Custom Speech Classifier

Submitted by

Justine Jacob 16BLC1061

Sandeep B 16BLC1073

Aditya P Varma 16BLC1107

J Component - Report

ECM2002 – Machine Learning Algorithms

BACHELOR OF TECHNOLOGY

in

ELECTRONICS AND COMPUTER ENGINEERING

October 2018

TABLE OF CONTENTS

Abstract

Fake news is a major issue in today’s world. We often come across news where a statement supposedly said by the person destroys his credibility or public image. But later turns out that he/she never said it and it was a fake news.

Our Project aimed at developing an efficient speech classifier which sorts out if a particular speech was actually given out by a person or not using our own algorithms instead of the generic ones so that greater accuracy with a small training set can be achieved.

To test and validate our classifier, we decided to work with Barack Obamas Speech transcripts since he was President for 2 terms and it was easier to obtain sufficient datasets. So on feeding the classifier the speech data, it would ideally classify it into Obama or Non Obama speech.

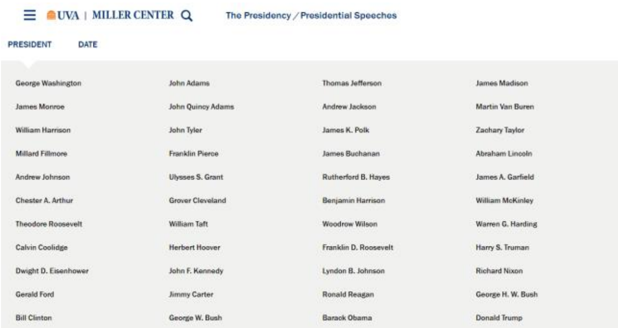
(2.1) - Dataset Description:

The Dataset is a mix of Obama and Non-Obama Speeches.

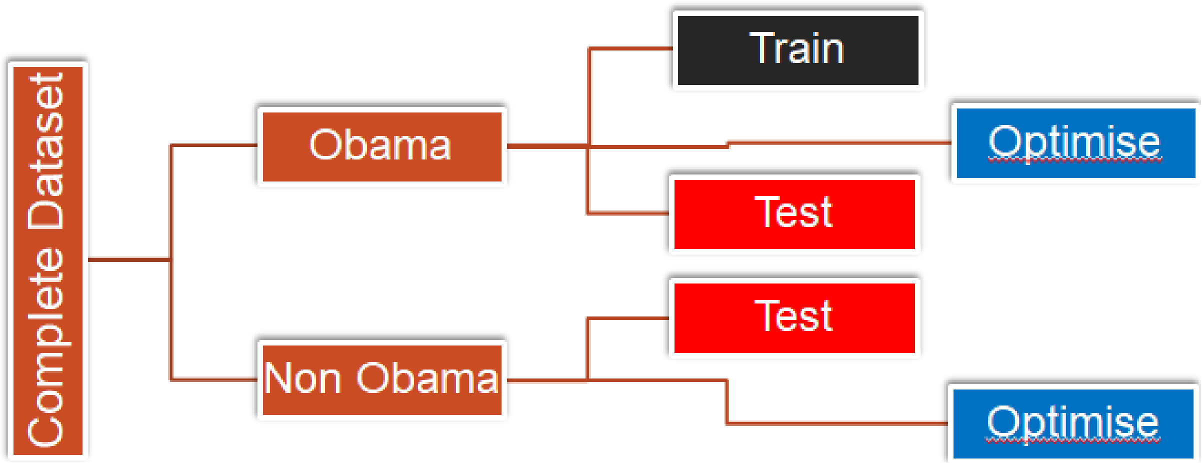
Source:

American Rhetoric

Miller Center

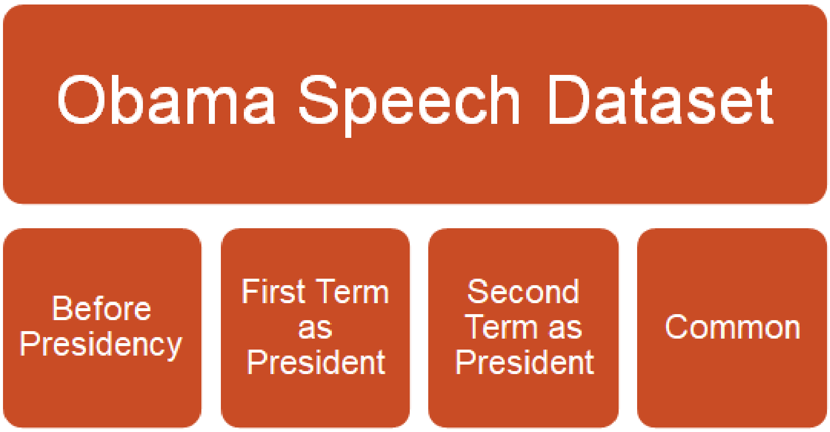


Dataset Split:



(2.2)
To test our model against different types of dataset, we sub-divided the Obama Speeches into 4 types:

1. The Speeches he gave before his Presidency
2. The Speeches he gave during his First Term as President
3. The Speeches he gave during his Second Term as President
4. The Entire Speech Set (Common)



(3.1) - Algorithm:

Since our classifier is built to work with Speech transcripts, the first major hurdle was to figure out the predictor/ metric of classifying speeches of 2 different persons. After a lot of brainstorming, we decided to go for 2 Word Probability Links as our metric for classification.

So the classifier works in Three Levels : Training, Optimising and Testing.

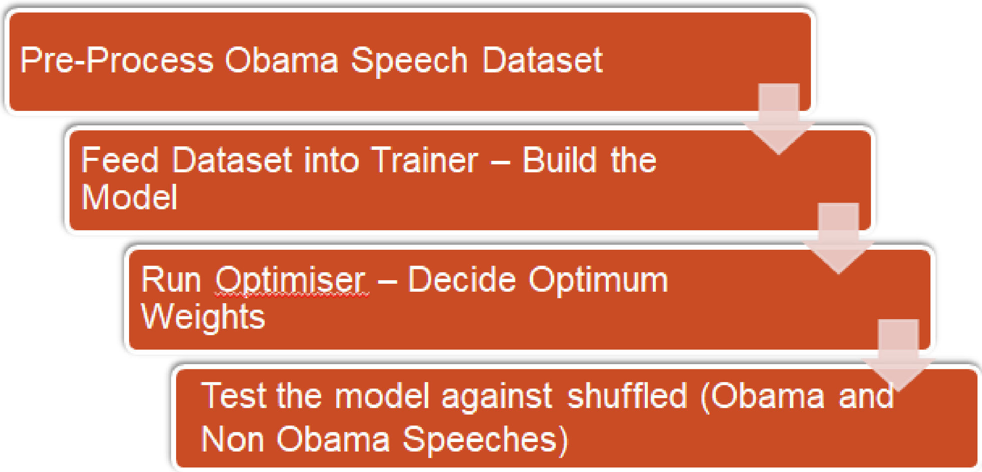
Training:
The training includes feeding the model with random speeches from a pool of Obama Speeches.
The model reads through the speeches and pre-processes the text by tokenizing, preserving quotes and removing punctuations. Then it iterates through the whole speech and generates a Dynamic Word Web or Neural Nodes where each word acts as a sub-node with a directional link connecting the former and latter word and the weight of the link being the probability of occurrence.
In Python, the structure is modeled using dictionary as the data structure.
The dictionary, after training, is pickled and saved into a file for use.

Optimising:
This makes sure the weights are properly assigned to the model. Supervised learning is applied here. A mix of Obama and Non Obama speeches is selected at random from the pool of speeches and fed into the classifier model. Both positive and negative weights are varied and the pair of weights which produce the maximum degree of separation is chosen for the test set to be followed.

