Machine Learning Engineer Nanodegree
Capstone Project Report

Toxic Comment Classification

Kuan - Han Chen
July 15th, 2018

Definition

Project Overview

With the rapid development of online media, People are free to express their opinions on the online media. But it's hard to stop people from saying something malicious. Discussing things will be difficult. The threat of abuse and harassment online means that many people stop expressing themselves and give up on seeking different opinions. Many Platforms struggle to effectively facilitate conversations, leading many communities to limit or completely shut down user comments.
The goal of this project is to build a model to detect probability of different levels of toxicity like threats, obscenity, insults, and identity hate on any textual comments. This model will help platform to monitor people's comments, and may filter the malicious comment immediately. Create a platform for people to speak their own opinions freely and respect other's.
This project is taken from Kaggle platform. I will build a multi-headed model that's capable of detecting different types of of toxicity like threats, obscenity, insults. The field of study that focuses on the interactions between human language and computers is called Natural Language Processing. This process is massive and complex, so we must rely on the algorithms behind machine learning to solve the problem.

Ref:
1. https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge
2. https://machinelearningmastery.com/tactics-to-combat-imbalanced-classes-in-your-machine-learning-dataset/
3.

Problem Statement

Short-text classification is an important task in natural language processing, including selective dissemination of information to information consumers, spam filtering, or sentiment analysis. Many different approaches have been developed for short-text classification. In machine learning models, such as Naive Bayes Classifier, Support Vector Machine(SVM), Decision Tree Classifier. In deep Neural Networks, such as Convolutional Neural Network (CNN), Long Short Term Model (LSTM), Gated Recurrent Unit (GRU), Recurrent Convolutional Neural Network (RCNN). Among these different models, Convolutional Neural Network (CNN) architecture have also demonstrated profound efficiency in NLP tasks including sentiment classification. The method of classifier we will use in this project is TextCNN, which was proposed by Y Kim in ''Convolutional Neural Networks for Sentence Classification'', 2014.

First, I will analyze the datasets we got from Kaggle. Then preprocess the data, remove low value data, it will increase the efficiency of the training model. Last, build the model and analyze the result.

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

Metrics

- ROC: The AUC(Area Under the Curve) 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：

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

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

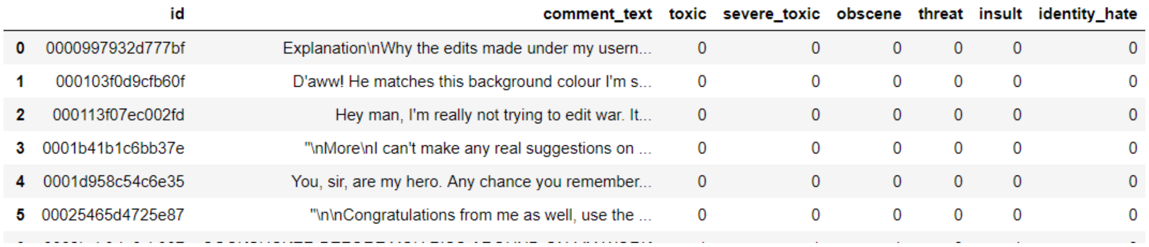
Accuracy is the measure that can be useful when the problem has well balanced. After resample data, data approaching balanced. I still take accuracy as the performance metric. The **ROC curve** (or **receiver operating characteristics curve**) is a popular graphical measure for assessing the performance or the accuracy of a classifier, which corresponds to the total proportion of correctly classified observations. This metric is more truthful than the metric of accuracy in an imbalanced data. The **Area Under the Curve** (**AUC**) summarizes the overall performance of the classifier, over all possible probability cutoffs. Despite the processing by resampling, the data is still imbalanced, so I will use ROC/AUC score to measure the performance of the model.

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. train.csv visualized by pandas, shown in Fig. 1.
- 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.

.



	id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
0	0000997932d777bf	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0
2	000113f07ec002fd	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0
3	0001b41b1c6bb37e	"\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	0
4	0001d958c54c6e35	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0
5	00025465d4725e87	"\n\nCongratulations from me as well, use the ...	0	0	0	0	0	0

Fig. 1. train.csv visualized by pandas

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 algorithms have

less affected from imbalance data.

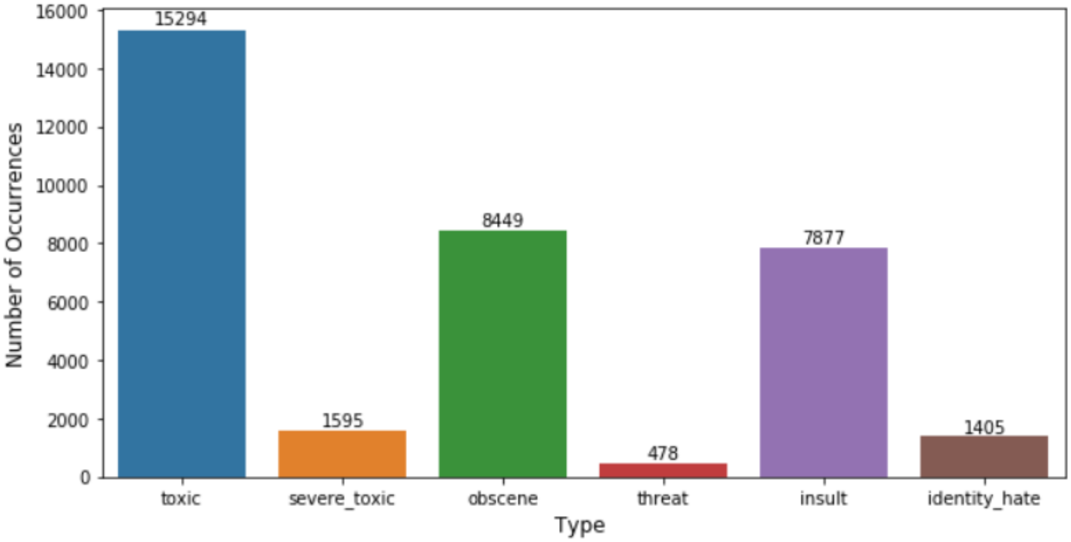


Fig 1. The number of each class from train set.

