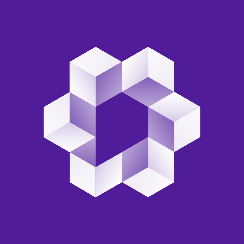




**Data Scientist Assignment**

**Kaggel Toxic Comment Classification**



Yonatan Faigenbaum

T able of content

[Table of content 2](#_Toc512814221)

[Assignment Definition 3](#_Toc512814222)

[Workflow 3](#_Toc512814223)

[Data Investigation 4](#_Toc512814224)

[Preprocessing 7](#_Toc512814225)

[Architecture 8](#_Toc512814226)

[Final Optimization 11](#_Toc512814227)

[Evaluation 11](#_Toc512814228)

[Code Arrangement 12](#_Toc512814229)

Assignment Definition

* Discussing things you 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.
* Using labeled dataset from the Wikipedia's talk page build a multi-headed model that’s capable of detecting different types of toxicity like threats, obscenity, insults, and identity-based hate.
* The evaluation of the model is based on AUC score.
* [Current models](https://github.com/conversationai/unintended-ml-bias-analysis) used by [Perspective API](https://perspectiveapi.com/) achieve an AUC of 0.95
* For this assignment show only a general approach for the problem without optimizations

Workflow

**Model**

Examine Nets

Embedded

CNNs

LSTM/GRU

Preprocess / NLP

Dataset Investigation

**Dataset**

Hyper parameter Tuning

Data Investigation

The training dataset contain –

* 159,571 samples
* Each sample can be 6 different toxic comments or non harmful

*Example*, The comment –

*FUCK YOUR FILTHY MOTHER IN THE ASS, DRY!*

Has 3 labels - toxic, obscene and an insult

* The labels are highly skewed, Only 10% of the samples is some toxic (see figure)
* The mean comments length is 68 words, The median is 36 words (see figure)
* Looking at the tf-idf per label we can see the most popular curses
* Many of the comments have spelling mistakes

*Examples*, Fuck (taking from the competition discussion board) –

'fuck', 'fucking', 'fucked', 'fuckin', 'fucka', 'fucker', 'fucks', 'fuckers',

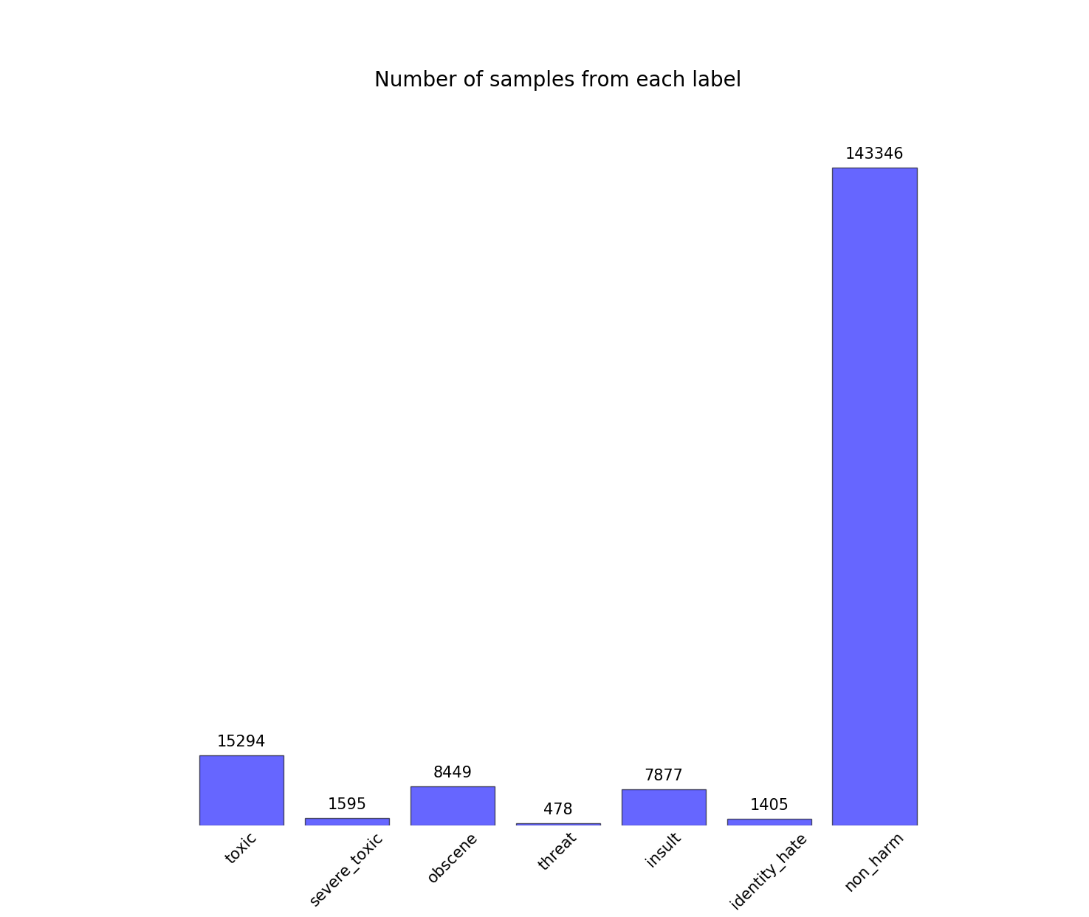
'fck', 'fcking', 'fcked', 'fckin', 'fcker', 'fcks',

