TOXIC COMMENT CLASSIFICATION

Asha Jose, Prabhakar Kumar, Parvathy S Kumar

ABSTRACT

A greater share of the human population uses social media and this number is increasing tremendously day-by-day. As a result, the number of derogatory comments associated with social media is also increasing, even though they only constitute a very small share of the total comments. Identifying and classifying these toxic comments is an active research task and a lot of research is happening in this area. This paper aims to do a multi-label classification on these comments into 6 labels namely toxic, severely toxic, identity-hate, threat, obscene using different deep learning and shallow learning techniques and compare their performance and limitations.

INTRODUCTION

A toxic comment is one that is unpleasant, disrespectful, or profane and causes other individuals to become uncomfortable, causing them to leave the conversation.

The problem's origins can be traced back to a plethora of internet forums where individuals actively interact and leave comments. Because some comments may be aggressive, insulting, or even hateful and result, and since these clearly constitute a risk of online abuse and harassment, it is the hosting organizations' responsibility to ensure that these discussions are not negative. The goal is to, hence create a model that could anticipate which categories the comments would fall into.

The dataset we used are the comments on Wikipedia talk pages presented by Google Jigsaw during Kaggle's Toxic Comment Classification Challenge. We have used Logistic Regression, LSTM and CNN for classification on these dataset,compared the performance of these models and the relevant scores have been calculated and noted.

DATASET

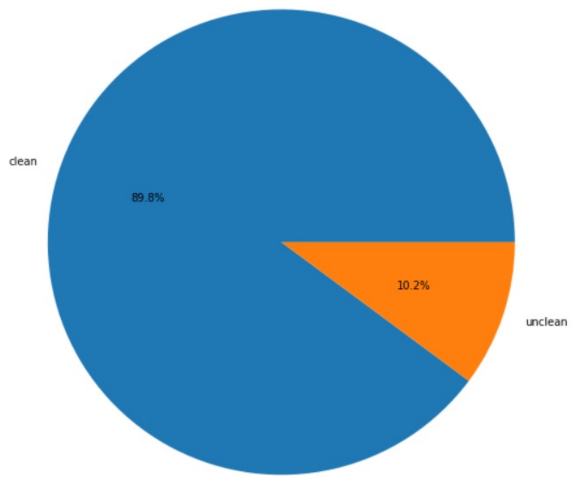
We analyzed the dataset published by Google Jigsaw in December 2017. It is the largest publicly available dataset for the job.It has 1,59,571 annotated user comments collected from Wikipedia talk pages as train data and 1,53,164 as test data.

MULTILABEL CLASSIFICATION

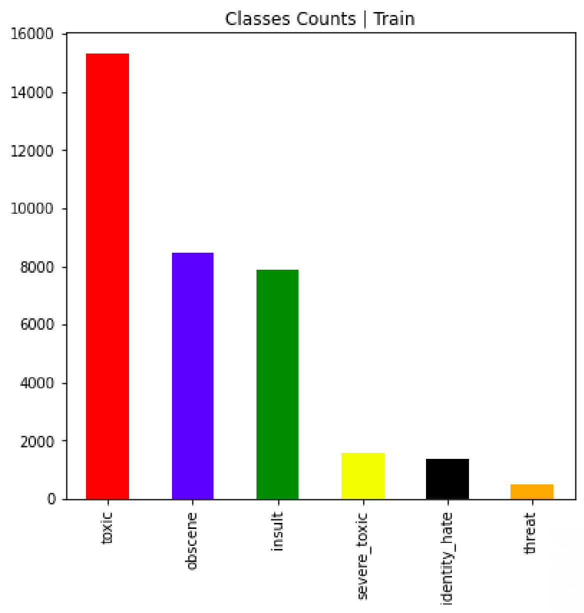
The job is framed as a multi-label classification issue because comments can be associated with numerous classes at once.Human raters assigned the labels 'toxic,''severe toxic,' 'insult,' 'threat,' 'obscene,' and 'identity hate' to these comments. Official definitions for the six classes have yet to be released by Jigsaw. However, they do note that a toxic comment is defined as "a nasty, disrespectful, or irrational comment that is likely to cause you to quit a debate."

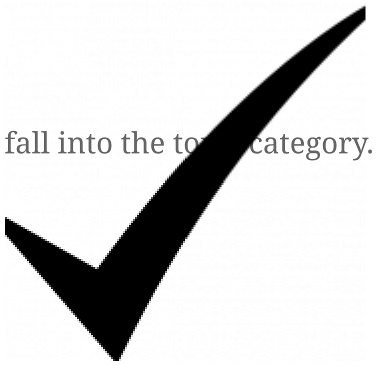
DATA ANALYSIS

A large percentage of the given data is clean i.e.1,43,343 records in the train dataset is non-toxic, i.e. it does not fall into any of the 6 categories.

