Project 2: Exploring discourse in Reddit.

Rishab Katta
Akhil Karrothu

April 29, 2019

1. **Introduction :**
The goal of our project is to collect posts from two subreddits and develop a machine learning agent that can predict from which subreddit the unlabeled post has been taken. So for this we have taken two different subreddits that are unpopular opinion and world news. We have taken entirely different subreddits because we could enhance the prediction percentage of our machine learning agent. We have used SVM classifier, Random forest classifier and LSTM for training the data and predicting from which subreddit the post came from. So for the training and testing purposes we have divided the downloaded data from the subreddits. So finally after the classifier are developed we have used a portion of our data to evaluate our classifier in terms of precision and recall for each of the subreddits.

2. **Methods :**

Selection of Subreddits and Data Extraction:
The subreddits we have chosen are unpopular opinion and world news. We have chosen these two subreddits because the subreddits are entirely different from each other. So for the purpose of our project we decided to download 500 posts from each of the subreddit. Here we are downloading the data in json format. So in order to download data from subreddits in json format we have used the url of the subreddit by giving it a .json extension. But the problem we encountered here is that for one request we can be only able to download 25 posts of a subreddit. So inorder to download 500 posts from a subreddit we pinged the url for 19 more times. One more problem is that if we dont have a pointer specifying the point untill which post we have downloaded the data in the previous loop we will end up in downloading the same 25 posts every time so in order to overcome this we have used a parameter called after. This parameter ensures that we traverse through the entire posts of the particular subreddit in a order from top to bottom. Such that by not printing the same set of 25 posts every time. As we had downloaded the data in json format we received the data in messy format with number of keys. But by keen observation we figured out that the actually data related to posts is under the keys [data] [children]. So we have stored those posts into a list. Later we created a data frame using that list. In the same manner we have downloaded the posts for both the subreddits.

Data cleaning and Splitting :
Once we have created data frames for each of the subreddit posts we have clubbed both the dataframes into one dataframe.The next we have done is like cleaning of the obtained data by removing the punctuation, stop words and unicode characters for this purpose we used **scikit-learn** package. Later for modelling purpose we named the both subreddits 1 and 0 respectively. For conversion of the data

from text to variables while cleaning we have used **TFIDFVectorizer**. While conversion the vectorizer counts how many times the particular word occured in a post and importance of the particular word in the total posts. We have used **sklearn.model-selection** to split the data into three parts like 50 percent for training, 25 percent for development and 25 percent for testing.In this way we cleaned the data and divided it for further use for classifiers.

Learning Algorithms :
The learning algorithms that we have used are,
1) Random Forest Classifier
2) Support Vector Classifier
3) Long Short Term Memory

Firstly we have implemented **Random Forest classifier** which works on the Random forest algorithms. Basically this a ensemble classification algorithm. It means that basically in classification we will be using only one classifier to predict but in ensemble classifier we use group of classifiers to predict. So for this purpose in random forest the ensemble classifiers used are decision trees. All these trees are randomly created. once they are created after they predict we take the prediction that has been commonly given by most number of trees. In this way the Random forest classifier works. In our project,
1) Firstly we have used **Randomforestclassifier** method from **scikit-learn** in order to train the model using the train data which we have obtained while splitting. we have done fitting and predicting using the train data set.
2) We have used **average-precision-score** method to calculate the average precision score of our model on development data set. In order to calculate the Average precision score we gave prediction on development data set and the original development data set as parameters.
3) We have used **accuracy-score** method to calculate the accuracy of our model on test data set. In order to calculate the accuracy score we gave prediction on test data set and the original test data set as parameters.
4) We have used **Confusion-matrix** method on test data set to figure out how our model has been predicting or simply to visualize the performance of our classifier. It gives us the values of true negative and true positive along with false negative and false positive.
5) We have plotted the precision recall curve using the **precision-recall-curve** method in the scikit learn.
These are steps followed by us to predict using random forest classifier.

