Toxic Comment Classification

EE5180 : Introduction to Machine Learning

Anirudh Sagar Gollapalli (EE18B007), Jayadev Joy (EE18B011), Naga Deepthi Chimbili (EE18B043), Varshitha Vadapally (EE18B065)

Introduction

- The social media is a unique place where people can express themselves without compromising their identity. But unfortunately, there are some people who misuse this freedom to spread negativity. Such a situation leads to a toxic environment and should be dealt with in order to enforce safe and healthy conversations. Taking the size and scale of the social media, it is not practical to use manual moderation. Which is why social media giants use Machine Learning to automate this process.

- We aim to build a classifier that detects the level of toxicity for a comment which would be useful to detect toxic comments and hence can be removed.

Problem Statement

In this Project, what we attempt to do is to use different classifier models, which predicts the probability of each category of toxicity for each comment from the dataset, and appropriately label the comments with the corresponding type of toxicity. We will compare the accuracy score and the time taken for different classifiers. This is a multi-label classification problem, where one comment can belong to more than one label simultaneously. For example, a comment maybe toxic, obscene and insulting at the same time. A comment can also be non-toxic and hence does not belong to any of the six labels.

Data-Set Description

The Data-set used for this task is sourced from a Kaggle competition and is split into training data and test data. It is composed of comments from Wikipedia talk page edits.

The training data-set consists of a total of 159,571 instances with comments and the corresponding multiple binomial labels:

- Toxic
- Severely Toxic
- Obscene
- Threat
- Insult
- Identity Hate

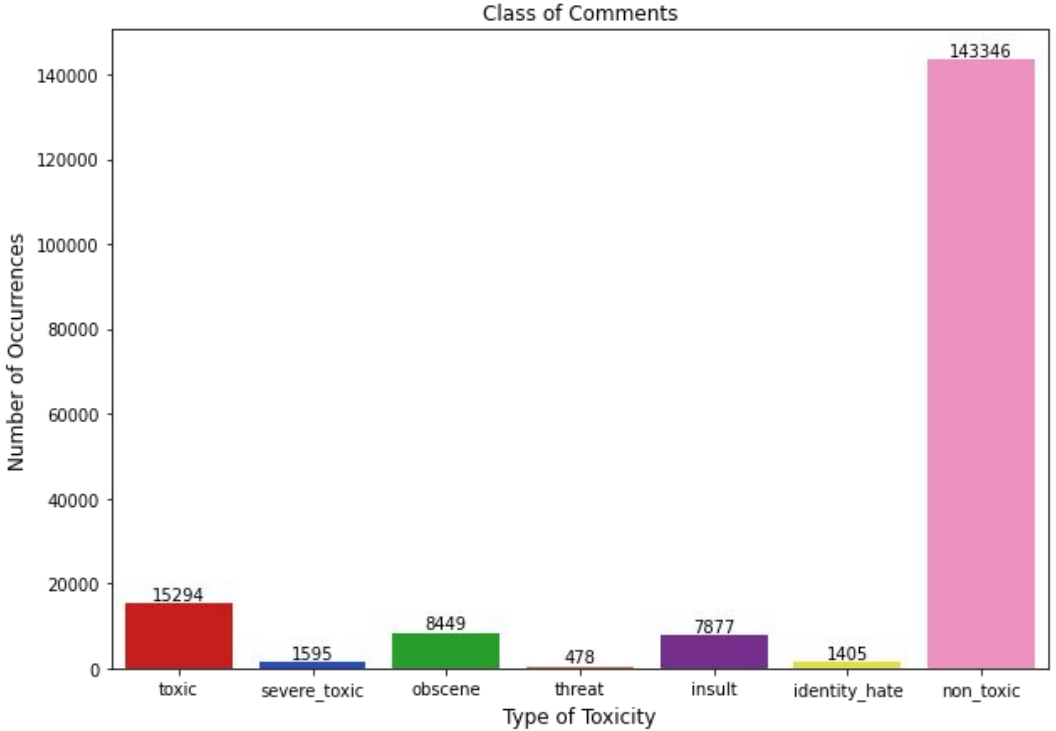
Data-Set Description

Sample instances of the data-set are shown below in the figure :

