TOXIC COMMENT CLASSIFICATION

Manish Marwah
Department of Computer Science
Santa Clara University
mmarwah@scu.edu

Coauthor
Nirali Patel
NRPatel@scu.edu

Coauthor
Soumya Parvatikar
SParvatikar@scu.edu

Coauthor
Priyanka Advani
PAdvani@scu.edu

Abstract

Discussing things we care about can be difficult. The threat of abuse and harassment online means that many people stop expressing themselves and give up on seeking different opinions. This paper proposes a comparison of different models for multi-label text classification, and compare them on basis of their f-scores, with a resulting classification learning accuracy of 96.6%. Github link of code is https://github.com/PriyankaAdvani/Toxic-Comments-Classifier

1 Introduction

Keeping online conversations constructive and inclusive is a crucial task for platform providers. Automatic classification of toxic comments, such as hate speech, threats, and insults, can help in keeping discussions fruitful. In addition, new regulations in certain European countries have been established enforcing to delete illegal content in less than 72 hours..

This is a multi-label classification problem. The different classes are:

- toxic
- severe_toxic
- obscene
- threat
- insult
- Identity_hate
- clean

Difference between multi-class classification & multi-label classification is that in multi-class problems the classes are mutually exclusive, whereas for multi-label problems each label represents a different classification task, but the tasks are somehow related.

In multi-label classification, data can belong to more than one label simultaneously. For example, in our case, a comment may be toxic and insulting at the same time. It may also happen that the comment is non-toxic and hence will be classified as clean.

1.1 Motivation

Maintaining a decorum on an online forum is the need of the hour. Defined as "willful and repeated harm inflicted through the medium of electronic text", cyberbullying puts targets under attack from a barrage of degrading, threatening, and/or sexually explicit messages and images conveyed using websites, instant messaging, blogs, chat rooms, cell phones, websites, email, and personal online profiles. Thus, the task of identifying and removing toxic communication from public forums is critical and is infeasible for human moderators.

1.2 Problem Statement

Text arising from online interactive communication hides many hazards such as fake news, online harassment and toxicity. Toxic comment is not only the verbal violence but a comment that is rude, disrespectful or otherwise likely to make someone leave a discussion.

Toxic comment classification is a multi-label text classification problem with a highly imbalanced dataset. Toxic comment classification is about detection of different types of toxic comments by classifying them into one or more labels from 'toxic', 'severe_toxic', 'obscene', 'threat', 'insult', 'identity_hate' and 'clean'. We need to create a model which predicts a probability of each type of toxicity for each comment.

1.3 Challenges

The challenge is particularly interesting because of the heated discussions that toxic content online has influenced the overall health of the society. It is also interesting that the service providers are finally leveraging deep learning to supervise their service in a scalable way.

1.4 Related Work

https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge

2 Dataset

train.csv - The training set, contains comments with their binary labels

Size: 160k x8

Columns: id, comments, toxic, severe_toxic, obscene, threat, insult, identity_hate

test.csv - The test set, predict toxicity probabilities for these comments.
Size: 153k x 2

