Toxic Comment Classification

Shivani Bhoite
Department of Computer Science
Georgia State University
Atlanta, GA 30324
sbhoite1@student.gsu.edu

Abstract

This project aims to build a trainable deep learning model that helps in the classification of the text, based upon the toxicity score of the emotions conveyed through the text. This project will help classify and train the neural network for identification of the toxicity scores of any given text.

1 Problem

Everyone has a phone these days, and with the phone, EVERYONE has access to one of the social media platforms. (97 percent of the teens are online and the vast majority of them have access to internet) These platforms provide an environment where people can freely engage in discussions and express their opinions with regards to the events happening around them. These events can be anything ranging from news articles, politics to the weather. The people actively engage in this discussion, and sometimes the comments refer to each other. This leads to active one on one discussions arise.

When views of people do not match, it leads to offensive and abusive talks via such comments, beginning to severe cases of cyberbullying.

2 Goal

Passing on toxic text which leads to cyberbullying cases. It is a real problem in society today. (34 percent of people feel they have been cyberbullied in their lifetime). To monitor that the cases of cyberbullying do not occur, the owner companies of chat forums had to deploy a human interference for monitoring. However, as human moderation is expensive and time- consuming, mostly the companies have deployed an automated machine learning model for the classification of the toxic text.

The goal of this project is to identify the toxicity level of the text and eliminate the human intervention as little as possible

3 Data Analysis

3.1 What kind of Data?

The data is obtained from a Kaggle competition - 'Toxic Comment Classification'.

Data source Link:

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

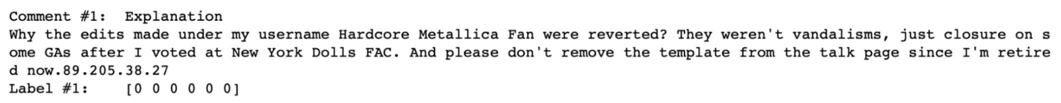
The dataset which is provided has a large number of Wikipedia comments which have been labeled by human raters for toxic behavior. The data is labelled for the following types of toxicity, which are:

- Toxic
- Severe Toxic
- Obscene
- Threat
- Insult
- Identity Hate

The dataset includes 2 files

- Train.csv – includes the training data with comments and binary labels to identify which form of toxicity the lie into
- Test.csv - includes the testing data with comments and binary labels to identify which form of toxicity the lie into.

The below is an example of a comment in a data. The label values are 0 or 1 indicated if the data belongs to 1 on the 6 toxic levels



```
Comment #1:  Explanation
Why the edits made under my username Hardcore Metallica Fan were reverted? They weren't vandalisms, just closure on s
ome GAs after I voted at New York Dolls FAC. And please don't remove the template from the talk page since I'm retire
d now.89.205.38.27
Label #1:    [0 0 0 0 0 0]
```

Figure 1: A example of data.

3.2 Data Analysis

There are in total 159571 comments available in the data.

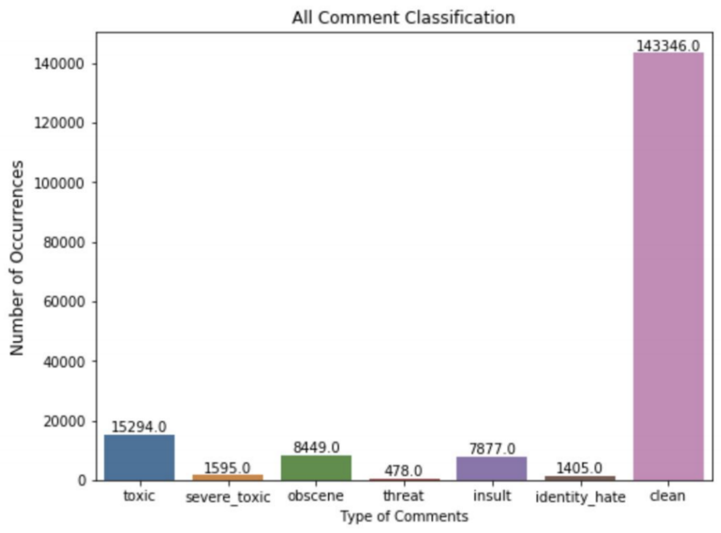


Figure 2: All comment classification.

