Toxic Comment Classification

Anshul Rao
Khoury College of Computer Sciences
Northeastern University
Boston, MA 02115
rao.ans@northeastern.edu

Abstract

I implemented two multi-label classification models namely Multinomial Naive Bayes (MNB) and Long Short-Term Memory (LSTM) models to classify Wikipedia comments into six categories - toxic, severe_toxic, obscene, threat, insult and identity_hate. The MNB model produced an accuracy of 86 percent on the test data and hamming loss of 0.031 whereas the LSTM model produced an accuracy of 99 percent on the test data and loss of 0.259.

1. Introduction

The presence of threats online in the form of abuse and harassment has been a consistent issue for a long time. Since checks for putting hateful comments/posts on social media hardly exist, the presence and spread of toxic content online is extremely rampant. Conversational toxicity can lead people to both stop genuinely expressing themselves and to stop seeking others' opinions out of fear of abuse/harassment.[1] Platforms struggle to effectively facilitate conversations, leading many communities to limit or completely shut down user comments.[2] Hence, the need for improvement in online conversation is the primary motivation behind this project which aims at building models that can classify toxic comments.

The dataset has been taken from Kaggle and has 159,571 Wikipedia comments (each identifiable by an *id*) which have been labeled by human raters for toxic behavior. The types of toxicity are: *toxic*, *severe_toxic*, *obscene*, *threat*, *insult* and *identity_hate*.[2] For example, '*Man, you are really a dishonest person, disgusting!!!!*' is classified as *toxic*.

In this project, I implemented two models that performed multi-label classification, one was Multinomial Naive Bayes and the other was Long Short-Term Memory model, a type of recurrent neural network.

2. Methods

2.1. Data Cleaning and Preprocessing

The text field 'comment_text' was cleaned by implementing a **clean_comment()** function. For cleaning the comment, hyperlinks, stopwords and punctuations were removed and WordNet was used for lemmatizing. Lemmatizing is the process of grouping together the inflected forms of a word so they can be analyzed as a single item.[5]

The *string* library was used for removing punctuations and stopwords from the *nltk* package were used to filter them out from the corpus. Hyperlinks were removed via regular expression and the *wordnet* library in *nltk* was used for lemmatization.

The dataset was split into training and test dataset in 4:1 proportion meaning that 80 percent was used for training and 20 percent for testing.

Apart from cleaning the comments, it was seen that the data was highly imbalanced. Around 90 percent of the comments were non-toxic in nature, meaning all the six labels had value 0. Class imbalance can be really harmful as it can make our model give poor results especially for the minority class, which happens to be the more important class we want to be able to classify. Hence, class imbalance was handled by upsampling the minority class, i.e., all the toxic comments in the training dataset.

The data was also checked for any missing values but there were none. The comments w.r.t. their lengths were very right-skewed with the majority of the comments having length less than 500. To ensure these outlier data points (i.e. comments of length more than 500) do not negatively impact the model performance, they were removed from the training dataset.

Note that both handling of class imbalance and removing long comments were done on the training dataset and not the test dataset.