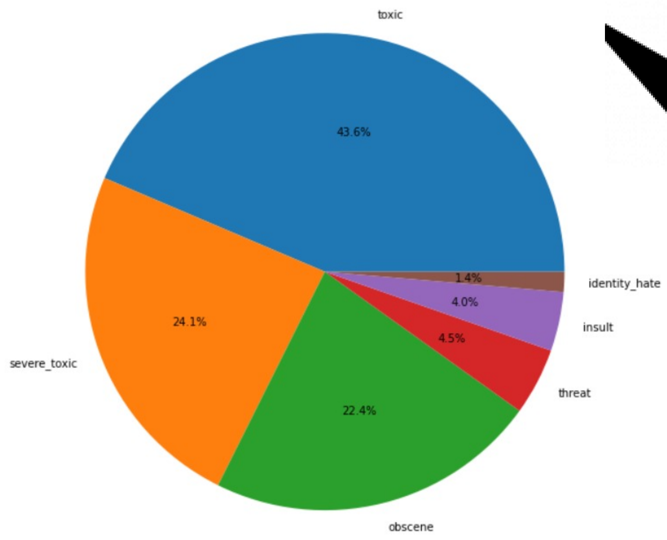


The data distribution between the different labels also is highly imbalanced.





Among the unclean comments, the most fall into the to category.



Distribution of labeled data.

CHALLENGES

- Out-of-vocabulary words - purposefully obfuscated text, slangs and misspellings fall into this category. FastText is used to deal with the problem
- Multi-word phrases - There are occurrences of multi-word phrases in both the datasets which could potentially lead to a decrease in performance of the model. For the algorithms to truly predict these as toxic, the model should recognize this as single words. For this purpose, we are using Lemmatizer here.
- Long-Range Dependencies- The toxicity of a statement is frequently determined by expressions used early in the comment. This is particularly problematic for lengthy remarks, when the impact of previous portions on the final result can be lost.
- Doubtful Labels - There are a large number of sentences whose label we cannot predict correctly when taking the respective class definition into account.
- We also encountered a problem of gpu out of memory error in the given dataset even for the most basic methods wherein the code would run for some time and give “out of memory” error.By reducing the number of features drastically we were able to overcome this error, Only a slight change in F1 score was observed.
- BIAS DATA- We have to carefully check the train and test dataset. If the test data is entirely different from the train data, it can lead to a reduction in the f1 score of the model.

PREPROCESSING

- **Removal of links & punctuations -** Links and punctuations were removed from both the train and test data
- **Lemmatizer -** The lemmatizations of words were done using the WordNet Lemmatizer
- **Filtering non character -** Non-characters including numbers, extra spaces, symbols were successfully filtered from our datasets
- **Printable-** Only printable characters were used.

METHODOLOGIES USED

We used 2 different neural network models (LSTM and CNN) and one shallow learning model (Logistic regression). These models are very different in the aspect of comment classification and have their own merits and demerits.

LOGISTIC REGRESSION

For binary classification tasks, the Logistic Regression (LR) technique is commonly employed. This is because the sigmoid function used in LR enables the probabilities to fall into a particular category as opposed to linear regression. For multilabel classification, one-vs-rest technique is used to classify all labels. For this reason LR is very good for any classification problem. It necessitates human feature engineering, as opposed to deep learning algorithms. LR, unlike Deep Learning approaches, allows for the extraction of model insights such as observed coefficients. But logistic regression does not take into account sequence of words or their long term dependencies. One of the most suggestive features for the task of hate speech detection is word and character n-grams. As a result, we look into using word and character n-grams in our models.Char n-gram seems to work better than word n-gram. This may be because of their intrinsic capability of char n-gram to capture notions that surpass mere vocabulary and relate to the daily, even informal use of language, making them ideal for user generated content data models.

LSTM

LSTM (Long Short Term Memory) is a commonly used neural model in general NLP problems. LSTM is a variation of RNN and RNN is better than simple neural network because RNN takes into consideration the sequence of words and it takes previous word also to form weights, but due to vanishing and exploding gradient, the long range dependencies aren't accounted for and therefore the memory is short term. LSTM solves this issue by storing keywords and thereby allowing long term memory also. For this reason, LSTM is very good in problems involving continuous or sequential data. But LSTM takes longer time to train and is easy to overfit and it is very sensitive to weights. We tried LSTM separately with fastText and glove (libraries for efficient learning of word representations and sentence classification). FastText seems to work slightly better than glove. This could be because fastText works on char n-gram and glove works on word n-gram and we already saw char n-gram works better for our dataset.