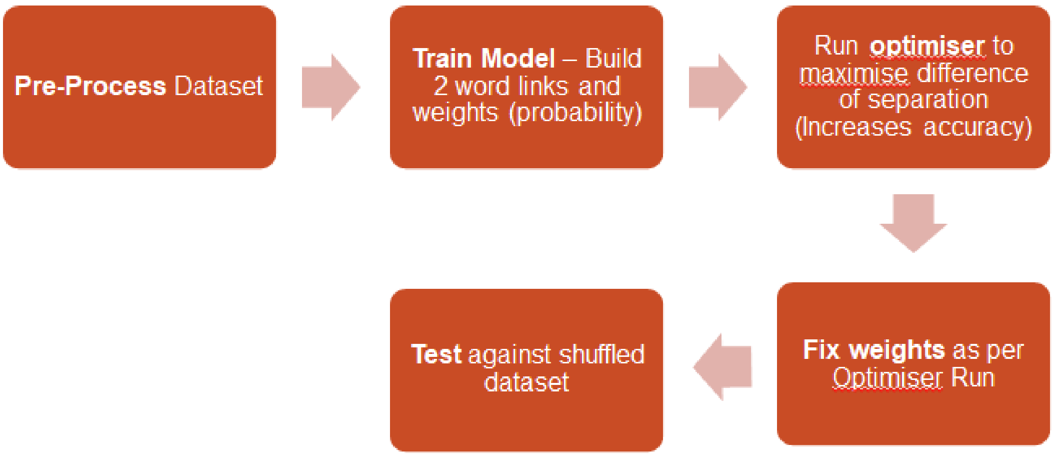
Testing:
After the optimum weights are selected and set into the model, the test set which consists of a mix of Obama and Non-Obama speeches are fed into the model and the model is allowed to classify the speeches based on the neural nodes and the link weights.

(3.2) - Initial Design Flow:

Updated 4 Tier Design:



Summarized Algorithm:



Detailed Explanation of Algorithm:

The entire classifier was developed in Python3.5.

(3.3) - Text Pre-Processing:
We wrote a custom pre-processor to make sure the nodes are efficiently made. Various checks and operations performed in this stage includes

- Tokenizer
- Preserves Quotes
- Maintains Uniformity for ease of comparison

Data Structure:
The Dynamic Word-Web/Neural nodes were modeled using a hybrid List in Dictionary structure.

Sample Data in the dictionary

```
child : [6, ['turns', 1], ['care', 2], ['the', 1], ['who', 1], ['in', 1]]
better : [6, ['america', 1], ['pay', 1], ['treatment', 1], ['way', 1], ['job', 1], ['day', 1]]
around : [7, ['their', 1], ['the', 4], ['our', 1], ['lunchtime', 1]]
didn't : [2, ['just', 1], ['expect', 1]]
struggles : [1, ['to', 1]]
exchange : [1, ['for', 1]]
west : [3, ['come', 1], ['we', 1], ['all', 1]]
underground : [1, ['railroad', 1]]
internet : [2, ['connection', 1], ['possible', 1]]
veterans : [1, ['who', 1]]
lincoln : [5, ['once', 1], ['understood', 1], ['was', 1], ['organized', 1], ['before', 1]]
commitment : [1, ['and', 1]]
capitol : [1, ['where', 1]]
wonder : [2, ['what', 1], ['--', 1]]
paycheck : [2, ['to', 1], ['despite', 1]]
makes : [1, ['future', 1]]
speeches : [1, ['all', 1]]
other : [8, ['we', 1], ['party', 1], ['thing', 1], ['senators', 1], ['eras', 1], ['the', 1], ['things', 1], ['way', 1]]
99th : [2, ['in', 2]]
abolitionists : [1, ['emerged', 1]]
```

Format :
<former_word> : [<Total occurrence of former>, [<latter word_1>,<no of occurences>], [<latter word_2>, <no of occurences>],….]

(3.4) - Trainer:
Once the speech data (Only Obama Speeches) is read and pre-processed, the whole speech is iterated one pair of word at time and the corresponding nodes and links are made by updating the dictionary entries. The whole process is repeated for n speeches at random.
At the end, the file is pickled and saved for future use.