2.2. EDA

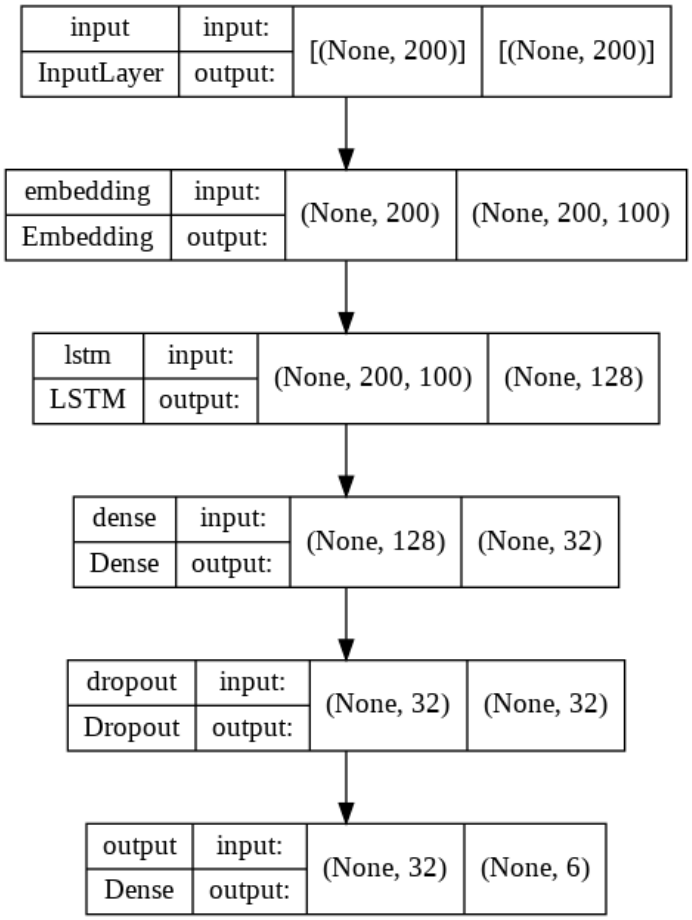
Exploratory data analysis was done on the dataset to gather important insights about the data. For example, the class imbalance and right-skewness of data distribution based on comment length was inferred via the visualizations created as part of EDA. The *seaborn* and *matplotlib* libraries were used to create the visualizations.

2.3. Multinomial Naive Bayes Model

Multinomial Naive Bayes is a simple model based on Bayes theorem and calculates the likelihood of the label. The input X to the Multinomial Naive Bayes model is the TF-IDF matrix representation of the comments. 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.[6] Six classifiers were fit, one for each label. *MultinomialNB* and *TfidfVectorizer* packages from *sklearn* were used while creating this model.

2.4. Long Short-Term Memory Model

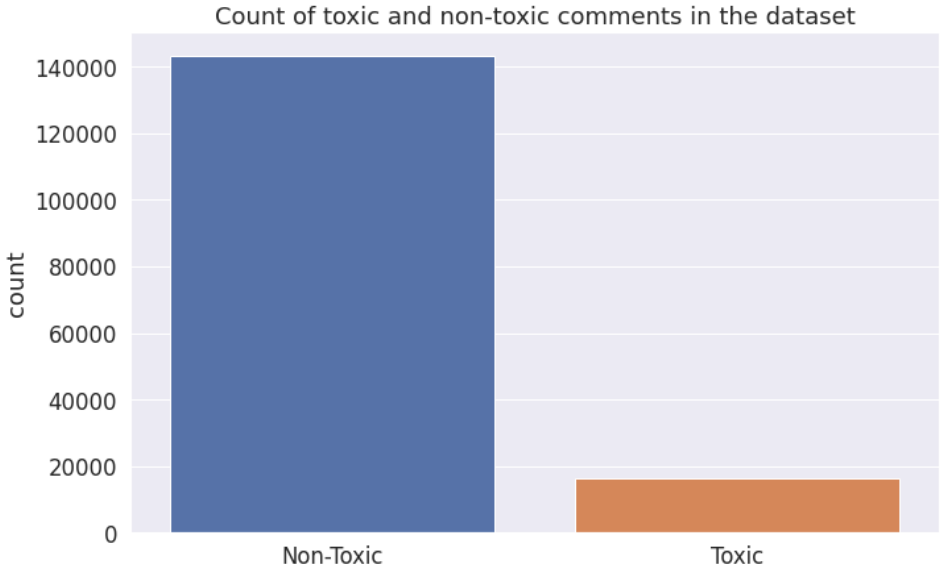
For the LSTM model, an embedding matrix was created using GloVe. GloVe is an unsupervised learning algorithm for obtaining vector representations for words.[4] The libraries used while creating this model were *keras* and *tensorflow*. A 5-layer neural network was built with an embedding layer, followed by an LSTM layer, followed by a dense layer with ReLU activation function, followed by a dropout layer and then finally an output layer which is a dense layer with sigmoid activation function.



