = 2.6, ‘discombobulated’ = 1 x 1 = 1. After calculating these terms, they are normalised and used as weights in their vector representation.

8 Results

Now, we are going to list all the results from all the models namely each confusion matrix, count of different categories and accuracies.

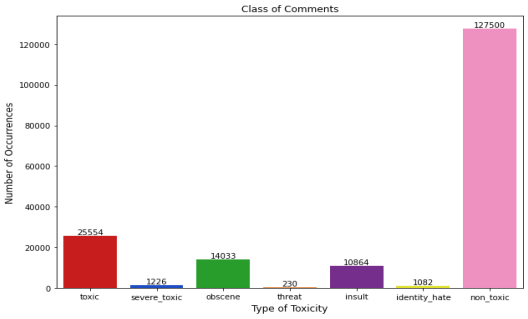


(a) Number of comments in each category of the output

toxic		severe_toxic		obscene		threat		insult		identity_hate	
35921	157	39460	27	37641	109	39788	0	37690	192	39529	7
1862	1953	340	66	1032	1111	105	0	1143	868	337	20

(b) Confusion Matrix

Figure 5: Naive Bayes Model

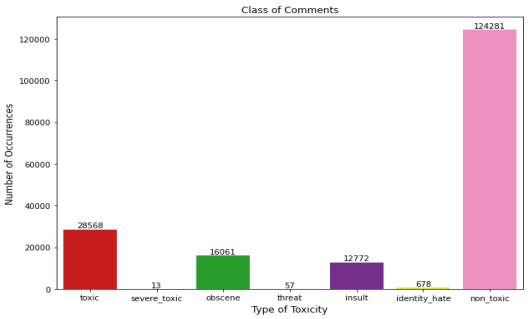


(a) Number of comments in each category of the output

toxic		severe_toxic		obscene		threat		insult		identity_hate	
35803	275	39415	72	37625	125	39781	7	37662	220	39512	24
1492	2323	302	104	815	1328	89	16	992	1019	294	63

(b) Confusion Matrix

Figure 6: Logistic Regression Model

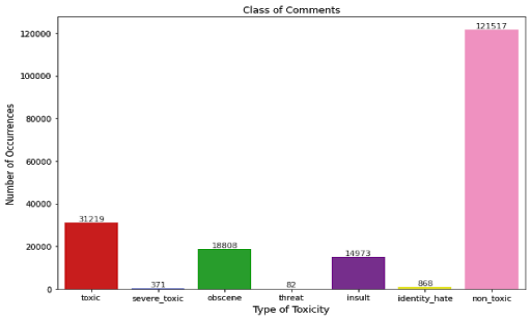


(a) Number of comments in each category of the output

toxic		severe_toxic		obscene		threat		insult		identity_hate	
35763	315	39485	2	37581	169	39786	2	37589	293	39526	10
1393	2422	402	4	720	1423	96	9	899	1112	314	43

(b) Confusion Matrix

Figure 7: SVM Model



(a) Number of comments in each category of the output

toxic		severe_toxic		obscene		threat		insult		identity_hate	
35535	510	39436	42	37470	296	39757	7	37435	418	39529	20
1374	2474	381	34	567	1560	119	10	897	1143	290	54

(b) Confusion Matrix

Figure 8: Random Forest Model

Model	toxic	severe_toxic	obscene	threat	insult	identity_hate
Naïve Bayes	0.949389	0.990800	0.971398	0.997367	0.966535	0.991376
Logistic Regression	0.955706	0.990624	0.976436	0.997593	0.969618	0.992028
SVM	0.957185	0.989872	0.977715	0.997543	0.970120	0.991878
Random Forest	0.952773	0.989396	0.978367	0.996841	0.967036	0.992229

(a) Accuracy in each category of each model

Model	Final Accuracy
Naïve Bayes	0.8707
Logistic Regression	0.8843
SVM Classifier	0.8871
Random Forest	0.8821
LSTM	0.9739

(b) Final accuracy of all the models

Model	Computation Time
Preprocessing	3m 27s
Naïve Bayes	1m 2s
Logistic Regression	1m 11s
SVM	49m 49s
Random Forest	19m 52s
LSTM	34m 48s

(c) Computation time of each process

Figure 9: Accuracies and Computation times

Every model has classified most comments as clean, next most comments as toxic, as expected. But, note that bayes model did not identify any comments as threat. This could possibly due to the threat comments being too low in our train dataset(as we saw in fig.1a) and Naive Bayes being baseline model was not enough to learn with small threat data. We can see that other models did not perform as bad as Naive Bayes. We can see the final accuracy table(fig.9b) that the most accurate model is LSTM and the least accurate model is Naive Bayes(as expected) and the remaining three models' accuracies are around same. But, when we talk about model performance, we also need to consider the computation time. SVM(49min) and LSTM(34min) have very high computation times. Random Forest(19min) has a high computation time as well. Logistic Regression(1min) has a very low computation time and also has good accuracy. So, Logistic Regression seems to be our best bet.

9 Conclusion

Some of the important insights that we developed by executing this project are listed below :

- Since the testing data-set did not have the correct labels associated with them (as expected in a practical test data-set), there was no way to compute the final accuracy of the classifier, by training the classifier from the entire training data-set.
- For the purpose of calculating final accuracy and the confusion matrix, the training data-set was split into train data (0.75) and test data (0.25).
- But this act of splitting the training data-set is not preferred here due to the non-uniform distribution of the different categories of toxicity. Splitting the data-set could result in the loss of information regarding the distribution of the nature of toxicity. It is always preferred to train the classifier from the entire training data-set for best results.

- The high accuracies received from even simple classification models is due to the non-uniform distribution of the different categories of toxicity, and not due to the good performance of the classifier.
- Models that are implemented from scratch often results in high computation time, due the large size of the data-sets. Hence it is preferred to use the in built library functions. (Execution of our Naive Bayes model which was developed from scratch took more than 2 hours and hence the same idea was dropped due to the high computation time involved)
- Hyper-parameter tuning was also not preferred due to the large size of the data-set, resulting in high computation time.

10 Further Developments

- It is said that Neural Networks and LSTM models give much better results. Although we were able to implement a basic LSTM model, due to the time limitation, we were not able to dwell deep into LSTM and Neural Networks.
- The task of toxic comment classification could be combined with computer speech recognition to expand the application areas of this classification. A few possible application areas would be : Censorship, plagiarism detection, creating a scoring system for debate competitions, implementing parental controls for the content in the internet etc.
- Taking a step back from toxic comments, the underlying principle of text classification can be used for more complex classifications like, classifying books into their appropriate genres, based on the contents of the book, sentiment analysis, classification of news articles into their appropriate categories etc.

11 References

1. Multi-label Toxic Comment Classification Using Supervised Machine Learning Algorithms
2. Detecting Abusive Comments Using Ensemble Deep Learning Algorithms
3. Abusive Comments Classification in Social Media Using Neural Networks
4. Toxic Comment Classification
5. Convolutional Neural Networks for Toxic Comment Classification
6. Toxic Comment Classification