The Second learning algorithm that we have used for the prediction is **Support vector classifier**. It is a supervised learning technique. As the data we have downloaded contains both features and labels we are using the SVM classifier to build a model to predict the subreddit to which the Unlabeled post belongs. As we are having 2 classes so it is called as Binary SVM Classifier. When we plot the data in training data set there is apparent gap between the both sets of data so we are using the **Linear SVM Classifier**. In Linear SVM we will be having a straight hyperplane that is predicted for dividing the 2 classes. In the process of drawing the hyperplane the main motto is to maximize the distance of the hyperplane from nearest data point from either of the classes. So in order to implement this we are using the library **Sklearn** which has the function **SVC** and we pass the parameter as **Linear**. In our project,
1) Firstly we have used **SVC** method from **sklearn** in order to train the model using the train data which we have obtained while splitting. we have done fitting and predicting using the train data set.

2) We have used **average-precision-score** method to calculate the average precision score of our model on development data set. In order to calculate the Average precision score we gave prediction on development data set and the original development data set as parameters.
3) We have used **accuracy-score** method to calculate the accuracy of our model on test data set. In order to calculate the accuracy score we gave prediction on test data set and the original test data set as parameters.
4) We have used **Confusion-matrix** method on test data set to figure out how our model has been predicting or simply to visualize the performance of our classifier. It gives us the values of true negative and true positive along with false negative and false positive.
5) We have plotted the precision recall curve using the **precision-recall-curve** method in the scikit learn.

The third learning algorithm that we have used is **Long short term memory (LSTM)** which is an extension of **recurrent neural network (RNN)** in terms of memory basically RNN uses the short term memory to predict the memory but by using the LSTM we have long term memory where we can can predict. LSTM makes RNN to remember their inputs for long time for further use. Basically in LSTM we have three gates : Input, Output and Forget. Where each of the gates have their own function. These gates decides whether to allow the new input in or not and to delete the unimportant information or to allow to have it effect on output. These gates are analog that they range from 0 to 1. In order to know how it works we have to know about sequential data which means that related stuff follow each other here we use time series data. Basically in RNN we make a decision based on the learning's until that point and the input at that point.Lets see how we have implemented the algorithm for the predicting the the name of subreddit from which the unlabeled post comes from.
1) We have downloaded the data from two different subreddits as we have done earlier.
2) Here we have Splitted the data bit different it means we have divided the data into train and test data like 75 and 25 percent. It is done using **train-test-split** from **sklearn.model-selection**.
3) Once we have obtained the data in the form of data frame we have to make slight transforms to it by making a target column in the for of integer or float.
4) Then we have performed binning technique on the data to look out for the length of posts and we found out that among the 998 posts we have only 16 posts with bin size greater than 500 so we make 500 as out maximun length.
5) Once the binning is done we have to preprocess the data by tokenizing. Because we cannot use the raw data as input for the model so we need to encode it so for that purpose we use **fit-on-texts** which creates the vocabulary index based on frequency that is like each of the word in text has its own integer.
6) After the above step we pass data to conversion of the text to sequence using **tokenizer.texts-to-sequences**. After we have obtained the number sequence we transform it into a 2D Numpy array. Later we calculate the Vocab size by finding the length of the word index.
7) In order to construct the model first we construct the **embedding layer** which is our first hidden layer by giving it max length, size of vector space and the length of sequence. we connect this embedding layer to the Input.
8) Later we connect this embedding layer to the **Dense layer.** Then we construct the model using the model method by passing the input and predictions as the final dense formed dense layer. Once the model is constructed we look out at the summary.
9) After the model creation we have plotted the graph between accuracy and epochs. Finally done our prediction using the model on test data.
This how we have done prediction using LSTM.