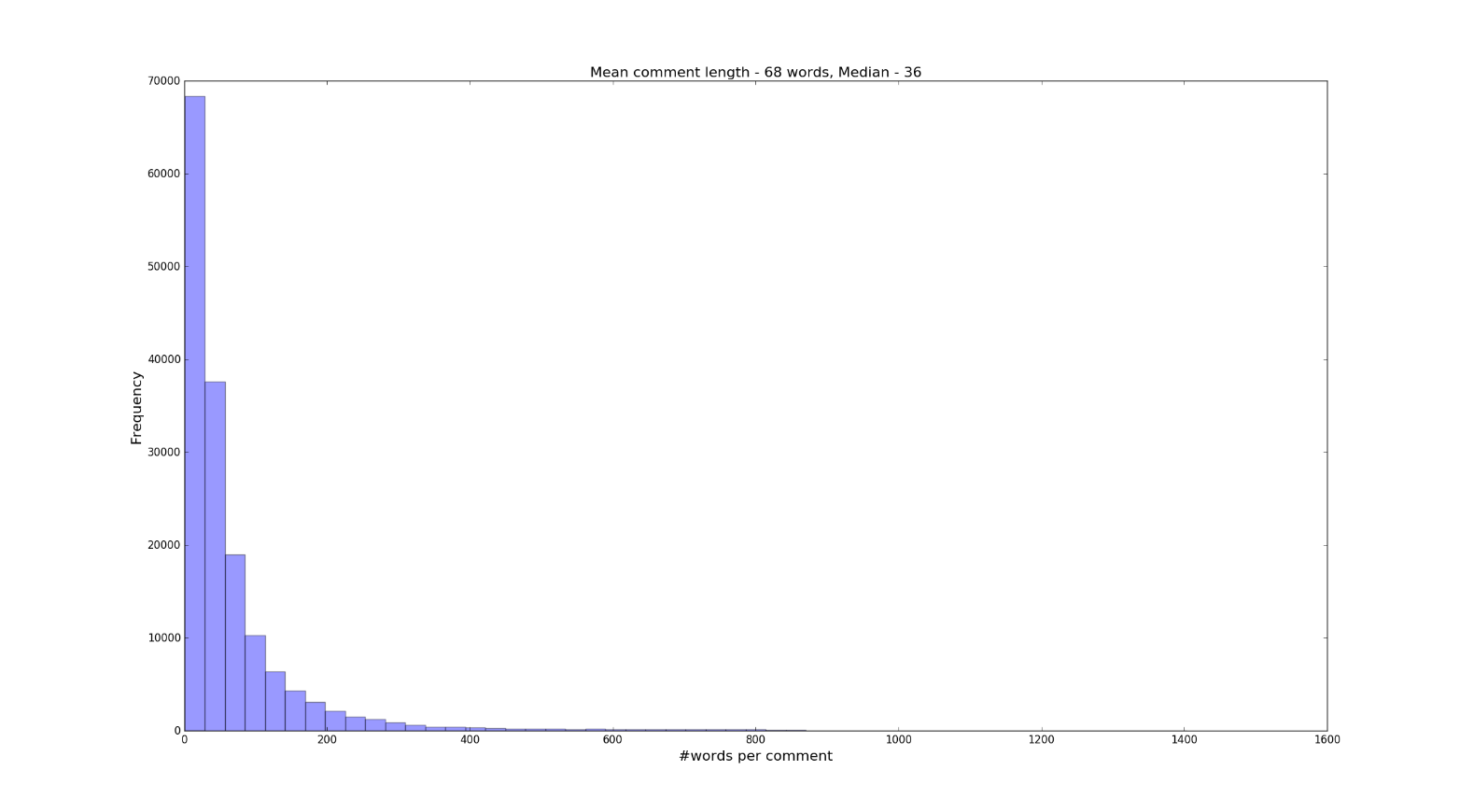
'fuk', 'fuking', 'fuked', 'fukin', 'fuker', 'fuks', 'fukers',