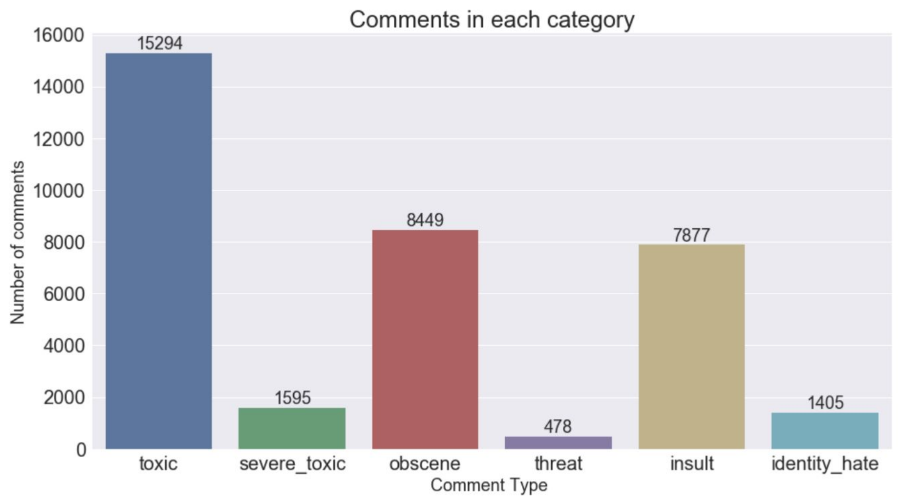
Columns: id, comments

3 Solutions and Result

3.1 Exploratory Data Analysis (EDA):

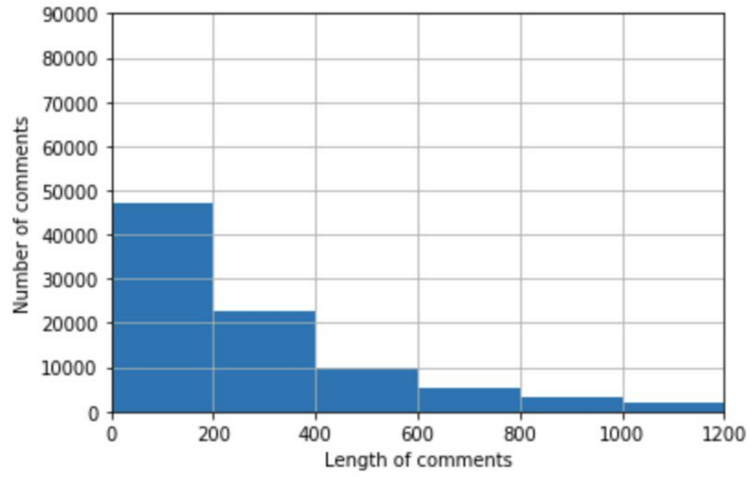
Exploratory Data Analysis is one of the important steps in the data analysis process. Here, the focus is on making sense of the data in hand — things like formulating the correct questions to ask to your dataset, how to manipulate the data sources to get the required answers, and others.

1. We counted the number of comments under each label. Because some classes do not have enough number of data points the f-score for those classes will be very low. We still evaluate those classes to prove that the model cannot learn well from a low number of data points.

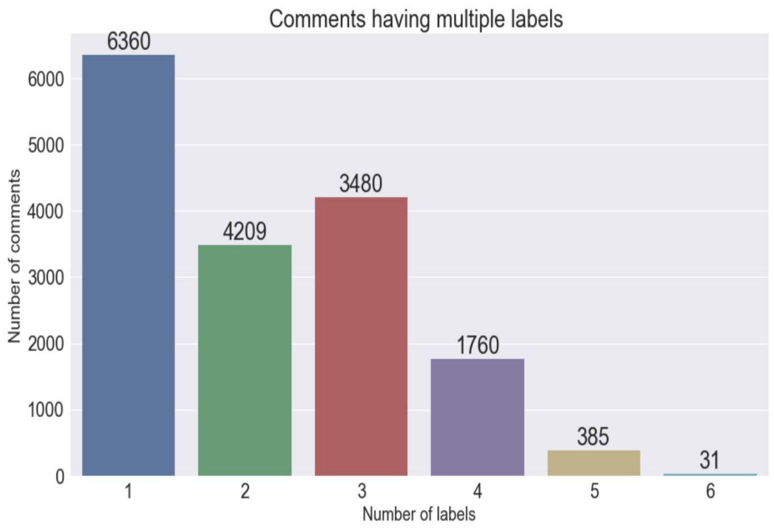


2. From this visualisation, we observed that comments were of varying lengths from less than 200 characters to 1200 characters. The majority of comments had a length up to 200.

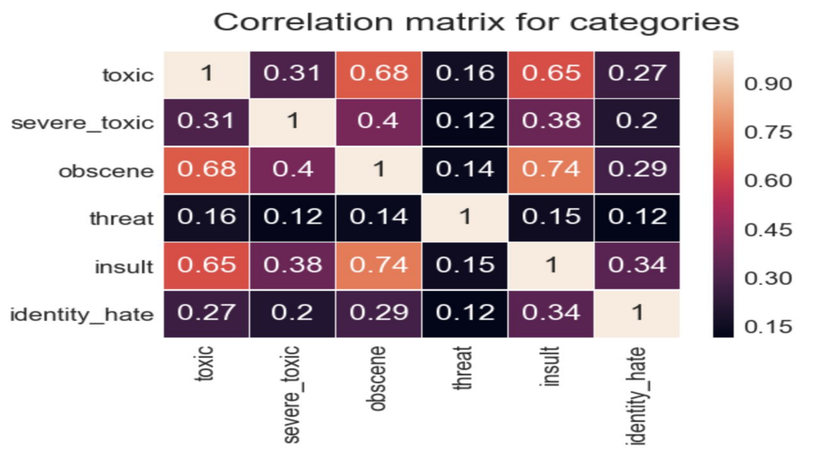
```
average length of comment: 395.342
```



3. We further did some more EDA to find how many comments have multiple labels associated with them:



4. The Correlation matrix gave us further insight into data:



a. 'toxic' is clearly correlated with 'obscene' and 'insult' (0.68 and 0.65)
b. 'toxic' and 'severe_toxic' are only got a 0.31 correlation factor
c. 'insult' and 'obscene' have a correlation factor of 0.74

3.2 Data Pre-processing:

The challenges faced during an attempt to solve the problem were with the comments which included slang words or non-english words which did not have training data to classify them correctly.

Under preprocessing we applied tokenization, lemmatization and removal of special characters. Stop words were not removed at this stage since we used n-gram parameter while considering the features. 30% of the Training data is separated for validation set on which we tune the hyperparameters and evaluate the model. We used TF-IDF scores to limit our number of features.

3.2.1 TFIDF

TF-IDF stands for *term frequency-inverse document frequency*, and the TF-IDF weight is a weight often used in information retrieval and text mining. This weight is a statistical measure used to evaluate how important a word is to a document in a collection or corpus. The importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus. Variations of the TF-IDF weighting scheme are often used by search engines as a central tool in scoring and ranking a document's relevance given a user query.