Results and Comparison :
NOTE : The results that included are the outcomes of the instance the accuracy may differ by a point or so after every execution.
When we ran our code for **SVM binary classifier**, these are the results that it produced.

```
True Negatives: 182
False Positives: 2
False Negatives: 3
True Positives: 187
                       |Pred Unpopular Opinion     |Pred World news
-----------------------|---------------------------|--------------------
Act Unpopular Opinion  |                   182     |        2
Act World news         |                   3       |        187

Average precision-recall score: 1.00
Accuracy: 0.9866310160427807
```

Let's talk about what each block above is and what it represents.
A **true positive** is one that detects the condition when the condition is present. The number of true positives in our prediction is 187.
A **true negative** is one that does not detect the condition when the condition is absent. The number of true negatives in our prediction is 182.
A **false positive** is one that detects the condition when the condition is absent. The number of false positives in our prediction is 2.
A **false negative** is one that does not detect the condition when the condition is present. The number of false negatives in our prediction is 3.

The next block which is represented in a tabular format is called a **confusion matrix**.It's also called an error matrix. It's a visualization of the performance of the algorithm. It compares the actual class vs predicted class for each of the examples. They way you interpret the matrix is as follows, When it's a post from Unpopular opinion subreddit, our classification algorithm predicted that it is from unpopular opinion 182 times. When it's a post from world news, our classification algorithm predicted that it's from unpopular opinion 3 times. When it's a post from unpopular opinion, our classification algorithm predicted that it's from world news 2 times. And lastly, When it's a post from world news, our classification algorithm predicted that it's from world news 187 times.

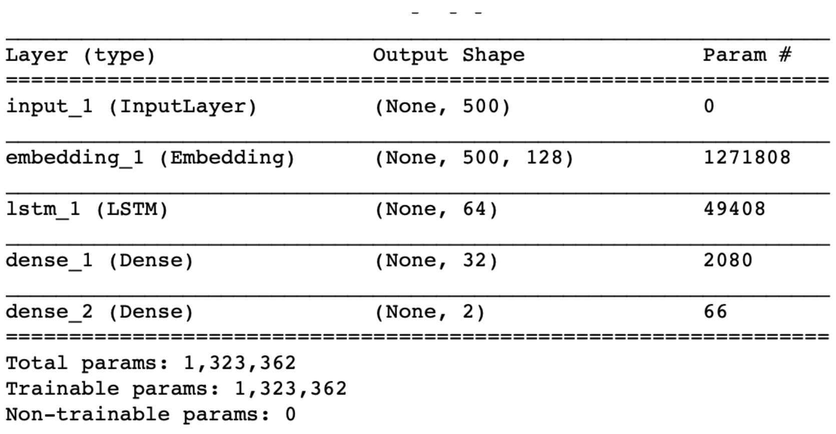
Precision is the ratio of the number of relevant items found to the total number of items found $-- >$ Precision = True Positives / (True Positives + False Positives), whereas recall is the ratio of number of relevant items found to the total number of relevant items $-- >$ Recall = True Positives / (True Positives + False Negatives) **Average precision score** is the weighted mean of precisions achieved at each threshold, with the increase in recall from the previous threshold used as the weight. The average precision score for this classifier is 1.00

Accuracy: For each index position of y_test and y_pred lists, the values are compared and then the matches found are divided by the total number of samples. That is accuracy in a nutshell. So the total accuracy of our classification algorithm is 98.66%

When we ran our code for **Random Forest Classifier**, these are the results it produced.

```
True Negatives: 122
False Positives: 0
False Negatives: 7
True Positives: 120
                       |Pred Unpopular Opinion     |Pred World news
-----------------------|---------------------------|--------------------
Act Unpopular Opinion  |                   122     |        0
Act World news         |                   7       |        120

Average precision-recall score: 1.00
Accuracy: 0.9718875502008032
```