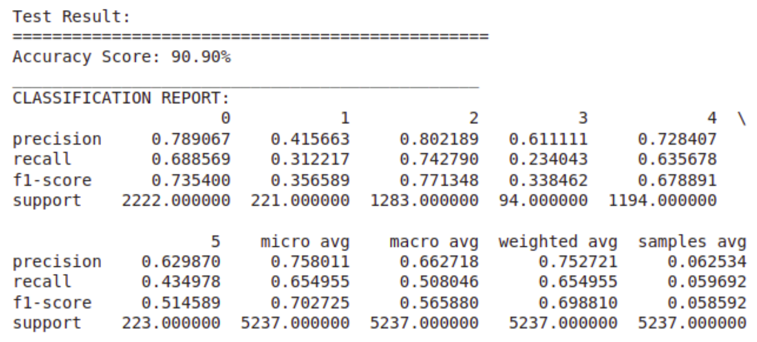
CNN

CNN (Convolutional Neural Network) is generally used for data in matrix formThe convolutional layer of CNN is different from other neural networks and makes it best for feature detection no regardless of the position. RNN is trained to recognize patterns across time, while a CNN learns to recognize patterns across spaces. Text data after tokenizing is a one dimensional matrix where the values are the density of the word vector. Here, a one dimensional convolutional layer is used. CNN keeps track of whether a feature is detected or not but not where it is detected but this can be good in case of our problem scenario where each filter of the convolutional layer can work as a specific feature for detecting negation, disappointment etc. Hence, CNN in NLP works best on classification problems involving sentiment analysis and toxic comment classification has an [illegible]ement of sentiment analysis involved. But CNN needs an extremely large data set a[illegible] also doesn't take into account dependencies. Once again, FastText works better than [illegible]e.

PERFORMANCE AND SCORES

LOGISTIC REGRESSION

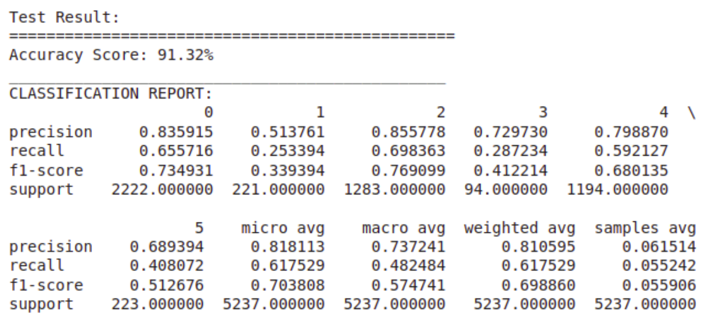
Logistic regression was done for both word and char n-grams.The vectorization was done using the `TfidfVectorizer`. The maximum_features was set keeping in mind the performance of the model and the "out of memory error". Using the LogisticRegression function available in the scikit-learn library, the model was developed and the associated predictions and scores were noted.



```
Test Result:
================================================
Accuracy Score: 90.90%
_______________________________________________
CLASSIFICATION REPORT:
                     0           1            2          3            4  \
precision     0.789067    0.415663     0.802189   0.611111     0.728407
recall        0.688569    0.312217     0.742790   0.234043     0.635678
f1-score      0.735400    0.356589     0.771348   0.338462     0.678891
support    2222.000000  221.000000  1283.000000  94.000000  1194.000000

                    5    micro avg    macro avg  weighted avg  samples avg
precision    0.629870     0.758011     0.662718      0.752721     0.062534
recall       0.434978     0.654955     0.508046      0.654955     0.059692
f1-score     0.514589     0.702725     0.565880      0.698810     0.058592
support    223.000000  5237.000000  5237.000000   5237.000000  5237.000000
```

Output for the Logistic Regression word n-grams



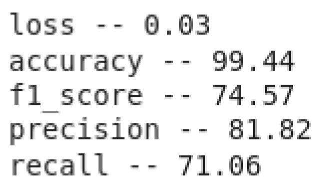
```
Test Result:
================================================
Accuracy Score: 91.32%
_______________________________________________
CLASSIFICATION REPORT:
                     0           1            2          3            4  \
precision     0.835915    0.513761     0.855778   0.729730     0.798870
recall        0.655716    0.253394     0.698363   0.287234     0.592127
f1-score      0.734931    0.339394     0.769099   0.412214     0.680135
support    2222.000000  221.000000  1283.000000  94.000000  1194.000000

                    5    micro avg    macro avg  weighted avg  samples avg
precision    0.689394     0.818113     0.737241      0.810595     0.061514
recall       0.408072     0.617529     0.482484      0.617529     0.055242
f1-score     0.512676     0.703808     0.574741      0.698860     0.055906
support    223.000000  5237.000000  5237.000000   5237.000000  5237.000000
```