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.

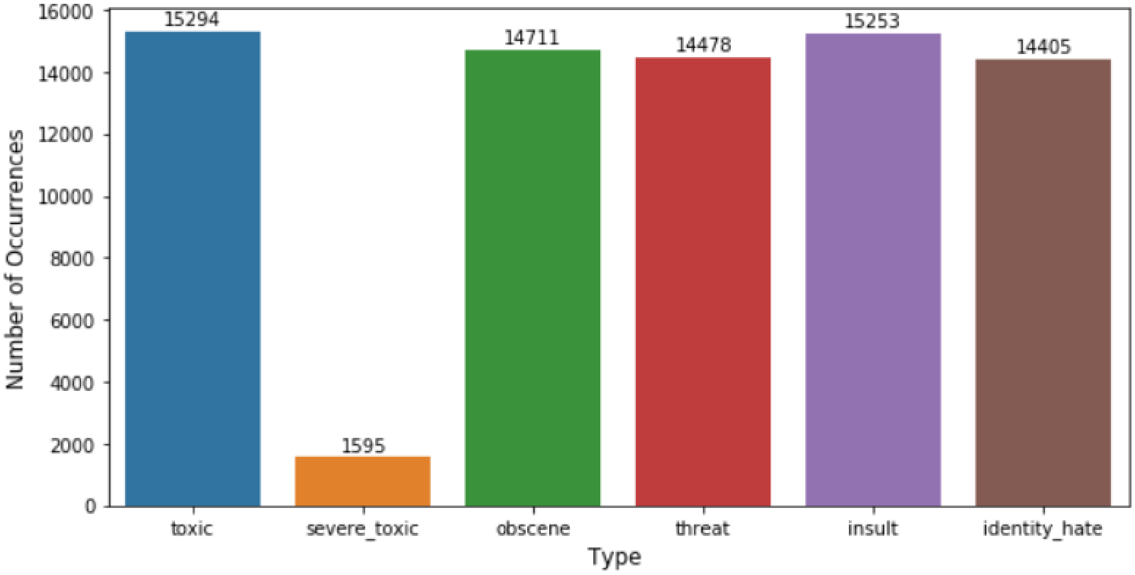
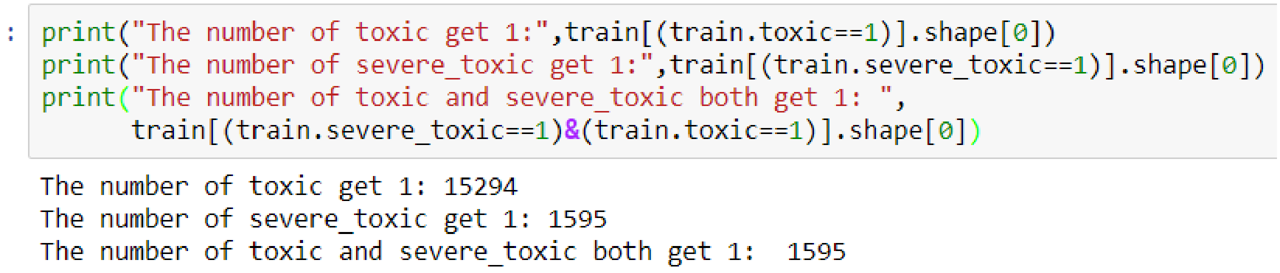


Fig. 2. After sampling, The number of each class train set.

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.