Execution of Trainer

```
Enter Pickle File name :com_trl_05
Enter (train:total) file ratio :0.5
Starting Training


No of Files used for Training : 166
Training with file : o_f_38.txt  completed!
Training with file : o_f_13.txt  completed!
Training with file : o_f_79.txt  completed!
Training with file : o_s_55.txt  completed!
Training with file : o_s_67.txt  completed!
Training with file : o_s_34.txt  completed!
Training with file : o_s_108.txt  completed!
Training with file : o_f_81.txt  completed!
Training with file : o_s_140.txt  completed!
Training with file : o_s_21.txt  completed!
Training with file : o_f_120.txt  completed!
Training with file : o_f_63.txt  completed!
Training with file : o_f_60.txt  completed!
Training with file : o_s_48.txt  completed!
Training with file : o_f_22.txt  completed!
Training with file : o_s_131.txt  completed!
Training with file : o_f_33.txt  completed!
Training with file : o_f_155.txt  completed!
Training with file : o_s_122.txt  completed!
Training with file : o_f_133.txt  completed!
Training with file : o_s_60.txt  completed!
Training with file : o_s_42.txt  completed!
Training with file : o_s_46.txt  completed!
Training with file : o_f_70.txt  completed!
Training with file : o_f_124.txt  completed!
Training with file : o_s_142.txt  completed!
Training with file : o_s_32.txt  completed!
Training with file : o_f_43.txt  completed!
Training with file : o_f_66.txt  completed!
Training with file : o_f_123.txt  completed!
Training with file : o_f_128.txt  completed!
```

(3.5) - Optimiser:

Once the training is completed, the next stage in the classifier is the optimizer where the weights are optimized using Supervised learning and calculating the degree of separation.

Execution of Optimiser

```
Starting Optimiser


Enter Range for Positive Weight (1 to x) :5
Enter Range for Negative Weight (-y to 0) :5
Enter Step value for weight :1
No of Files Used in Optimiser : 105
pos_wt = 1.0   neg_wt = 0
pos_wt = 1.0   neg_wt = 1.0
pos_wt = 1.0   neg_wt = 2.0
pos_wt = 1.0   neg_wt = 3.0
pos_wt = 1.0   neg_wt = 4.0
pos_wt = 1.0   neg_wt = 5.0
pos_wt = 2.0   neg_wt = 0
pos_wt = 2.0   neg_wt = 1.0
pos_wt = 2.0   neg_wt = 2.0
pos_wt = 2.0   neg_wt = 3.0
pos_wt = 2.0   neg_wt = 4.0
pos_wt = 2.0   neg_wt = 5.0
pos_wt = 3.0   neg_wt = 0
pos_wt = 3.0   neg_wt = 1.0
pos_wt = 3.0   neg_wt = 2.0
pos_wt = 3.0   neg_wt = 3.0
pos_wt = 3.0   neg_wt = 4.0
pos_wt = 3.0   neg_wt = 5.0
pos_wt = 4.0   neg_wt = 0
pos_wt = 4.0   neg_wt = 1.0
pos_wt = 4.0   neg_wt = 2.0
pos_wt = 4.0   neg_wt = 3.0
pos_wt = 4.0   neg_wt = 4.0
pos_wt = 4.0   neg_wt = 5.0
pos_wt = 5.0   neg_wt = 0
pos_wt = 5.0   neg_wt = 1.0
pos_wt = 5.0   neg_wt = 2.0
pos_wt = 5.0   neg_wt = 3.0
pos_wt = 5.0   neg_wt = 4.0
pos_wt = 5.0   neg_wt = 5.0
Optimised Weights
Positive Weight : 1.0
Negative Weight : 0
```

(3.6)Tester:
Once the weights are optimized, the random speech set is fed into the classifier and a confusion matrix is generated which gives the accuracy of the model.

Execution of Tester

```
No of test file : 103
o_s_115.txt  is an Obama File
o_f_20.txt  is an Obama File
o_f_49.txt  is an Obama File
o_s_124.txt  is an Obama File
o_f_52.txt  is an Obama File
o_s_4.txt  is an Obama File
o_s_71.txt  is an Obama File
o_f_1.txt  is an Obama File
o_f_59.txt  is not an Obama File
o_s_49.txt  is an Obama File
o_f_143.txt  is an Obama File
o_s_77.txt  is an Obama File
o_s_113.txt  is an Obama File
o_s_65.txt  is an Obama File
o_f_109.txt  is an Obama File
n_25.txt  is not an Obama File
o_s_22.txt  is an Obama File
o_s_125.txt  is an Obama File
o_f_34.txt  is an Obama File
n_5.txt  is not an Obama File
n_19.txt  is an Obama File
o_s_5.txt  is an Obama File
o_s_120.txt  is an Obama File
n_23.txt  is not an Obama File
o_f_91.txt  is an Obama File
o_s_69.txt  is an Obama File
o_s_28.txt  is an Obama File
o_b_3.txt  is an Obama File
o_f_46.txt  is an Obama File
o_f_129.txt  is an Obama File
n_27.txt  is not an Obama File
o_f_25.txt  is an Obama File
o_b_27.txt  is an Obama File
o_s_141.txt  is an Obama File
n_40.txt  is not an Obama File
n_33.txt  is not an Obama File
o_f_5.txt  is an Obama File
```

```
Confusion Matrix
Actual ->
        Yes     No
Yes      78      1
No       5       19


Accuracy : 94.1747572815534 %
```