From the above figure, out of the comments 143346 are the clean comments where the binary value for each toxicity is 0.

For the comments which are toxic in a way are classified as follows:

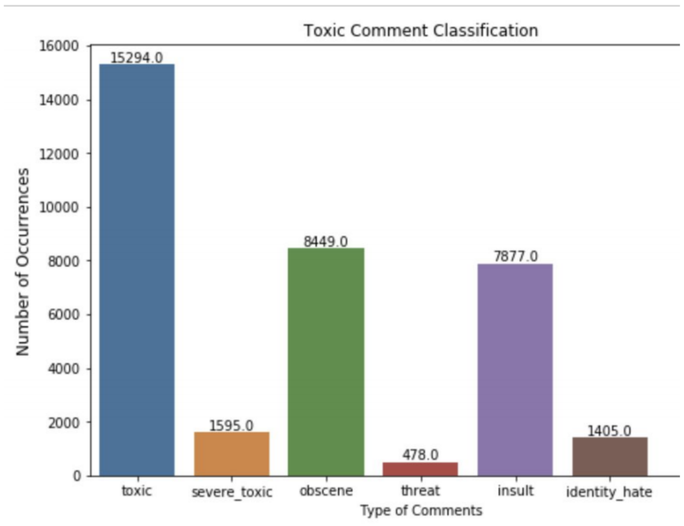


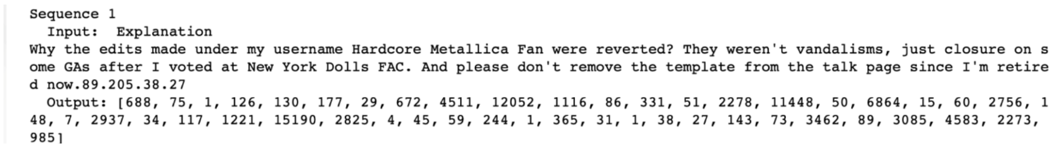
Figure 3: Count of Toxic Comments.

4 Data Preprocessing

4.1 Data Cleaning

The data consists of 10734904 words in total. The unique words in the data 532299. But after finding out the most unique words, it is observed that the words are ”the”, ”to”,”of”,”and”,”is”,”you”. Also since all the words are not in the same case, some words are counted more than once. Hence to make the data more relevant and clean the following data cleaning was done.

- Removal of irrlevant character - All the special characters were removed
- Convert all letters to lowercse
- Tokenize the words



```
Sequence 1
  Input:  Explanation
Why the edits made under my username Hardcore Metallica Fan were reverted? They weren't vandalisms, just closure on s
ome GAs after I voted at New York Dolls FAC. And please don't remove the template from the talk page since I'm retire
d now.89.205.38.27
  Output: [688, 75, 1, 126, 130, 177, 29, 672, 4511, 12052, 1116, 86, 331, 51, 2278, 11448, 50, 6864, 15, 60, 2756, 1
48, 7, 2937, 34, 117, 1221, 15190, 2825, 4, 45, 59, 244, 1, 365, 31, 1, 38, 27, 143, 73, 3462, 89, 3085, 4583, 2273,
985]
```

Figure 4: Example of a tokenized word.

- Standadize the input length with padding

After preprocessing of the data of vocab size falls from 10734904 words to 210,337 words.

4.2 Embedding

The challenge faced here was, first words were converted to numbers using the one-hot encoding. Every word in the text is converted to a vector with a number corresponding to its location. But since the vocabulary size was 210,000, doing one-hot encoding resulted in vectors that have more zeros in them. The output was high-dimensional and sparse. So instead of using the one-hot encoding, the ’word2vec’ approach is taken into consideration.