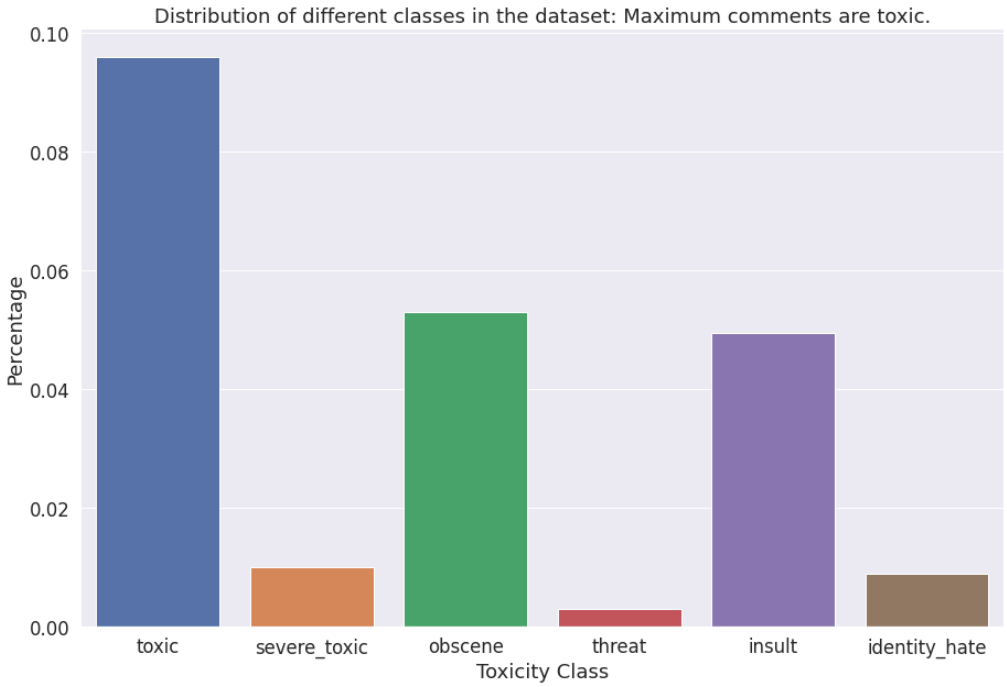
3. Results

3.1. EDA

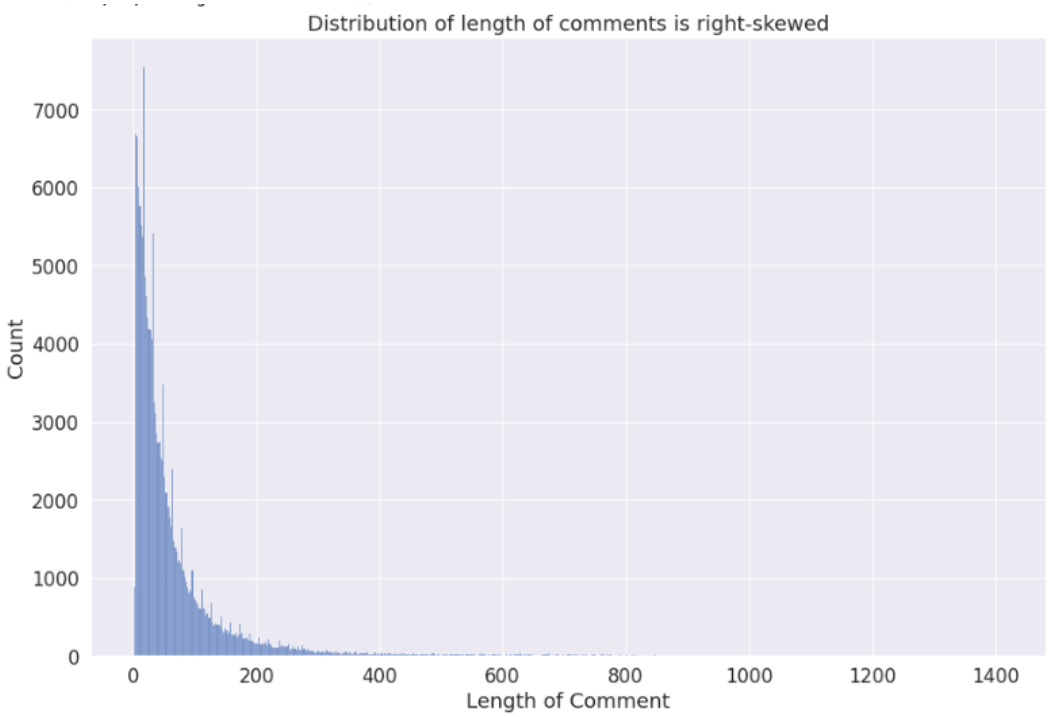
3.1.1. Class imbalance in the dataset: About 90 percent of the comments in the dataset are non-toxic in nature, i.e., all the six labels have values 0.