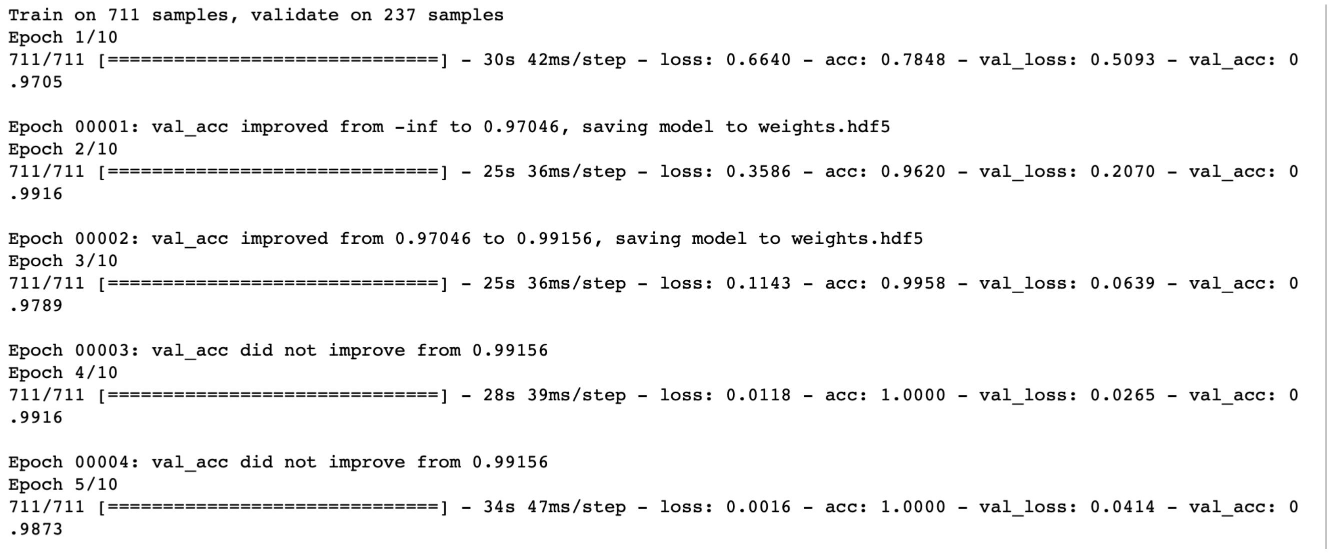
When we ran our code for **LSTM**, these are the results it produced.
The following picture shows the summary of the model by showing its layers.
For the embedding layer the params are 1271808.
For the lstm layer the params are 49408.
For the dense-1 layer the params are 2080.That is 64*32 + 32 = 2080.
For the dense-2 layer the params are 66. That is 2*32 + 2 = 64.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 500)               0
_________________________________________________________________
embedding_1 (Embedding)      (None, 500, 128)          1271808
_________________________________________________________________
lstm_1 (LSTM)                (None, 64)                49408
_________________________________________________________________
dense_1 (Dense)              (None, 32)                2080
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 66
=================================================================
Total params: 1,323,362
Trainable params: 1,323,362
Non-trainable params: 0
```

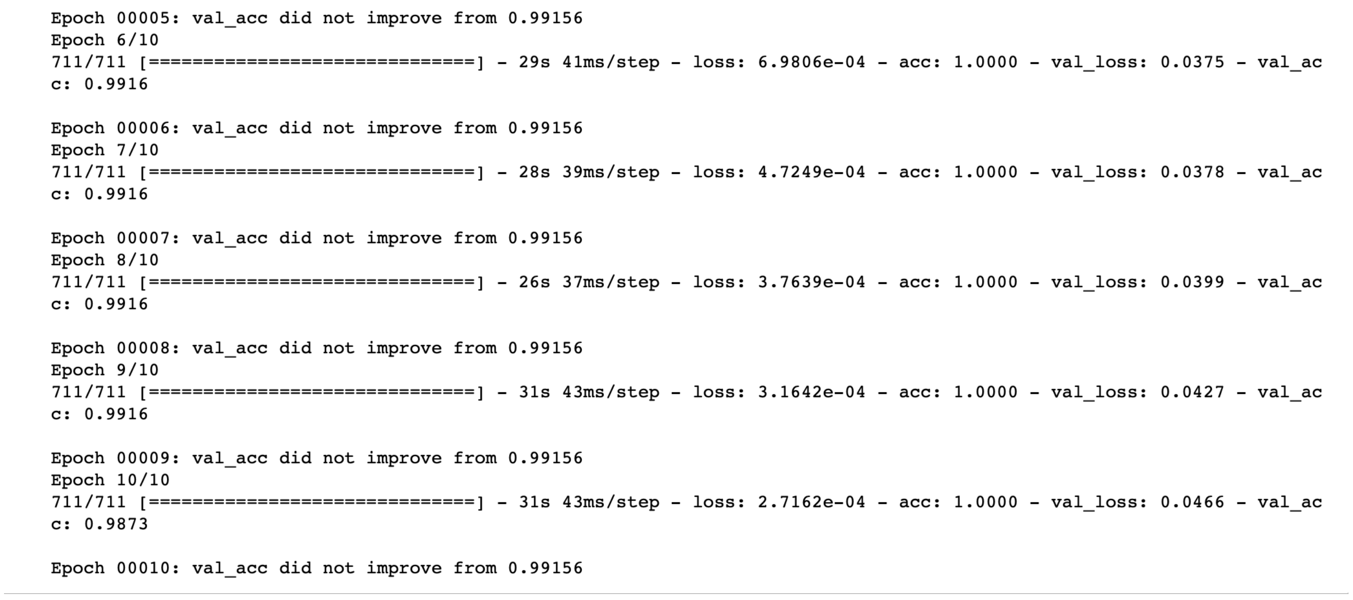
Figure 1: summary

The following figures show the training of our model on 711 samples in 10 epochs. In each of the epoch we can see the value loss and the accuracy and the validation accuracy.