Typically, the TF-IDF weight is composed of two terms: the first computes the normalized Term Frequency (TF), aka. the number of times a word appears in a document, divided by the total number of words in that document; the second term is the Inverse Document Frequency (IDF), computed as the logarithm of the number of the documents in the corpus divided by the number of documents where the specific term appears.

- **TF: Term Frequency**, which measures how frequently a term occurs in a document.

Since every document is different in length, it is possible that a term would appear much more times in long documents than shorter ones. Thus, the term frequency is often divided by the document length (aka. the total number of terms in the document) as a way of normalization:

TF(t) = (Number of times term t appears in a document) / (Total number of terms in the document).

- **IDF: Inverse Document Frequency**, which measures how important a term is. While computing TF, all terms are considered equally important. However, it is known that certain terms, such as "is", "of", and "that", may appear a lot of times but have little importance. Thus we need to weigh down the frequent terms while scaling up the rare ones, by computing the following:

IDF(t) = log_e(Total number of documents / Number of documents with term t in it).

The challenges faced during an attempt to solve the problem were with the comments which included slang words or non-english words which did not have training data to classify them correctly.

3.3 Model Training

Our train data was split into train and validation set 70:30. All our models were trained on this train set. validation set was used to tune the Hyperparameters. After tuning the train and validation sets were merged and trained again. This model was used to predict on the test set. This additional step was applied in order to get more data points for the model training.

3.3.1 Naive Bayes

MultinomialNB implements the naive Bayes algorithm for multinomially distributed data and is one of the two classic naive Bayes variants used in text classification (where the data are typically represented as word vector counts, although TF-IDF vectors are also known to work well in practice). The distribution is parametrized by vectors θy=(θy1,…,θyn) for each class y, where n is the number of features (in text classification, the size of the vocabulary) and θyi is the probability P(xi∣y) of feature i appearing in a sample belonging to class y.
The parameters θy is estimated by a smoothed version of maximum likelihood, i.e. relative frequency counting:

θ^yi=Nyi+αNy+αn

where Nyi=∑x∈Txi is the number of times feature i appears in a sample of class y in the training set T, and Ny=∑i=1nNyi is the total count of all features for class y.
The smoothing priors α≥0 account for features not present in the learning samples and prevent zero probabilities in further computations. Setting α=1 is called Laplace smoothing, while α<1 is called Lidstone smoothing.

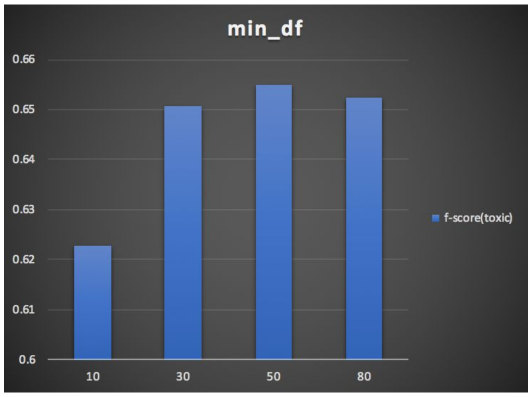
Results for validation set with default parameters

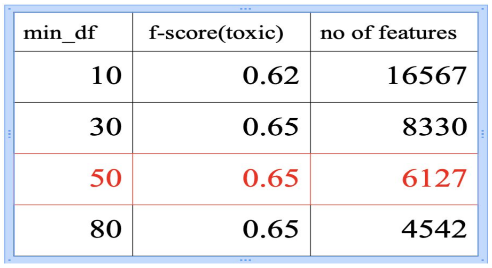
	toxic	Severe _toxic	obscene	threat	insult	Identit y_hate	clean
accuracy	0.9197	0.9894	0.9519	0.9969	0.9525	0.9909	0.9173
precision	0.9948	0	0.9924	0	0.9897	0	0.9157
recall	0.1683	0	0.1027	0	0.04094	0	0.9999
F-Score	0.2879	0	0.18619	0	0.07863	0	0.9560

3.3.1.1 Parameter tuning

For a simplicity, all the numbers are for toxic class. Similar parameter tuning can be done for other classes. We evaluated this mode with binary averaging.

min-df : When building the vocabulary this ignores terms that have a document frequency strictly lower than the given threshold. as the mi_df is increased, a number of features decreases. The best f-score is obtained at 50.