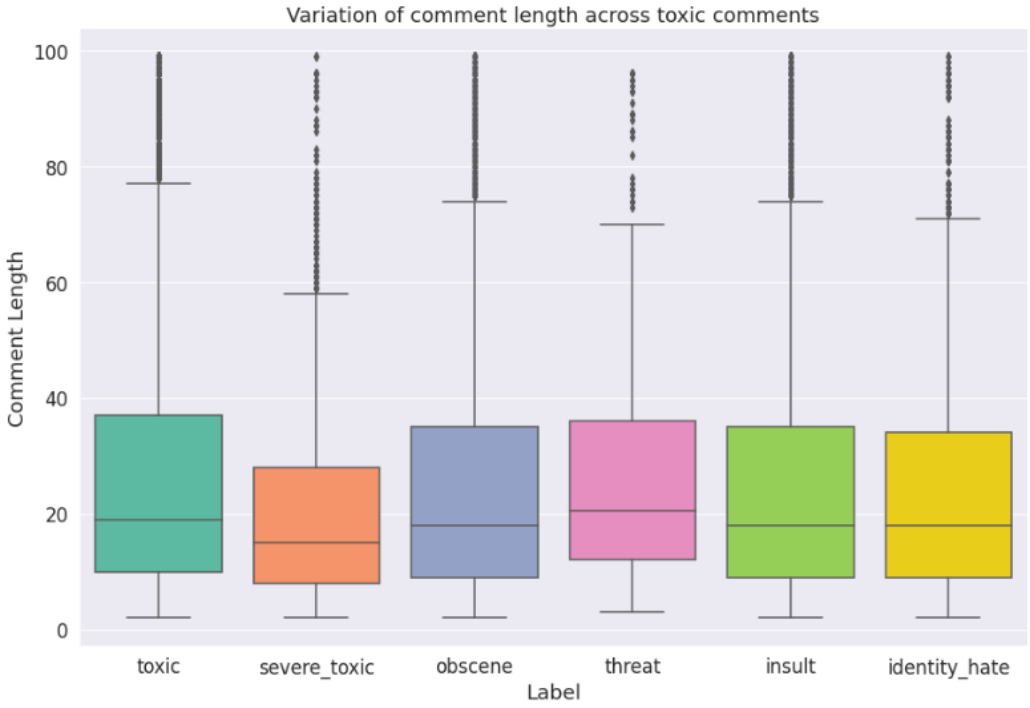
3.1.2. Percentage distribution of the six classes in the dataset: Out of the six classes present in the dataset, the most prominent presence is that of toxic class. Basically, the distribution of classes is as follows: *toxic* > *obscene* > *insult* > *severe_toxic* > *identity_hate* > *threat*.