	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

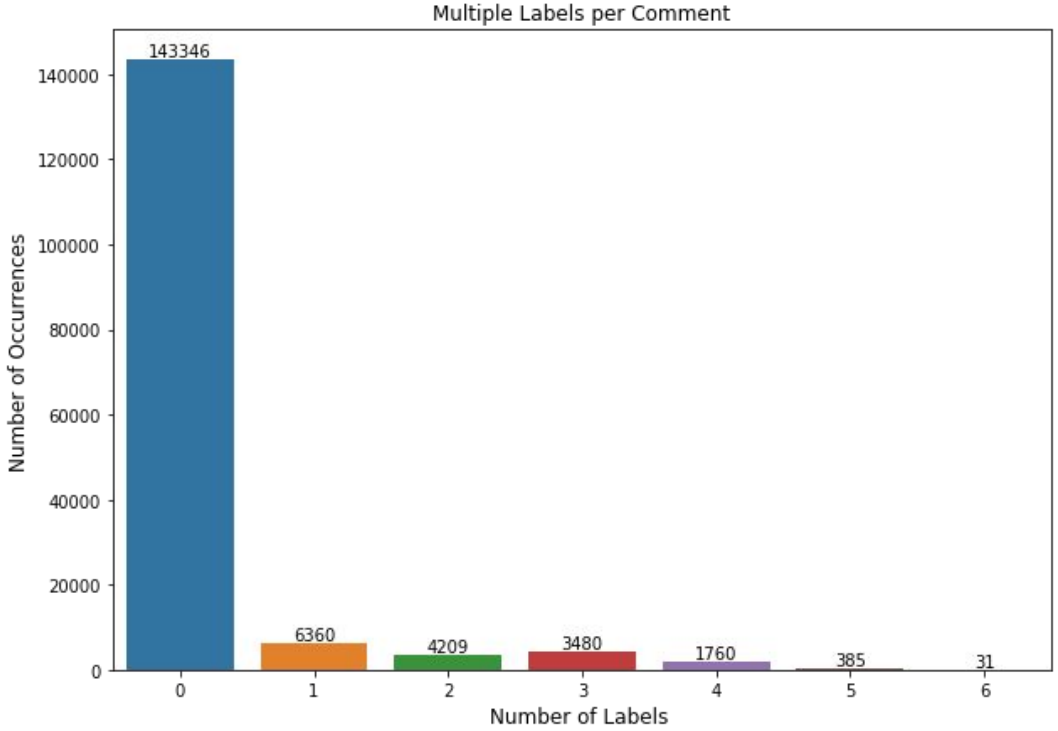
Exploring The Data

Visualizing the count of different categories of toxicity of comments :



Exploring The Data

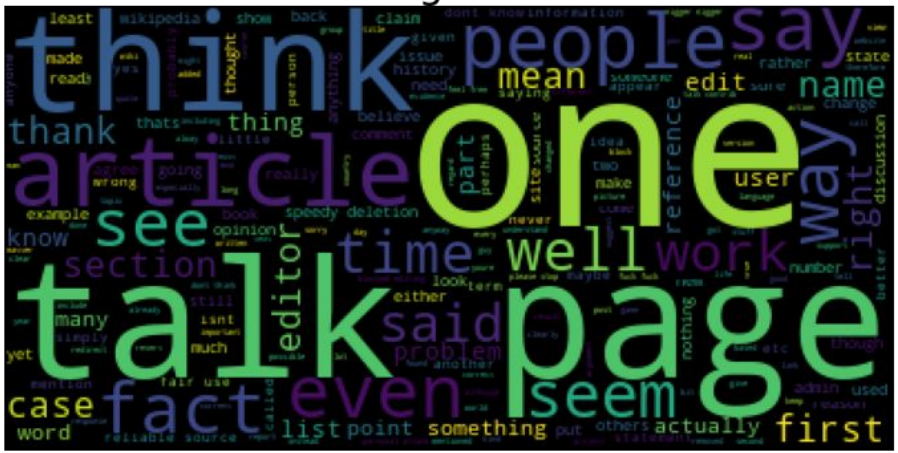
Visualizing the count of different number of labels per comment :