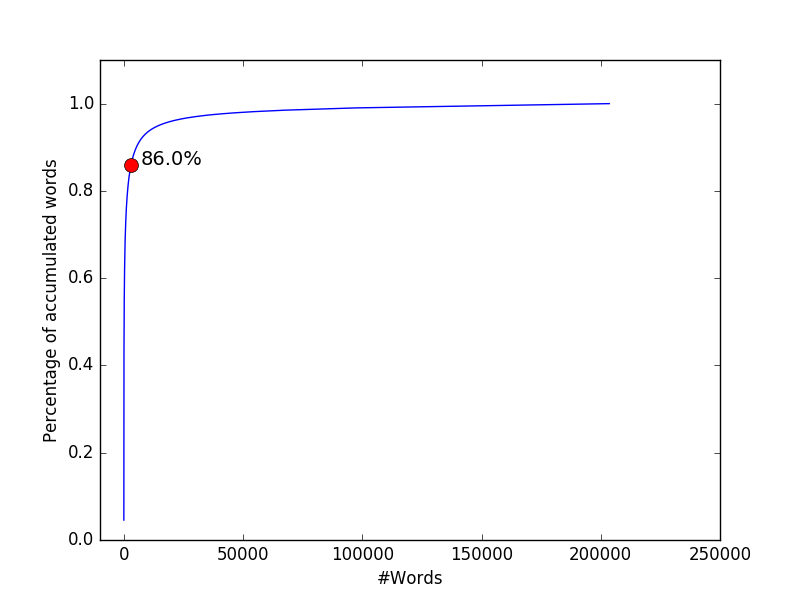
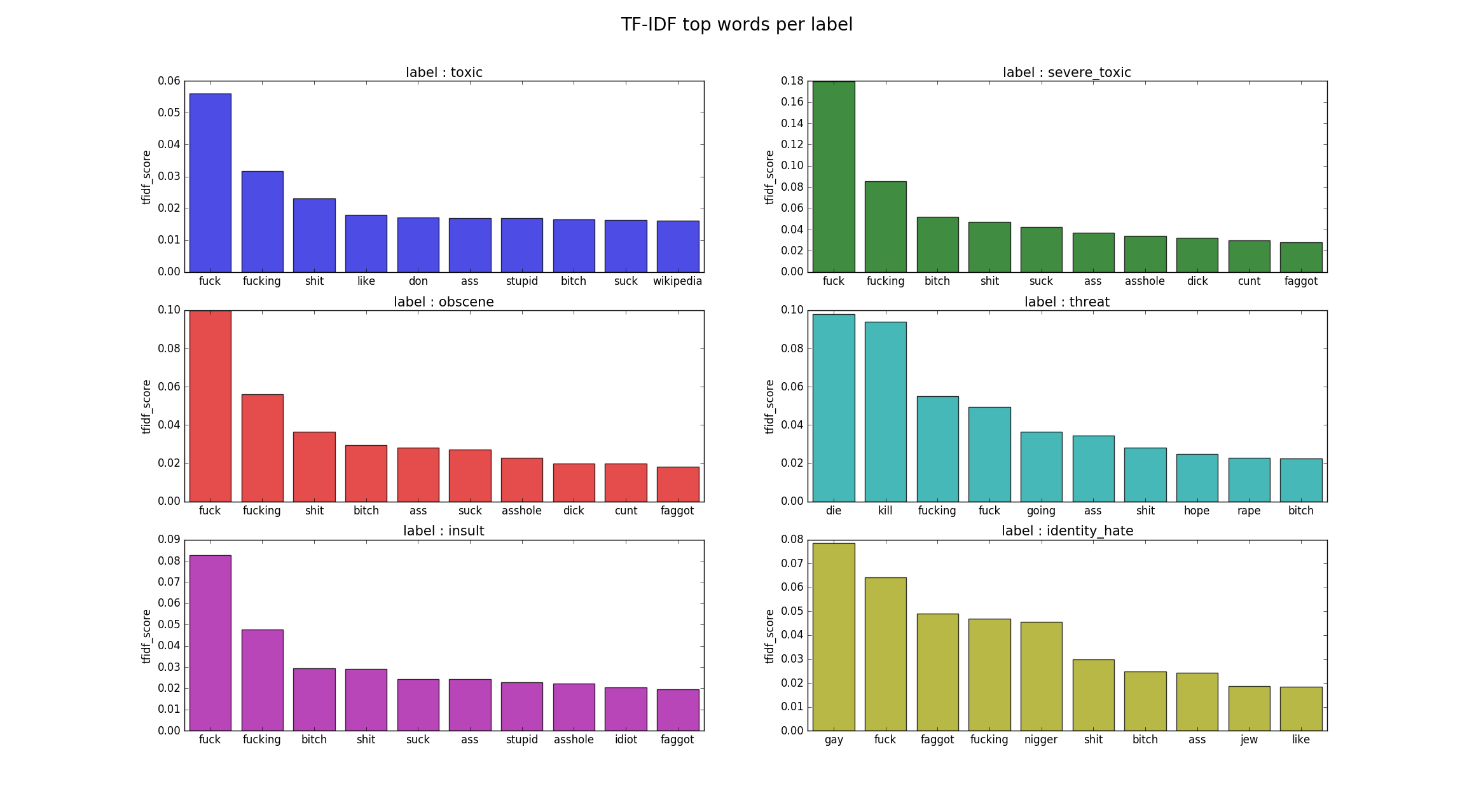
'fk', 'fking', 'fked', 'fkin', 'fker', 'fks',