```
Train on 711 samples, validate on 237 samples
Epoch 1/10
711/711 [==============================] - 30s 42ms/step - loss: 0.6640 - acc: 0.7848 - val_loss: 0.5093 - val_acc: 0
.9705

Epoch 00001: val_acc improved from -inf to 0.97046, saving model to weights.hdf5
Epoch 2/10
711/711 [==============================] - 25s 36ms/step - loss: 0.3586 - acc: 0.9620 - val_loss: 0.2070 - val_acc: 0
.9916

Epoch 00002: val_acc improved from 0.97046 to 0.99156, saving model to weights.hdf5
Epoch 3/10
711/711 [==============================] - 25s 36ms/step - loss: 0.1143 - acc: 0.9958 - val_loss: 0.0639 - val_acc: 0
.9789

Epoch 00003: val_acc did not improve from 0.99156
Epoch 4/10
711/711 [==============================] - 28s 39ms/step - loss: 0.0118 - acc: 1.0000 - val_loss: 0.0265 - val_acc: 0
.9916

Epoch 00004: val_acc did not improve from 0.99156
Epoch 5/10
711/711 [==============================] - 34s 47ms/step - loss: 0.0016 - acc: 1.0000 - val_loss: 0.0414 - val_acc: 0
.9873
```

Figure 2: training



```
Epoch 00005: val_acc did not improve from 0.99156
Epoch 6/10
711/711 [==============================] - 29s 41ms/step - loss: 6.9806e-04 - acc: 1.0000 - val_loss: 0.0375 - val_ac
c: 0.9916

Epoch 00006: val_acc did not improve from 0.99156
Epoch 7/10
711/711 [==============================] - 28s 39ms/step - loss: 4.7249e-04 - acc: 1.0000 - val_loss: 0.0378 - val_ac
c: 0.9916

Epoch 00007: val_acc did not improve from 0.99156
Epoch 8/10
711/711 [==============================] - 26s 37ms/step - loss: 3.7639e-04 - acc: 1.0000 - val_loss: 0.0399 - val_ac
c: 0.9916