Exploring The Data

Visualizing the frequently appearing words in the dataset using Wordcloud :

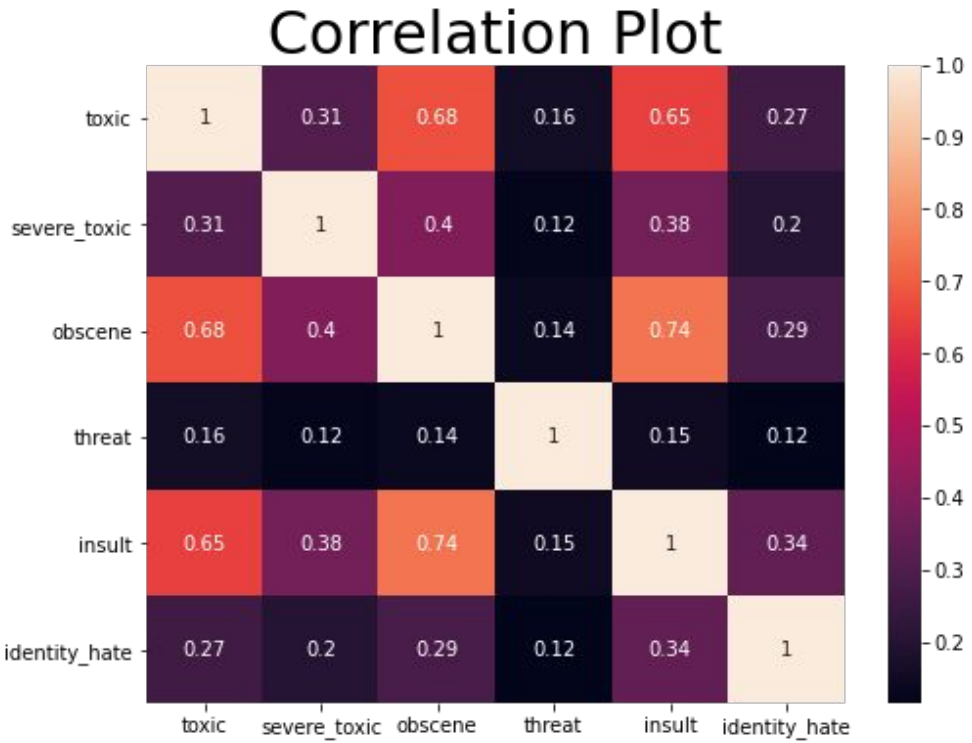
Training Dataset



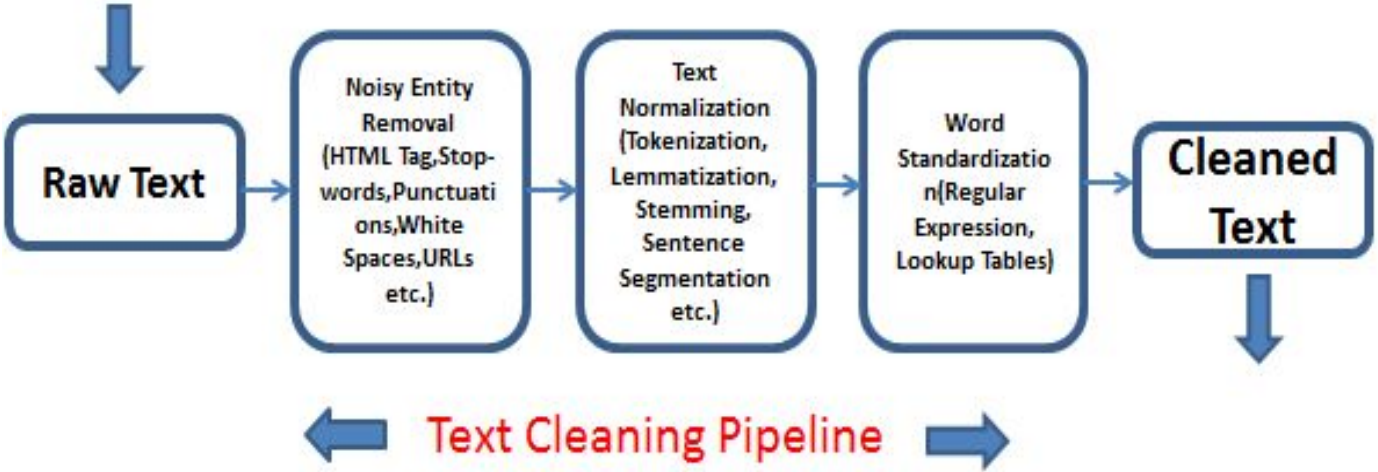
Testing Dataset

Exploring The Data

The Correlation Plot for the training dataset is displayed below :