* There are some non-English words
* There are ~200,000 unique words (including spelling mistakes, non-English, numbers ext.)
* To evaluate the number of words to use in the model we can plot cdf for the number of accumulated words

*Example*, taking the 3000 most popular words will get a coverage of 86% from all the words.







Preprocessing

Preprocessing is the most important part of the work. It has a great effect on the behaver of the model.

Since text preprocessing can take a lot of time and investigation which is out of the scope of this assignment, I will describe some preprocessing techniques -

* Delete all punctuation marks – I assume punctuation marks has no effect on the comment type
* Fix spelling mistakes – to reduce the dictionary
* Delete digits like IPs, dates, phone numbers ext. – I assume digits has no effect on the comment type
* Using word segmentations – some people forget to use spacebar
* Using lemmatization and/or stemming – to reduce the dictionary

Of course each action must be checked to see if the model improved

Architecture

The nature of the problem is of a time series, hence an RNN is a good starting point.

Since it's not a real time problem we can use information from the "future" and use Bidirectional layers.

The each input is a word so we need an embedded lyre at the beginning.

For this assignment I cheeked only one architecture –

1. Embedded layer converting a dictionary of 20,000 (we saw that even 3000 words is a nice covarage) words into a vector of size of 50.

The length of each comments will be maximum of 100 words

The initial values for the matrix were taking from *glove.6B.50d* model

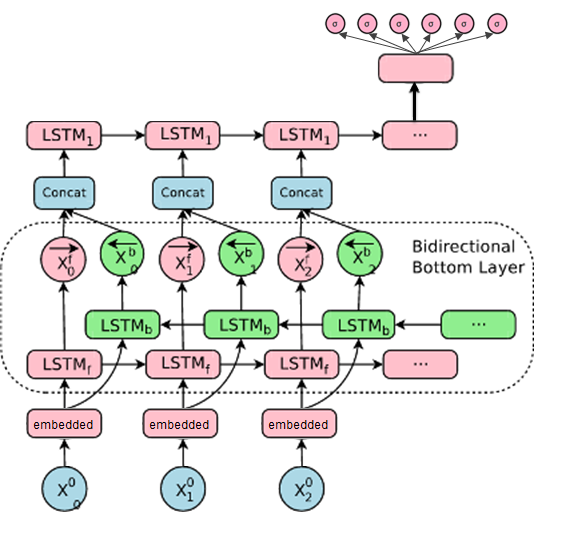
1. Bidirectional LSTM with cell size of 50 and dropout of 0.1
2. Dropout layer of 0.1
3. Batch normalization layer
4. Unidirectional LSTM with cell size of 50 and dropout of 0.1
5. Dropout layer of 0.1
6. Dense layer to 50
7. Final layer, dense to 6 sigmoid activation functions

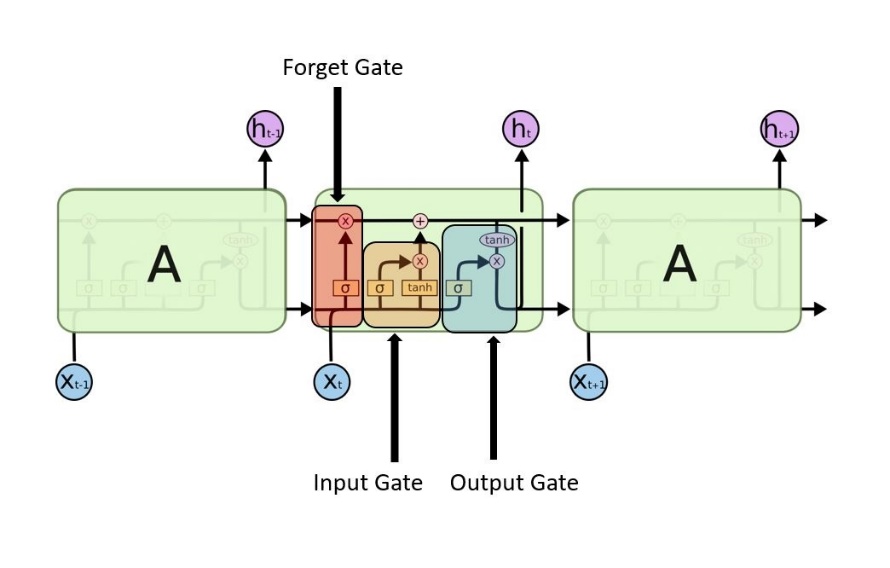