Epoch 00008: val_acc did not improve from 0.99156
Epoch 9/10
711/711 [==============================] - 31s 43ms/step - loss: 3.1642e-04 - acc: 1.0000 - val_loss: 0.0427 - val_ac
c: 0.9916

Epoch 00009: val_acc did not improve from 0.99156
Epoch 10/10
711/711 [==============================] - 31s 43ms/step - loss: 2.7162e-04 - acc: 1.0000 - val_loss: 0.0466 - val_ac
c: 0.9873

Epoch 00010: val_acc did not improve from 0.99156
```

Figure 3: Training

The below is the graph that contains both the validation accuracy curve and the accuracy curve that has a been obtained during the each of 10 epochs. Here epoch is nothing but the iteration in each of the iteration we train the model for all the 711 samples and we print out the val loss and accuracy along with validation accuracy .

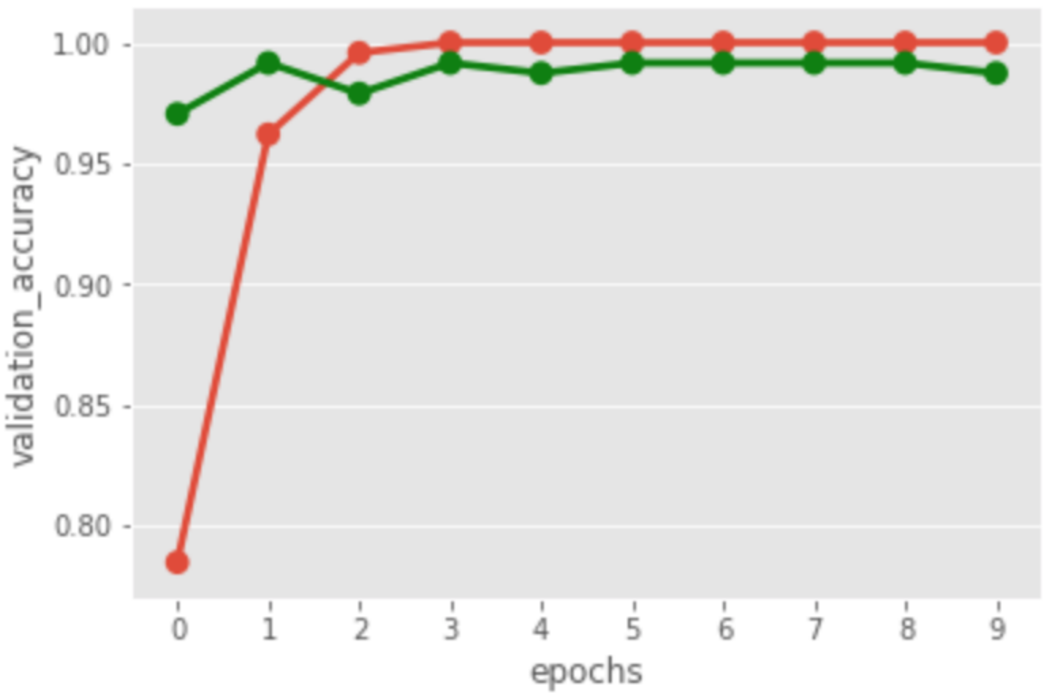
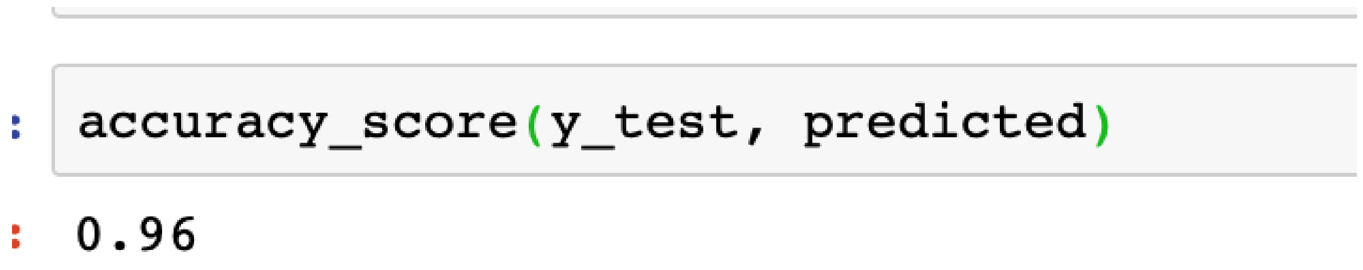


Figure 4: Graph accuracy vs epochs