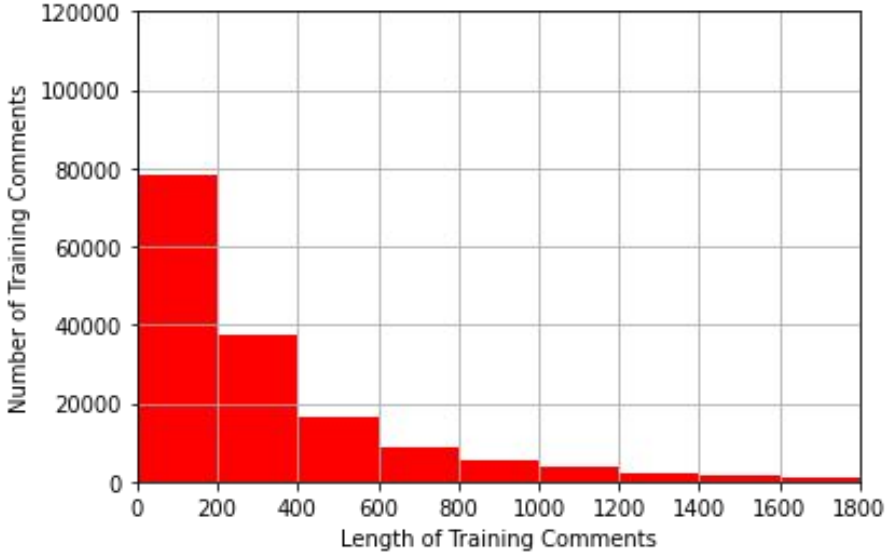


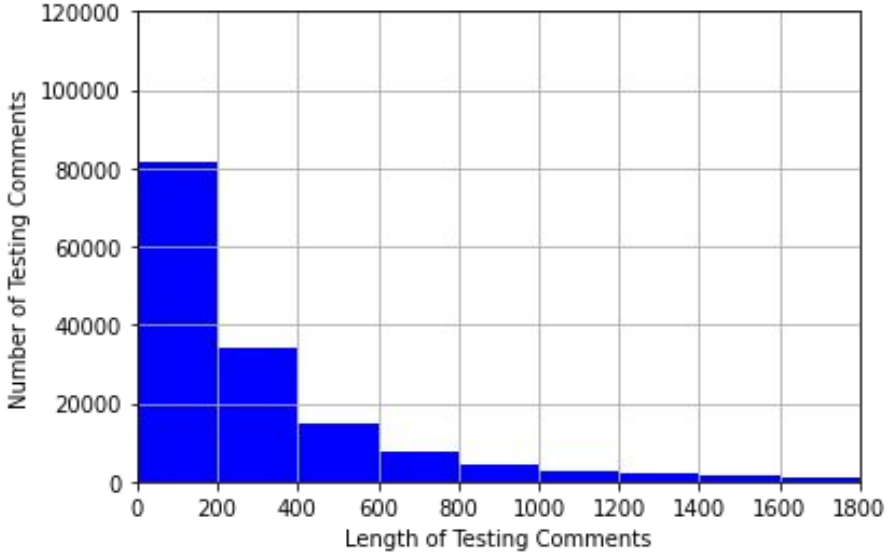
Text Preprocessing



Exploring The Data

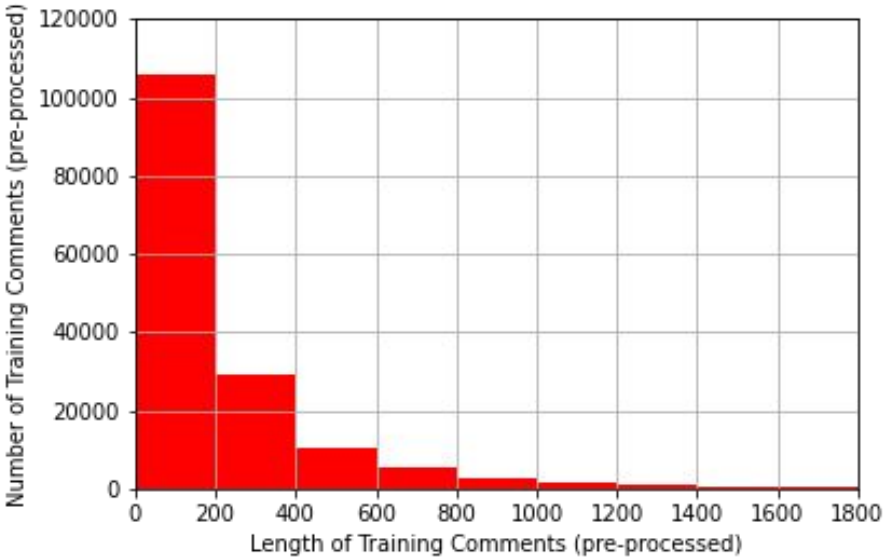