The training dataset was split to 60% training, 20% validation and 20% testing.

The splitting saved the ratio between no toxic at all to any toxic (1/10)

A smarter split might preserve the ratio of all 6 labels.

During the training we need to see that the model score converge, However since the training time is quite a lot I did only one epoch but it seems to converge during all the batches.





LSTM Cell

Model Summery –

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type)                 Output Shape Param #

=================================================================

input\_1 (InputLayer)         (None, 100) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

embedding\_1 (Embedding)      (None, 100, 50) 1000000

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

bidirectional\_1 (Bidirection (None, 100, 100)          40400

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_1 (Dropout)          (None, 100, 100) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

batch\_normalization\_1 (Batch (None, 100, 100)          400

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

lstm\_2 (LSTM)                (None, 50) 30200

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_2 (Dropout)          (None, 50) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_1 (Dense)              (None, 50) 2550

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_2 (Dense)              (None, 6) 306

=================================================================

Total params: 1,073,856

Trainable params: 1,073,656

Non-trainable params: 200

Final Optimization

For this assignment I didn’t do any optimizations.

Some optimizations to be done –

* Change embedding layer – there are more pre-trained embedded layers. Some might be more appropriate.
* Change the words dictionary size and the maximum comment length
* Change the cells to GRU
* Add 1D CNN after the embedded layer, it find some more information
* Play with all the hyper parameters