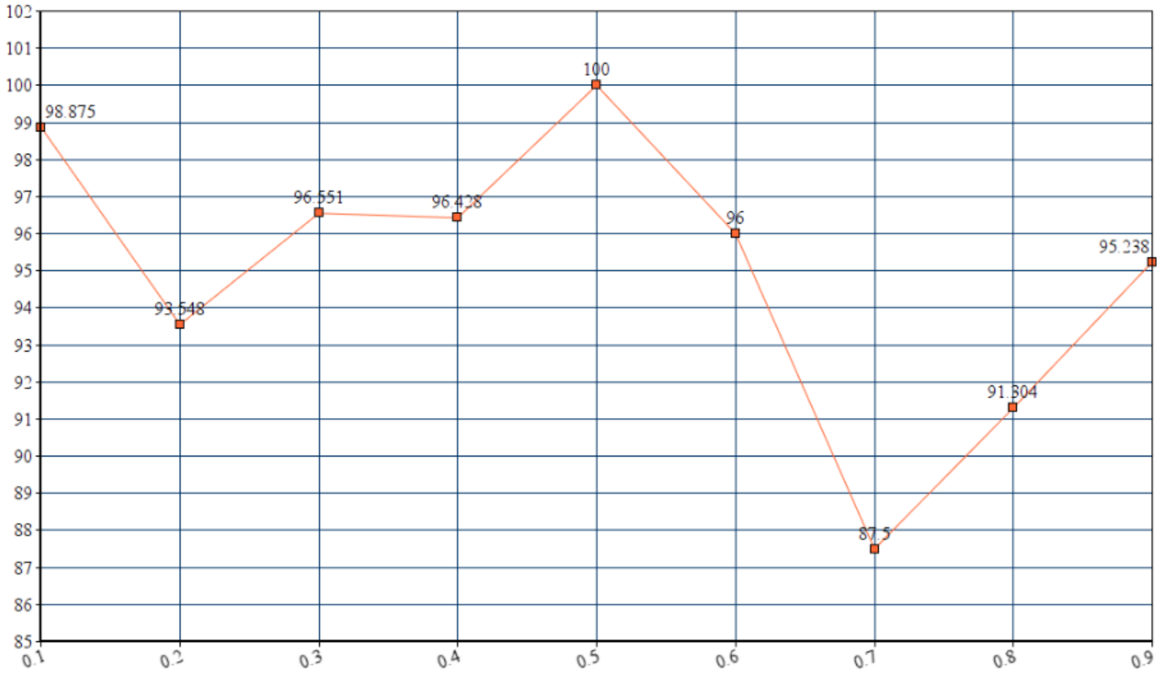
(4.1) - Variations Introduced

The entire code is flexible. Hence the following variations were introduced and the results were recorded.

- Variation in the Dataset used – Before, 1st term, 2nd term or common set
- Variation in (train:total) ratio for the training data (similar to k-fold cross validation)
- Variation in total no of Optimiser executions.