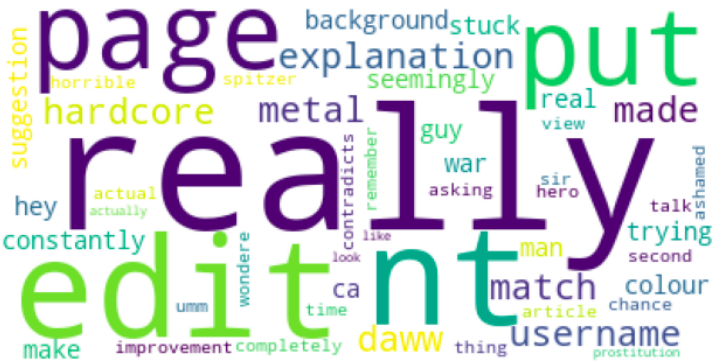
3.1.3. Length of comments: Most of the comments have length below ~300 with only a handful of comments having length more than it.



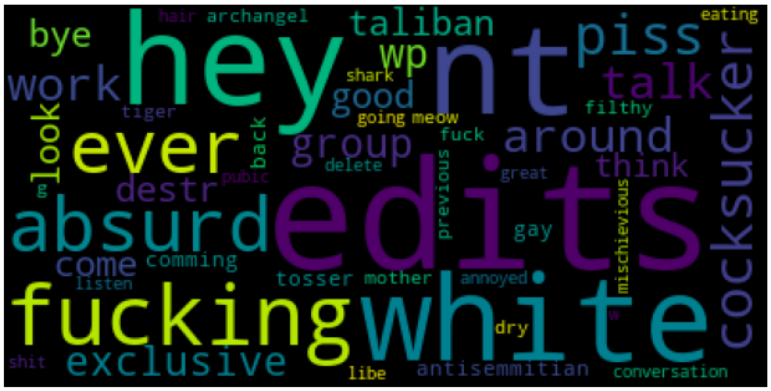
3.1.4. Variation of comment length across toxic comments: The *severe_toxic* comments seem to be relatively shorter compared to others. Other than that, there is no specific variation in length of comments based on toxicity.



3.1.5. Word could for non-toxic comments: No specific word stands out and all seem random.

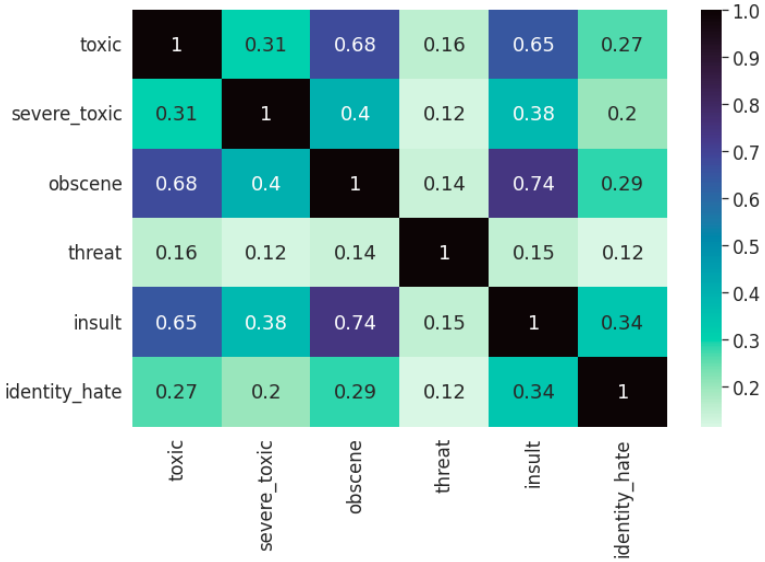


3.1.6. Word cloud for toxic comments: There are many words that exude abuse and negativity present in this word cloud.



The word edit/edits is present in both word clouds. This word has no sentiment as such attached to it so maybe it's specific to the dataset and Wikipedia comments.