I have used the pre-trained FastText embeddings. FastText is a library for the learning of word embeddings and text classification created by Facebook’s AI Research lab. The model allows to create unsupervised learning or supervised learning algorithm for obtaining vector representations for words. Facebook makes available pre-trained models for 294 languages. FastText supports continuous training bag of words (CBOW) or Skip-gram models using negative sampling, softmax, or hierarchical softmax loss functions.

The other option was using GloVe: Global Vectors for Word Representation. The GloVe is built on two main methods – global matrix factorization and local context window. Instead of extracting the embeddings from a neural network that is designed to perform a different task like predicting neighboring words (CBOW) or predicting the focus word (Skip-Gram), the embeddings are optimized directly, so that the dot product of two-word vectors equals the log of the number of times the two words will occur near each other.

But when rare words occur, fastText works well with rare words. So even if a word wasn't seen during training, it can be broken down into n-grams to get its embeddings. This is the main advantage and reason for using fastText over GloVe.

After applying the fastText, a 300-dimensional vector for each word in the vocabulary is obtained. FastText is able to achieve excellent performance for word representations and sentence classification.

5 Deep Learning Models

For the classification of comments via Neural Networks I have taken into consideration the following 2 models -

- Convolution Neural Network (CNN with word embedding)
- Recurrent Neural Network (RNN) with Long Short Term Memory (LSTM) cells

5.1 Convolution Neural Network

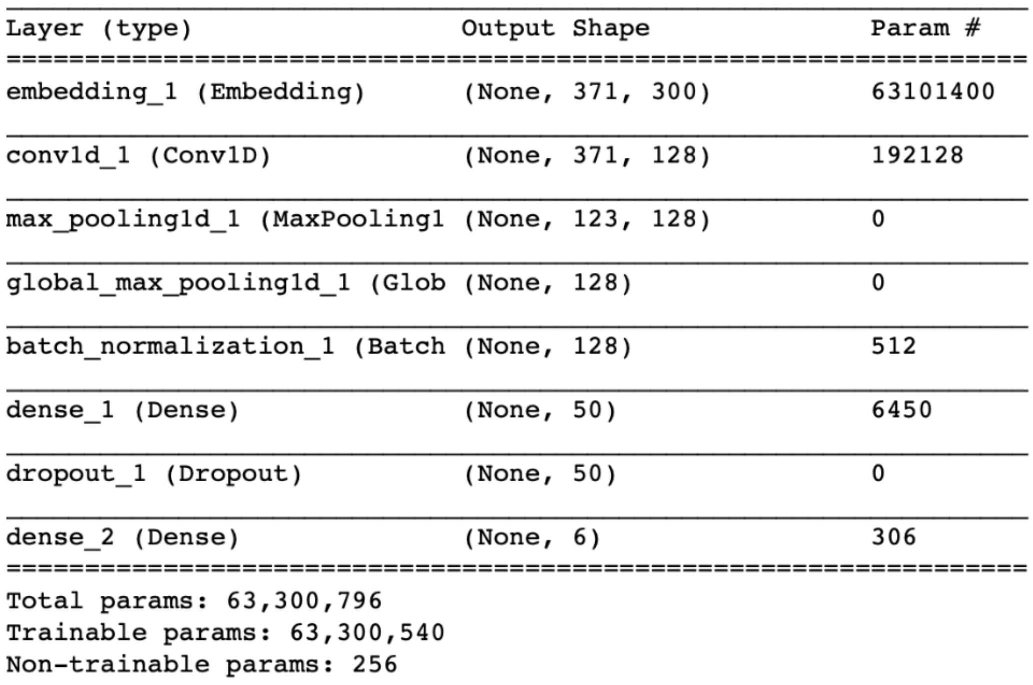
The CNN model used consists of the 1-dimensional neural network across concatenated word embeddings for each input comment. The convolutional layer has 128 filters with a kernel size of 5 so that each convolution considers a window of 5-word embeddings. The next layer is a fully-connected layer with 50 units. Adding a Fully-Connected layer is a way of learning non-linear combinations of the high-level features as represented by the output of the convolutional layer. An output layer then follows the fully connected layer.