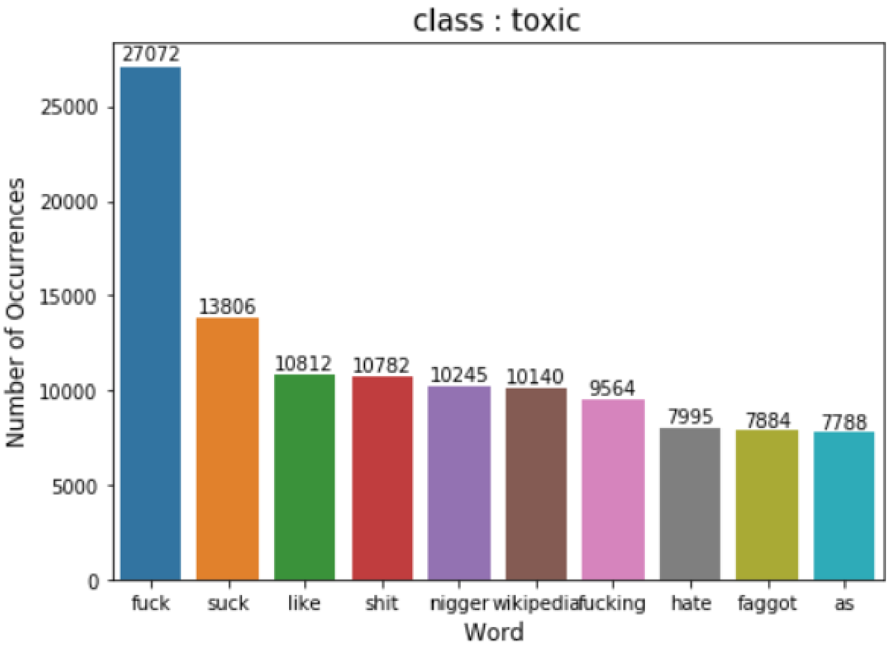


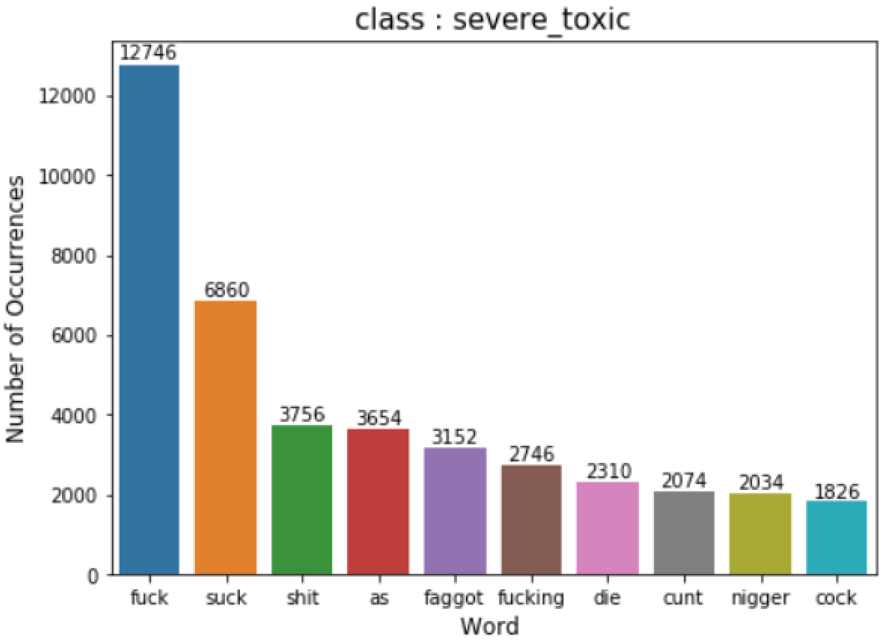
```
print("The number of toxic get 1:",train[(train.toxic==1)].shape[0])
print("The number of severe_toxic get 1:",train[(train.severe_toxic==1)].shape[0])
print("The number of toxic and severe_toxic both get 1: ",
      train[(train.severe_toxic==1)&(train.toxic==1)].shape[0])
```

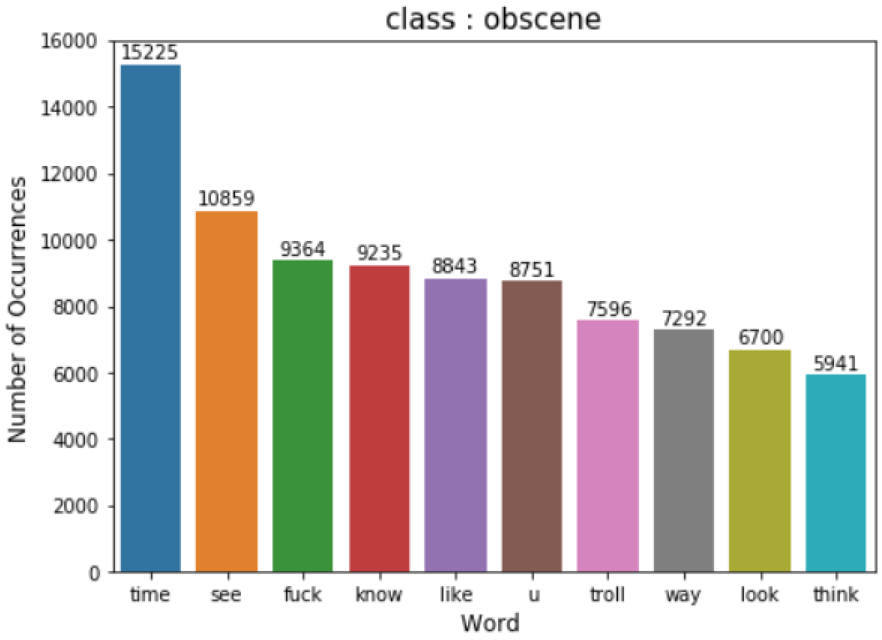
```
The number of toxic get 1: 15294
The number of severe_toxic get 1: 1595
The number of toxic and severe_toxic both get 1:  1595
```

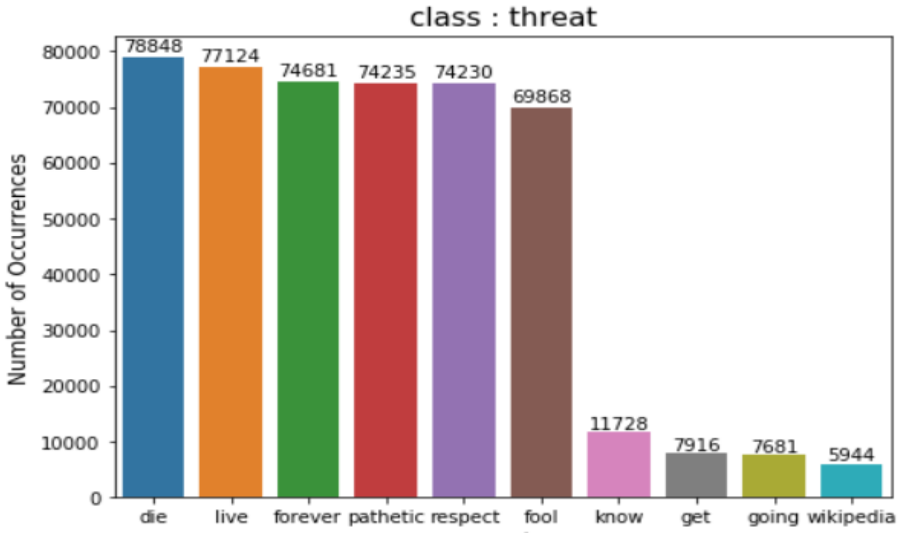
Fig. 3. analyze the relationship between the class of toxic and severe_toxic