```
accuracy_score(y_test, predicted)
```

```
0.96
```

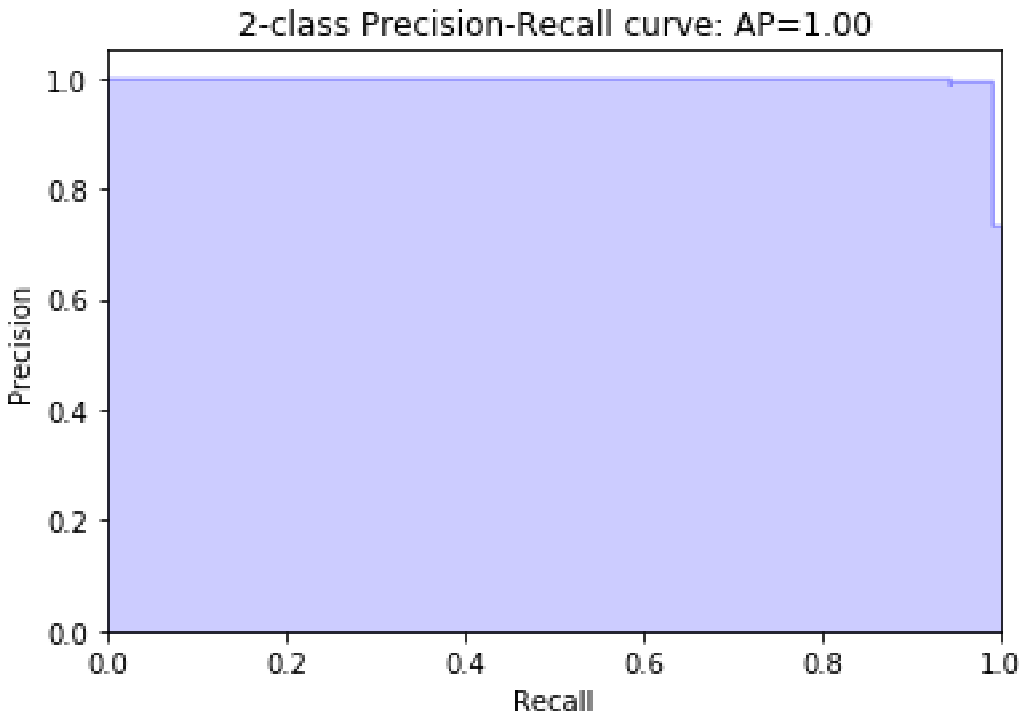
Figure 5: Accuracy of LSTM

The results for the Random Forest Classifier can be interpreted same as described for the SVM classifier. If we look at the accuracy of two classfiers we can see that SVM classfier has more accuracy than the Random Forest classifier. This is because Random forest classifiers run different decision trees on subset of features and use the mode as the final classification. So using a subset of features requires the classification problem to be MULTICLASS. So we are basically saying that Random Forest Classifiers are better at multi-class classifications. As opposed to SVM, which performs gen-

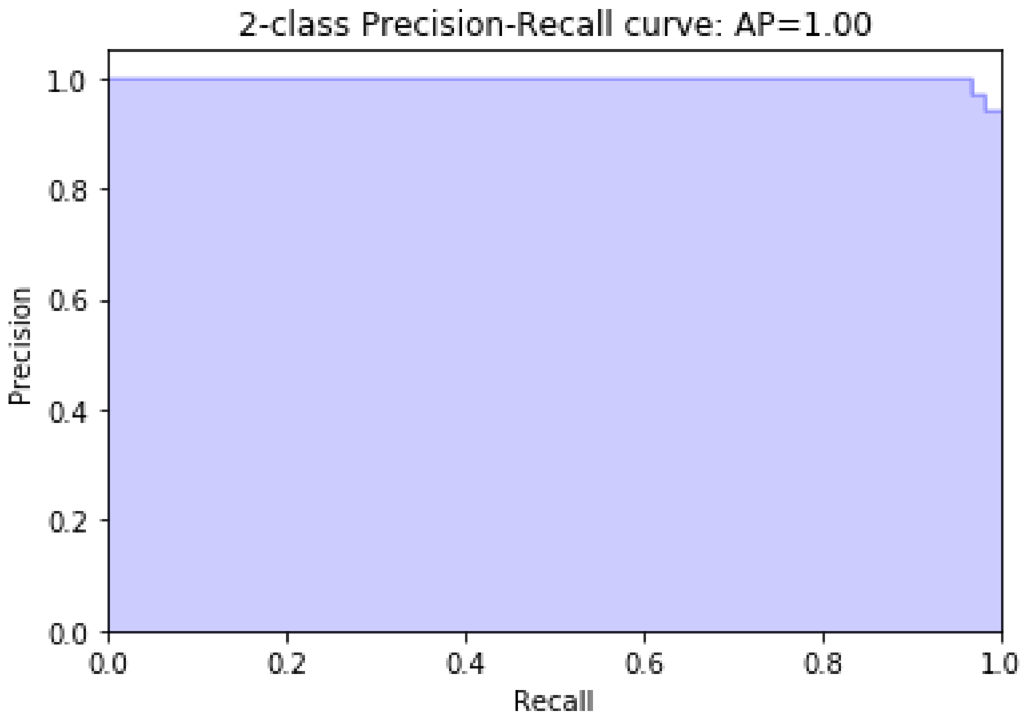
erally well in binary classification. SVM maximizes the "margin" and thus relies on the concept of "distance" between different points. So we can say that for this particular problem, SVM performs better than Random Forest classifier.
When we compare the results of the SVM and Random forest with LSTM which has the accuracy of 96 then also the SVM performs the best among the three. The thing with LSTM is more time consuming it takes time to train the model but prediction rate is fast than the remaining three. Here we dont have to construct the confusion matrix.

Precision Recall Curves:
Precision recall curve is a plot of precision on the y-axis vs recall on the x-axis. PRC curves are a metric to evaluate the classifier's quality. Precision and Recall are inversely related and balance between these two needs to be achieved. To achieve this and to measure performance PRC curves come in handy. The PRC curve shows the trade-off between precision and recall. A large area under the curve represents both high precision and recall. This is the best case scenario for a classifier. It shows that the model returns accurate results for majority of classes it selects.

Precision Recall Curve for Random Forest Classifier.



Precision Recall Curve for SVM Classifier.



As mentioned above, the more is the filled in area in the PRC curves, the stronger the classifier is. Since we picked two subreddits which are very different from each other, it is not a surprise that both our classifiers do a very good job in predicting the subreddit a post comes from.

Conclusion:
In conclusion, we can say that all three classifiers did a very good job in predicting the subreddit of the posts as evident from the PRC curves and accuracy metrics. SVM classifier performed better than Random Forest and LSTM because SVM will generally perform better on linear dependencies and when the data is linearly separable. Random Forests and LSTM on the other hand perform well in multi-class classifications. Since our problem is binary classification and the bag of words from each subreddit is linearly separable, SVM performs much better at classifying the posts from two subreddits. In future we would like to perform prediction using these three classifiers on multi classification problem and check whether if the Random forest and LSTM perform better than the SVM.