Machine Learning Engineer Nanodegree
Capstone Project Report

Toxic Comment Classification

Kuan - Han Chen
July 15th, 2018

Definition

Project Overview

In this project, I will build a multi-headed model that's capable of detecting different types of of toxicity like threats, obscenity, insults. I will use naive bayes as my benchmark model. And get a baseline score through naive bayes model for my dataset, and compare with Text-CNN, which algorithm has higher AUC, ROC and accuracy.

Problem Statement

In this competition, you're challenged to build a multi-headed model that's capable of detecting different types of of toxicity like threats, obscenity, insults, and identity-based hate better than Perspective's current models. You'll be using a dataset of comments from Wikipedia's talk page edits. Improvements to the current model will hopefully help online discussion become more productive and respectful.

Metrics

- ROC: The AUC value is equivalent to the probability that a randomly chosen positive example is ranked higher than a randomly chosen negative example.
- AUC: In signal detection theory, a receiver operating characteristic (ROC), or simply ROC curve, is a graphical plot which illustrates the performance of a binary classifier system as its discrimination threshold is varied.

- Accuracy: How many number of correct prediction in total number of correct predictions.
- Calculation formula：

$$\mathrm{ROC}/_{\mathrm{AUC}} \quad : \quad \mathrm{TPR} = \frac{TP}{TP + FN} \quad , FPR = \frac{FP}{FP + TN}$$

$$\text{Accuracy} = \frac{Number\ \ of\ \ correct\ \ predictions}{Total\ \ number\ \ of\ \ predictions}$$

Datasets and Inputs

The datasets are provided by Jigsaw and Google for competeion in Kaggle. Them contain training set and testing set.

- train.csv(160k rows x 8 columns) - the training set, contains 8 columns, namely id, comment_text, toxic, severe_toxic, obscene, threat, insult and identity_hate, and last six columns comments with their binary labels
- test.csv(153k rows x columns) - the test set, contains 2 columns, namely id and comment_text .You must predict the toxicity probabilities for these comments. To deter hand labeling, the test set contains some comments which are not included in scoring.
- test_labels.csv(153k rows x 7 columns) - labels for the test data; value of -1 indicates it was not used for scoring. It contains 7 columns, namely id, toxic, severe_toxic, obscene, threat, insult and identity_hate.

Note: I am not sure if the training set is imbalanced dataset. If it is, I will take following approach, copy the few materials to make it equivalent to the majority of the data.

Analysis

Data Exploration and Visualization

There datasets contain a large number of text comments and classified into which type of toxicity like threats, obscenity, insults. When we import train sets and Visualize this data. Fig 1. Shows the number of each class. Then we found that data is an imbalanced data, the number of each type varies greatly. Use imbalance datasets to train your model, the accuracy measures may tell you that you have good accuracy, but the accuracy is reflecting the underlying each class exist non-uniform distributed. If we get the imbalance data, we can take following approach.

1. Change performance metric. Accuracy is not the appropriate metric to use in an imbalanced data, that would mislead us. We can use Precision, Recall, F1 – score and AUC/ROC as performance metric
2. Resample dataset. We can add copies of instances from the under-represented class, called over-sampling. And delete instances from the over-represented, called under-sampling.
3. Use different algorithm, like decision tress and SVM. These algorithm have less affected from imbalance data.