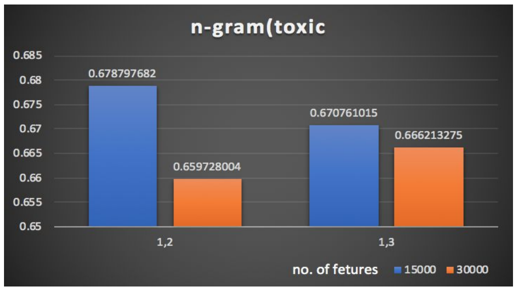
min_df	f-score(toxic)	no of features
10	0.62	16567
30	0.65	8330
50	0.65	6127
80	0.65	4542

max-df: When building the vocabulary this ignores terms that have a document frequency strictly higher than the given threshold. This parameter will remove all the stop words.

max_features: Lets us limit the number of features used in model training.

max_features	f-score(toxic)
1000	0.61
3000	0.64
5000	0.65
5500	0.65
5300	0.65
6127	0.65

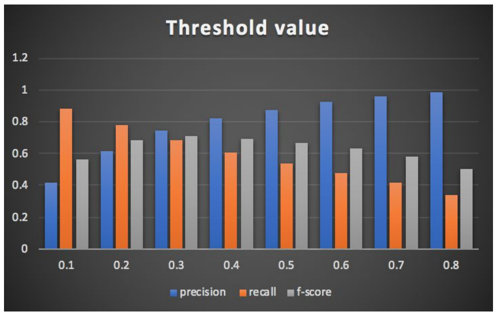
ngram_range: a contiguous sequence of *n* words from a text or speech. With the increase in ngram range, the number of features increases. we need to limit the features by tuning max_features.

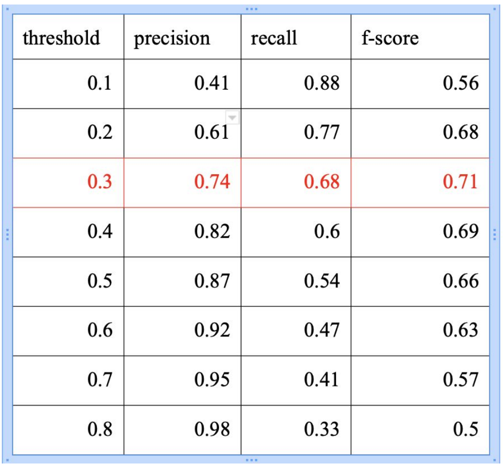


ngram_range	toxic - f-score
1,2	0.6788
1,3	0.67076
1,4	0.6577

	15000	30000
1,2	0.6788	0.65973
1,3	0.67076	0.66621

Threshold: Predicted value is 1 if the probability is higher than threshold.

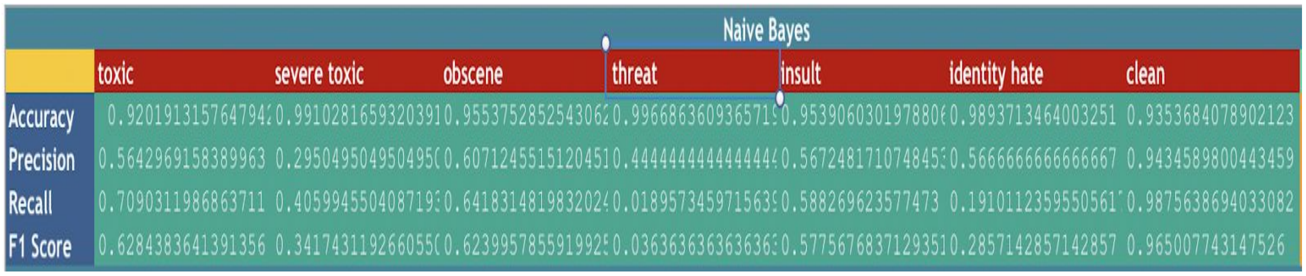




threshold	precision	recall	f-score
0.1	0.41	0.88	0.56
0.2	0.61	0.77	0.68
0.3	0.74	0.68	0.71
0.4	0.82	0.6	0.69
0.5	0.87	0.54	0.66
0.6	0.92	0.47	0.63
0.7	0.95	0.41	0.57
0.8	0.98	0.33	0.5

With parameter tuning the f-score for validation went up from 0.28 to 0.71.

3.3.2. Test Results



Naive Bayes							
	toxic	severe toxic	obscene	threat	insult	identity hate	clean
Accuracy	0.920191315764794	0.991028165932039	0.955375285254306	0.996686360936571	0.953906030197880	0.9893713464003251	0.9353684078902123
Precision	0.5642969158389963	0.295049504950495	0.6071245515120451	0.444444444444444	0.567248171074845	0.5666666666666667	0.9434589800443459
Recall	0.7090311986863711	0.405994550408719	0.641831481983202	0.018957345971563	0.588269623577473	0.191011235955056	0.9875638694033082
F1 Score	0.6284383641391356	0.341743119266055	0.623995785591992	0.036363636363636	0.5775676837129351	0.2857142857142857	0.965007743147526

The f-score on test set is smaller than train and validation sets. At first, we suspected that this was a result of over-fitting. However, as we changed the validation set, we observed this pattern repeatedly, the performance on train and validation sets matched, and the performance on test set was low. This means that the distribution of test data set is likely different from that of the train set. Although the Kaggle website didn't list that this was the case, for a different problem, the test set was a set of tweets from a different year than the set from which train set was built. There were discussions on how language usage patterns change from time to time to cause these patterns to change. This might be a reason here as well.