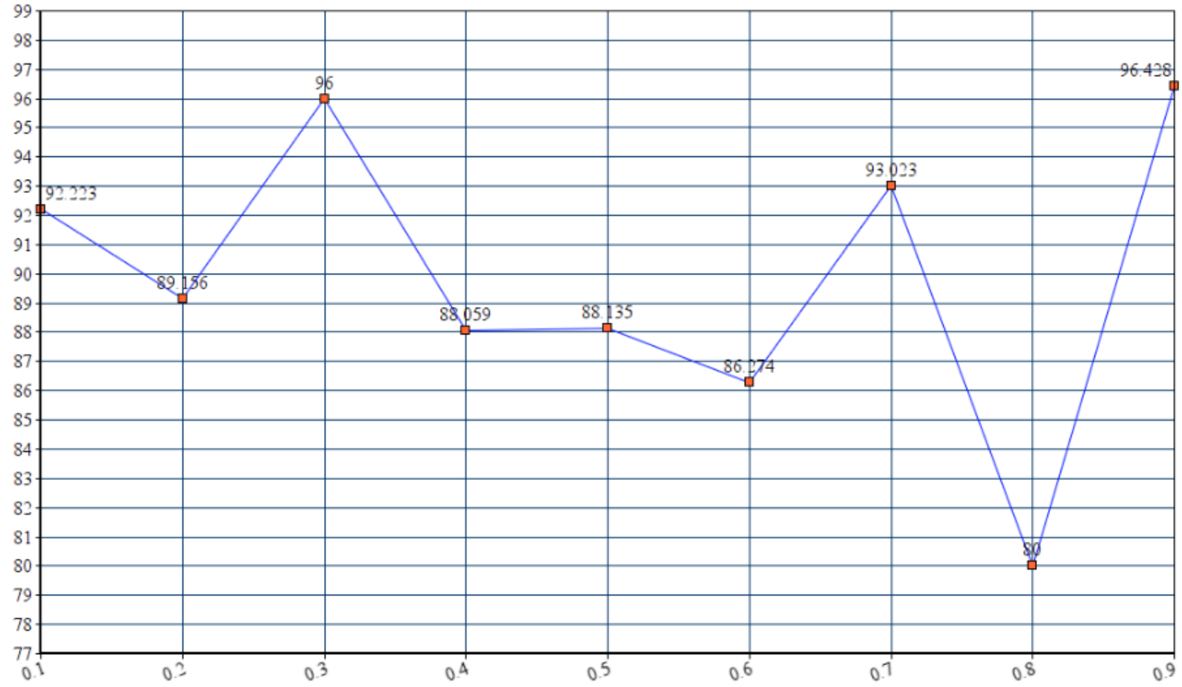
Variation in (train:total) for different types of datasets
Each of the dataset was used to train the model by varying the (train:total) ratio from 0.1 to 0.9.