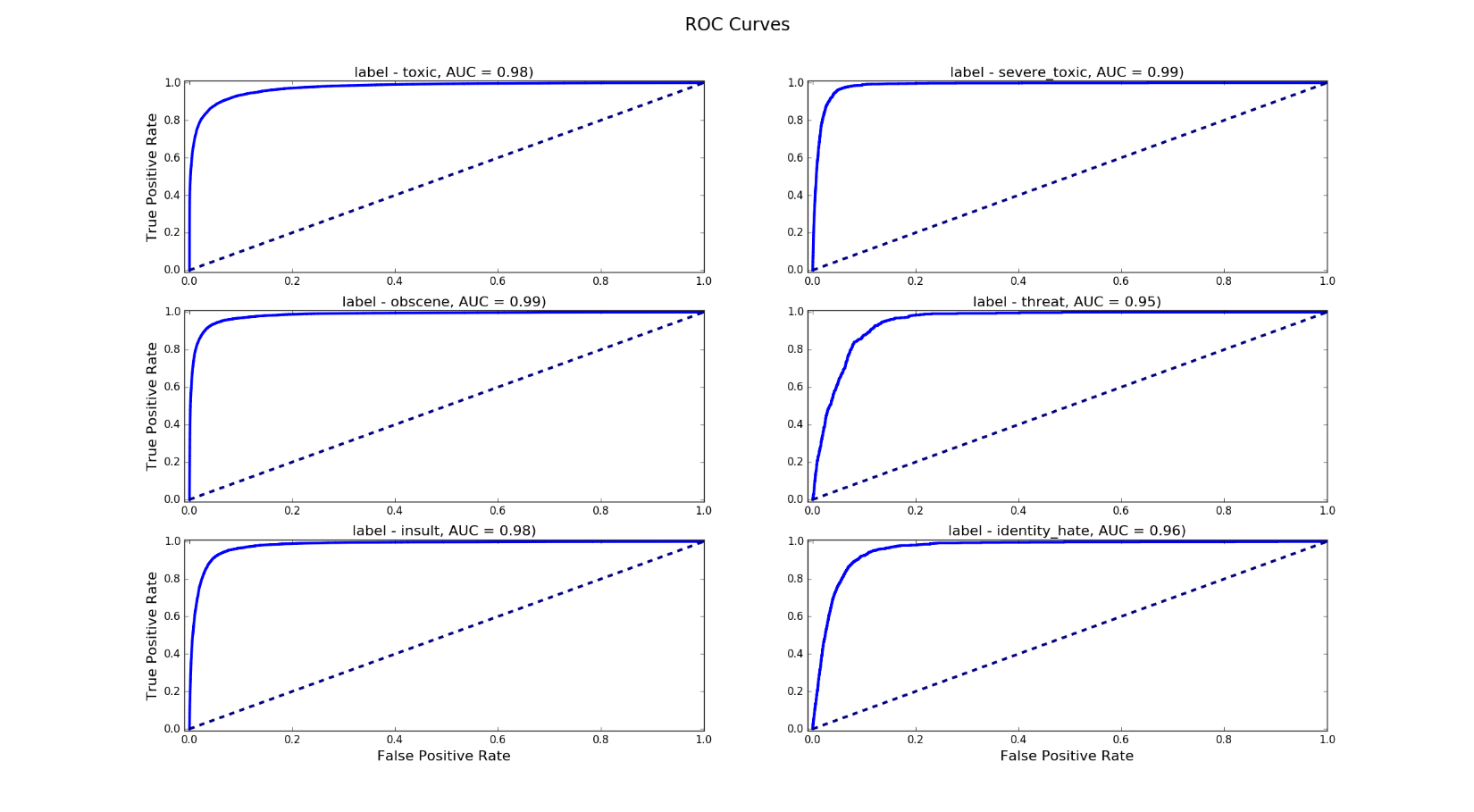
Evaluation

Since the dataset is highly skewed then a metric of accuracy is not enough (a model which will return 0a's will have accuracy of 90 %!!)

For the competition the evaluation metric was the average of all 6 AUC.

The average AUC of my model is **0.97**

I checked it on the training data + the testing data.



Code Arrangement

The code is split in to 6 folders –

1. data – contains all the datasets and glove model
2. docs – contains some references and this report
3. modeling – contains training file and evaluating file
4. plotting\_utils – contains all the plotting files
5. preprocessing – contain all the preprocessing files and a spelling mistake file which I downloaded from the internet
6. trained\_models – all the saved models

There are 3 file to run the code –

1. data\_analyze – do all the dataset investigation
2. train\_model – do the training, has hyper parameters configuration
3. evaluate\_model – calculate the AUC