Visualizing the length of comments in the dataset :

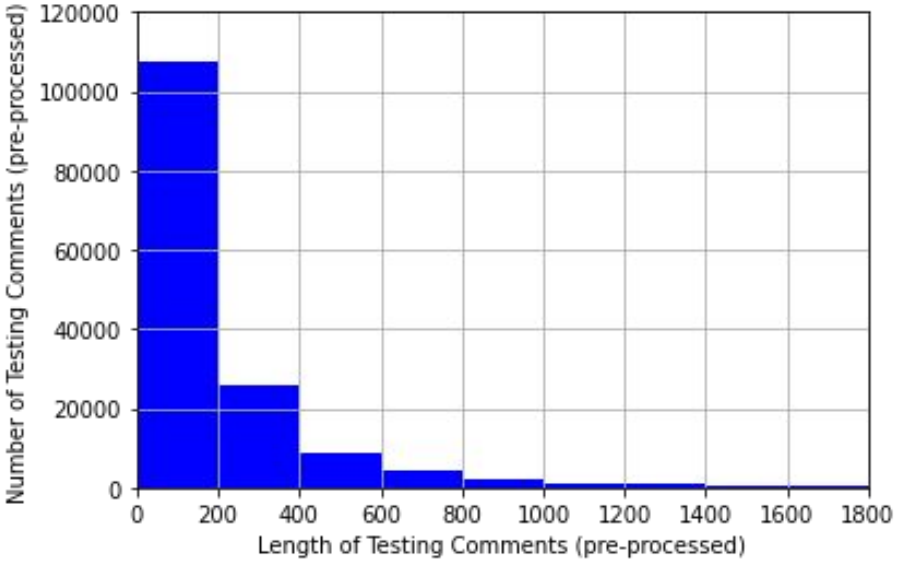




Text Preprocessing

Visualizing the length of comments in the dataset after preprocessing :