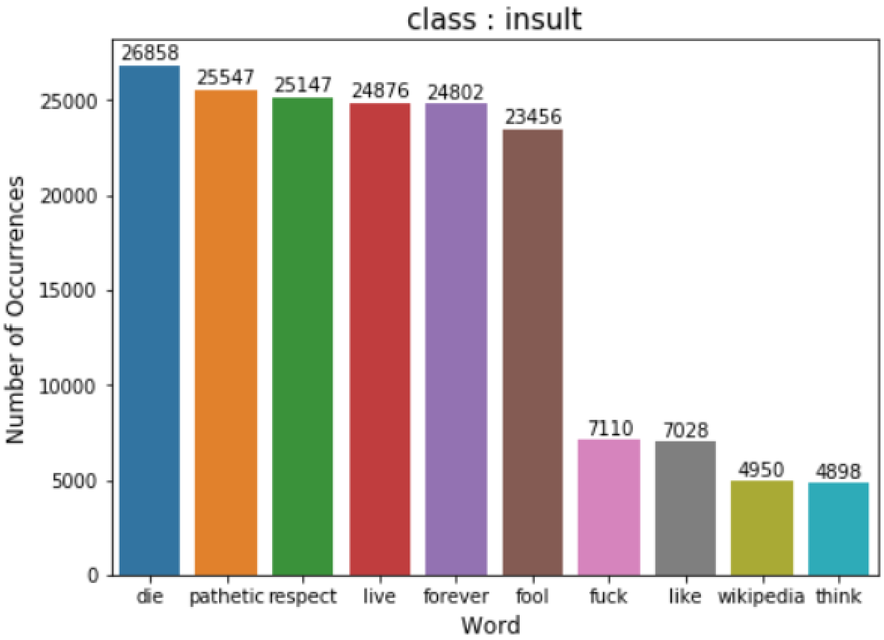
We find out which words are often found in specific classes and visualize these top 10 words from each class in train set are shown below.

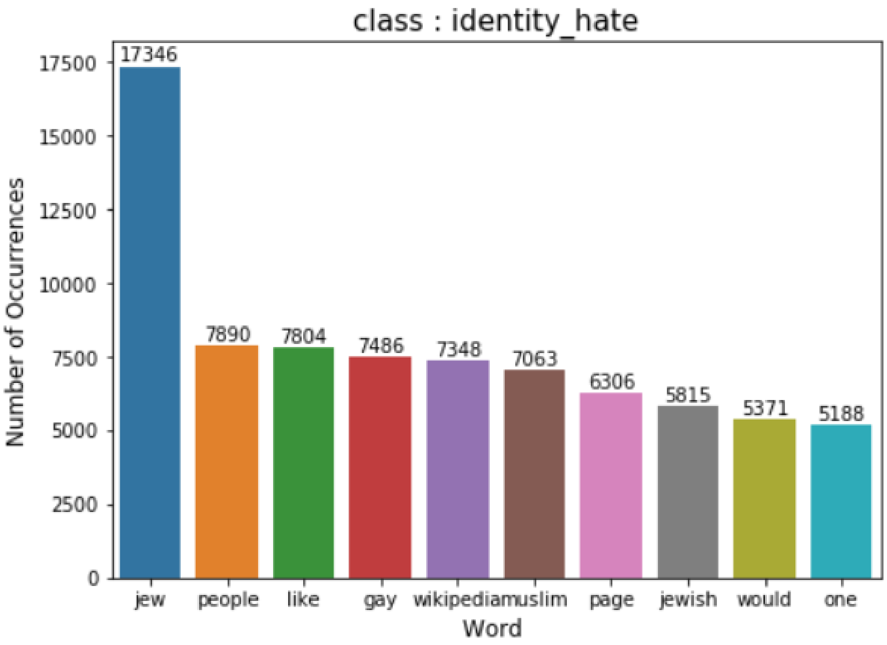



class : severe_toxic
Number of Occurrences
0
2000
4000
6000
8000
10000
12000
Word
fuck 12746
suck 6860
shit 3756
as 3654
faggot 3152
fucking 2746
die 2310
cunt 2074
nigger 2034
cock 1826


class : obscene
Number of Occurrences
0
2000
4000
6000
8000
10000
12000
14000
16000
Word
time 15225
see 10859
fuck 9364
know 9235
like 8843
u 8751
troll 7596
way 7292
look 6700
think 5941


class : threat
Number of Occurrences
0
10000
20000
30000
40000
50000
60000
70000
80000
78848
77124
74681
74235
74230
69868
11728
7916
7681
5944
die
live
forever
pathetic
respect
fool
know
get
going
wikipedia


class : insult
Number of Occurrences
0
5000
10000
15000
20000
25000
26858
25547
25147
24876
24802
23456
7110
7028
4950
4898
die
pathetic
respect
live
forever
fool
fuck
like
wikipedia
think
Word



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 Fig. 4.

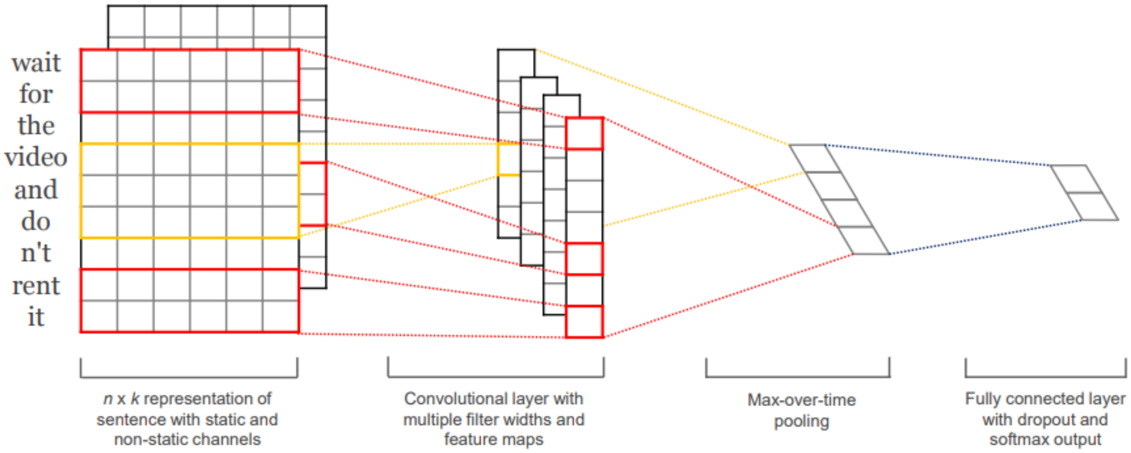

wait
for
the
video
and
do
n't
rent
it
n x *k* representation of sentence with static and non-static channels
Convolutional layer with multiple filter widths and feature maps
Max-over-time pooling
Fully connected layer with dropout and softmax output