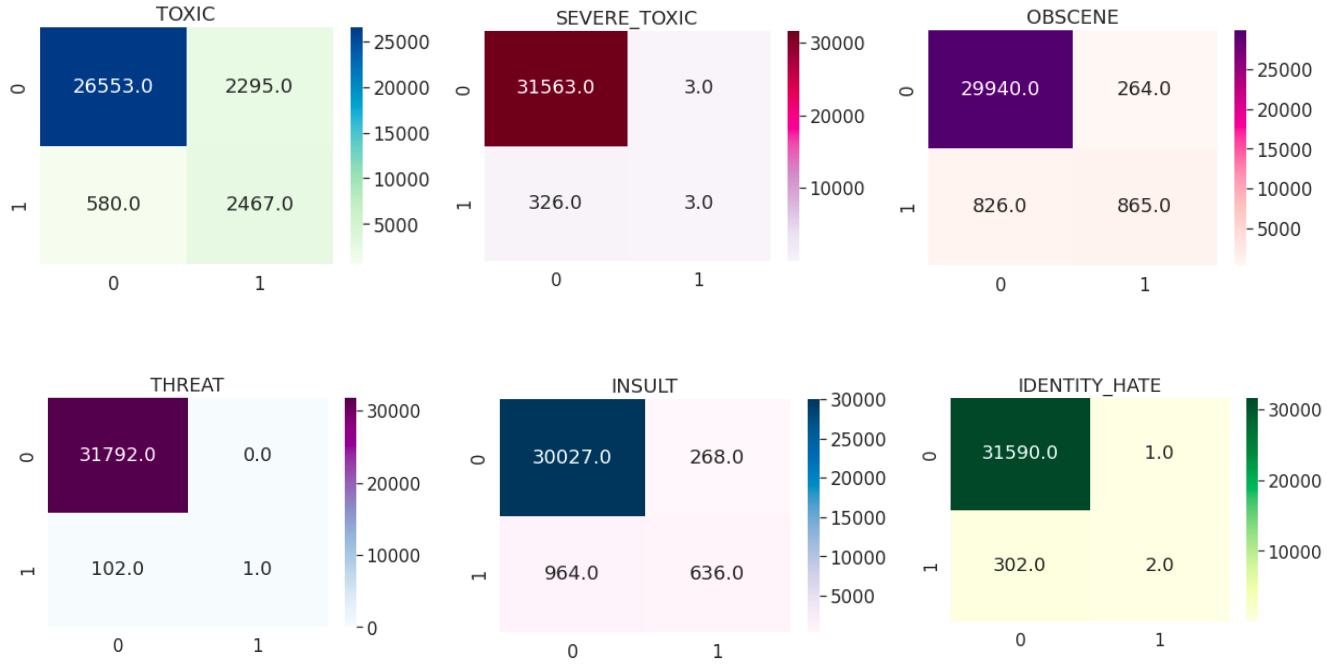
3.1.7. Correlation between different labels: Label *insult* has significant (though not extremely high) correlation with *toxic* and *obscene* comments.



It will be more difficult to classify labels that are highly correlated with another label since they will have similar words/text.