Word Embedding

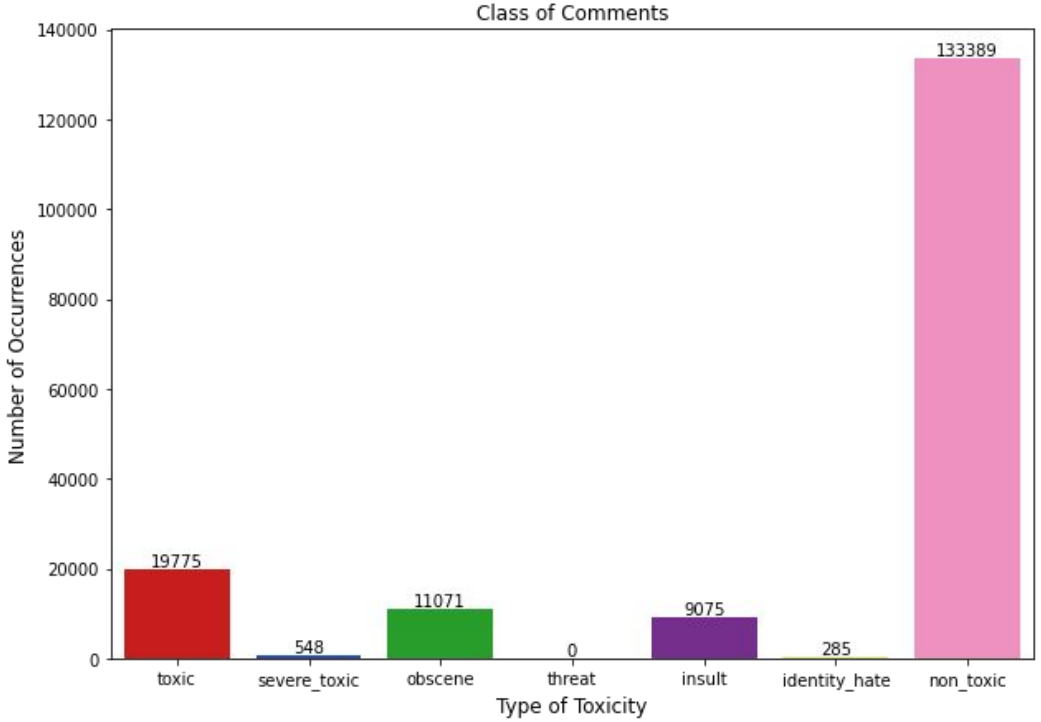
- Machine learning models are designed to work with numbers.
- So, in order to solve text based problems, we need to represent the text numerically.
- Word Embedding is a technique where we transform our text data in to a sort of numerical vectors.
- There are numerous methods for performing Word Embedding.
- The most simple way to represent text as numbers is through 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 an example: 'The weather is nice today'. Here, vector representation of the word 'nice' is: [0,0,0,1,0]

TF-IDF Vectorizer

- TF abbreviates to Term Frequency and IDF abbreviates to Inverse Document Frequency.
- 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 probable that the corpus is about Tennis. Let's say we find a very rare word 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.

Naïve Bayes Classifier

Visualizing the count of different categories of toxicity of comments as determined by Naïve Bayes Classifier :