Case 1:
Before Presidency Speech set as Training Set



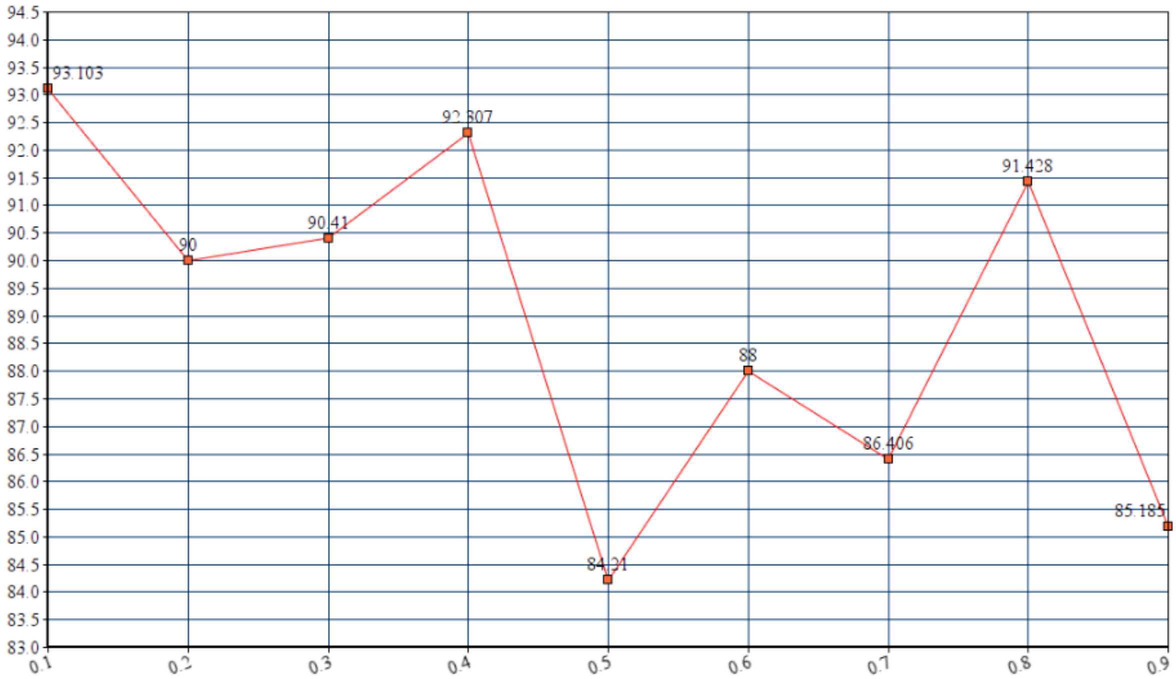
(train:total) ratio - - ->

(4.2) - Case 2:
First Term Speeches used as training set



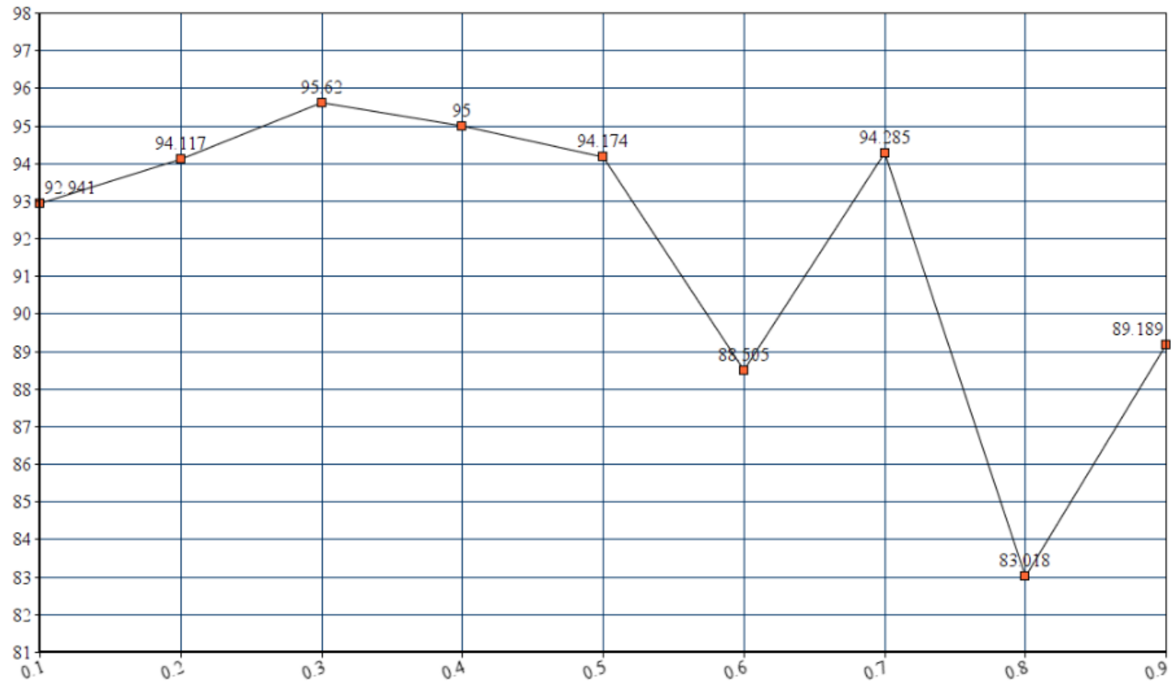
(train:total) ratio - - ->

Case 3:
Second Term Speeches used as training set



(train:total) ratio - - ->

(4.3) - Case 4:
Common Set used as Training set



(train:total) ratio - - ->

Inference
From the nature of the graphs, it was clear that a clear dip in accuracy was always present for the model after a (train:total) ratio of 0.5. To improve the model further to result in higher accuracy, the second variation was built into the code.

Example:
Observing the accuracy chart for Before presidency speech set as training set, a rapid accuracy dip occurred when (train:total) ratio was 0.7
Hence the Optimiser Variation, which varies the data used to optimize weight, was done on this ratio to check and see if the accuracy could be improved.