3.3.2 Logistic Regression

Parameter tuning

Results for the validation set with default parameters

table here

Same parameters as Naive Bayes were used to tune Logistic regression with additional two parameters.

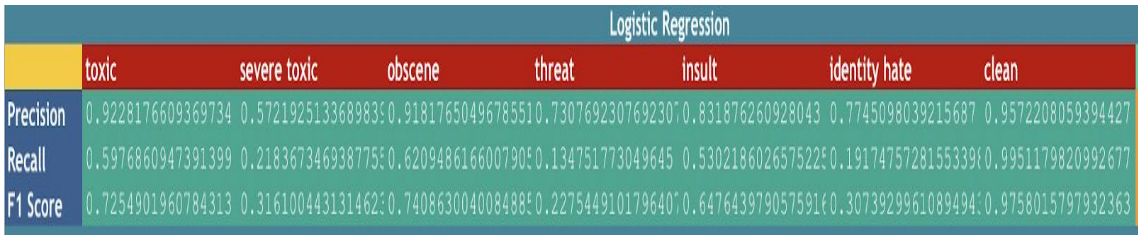
- C- regularization weights. For small values of C, we increase the regularization strength which will create simple models which underfit the data. For big values of C, we lower the power of regularization which implies the model is allowed to increase its complexity and therefore overfit the data
- penalty: L1 regularization or L2 regularization

		f-score
C=10.85,	penalty='l1'	0.77
C=5.85,	penalty='l1'	0.78
C=1.85,	penalty='l1'	0.77
C=1.0,	penalty='l1'	0.76
C=1.85,	penalty='l2'	0.73

max_df=0.8,
min_df=5,
ngram_range=(1,2)

C=1.0,	penalty='l1'	0.76
C=1.85,	penalty='l1'	0.77
C=5.85,	penalty='l1'	0.77

max_df=0.8,
min_df=5,
ngram_range=(1,3)



Logistic Regression							
	toxic	severe toxic	obscene	threat	insult	identity hate	clean
Precision	0.9228176609369734	0.572192513368983	0.918176504967855	0.7307692307692307	0.831876260928043	0.7745098039215687	0.9572208059394427
Recall	0.5976860947391399	0.218367346938775	0.620948616600790	0.134751773049645	0.530218602657522	0.191747572815533	0.9951179820992677
F1 Score	0.725490196078431	0.316100443131462	0.740863004008488	0.227544910179640	0.647643979057591	0.307392996108949	0.975801579793236

3.3 Evaluation Metrics:

Binary-averaging

- To measure a binary classifier. This will only report results for the class specified by pos_label. This is applicable only if targets (y_{true, pred}) are binary.

Micro-averaging & Macro-averaging (Label based measures):

- To measure a multi-class classifier we have to average out the classes somehow. There are two different methods of doing this called *micro-averaging* and *macro-averaging*.
- In *micro-averaging,* all TPs, TNs, FPs and FNs for each class are summed up and then the average is taken.

. Microaveraging Precision $Prc^{micro}(D) = \frac{\sum_{c_i \in C} TPs(c_i)}{\sum_{c_i \in C} TPs(c_i) + FPs(c_i)}$

. Microaveraging Recall $Rcl^{micro}(D) = \frac{\sum_{c_i \in C} TPs(c_i)}{\sum_{c_i \in C} TPs(c_i) + FNs(c_i)}$

- In the *micro-averaging* method, you sum up the individual true positives, false positives, and false negatives of the system for different sets and then apply them. And the micro-average F1-Score will be simply the harmonic mean of above two equations.

- *Macro-averaging* is straightforward. We just take the average of the precision and recall of the system on different sets.

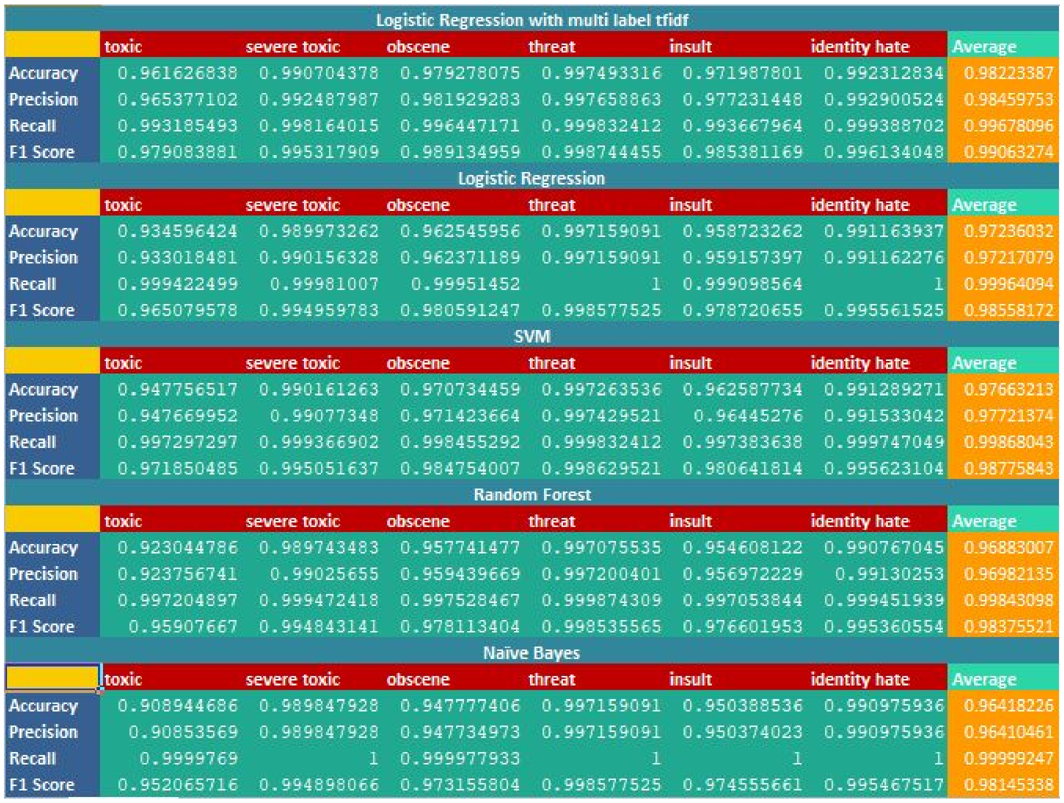
, Macroaveraging Precision $Prc^{macro}(D) = \frac{\sum_{c_i \in C} Prc(D, c_i)}{|\mathcal{C}|}$