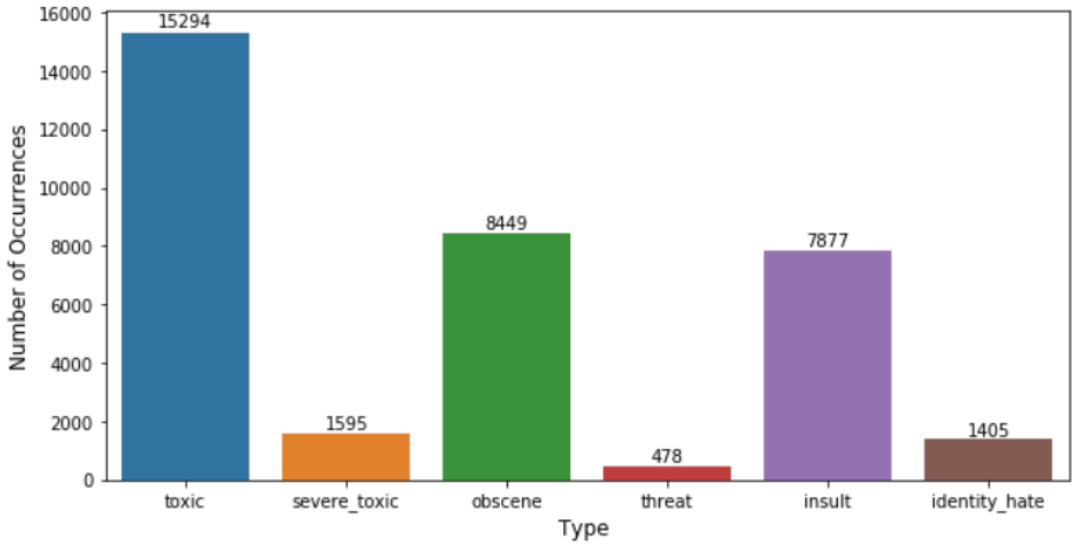


Fig 1. The number of each class

We take over-sampling to let the few materials to make it equivalent to the majority of the data. After sampling, shows in Fig 2.

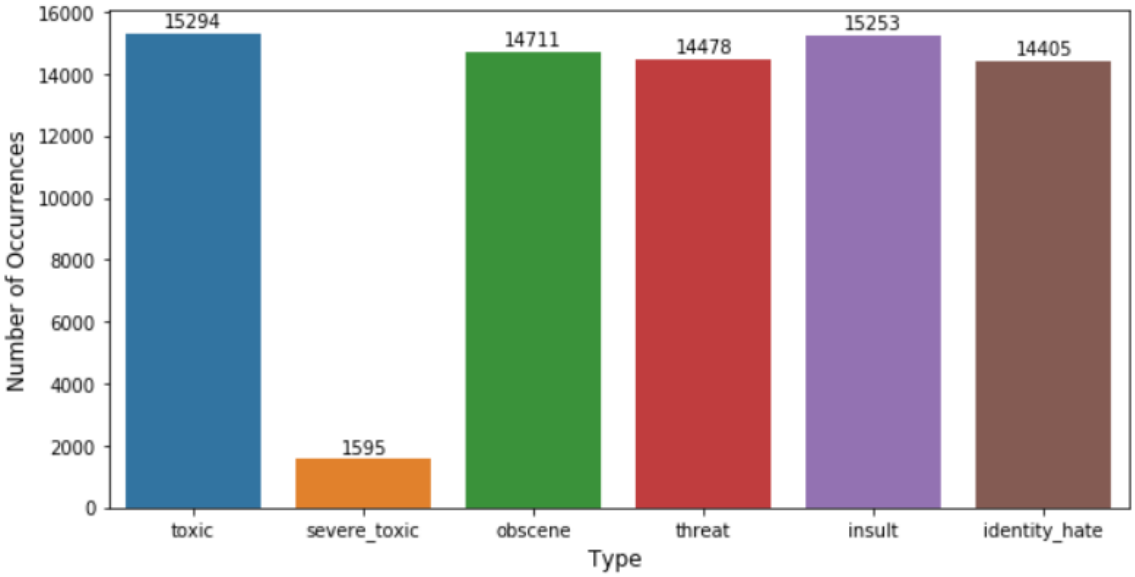


Fig. 2. After sampling, The number of each class

From Fig. 2. We still can found that the class of severe_toxic is still under-presented. We some simple way to analyze the relationship between the class of toxic and severe_toxic, shows in Fig. 3. We figure out that sever_toxic comments are always toxic, so that We increase the number of toxic and also increase the number of severe_toxic. So we don't take any action on the class of severe_toxic.

```
: print("The number of toxic get 1:",train[(train.toxic==1)].shape[0])
  print("The number of severe_toxic get 1:",train[(train.severe_toxic==1)].shape[0])
  print("The number of toxic and severe_toxic both get 1: ",
        train[(train.severe_toxic==1)&(train.toxic==1)].shape[0])
```

```
The number of toxic get 1: 15294
The number of severe_toxic get 1: 1595
The number of toxic and severe_toxic both get 1:  1595
```

Fig. 3. analyze the relationship between the class of toxic and severe_toxic

Algorithms and Techniques

The classifier is a TextCNN, which is state in “”Convolutional Neural Networks for Sentence Classification from Yoon Kim in 2014. The model architecture, shown in figure 4.