5.2 Recurrent Neural Network

The RNN model used is the LSTM model. The LSTM is a special kind of RNN which are capable of learning long-term dependencies. This model because the individual cell states in a model can remove or add information to the cell state through gates layers. This is useful in practice because it allows the model to remember insights derived from words throughout the comment. LSTMs are designed to avoid the long-term dependency problem. LSTMs also have this chain-like structure, but the repeating module has a different structure. Instead of having a single neural network layer, therefore interacting correctly. The LSTM model consists of one densely connected layer with 60 units across the concatenated word vectors for each of the words in a comment.

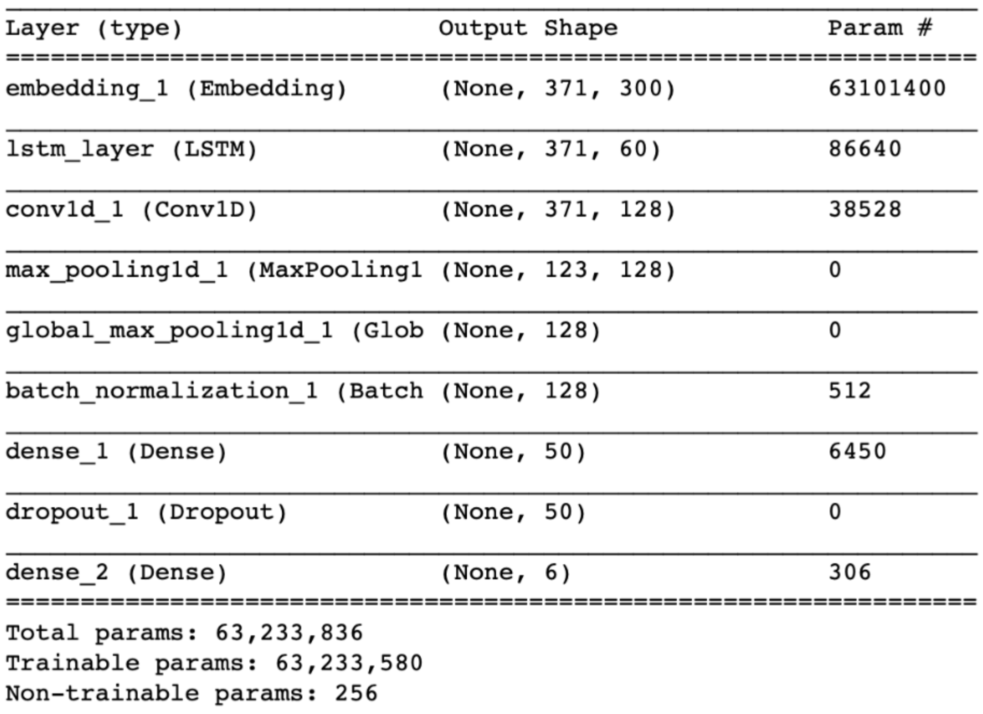
5.3 Model Summary

The Keras gives an representation of Model summary. The following is the model summary printed in Keras.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 371, 300)          63101400
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 371, 128)          192128
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 123, 128)          0
_________________________________________________________________
global_max_pooling1d_1 (Glob (None, 128)               0
_________________________________________________________________
batch_normalization_1 (Batch (None, 128)               512
_________________________________________________________________
dense_1 (Dense)              (None, 50)                6450
_________________________________________________________________
dropout_1 (Dropout)          (None, 50)                0
_________________________________________________________________
dense_2 (Dense)              (None, 6)                 306
=================================================================
Total params: 63,300,796
Trainable params: 63,300,540
Non-trainable params: 256
```

Figure 5: Model Summary of CNN.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 371, 300)          63101400
_________________________________________________________________
lstm_layer (LSTM)            (None, 371, 60)           86640
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 371, 128)          38528
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 123, 128)          0
_________________________________________________________________
global_max_pooling1d_1 (Glob (None, 128)               0
_________________________________________________________________
batch_normalization_1 (Batch (None, 128)               512
_________________________________________________________________
dense_1 (Dense)              (None, 50)                6450
_________________________________________________________________
dropout_1 (Dropout)          (None, 50)                0
_________________________________________________________________
dense_2 (Dense)              (None, 6)                 306
=================================================================
Total params: 63,233,836
Trainable params: 63,233,580
Non-trainable params: 256
```