3.2. Multinomial Naive Bayes

3.2.1. Confusion matrix of each label:-



3.2.2. Accuracy: **0.8593509954538329**

3.2.3. Hamming Loss: **0.030992318545226525**

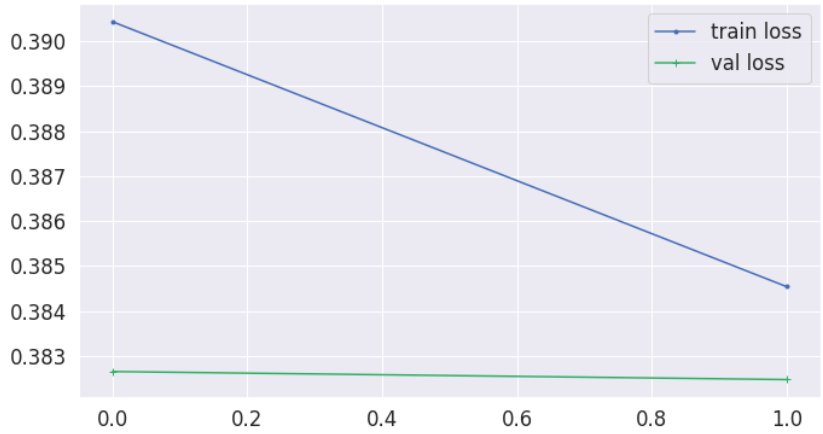
3.3. LSTM

3.3.1 Epoch cycles:-

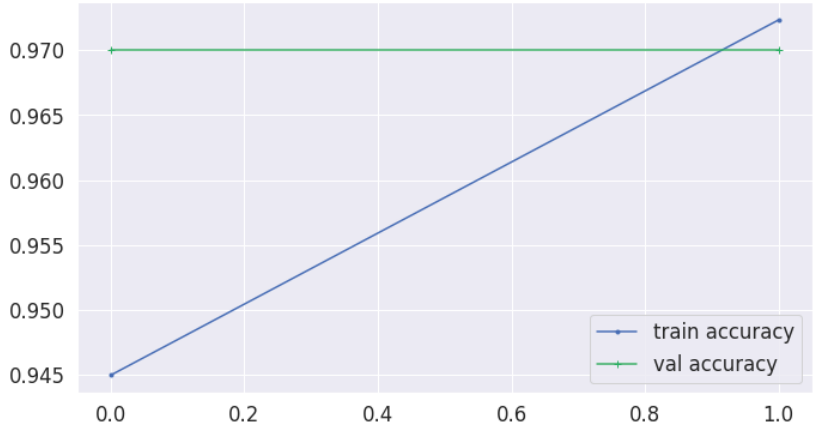
Training:

```
Epoch 1/2 1297/1297 [==============================] - 839s 645ms/step
- loss: 0.3904 - accuracy: 0.9450 - val_loss: 0.3827 - val_accuracy: 0.9700
Epoch 2/2 1297/1297 [==============================] - 829s 639ms/step
- loss: 0.3845 - accuracy: 0.9723 - val_loss: 0.3825 - val_accuracy: 0.9700
```

Training/Validation Loss



Training/Validation Accuracy



Testing:

```
997/997 [==============================] - 65s 65ms/step
- loss: 0.2585 - accuracy: 0.9944
```

3.3.2. Accuracy: **0.9943878054618835**
3.3.3. Loss: **0.2585**

4. Discussion

While working on this project I learned the difference between multi-label and multi-class classification and the nuances related to them. For example, the activation function in the output layer for a multi-label classification is sigmoid instead of a softmax because there is more than one correct class for multi-label classification. Similarly, I had to fit six different classifiers in Naive Bayes because the model is not predicting one class/label out of six but it can be more than that.

The results of the LSTM model were good and the accuracy on the test data was higher than that in the training data. Even the loss in test was lower than that of training.

Due to limited resources I could not successfully use GridSearchCV extensively to find the optimal values of hyperparameters. I was able to run it once though for two values of "units", 128 and 64, for the LSTM layer and the results can be found in the Appendix section.

5. Conclusion

As part of this project I was able to implement a Multinomial Naive Bayes model that gave 86 percent accuracy on the test dataset and an LSTM model with an accuracy as high as 99 percent on the test dataset.
In the future I would like to do some feature engineering and add features like number of capitalized words, number of exclamation marks, etc. I would also like to tune the hyperparameters of the LSTM model which I could not do thoroughly since they demand high computational power.