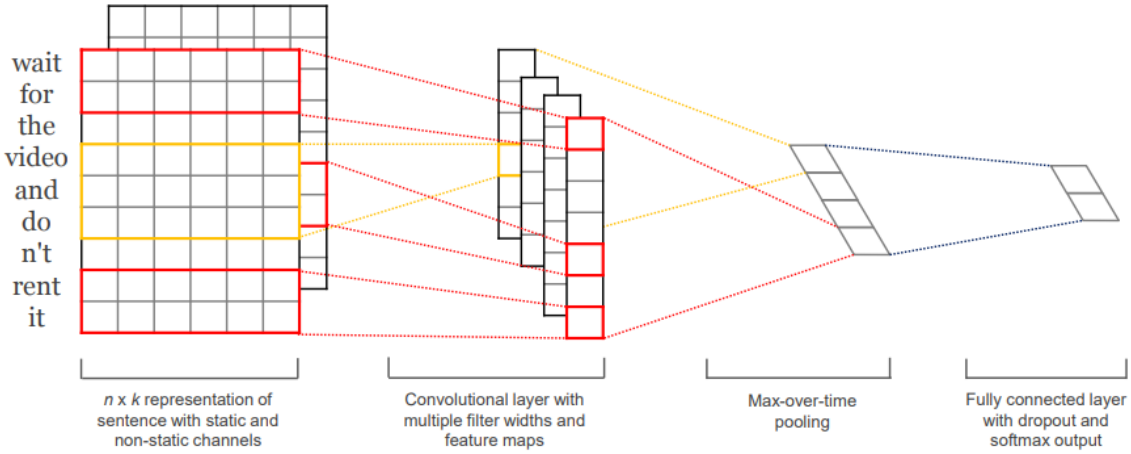


Fig. 4. Model architecture with two channels for an example sentence

Model architecture described as follows:

1. **Input layer**

As shown in the figure, the input layer is a matrix in which the word vectors corresponding to the words in the sentence are arranged in order (from top to bottom). Assuming that the sentence has n words and the dimension of the vector is k, then the matrix is n×k.

2. **Convolutional layer**

The input layer obtains several Feature Maps by convolution operation. The size of the convolution window is h×k, where h is the number of vertical words and k is the dimension of the word vector. Through such a large convolution, we will get several Feature Maps with a column number of 1.

3. **Pooling layer**

In the pooling layer, a method called Max-over-time Pooling is used in the text. This method simply proposes the largest value from the previous one-dimensional Feature Map, which explains that the maximum value represents the most important signal. It can be seen that this kind of filtering can solve the variable length sentence, which we

input. Because, no matter how many data from the Feature Map, we only need to extract the maximum value)

4. Fully connected layer + softmax

The output of the one-dimensional vector of the pooling layer is connected to a Softmax layer by means of full connection, and the Softmax layer can be set according to the needs of the task.

When we build TextCNN model, we will set some parameters. List some important parameter are as follows.

- max_features: the number of unique words.
- Maxlen: Unify the dimensions of all sentences. Make the shorter sentences has the same size with others by filling the shortfall by zeros.
- Batch size: the number of training examples in one forward/backward pass.
- Epochs: One Epoch is when an entire dataset is passed forward and backward through the neural network only once.

I will use Text-CNN as the main solution algorithm. Text-CNN has a good performance in natural language processing. The Text-CNN algorithm was published in this paper, Convolutional Neural Networks for Sentence Classification

Benchmark

To create an initial benchmark for the classifier, I use Decision Tree Classifier as benchmark model. We did not adjust any parameters from this benchmark model, we get the best accuracy is 0.846 and auc_roc score is 0.85

Methodology

Data Preprocessing

1. **Load Data**
2. **Data visualization**
3. **Data Exploration and Visualization**
4. **Resampling data(over-sampling)**
5. **Preprocessing**
    - Convert all letters to lowercase
    - Apostrophe replacement
    - Remove punctuation
    - Remove stopwords
    - Split into words