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

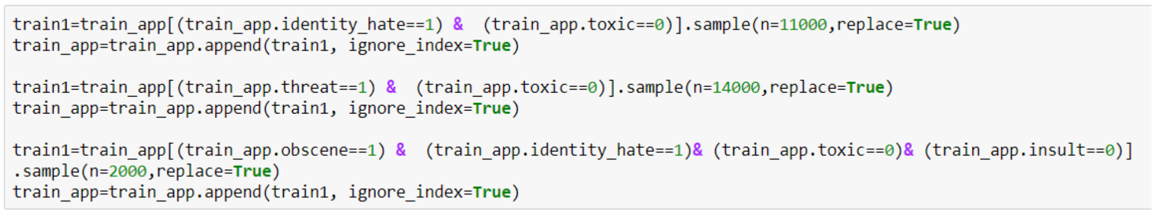
Methodology

Data Preprocessing

1. **Load Data**
2. **Data visualization**
3. **Data Exploration and Visualization**
4. **Resampling data(over-sampling)**

   I filter out the data that the class of identity_hate is equal to 1 and the class of

toxic is equal to 0, random copy and add to my train set.
Here is the process of resample data. No fixed practice, you can try other combinations.

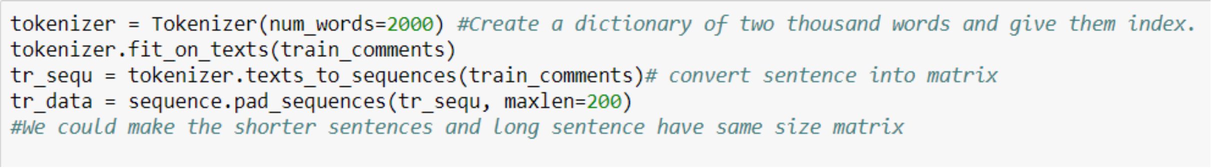


```
train1=train_app[(train_app.identity_hate==1) &  (train_app.toxic==0)].sample(n=11000,replace=True)
train_app=train_app.append(train1, ignore_index=True)

train1=train_app[(train_app.threat==1) &  (train_app.toxic==0)].sample(n=14000,replace=True)
train_app=train_app.append(train1, ignore_index=True)

train1=train_app[(train_app.obscene==1) &  (train_app.identity_hate==1)& (train_app.toxic==0)& (train_app.insult==0)]
.sample(n=2000,replace=True)
train_app=train_app.append(train1, ignore_index=True)
```

5. **Data - Preprocessing**
   - Convert all letters to lowercase
   - Apostrophe replacement
     you're --> you are
     what's --> what is
     We turn “you’re ” into “you are”, because “you’re” is not in the stopwords package.
   - Remove punctuation
   - Remove stopwords
     - ➢ Stopwords are very common words in a language, like “the”, “is”, “she”. The value of stopwods in txt-classification is low, we will be drop them from the data. We get a set of English stop words from nltk package.
   - Tokenization
     - ➢ Tokenization is the process of breaking up the given text into units called tokens.
       I love Udacity -> [“I”, ”love”, ”Udacity”]
   - Lemmatization
     - ➢ In English, words appear in different forms. Look as these two sentence: I have a cat & I have two cats, both sentence talks about cat, but the use in different inflection. Turn these two into the same one by Lemmatization step.
       dogs -> dog, cats -> cat
       doing -> do, done -> do
6. Transforming words to Sequences
   - ➢ Building a dictionary, coverting word into vector and every word has same shape of matrix.