6. References

1. CS224N: Detecting and Classifying Toxic Comments: https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1184/reports/6837517.pdf
2. Kaggle: Toxic Comment Classification Challenge: https://www.kaggle.com/competitions/jigsaw-toxic-comment-classification-challenge/data
3. A Survey of Toxic Comment Classification Methods: https://arxiv.org/pdf/2112.06412.pdf
4. GloVe: Global Vectors for Word Representation: https://nlp.stanford.edu/projects/glove/
5. Lemmatisation: https://en.wikipedia.org/wiki/Lemmatisation
6. tf-idf: https://en.wikipedia.org/wiki/Tf%E2%80%93idf

Appendix

1. Code: https://gitlab.com/anshul.rao/toxic-comment-classification/-/tree/main/notebooks
2. Project: https://gitlab.com/anshul.rao/toxic-comment-classification
3. Tuning of "units" hyperparameter of LSTM layer using *GridSearchCV*:-

```
Fitting 5 folds for each of 2 candidates, totalling 10 fits

Epoch 1/2
5187/5187 [==============================] - 968s 186ms/step
- loss: 0.3864 - accuracy: 0.9683
Epoch 2/2
5187/5187 [==============================] - 953s 184ms/step
- loss: 0.3840 - accuracy: 0.9717
[CV] END ................................epochs=2, units=128; total time=33.8min
Epoch 1/2
5187/5187 [==============================] - 960s 185ms/step
- loss: 0.3843 - accuracy: 0.9670
Epoch 2/2
5187/5187 [==============================] - 926s 179ms/step
- loss: 0.3843 - accuracy: 0.9720
[CV] END ................................epochs=2, units=128; total time=32.8min
Epoch 1/2
5187/5187 [==============================] - 898s 173ms/step
- loss: 0.3867 - accuracy: 0.9595
Epoch 2/2
5187/5187 [==============================] - 878s 169ms/step
- loss: 0.3833 - accuracy: 0.9718
[CV] END ................................epochs=2, units=128; total time=30.7min
Epoch 1/2
5187/5187 [==============================] - 910s 175ms/step
- loss: 0.3876 - accuracy: 0.9480
Epoch 2/2
5187/5187 [==============================] - 909s 175ms/step
- loss: 0.3806 - accuracy: 0.9717
[CV] END ................................epochs=2, units=128; total time=31.4min
Epoch 1/2
5187/5187 [==============================] - 922s 177ms/step
- loss: 0.3872 - accuracy: 0.9618
Epoch 2/2
5187/5187 [==============================] - 922s 178ms/step
- loss: 0.3845 - accuracy: 0.9715
[CV] END ................................epochs=2, units=128; total time=32.1min

Epoch 1/2
5187/5187 [==============================] - 504s 97ms/step
- loss: 0.3878 - accuracy: 0.9521
Epoch 2/2
5187/5187 [==============================] - 486s 94ms/step
- loss: 0.3701 - accuracy: 0.9717
[CV] END ................................epochs=2, units=64; total time=17.9min
Epoch 1/2
5187/5187 [==============================] - 478s 92ms/step
- loss: 0.3861 - accuracy: 0.9662
```

```
Epoch 2/2
5187/5187 [==============================] - 492s 95ms/step
- loss: 0.3816 - accuracy: 0.9720
[CV] END ................................epochs=2, units=64; total time=17.0min
Epoch 1/2
5187/5187 [==============================] - 521s 100ms/step
- loss: 0.3870 - accuracy: 0.9684
Epoch 2/2
5187/5187 [==============================] - 522s 101ms/step
- loss: 0.3835 - accuracy: 0.9722
[CV] END ................................epochs=2, units=64; total time=18.0min
Epoch 1/2
5187/5187 [==============================] - 530s 102ms/step
- loss: 0.3865 - accuracy: 0.9677
Epoch 2/2
5187/5187 [==============================] - 521s 100ms/step
- loss: 0.3710 - accuracy: 0.9717
[CV] END ................................epochs=2, units=64; total time=18.2min
Epoch 1/2
5187/5187 [==============================] - 510s 98ms/step
- loss: 0.3877 - accuracy: 0.9579
Epoch 2/2
5187/5187 [==============================] - 507s 98ms/step
- loss: 0.3844 - accuracy: 0.9718
[CV] END ................................epochs=2, units=64; total time=17.5min
```