- Stem words

Implementation

1. **Load Data**
2. **Data visualization**
3. **Data Exploration and Visualization**
4. **Resampling data(over-sampling)**
5. **Preprocessing**
   - Convert all letters to lowercase
   - Apostrophe replacement
   - Remove punctuation
   - Remove stopwords
   - Split into words
   - Stem words
6. Transforming words to Sequences
7. TexCNN model
8. Result(accuracy, AUC/ROC score)

Refinement

I found that model performance did not perform well when we constantly increasing convolution-layer. The performance of three convolution-layer structure is better than two convolution-layer structure. But the performance of four convolution-layer structure is almost equal to three convolution-layer structure, so we choose three convolution-layer structure. As dense layer, I think use too many dense layer will cause over-fitting. I use one dense layer structure as my final model.

Results

Model Evaluation and Validation

We constructed three different models architecture below and its performance. Compare the performance of these three models, model 1 and model 2 both have higher performance than model 3. So we will pick one between model 1 and model 2. But model 2 has much more total parameters than model 1, it means it will occupy more resources lead to lower efficiency, when we train model. In comprehensive survey, I choose model 1 as quasi-final model, and I will make some fine-tuning to this model to make perform better than original one.

Model 1		
Layer	**Output Shape**	**Parameters**
InputLayer	(None, 100)	0
Embedding	(None, 100, 256)	512000
Conv1D_1	(None, 100, 128)	65664
Conv1D_2	(None, 100, 128)	98432
Conv1D_3	(None, 100, 128)	131200
GlobalMP1D_1	(None, 128)	0
GlobalMP1D_2	(None, 128)	0
GlobalMP1D_3	(None, 128)	0
Concatenate	(None, 384)	0
Dropout	(None, 384)	0
Dense	(None, 128)	49280
Dense	(None, 32)	4128
Dense	(None, 6)	198
Total Parameters		860,902

Model 1	
Training Accuracy	0.95
Training ROC/AUC Score	0.93
Valid Accuracy	0.92
Valid ROC/AUC Score	0.92

Model 2		
Layer	**Output Shape**	**Parameters**
InputLayer	(None, 100)	0
Embedding	(None, 100, 256)	512000
Conv1D_1	(None, 100, 128)	65664
Conv1D_2	(None, 100, 128)	98432
Conv1D_3	(None, 100, 128)	131200
Conv1D_4	(None, 100, 128)	163968
GlobalMP1D_1	(None, 128)	0
GlobalMP1D_2	(None, 128)	0
GlobalMP1D_3	(None, 128)	0
GlobalMP1D_4	(None, 128)	0
Concatenate	(None, 512)	0
Dropout	(None, 384)	0
Dense	(None, 128)	65664
Dense	(None, 32)	4128
Dense	(None, 6)	198
Total Parameters		1,041,254

Model 2	
Training Accuracy	0.95
Training ROC/AUC Score	0.93
Valid Accuracy	0.92
Valid ROC/AUC Score	0.92

Model 3		
Layer	**Output Shape**	**Parameters**
InputLayer	(None, 100)	0
Embedding	(None, 100, 256)	512000
Conv1D_1	(None, 100, 128)	65664
Conv1D_2	(None, 100, 128)	98432
GlobalMP1D_1	(None, 128)	0
GlobalMP1D_2	(None, 128)	0
Concatenate	(None, 512)	0
Dropout	(None, 384)	0
Dense	(None, 128)	32896
Dense	(None, 32)	4128
Dense	(None, 6)	198
Total Parameters		713,318

Final Model	
Training Accuracy	0.89
Training ROC/AUC Score	0.87
Valid Accuracy	0.86
Valid ROC/AUC Score	0.82

Justification

We choose the model has best performance among the three models. Then we make some fine-tuning with this model, take it as final model.