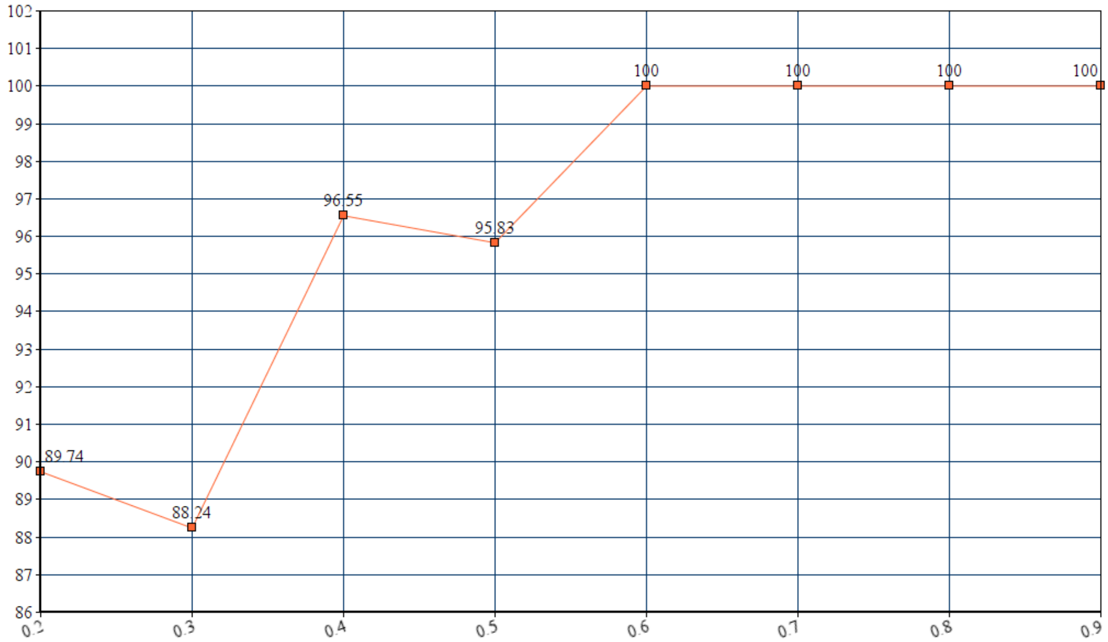
(4.4) - Results

```
Enter Pickle File name :bef_tr2_07_2
Enter (train:total) file ratio :0.7
Starting Training


No of Files used for Training : 19
###################
Data Dumped into  bef_tr2_07_2 .p File
Starting Optimiser Varations


Ratio : 0.2
No of Files Used in Optimiser : 10
..............................

Testing
No of test files : 39


***|******|******|********|************

Confusion Matrix
Actual ->
        Yes     No
Yes      2       0
No       4       33


Accuracy : 89.74358974358974 %



Ratio : 0.3
No of Files Used in Optimiser : 15
..............................

Testing
No of test files : 34
```

(4.5) - Summary Chart for Optimiser Variation

```
Ratio : 0.2
Pos_wt : 5.0
Neg_wt : 5.0
Accuracy : 89.74358974358974

Ratio : 0.3
Pos_wt : 5.0
Neg_wt : 5.0
Accuracy : 88.23529411764706

Ratio : 0.4
Pos_wt : 5.0
Neg_wt : 0
Accuracy : 96.55172413793103

Ratio : 0.5
Pos_wt : 1.0
Neg_wt : 0
Accuracy : 95.83333333333333

Ratio : 0.6
Pos_wt : 1.0
Neg_wt : 0
Accuracy : 100.0

Ratio : 0.7
Pos_wt : 1.0
Neg_wt : 0
Accuracy : 100.0

Ratio : 0.8
Pos_wt : 1.0
Neg_wt : 0
Accuracy : 100.0

Ratio : 0.9
Pos_wt : 1.0
Neg_wt : 0
Accuracy : 100.0
>>>
```

(4.6) –



ratio of files used for optimiser - - ->

(5.1) - Comparison with Existing Models

Pros

- Very high classification accuracy
- Even with a small training set as small as 8-12 speeches, the model is easily >90% accurate
- Can easily be extended to any person or context of speech (twitter, press conference transcripts etc.

Cons

- High Run time for huge data

Conclusion

After running various variation on the model and testing it against a wide variety of data, the classifier had an average accuracy of >90%.