Output for the Logistic Regression char n-gram

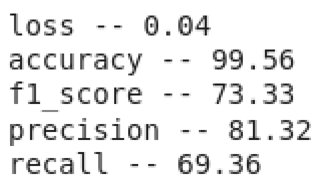
CNN

CNN was done using both fastText and Glove libraries. Words were tokenized using Tokenizer, a model in keras which is similar to one one hot encoder. Keras model CNN was used.



```
loss -- 0.03
accuracy -- 99.44
f1_score -- 74.57
precision -- 81.82
recall -- 71.06
```

Output for CNN(FastText)

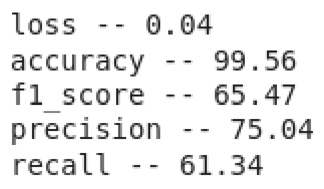


```
loss -- 0.04
accuracy -- 99.56
f1_score -- 73.33
precision -- 81.32
recall -- 69.36
```

Output for CNN(Glove)

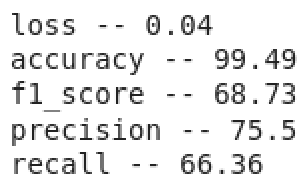
LSTM

LSTM was done using both fastText and Glove libraries. Words were tokenized using Tokenizer, a model in keras which is similar to one one hot encoder. Keras model LSTM was used.



```
loss -- 0.04
accuracy -- 99.56
f1_score -- 65.47
precision -- 75.04
recall -- 61.34
```

Output for LSTM (FastText)



```
loss -- 0.04
accuracy -- 99.49
f1_score -- 68.73
precision -- 75.5
recall -- 66.36
```

Output for LSTM (Glove)

METHOD	Macro Average of F1
Logistic Regression(Word n-gram)	0.565
Logistic Regression(Char n-gram)	0.574
LSTM(FastText)	0.675
LSTM(Glove)	0.687
CNN(FastText)	0.745
CNN(Glove)	0.733

SCOPE OF ERROR

- Doubtful labels can lead to both false positives and false negatives. There are some comments in the corpus, for which the correct labels are ambiguous for even humans. Hence such labels can fall under the category of both false positives and false negatives.
- Some comments might not explicitly consist of any toxic or swear words but can be classified as toxic comments(eg: You look like a dog). These words are potentials for false negatives.
- When in a comment, someone tries to quote some sentences or phrases that has some words that could lead the model to mark it as ‘positive’, even if the comment as a whole does not have any toxic element it could be a potential candidate for false positives.
- There are some words like swear words that the model might take as a strong indication of being positive. Even though in most sentences such words could imply a toxic meaning, in some sentences such words might not convey any derogatory idea.(eg; I was such a stupid back then)
- Idiosyncratic words are another potential source of Error. (Idiosyncrasies in spoken language are when someone uses conventional words or phrases in unusual ways—the word will be a true word in the speaker's original language, but it will not be related with whatever he or she is referring to.)
- Sarcasms and ironic sentences are another potential source of error since the textual meaning could be entirely different from the actual meaning.
- Rhetorical questions are another candidate for error since it is a common practice to wrap toxic comments within Rhetorical questions .

ETHICAL ASPECTS

The driving principle behind our project is to promote non-maleficence within online communities by spotting and responding to damaging comments. This will make the online platforms a safe and productive environment without negative distractions.

It's critical to reduce the number of false positives produced by our model and to promote all good discussion. A greater number of false positives would imply that a greater share of clear comments are marked as toxic and removed. This can potentially be a barrier in a fruitful discussion.

Our methodology also benefits platform hosts because it eliminates the need for manual discussion moderation, saving time and money. Because all comments will be handled on an equal level, using a machine learning model to filter comments promotes justice to all.

RESULT

Out of the 3 models with a total 6 ways, CNN with fastText worked better for our toxic comment classification data.

CONCLUSION

We carried out the methods of Logistic Regression, LSTM, and CNN on our dataset and noted the scores and performances. We also analyzed why some models overfitted the dataset compared to others. We also analyzed the scope of errors and limitations, and understood the challenges associated with the task.

RESOURCES

Data set: Jigsaw Kaggle Wikipedia Data set

Fast text: Wiki-news-300d-1m

Glove: Glove.6b.300d

Link for the resources: nlp_data

REFERENCES

Main- [1]Challenges for Toxic Comment Classification: An In-Depth Error Analysis, Betty van Aken , Julian Risch , Ralf Krestel , and Alexander Lose

[2]Toxic Comment Classification Pallam Ravi, Hari Narayana Batta, Greeshma S, Shaik Yaseen

[3]Multi Label Toxic Comment Classification using Machine Learning Algorithms Abhishek Aggarwal, Atul Tiwari