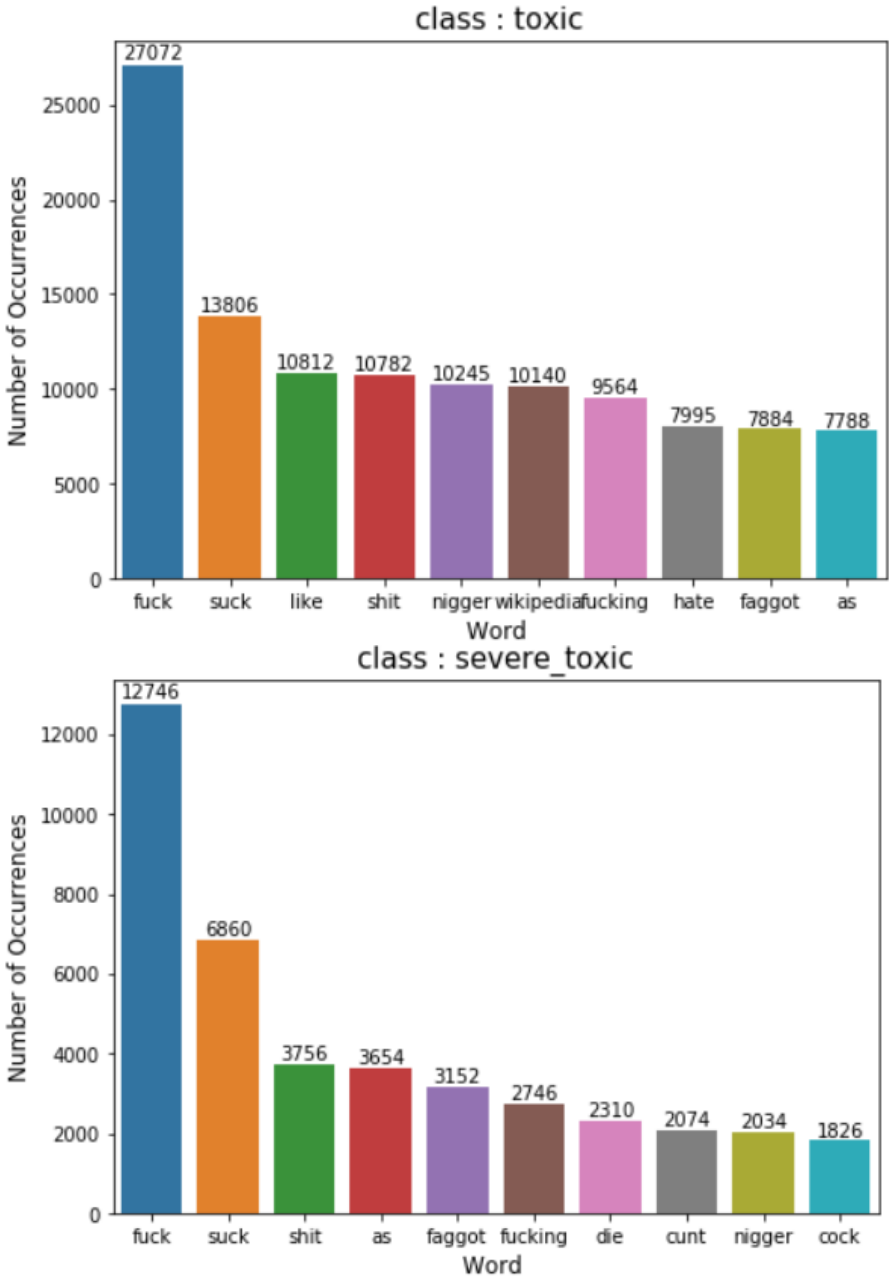
Final Model		
Layer	**Output Shape**	**Parameters**
InputLayer	(None, 100)	0
Embedding	(None, 100, 256)	512000
Conv1D_1	(None, 100, 256)	131328
Conv1D_2	(None, 100, 256)	196864
Conv1D_3	(None, 100, 256)	262400
GlobalMP1D_1	(None, 256)	0
GlobalMP1D_2	(None, 256)	0
GlobalMP1D_3	(None, 256)	0
Concatenate	(None, 384)	0
Dense	(None, 32)	24608
Dense	(None, 6)	198
Total Parameters		1,127,398