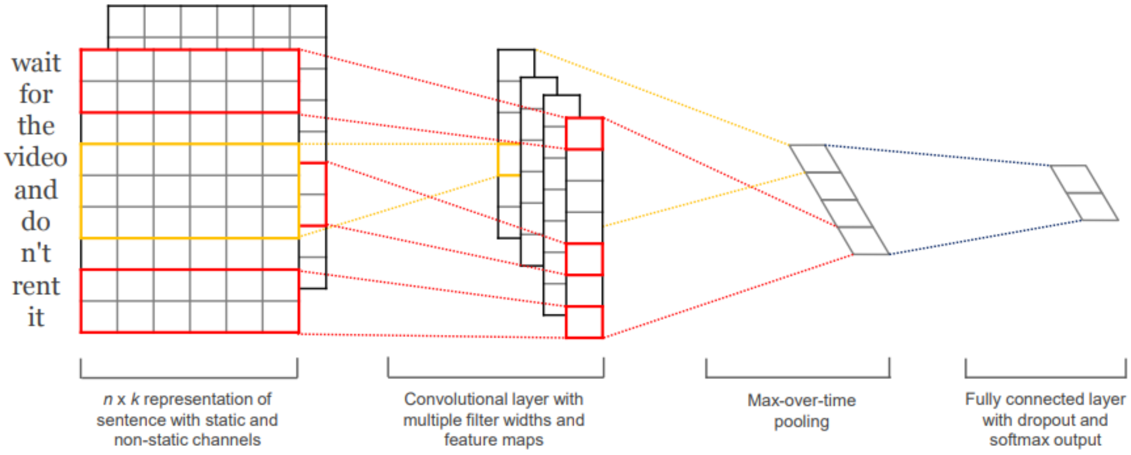


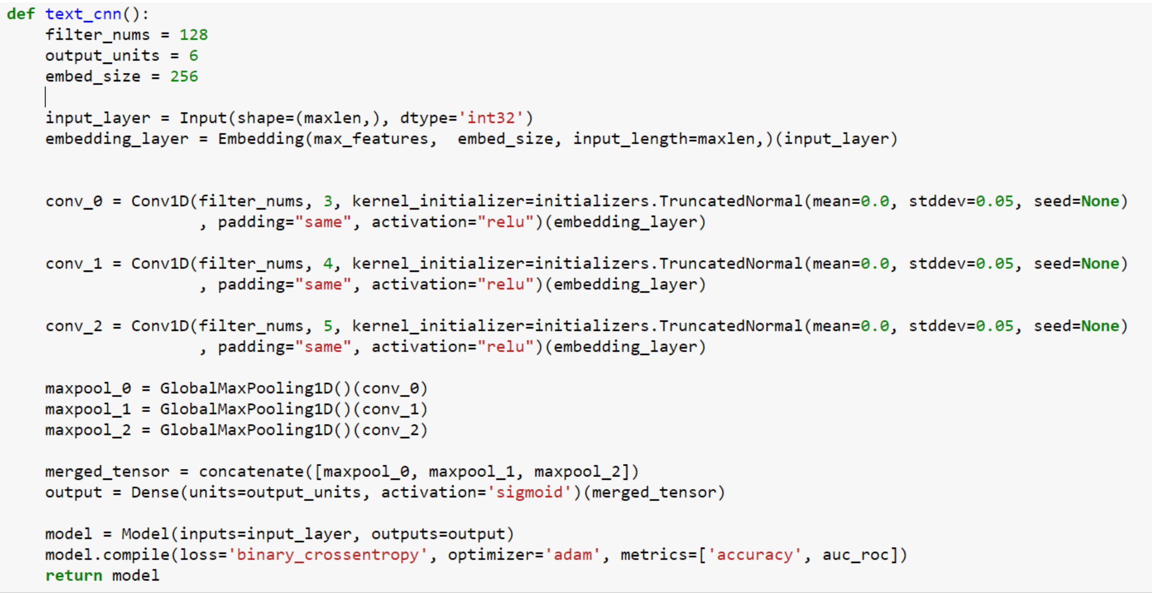
```
tokenizer = Tokenizer(num_words=2000) #Create a dictionary of two thousand words and give them index.
tokenizer.fit_on_texts(train_comments)
tr_sequ = tokenizer.texts_to_sequences(train_comments)# convert sentence into matrix
tr_data = sequence.pad_sequences(tr_sequ, maxlen=200)
#We could make the shorter sentences and long sentence have same size matrix
```

Implementation

7. TextCNN model

- ➢ I use Keras framework to build TextCNN model. My TextCNN model is built according to the model architecture below, this model architecture is proposed by Yoon Kim in “Convolutional Neural Networks for Sentence Classification” paper.


wait
for
the
video
and
do
n't
rent
it
n x *k* representation of sentence with static and non-static channels
Convolutional layer with multiple filter widths and feature maps
Max-over-time pooling
Fully connected layer with dropout and softmax output



```
def text_cnn():
    filter_nums = 128
    output_units = 6
    embed_size = 256
    
    input_layer = Input(shape=(maxlen,), dtype='int32')
    embedding_layer = Embedding(max_features,  embed_size, input_length=maxlen,)(input_layer)


    conv_0 = Conv1D(filter_nums, 3, kernel_initializer=initializers.TruncatedNormal(mean=0.0, stddev=0.05, seed=None)
                    , padding="same", activation="relu")(embedding_layer)

    conv_1 = Conv1D(filter_nums, 4, kernel_initializer=initializers.TruncatedNormal(mean=0.0, stddev=0.05, seed=None)
                    , padding="same", activation="relu")(embedding_layer)

    conv_2 = Conv1D(filter_nums, 5, kernel_initializer=initializers.TruncatedNormal(mean=0.0, stddev=0.05, seed=None)
                    , padding="same", activation="relu")(embedding_layer)

    maxpool_0 = GlobalMaxPooling1D()(conv_0)
    maxpool_1 = GlobalMaxPooling1D()(conv_1)
    maxpool_2 = GlobalMaxPooling1D()(conv_2)

    merged_tensor = concatenate([maxpool_0, maxpool_1, maxpool_2])
    output = Dense(units=output_units, activation='sigmoid')(merged_tensor)

    model = Model(inputs=input_layer, outputs=output)
    model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy', auc_roc])
    return model
```

- ➢ Embeding_layer: project the words to a defined vector space
- ➢ Convolution layer: You are free to increase or decrease the number of layers, and free to adjust the parameters.
- ➢ Max pooling layer: receive corresponding the output of convolution layer.
- ➢ Concatenate layer: concatenates a list of inputs from Max pooling layer.
- ➢ Dense layer: Also called fully connected layer. You are free to increase or decrease the number of layers, and free to adjust the parameters.
- ➢ model.complie: choose loos function, optimizer and metrics