, Macroaveraging Recall $Rec^{macro}(D) = \frac{\sum_{c_i \in C} Rcl(D, c_i)}{|\mathcal{C}|}$

- *The macro-averaging* method can be used when you want to know how the system performs overall across the sets of data. You should not come up with any specific decision with this average. On the other hand, *micro-averaging* can be a useful measure when dataset varies in size.

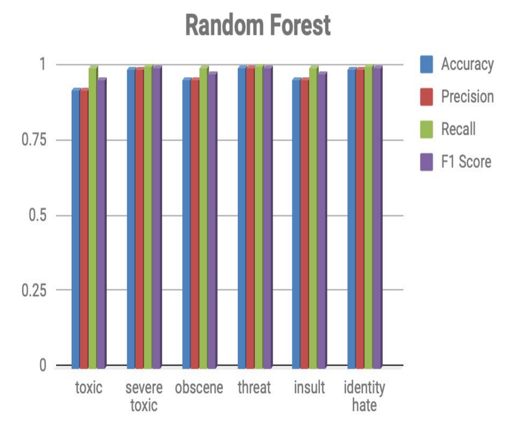
4 Result evaluation:

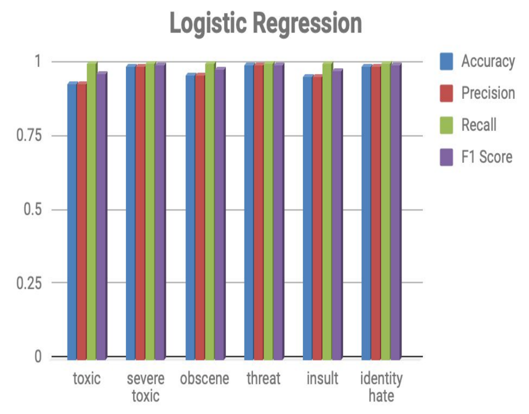
We got the following accuracy and F1-scores with macro averaging for the **validation** set :