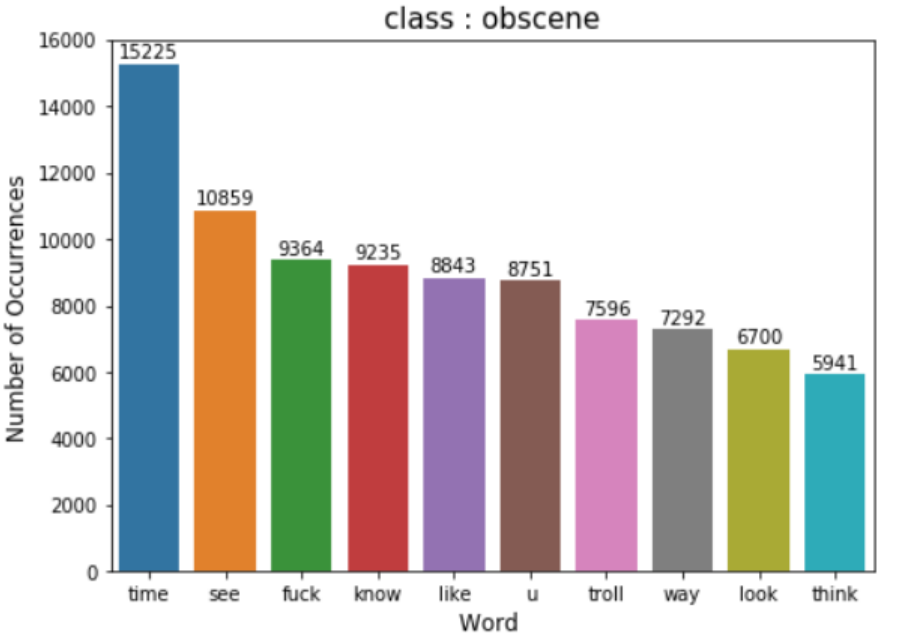
Final Model	
Training Accuracy	0.99
Training ROC/AUC Score	0.97
Valid Accuracy	0.96
Valid ROC/AUC Score	0.97