8. Result(accuracy, AUC/ROC score)

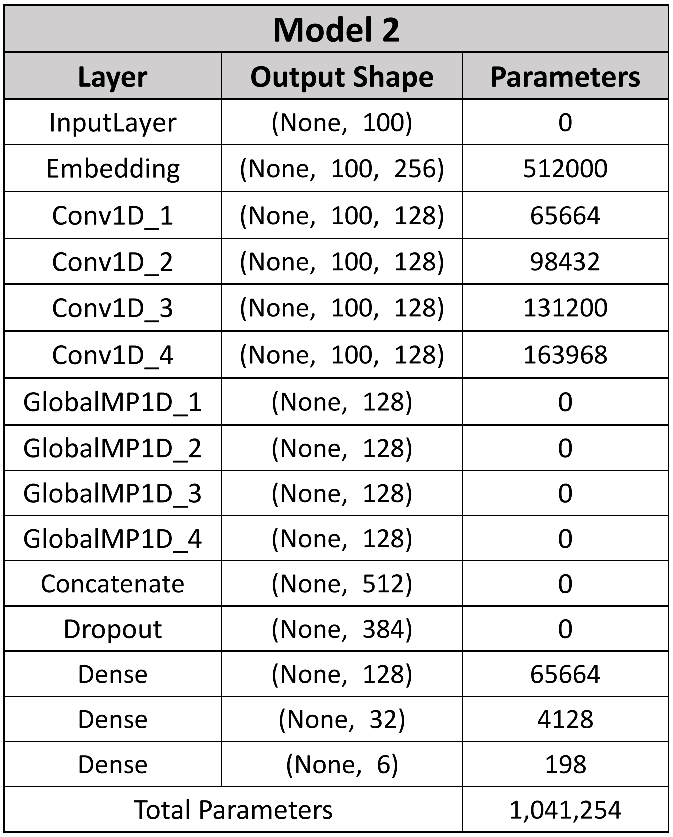
Refinement

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. Incomprehensive 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. The model architecture and its results are shown in Table 1.

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

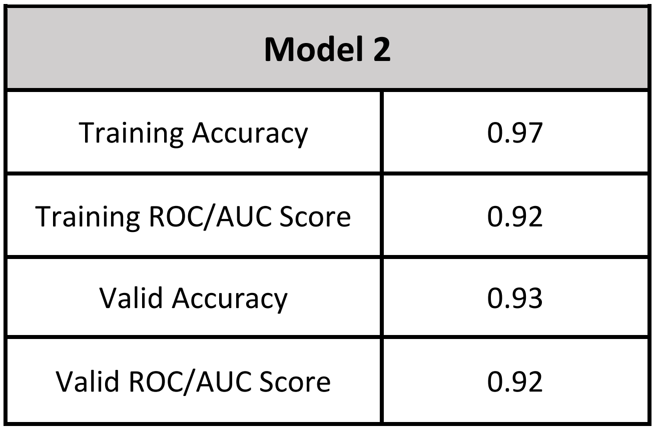
Model 1	
Training Accuracy	0.97
Training ROC/AUC Score	0.92
Valid Accuracy	0.93
Valid ROC/AUC Score	0.93

Table 1



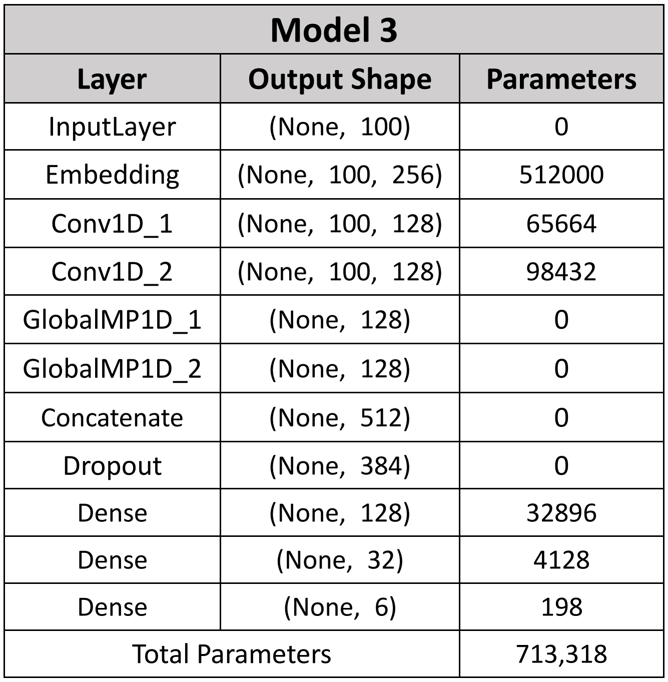
Model 2

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



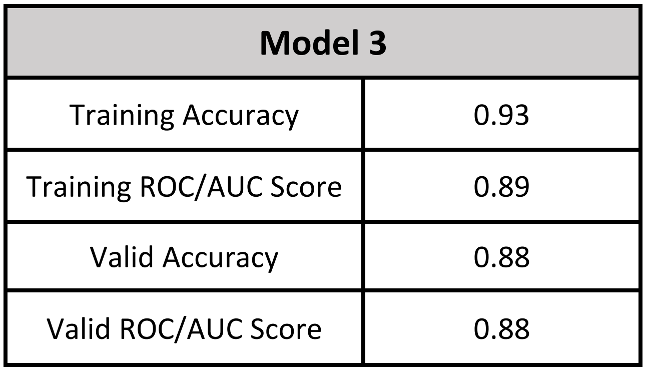
Model 2

Training Accuracy	0.97
Training ROC/AUC Score	0.92
Valid Accuracy	0.93
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



Model 3

Training Accuracy	0.93
Training ROC/AUC Score	0.89
Valid Accuracy	0.88
Valid ROC/AUC Score	0.88

I found that model performance did not perform well when we constantly increasing convolution-layer, the performance of three convolution-layer structure(model 1) and four convolution-layer structure(model 2) have very close performance. 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.

Results

Model Evaluation and Validation

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. I use k-fold cross validation to check the robustness model or not. I will provide the architecture of model and its performance, shown in Table 2. At k-fold cross validation, we collect the result from every fold validation and calculated average, show in Table 3. From the result of cross validation, we can sure that our model is robustness model.