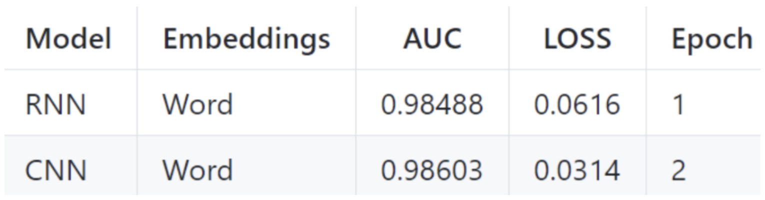
Figure 6: Model Summary of RNN.

6 Results

Since we are plotting a binary result for a model, to measure the performance of the models, AUC (Area Under The Curve) ROC (Receiver Operating Characteristics) curve score is considered. Higher the AUC, better the model is at predicting 0s as 0s and 1s as 1s.

As observed from the AUC ROC score, the CNN model has a higher accuracy over the RNN-LSTM model. The CNN model slightly outperformed the LSTM model with an accuracy of 98.6 percent with one epoch. On running this model for a second epoch, there was no increase in the accuracy, although there was an improvement in the loss factor, which went down to 3.14 percent.

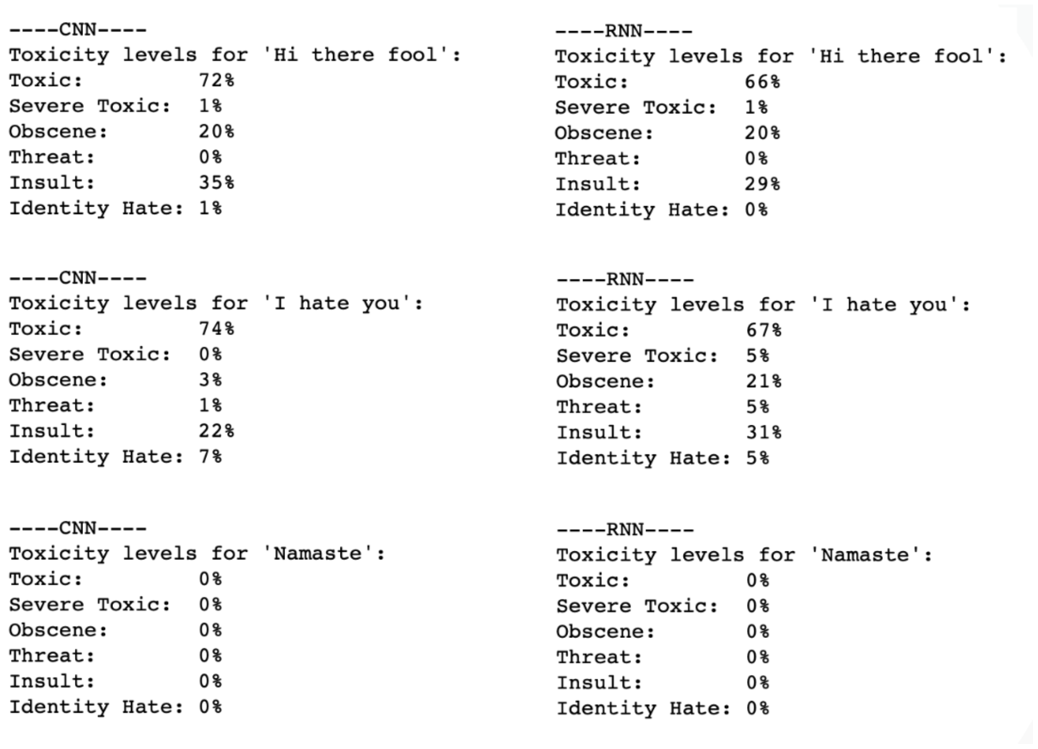
Note - As inferred from section 3.2, since the data consists of more clean comments (143346) and the toxic comments are few (15294), the achieved accuracy is higher.