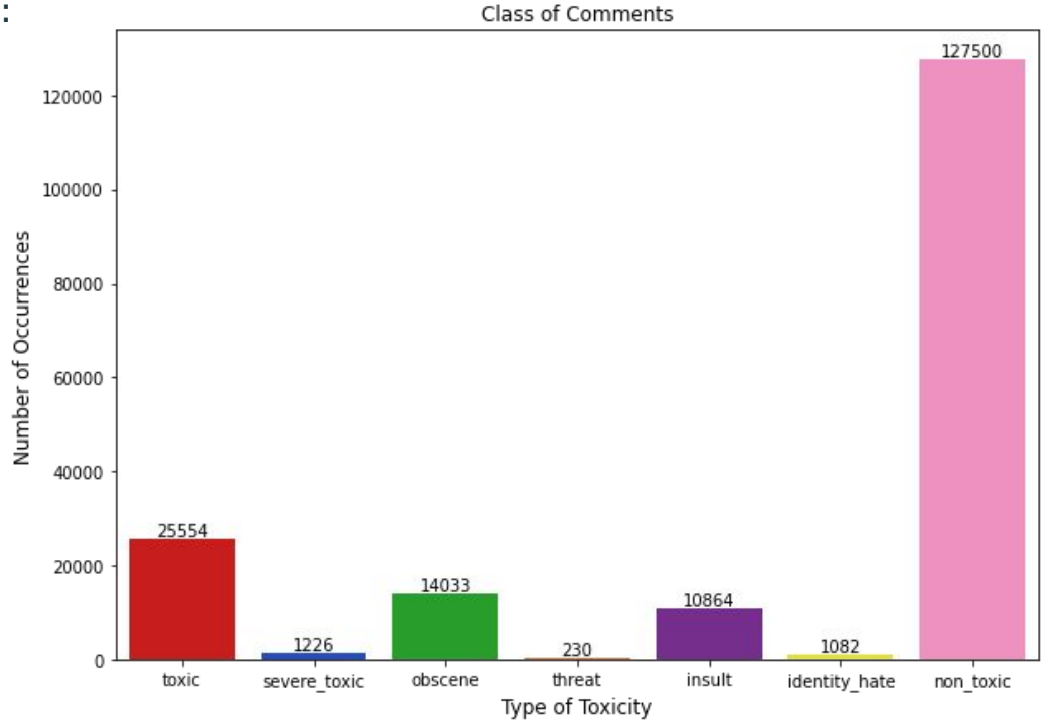
Naïve Bayes Classifier

The Confusion Matrix for the different toxic features for the Naïve Bayes Classifier is listed below :

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

Logistic Regression Classifier

Visualizing the count of different categories of toxicity of comments as determined by Logistic Regression Classifier :



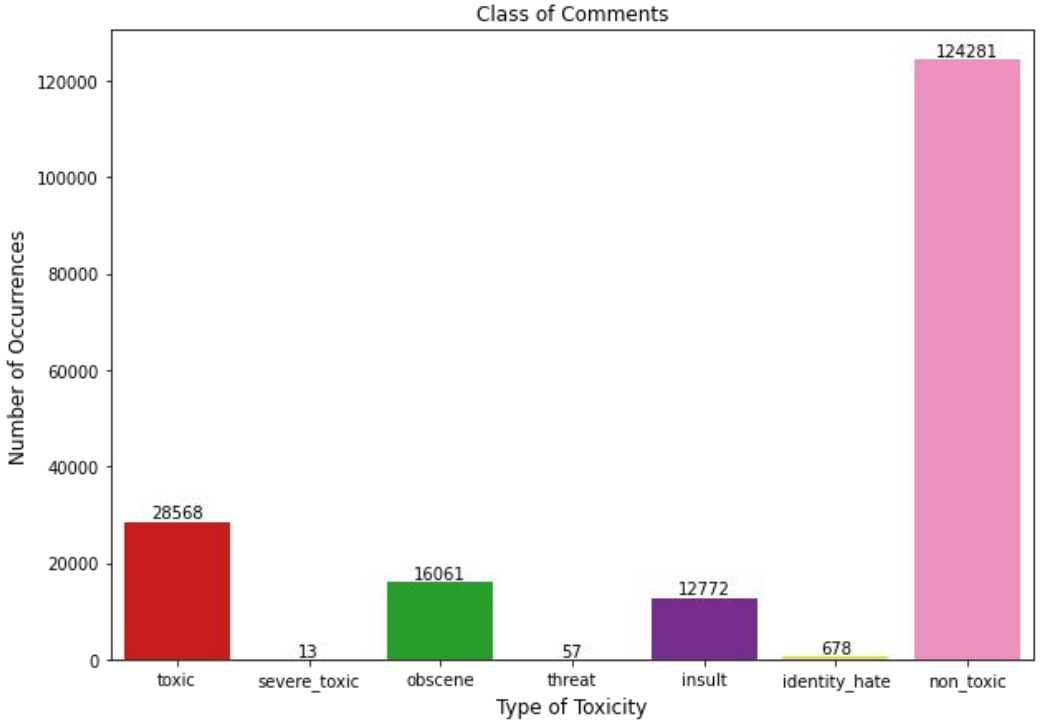
Logistic Regression Classifier

The Confusion Matrix for the different toxic features for the Logistic Regression Classifier is listed below :