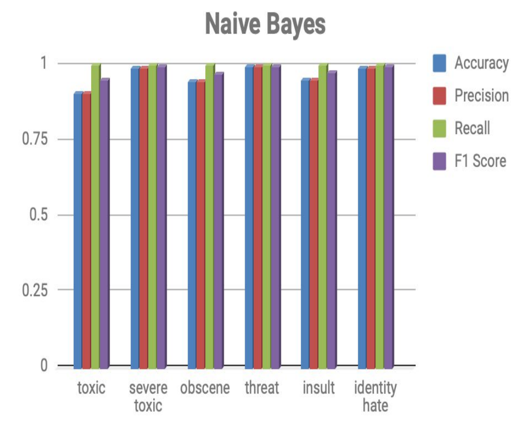


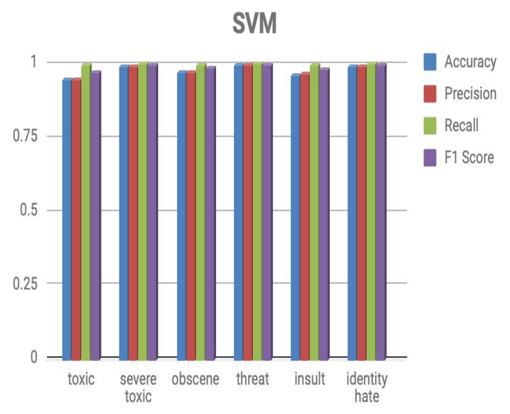
Logistic Regression with multi label tfidf							
	toxic	severe toxic	obscene	threat	insult	identity hate	Average
Accuracy	0.961626838	0.990704378	0.979278075	0.997493316	0.971987801	0.992312834	0.98223387
Precision	0.965377102	0.992487987	0.981929283	0.997658863	0.977231448	0.992900524	0.98459753
Recall	0.993185493	0.998164015	0.996447171	0.999832412	0.993667964	0.999388702	0.99678096
F1 Score	0.979083881	0.995317909	0.989134959	0.998744455	0.985381169	0.996134048	0.99063274
Logistic Regression							
	toxic	severe toxic	obscene	threat	insult	identity hate	Average
Accuracy	0.934596424	0.989973262	0.962545956	0.997159091	0.958723262	0.991163937	0.97236032
Precision	0.933018481	0.990156328	0.962371189	0.997159091	0.959157397	0.991162276	0.97217079
Recall	0.999422499	0.99981007	0.99951452	1	0.999098564	1	0.99964094
F1 Score	0.965079578	0.994959783	0.980591247	0.998577525	0.978720655	0.995561525	0.98558172
SVM							
	toxic	severe toxic	obscene	threat	insult	identity hate	Average
Accuracy	0.947756517	0.990161263	0.970734459	0.997263536	0.962587734	0.991289271	0.97663213
Precision	0.947669952	0.99077348	0.971423664	0.997429521	0.96445276	0.991533042	0.97721374
Recall	0.997297297	0.999366902	0.998455292	0.999832412	0.997383638	0.999747049	0.99868043
F1 Score	0.971850485	0.995051637	0.984754007	0.998629521	0.980641814	0.995623104	0.98775843
Random Forest							
	toxic	severe toxic	obscene	threat	insult	identity hate	Average
Accuracy	0.923044786	0.989743483	0.957741477	0.997075535	0.954608122	0.990767045	0.96883007
Precision	0.923756741	0.99025655	0.959439669	0.997200401	0.956972229	0.99130253	0.96982135
Recall	0.997204897	0.999472418	0.997528467	0.999874309	0.997053844	0.999451939	0.99843098
F1 Score	0.95907667	0.994843141	0.978113404	0.998535565	0.976601953	0.995360554	0.98375521
Naïve Bayes							
	toxic	severe toxic	obscene	threat	insult	identity hate	Average
Accuracy	0.908944686	0.989847928	0.947777406	0.997159091	0.950388536	0.990975936	0.96418226
Precision	0.90853569	0.989847928	0.947734973	0.997159091	0.950374023	0.990975936	0.96410461
Recall	0.9999769	1	0.999977933	1	1	1	0.99999247
F1 Score	0.952065716	0.994898066	0.973155804	0.998577525	0.974555661	0.995467517	0.98145338

Visualizing them through graph helped us more:

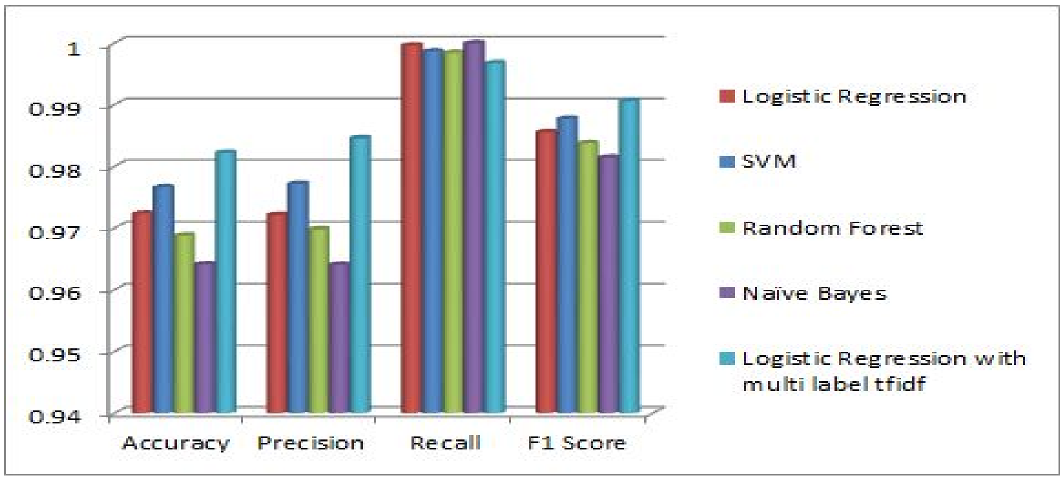




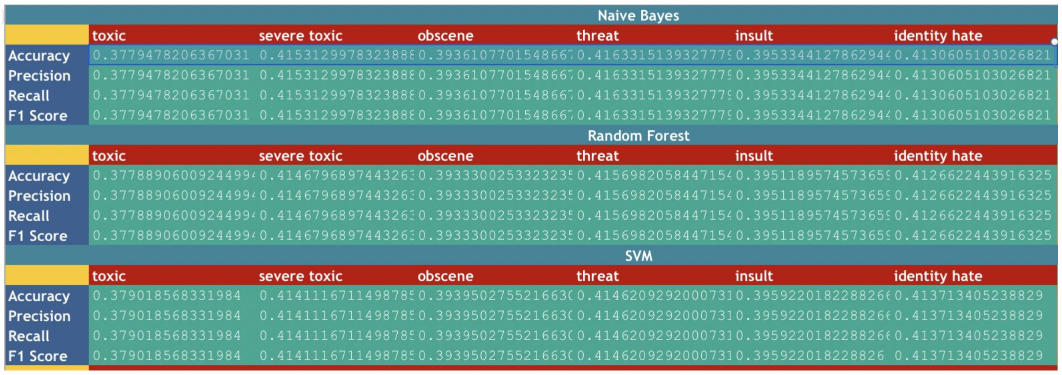




Final result comparison on Validation set:



For Test set Results, we got following:



Naive Bayes						
	toxic	severe toxic	obscene	threat	insult	identity hate
Accuracy	0.3779478206367031	0.4153129978323888	0.3936107701548667	0.4163315139327779	0.3953344127862944	0.413060510302682 1
Precision	0.3779478206367031	0.4153129978323888	0.3936107701548667	0.4163315139327779	0.3953344127862944	0.4130605103026821
Recall	0.3779478206367031	0.4153129978323888	0.3936107701548667	0.4163315139327779	0.3953344127862944	0.4130605103026821
F1 Score	0.3779478206367031	0.4153129978323888	0.3936107701548667	0.4163315139327779	0.3953344127862944	0.4130605103026821
Random Forest						
	toxic	severe toxic	obscene	threat	insult	identity hate
Accuracy	0.3778890600924494	0.4146796897443263	0.3933300253323235	0.4156982058447154	0.3951189574573659	0.412662244391632 5
Precision	0.3778890600924494	0.4146796897443263	0.3933300253323235	0.4156982058447154	0.3951189574573659	0.4126622443916325
Recall	0.3778890600924494	0.4146796897443263	0.3933300253323235	0.4156982058447154	0.3951189574573659	0.4126622443916325
F1 Score	0.3778890600924494	0.4146796897443263	0.3933300253323235	0.4156982058447154	0.3951189574573659	0.4126622443916325
SVM						
	toxic	severe toxic	obscene	threat	insult	identity hate
Accuracy	0.379018568331984	0.4141116711498785	0.3939502755216630	0.4146209292000731	0.3959220182288266	0.413713405238829
Precision	0.379018568331984	0.4141116711498785	0.3939502755216630	0.4146209292000731	0.3959220182288266	0.413713405238829
Recall	0.379018568331984	0.4141116711498785	0.3939502755216630	0.4146209292000731	0.3959220182288266	0.413713405238829
F1 Score	0.379018568331984	0.4141116711498785	0.3939502755216630	0.4146209292000731	0.395922018228826	0.413713405238829

5 Division of Labor

Soumya Parvatikar: Preprocessing, Naive Bayes, Logistic Regression

Nirali Patel: SVM, Logistic Regression with different feature extractions(TF- IDF and Multi- TFIDf) and binary classification

Priyanka Advani: RandomForest, Naive Bayes with modified preprocessing and with different feature extractions(TF- IDF and Multi- TFIDf)

6 Summary

Firstly, we started with data exploration techniques to see the nature of the data. After that, we started doing preprocessing of the data. In preprocessing we applied standard text processing techniques, tokenization, Lemmatization. After that, we applied the feature extraction techniques such as TFIDF and N-grams. After that, we applied Naive Bayes and Logistic Regression. To improve accuracy we did some parameter tuning.

After that, we wanted to know how preprocessing of the data impacts the result. So, we modified our preprocessing to know more about it. With different preprocessing, we applied the same algorithms logistic Regression and Naive Bayes. For further comparison, we applied random forest(with different estimators) and we also applied SVM. We also added Multi- TFIDF to see the changes in our results. For fun, we also converted the whole problem into Binary Classification problem. But it did not work out the best and we also faced a few problems while implementing it.

We also tried to apply different methods to for result evaluation. We tried to get values for precision, recall and f1 score with different average methods(binary, micro, macro and None). We ended up using micro. And we got good learning results. So, with this experience in future, we are thinking to learn and implement to tune parameters, to perform ensembling with current algorithms and other suitable algorithms to get good results on the **test** dataset.

7 References

http://nymag.com/intelligencer/2017/02/google-introduces-perspective-a-tool-for-toxic-comments.html

https://web.stanford.edu/class/cs224n/reports/2762092.pdf

https://medium.com/@nupurbaghel/toxic-comment-classification-f6e075c3487a

https://www.analyticsvidhya.com/blog/2017/08/introduction-to-multi-label-classification/

https://towardsdatascience.com/journey-to-the-center-of-multi-label-classification-384c40229bff

https://www.kaggle.com/clinma/eda-toxic-comment-classification-challenge

https://seaborn.pydata.org/tutorial.html

https://www.coursera.org/learn/machine-learning/home/welcome

https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

https://github.com/imrahulr/Toxic-Comment-Classification-Kaggle/blob/master/logistic_regression/svm/svm.ipynb