Table 2

Final Model		
Layer	**Output Shape**	**Parameters**
InputLayer	(None, 100)	0
Embedding	(None, 100, 256)	512000
Conv1D_1	(None, 100, 128)	131328
Conv1D_2	(None, 100, 128)	196864
Conv1D_3	(None, 100, 128)	262400
GlobalMP1D_1	(None, 128)	0
GlobalMP1D_2	(None, 128)	0
GlobalMP1D_3	(None, 128)	0
Concatenate	(None, 384)	0
Dense	(None, 6)	2310
Total Parameters		809,606

Final Model	
Training Accuracy	0.99
Training ROC/AUC Score	0.99
Valid Accuracy	0.98
Valid ROC/AUC Score	0.98

Table 3

	Accuracy	ROC/ACU Score
1-Fold	0.976	0.974
2-Fold	0.978	0.975
3-Fold	0.976	0.974
4-Fold	0.977	0.974
5-Fold	0.988	0.974
6-Fold	0.978	0.975
7-Fold	0.978	0.974
8-Fold	0.982	0.971
9-Fold	0.828	0.915
10-Fold	1.000	0.974
avg	0.965	0.968

Justification

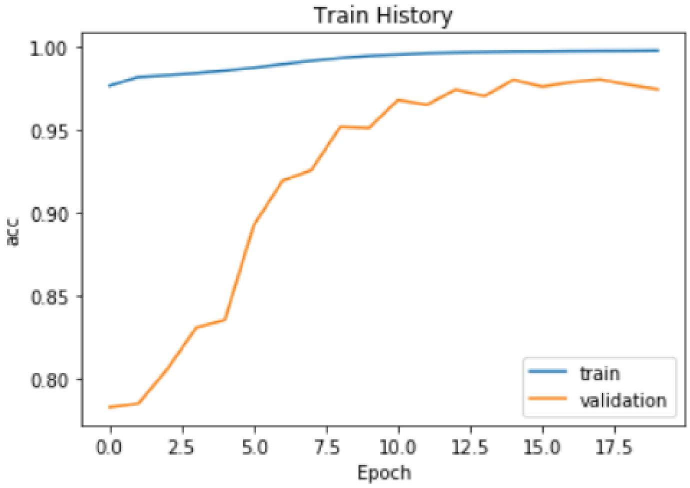
The performance of DecisionTreeClassifier and TextCNN, shown in table below.

DecisionTreeClassifier		TextCNN	
Training Accuracy	0.99	Training Accuracy	0.99
Training ROC/AUC Score	0..85	Training ROC/AUC Score	0.99
Valid Accuracy	0.84	Valid Accuracy	0.98
Valid ROC/AUC Score	0.84	Valid ROC/AUC Score	0.95

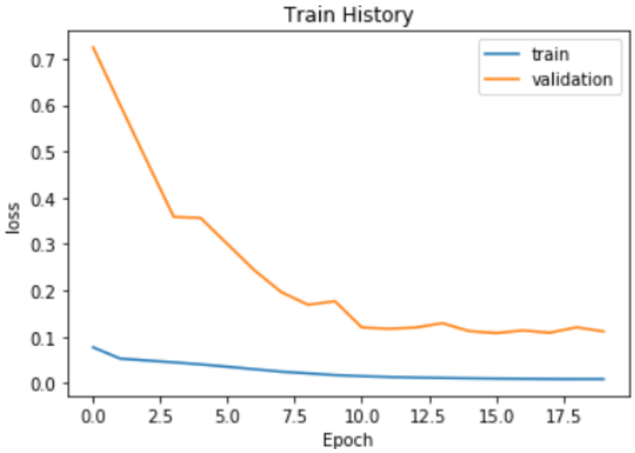
Conclusion

Free-Form Visualization

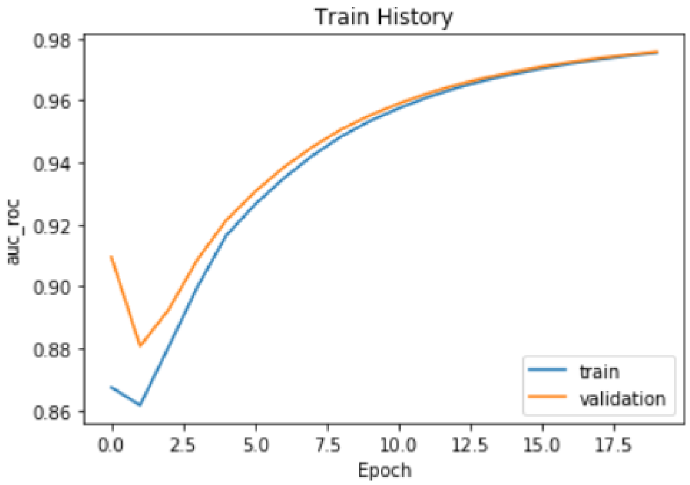
The accuracy of train set and validation set with increasing epoch.



The loss of train set and validation set with increasing epoch.



The AUC/ROC score of train set and validation set with increasing epoch.



Reflection

I spent the most time in cleaning data and I also think that this step is the most important and troublesome in text analysis. Because too much insignificant information can affect the accuracy of the model

The process used for this project can be summarized as following:

1. Understand the problem, imaging the problem solving steps, searching for information and setting the goals of the project.
2. Load the data
3. Data exploration and visualization, observing the distribution of data and try to find relationship between data. If data is an imbalanced data, list possible solutions. Observing the text, listing some word is needed to remove and unified text type will help us clean up the data.
4. Resample data if your data need it. Most classification data sets do not have exactly equal number of instances in each class, so imbalanced data is quite natural in classification
5. Data pre-processing is an important step in the data mining process. The phrase "garbage in, garbage out" is particularly applicable to data mining and machine learning projects.
6. Word embedding is the collective name for a set of language modeling and feature learning techniques in natural language processing (NLP) where words or phrases from the vocabulary are mapped to vectors of real numbers. https://en.wikipedia.org/wiki/Word_embedding
7. Search for related models of text classification, and pick the two models you are interested in. One model is as benchmark model, the other one is experiment model, and I will make the performance of model better and better through the adjustment of the parameters or the model architecture
8. Training model and check this model is robustness. I use k-fold cross volition to check my model.

Improvement

CNN has good performance in text classification, but there is a deficiency that is not intuitive enough, and the interpretability is not good. The Attention mechanism is a commonly used modeling long-term memory mechanism in the field of natural language processing. It can intuitively give the contribution of each word to the result and does help a lot on catching information in long sequences. So I think CNN-based attention model will have better performance than CNN model. I take this CNN-based attention model as my next practice.

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