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

SVM Classifier

Visualizing the count of different categories of toxicity of comments as determined by SVM Classifier :



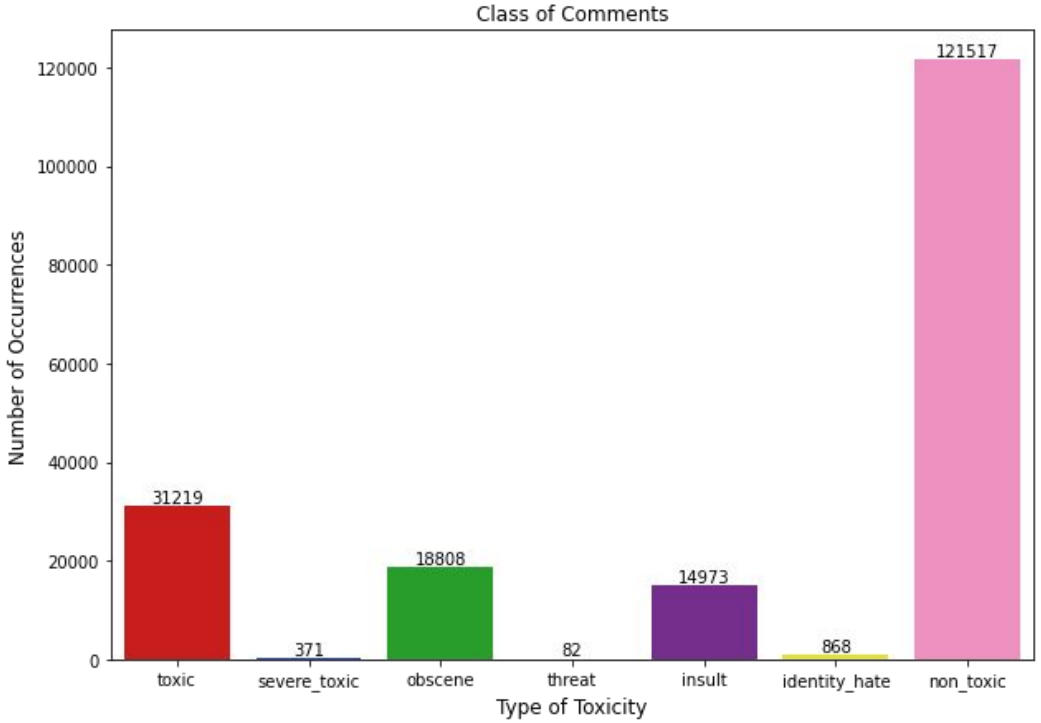
SVM Classifier

The Confusion Matrix for the different toxic features for the SVM Classifier is listed below :

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

Random Forest Classifier

Visualizing the count of different categories of toxicity of comments as determined by Random Forest Classifier :



Random Forest Classifier

The Confusion Matrix for the different toxic features for the Random Forest Classifier is listed below :

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

Computation Time

The Computation time taken for different processes/models are listed below :

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

Final Accuracy

The Final Accuracy for each feature under different models is listed below :

	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

Final Accuracy

The Final Accuracy for different models is listed below :

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

Further Developments

- Currently we are working on implementing Neural Networks and LSTM to the above problem, so as to improve the performance of the classification.
- 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.

References

- Multilabel Toxic Comment Classification Using Supervised Machine Learning Algorithms
- Detecting Abusive Comments Using Ensemble Deep Learning Algorithms
- Abusive Comments Classification in Social Media Using Neural Networks
- Toxic Comment Classification
- Convolutional Neural Networks for Toxic Comment Classification
- Toxic Comment Classification

Thank You