Model	Embeddings	AUC	LOSS	Epoch
RNN	Word	0.98488	0.0616	1
CNN	Word	0.98603	0.0314	2

Figure 7: Results.

To understand the results in the better way, consider the following output for the same text via different models.



```
----CNN----
Toxicity levels for 'Hi there fool':
Toxic:         72%
Severe Toxic:  1%
Obscene:       20%
Threat:        0%
Insult:        35%
Identity Hate: 1%
```

```
----RNN----
Toxicity levels for 'Hi there fool':
Toxic:         66%
Severe Toxic:  1%
Obscene:       20%
Threat:        0%
Insult:        29%
Identity Hate: 0%
```

```
----CNN----
Toxicity levels for 'I hate you':
Toxic:         74%
Severe Toxic:  0%
Obscene:       3%
Threat:        1%
Insult:        22%
Identity Hate: 7%
```

```
----RNN----
Toxicity levels for 'I hate you':
Toxic:         67%
Severe Toxic:  5%
Obscene:       21%
Threat:        5%
Insult:        31%
Identity Hate: 5%
```

```
----CNN----
Toxicity levels for 'Namaste':
Toxic:         0%
Severe Toxic:  0%
Obscene:       0%
Threat:        0%
Insult:        0%
Identity Hate: 0%
```

```
----RNN----
Toxicity levels for 'Namaste':
Toxic:         0%
Severe Toxic:  0%
Obscene:       0%
Threat:        0%
Insult:        0%
Identity Hate: 0%
```

Figure 8: Comparison of 2 Neural Network models for same text.

Therefore from the above results for same text from different models, the best model, in this case, was the CNN model. Although its performance accuracy is marginally better than the LSTM model, it could gain a better categorization of toxic comment accuracy.

7 Future Work

This model can not only be used for finding out the toxicity from the text on social media but also can be integrated with other platforms. For example, if this model is integrated with the email services, say Outlook, then before sending out any email, we can check what level of toxicity is contained in the email.

Basically, this model can be integrated with other platforms know the toxic levels of text. And if the text is highly toxic then let the user not able to publish/send that text.

Acknowledgments

I have used the idea and code given at 'Toxic Comment Classification Models Comparison and Selection' by Neha Bhangale

https://medium.com/@nehabhangale/
toxic-comment-classification-models-comparison-and-selection-6c02add9d39f

to learn about the models.

Where-ever needed, I have made changes to make the existing code better and more accurate.

References

[1] Hinduja, Ph.D, Sameer and Justin Patchin Ph.D. "Cyberbullying: Identification, Prevention, and Response." Accessed September 30, 2019. https://cyberbullying.org/Cyberbullying-Identification-Prevention-Response-2018.pdf

[2] 'Helping Teens Identify and Avoid Cyberbullying' by By Tyler Jacobson https:/www.sciencedaily.com/releases/2017/02/170221102036.html

[3] 'Word2Vec' by Google https://code.google.com/archive/p/word2vec/

[4] 'Convolutional Neural Networks for Sentence Classification' by Kim, Yoon https://www.aclweb.org/anthology/D14-1181/

[5] 'Convolutional Neural Networks for Text Classification' by David S. Batista http://www.davidsbatista.net/blog/2018/03/31/SentenceClassificationConvNets/

[6] 'Toxic Comment Classification Models Comparison and Selection' by Neha Bhangale

https://medium.com/@nehabhangale/toxic-comment-classification-models-comparison-and-selection

[7] 'GloVe and fastText — Two Popular Word Vector Models in NLP' by Satish Reddy GloVeandfastText\OT1\textemdashTwoPopularWordVectorModelsinNLP

[8] 'Understanding AUC - ROC Curve' Sarang Narkhede https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5

[9] 'A.I. Wiki' https://pathmind.com/wiki/word2vec

[10] 'Keras Tutorial' https://keras.io/