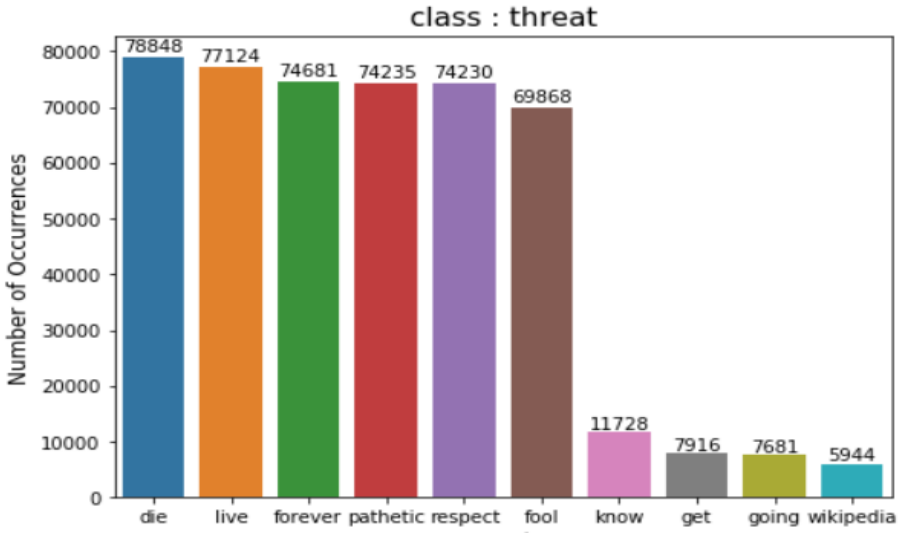
Conclusion

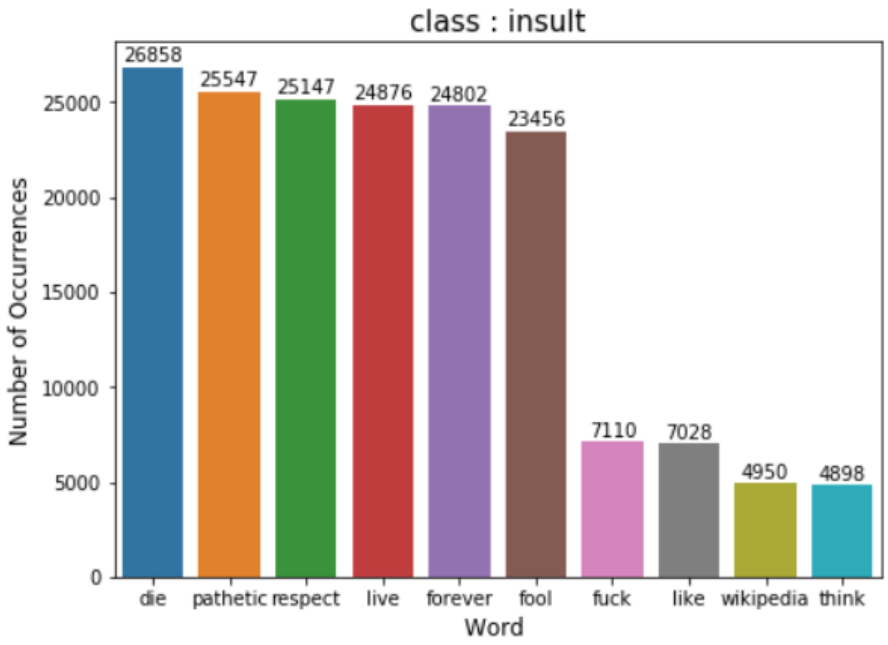
Free-Form Visualization

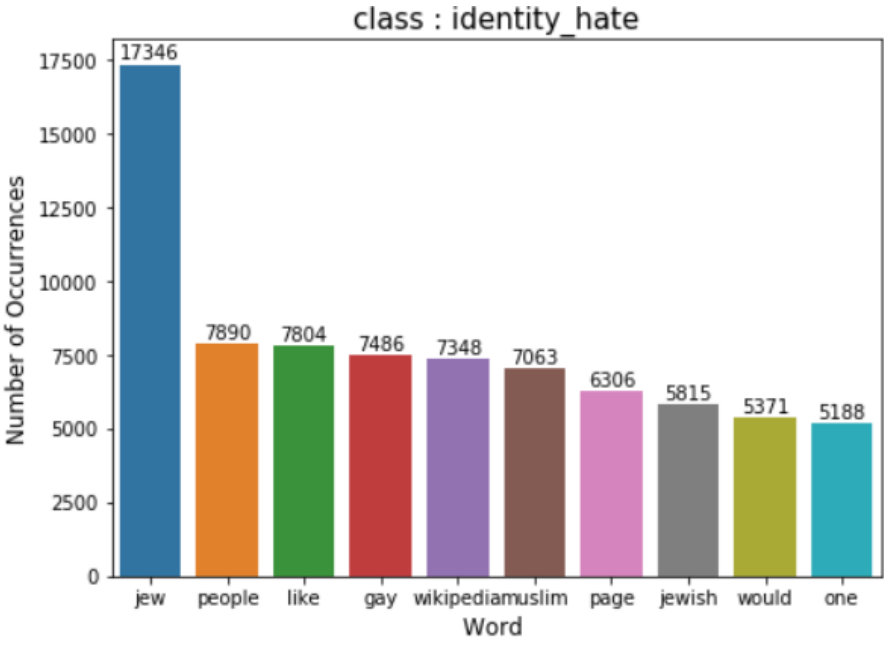
We find out which words are often found in specific classes and visualize these top 10 words from each class.




class : obscene
Number of Occurrences
Word
time
see
fuck
know
like
u
troll
way
look
think
15225
10859
9364
9235
8843
8751
7596
7292
6700
5941


class : threat
Number of Occurrences
die
live
forever
pathetic
respect
fool
know
get
going
wikipedia
78848
77124
74681
74235
74230
69868
11728
7916
7681
5944





Reflection

Because testing set doesn't come with correct answer that will affect accuracy and AUC/ROC score. Avoid the above things happening, we create a validation set from training set to validate your model.

Reference

1. https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge#description
2. http://www.aclweb.org/anthology/D14-1181
3. http://www.jeyzhang.com/cnn-apply-on-modelling-sentence.html
4. https://keras.io/layers/convolutional/
5. https://www.kaggle.com/jagangupta/stop-the-s-toxic-comments-eda/notebook
6. https://blog.csdn.net/fendouaini/article/details/79919322