Stephanie Nicole Garibay Lim
NLP – Coursework 1
220295309

Table of Contents

.ipynb file for Q1 to Q4 is named: NLP_Assignment_1_Q1_4.ipynb

Q1.

Parse_data_line method for simple data input

The data line parameter takes in each row of the data from the .csv file into a list, where each column from the .csv file is an index in the list. Therefore, to get the label and statement from the .csv file, the label will be the on the index 1 of the list. And the statement will be on the index 2 of the list.

After the label is indexed out from the list, there is a need to convert it to either 'FAKE' or 'REAL', hence the need to call the convert_label method.

After label conversion, this will then be used as the label, along with the statement in a (label, statement) tuple to be returned from the parse_data_line function.

Pre-process method for simple pre-processing

To convert the string to a list of tokens, the text is all converted to lower cases first. Followed by splitting the string into a list by splitting it from its whitespaces.

Looping within each word from the list, find words that has non word and digits character in it. This is with the use of the the regex [^\w] +, which means any character not in a-z, A-Z and 0-9 occurring once or unlimited times after each other in a word, will be checked. Using the package string punctuation function, the punctuation in each word will be removed. This would then result in empty strings. Example if word is … this will return an empty string after the strip punctation, while 'test.' will return a string without the punctuation which is 'test'. Empty strings are then removed from the token list. And the token list is returned.

13674 features is generated.

```
Now 0 rawData, 0 trainData, 0 testData
Preparing the dataset...
Now 10241 rawData, 0 trainData, 0 testData
Preparing training and test data...
After split, 10241 rawData, 8192 trainData, 2049 testData
Training Samples:
8192
Features:
13674
```

Q2.
For the simple feature extraction, after the sentiment is pre-processed, each word in token list is given an identification index and is recorded in the dictionary called global_feature_dict - keeping track of the words and its identification. If word is not found in the dictionary, a new index is generated, and new word is appended in the dictionary with its index.

Each word token index relating to the index from the word is given a weight. The weight in this case is the occurrence frequency of the word in the token list and this is recorded then in the dictionary called feature_vector. Dictionary, feature_vector, is then returned.

Q3.
The idea is that k-fold is achieved in cross validation. Where the test data length in each fold moves in a different area per fold as shown in Fig 3.1.

To make sure that the size of the test set remains the same in each fold, the integer result of the division of the size of each fold to the number folds is retrieved and is set as the size of the test set.

In order to achieve Fig 3.1, in each fold, upon knowing the size of the test data, all indexes from the start to the last index for the test data is identified and placed into the test indexes list. Within the fold, if the remaining indexes are not in the test indexes list, it is placed in the train indexes list.

The train and test data values are then identified from the dataset via the indexes in each respective train and data index list.

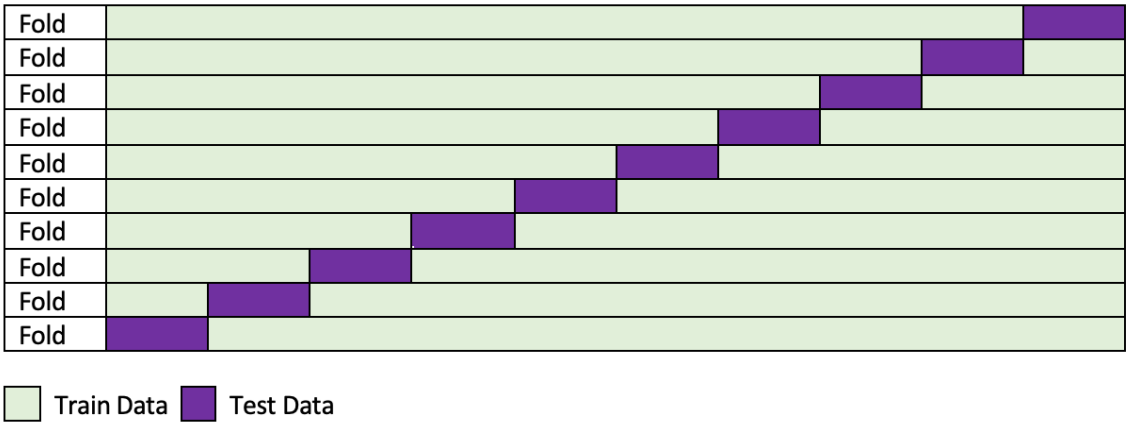


Fig 3.1 K-fold

Upon knowing the train and test data values for each fold, the training can be done at the train data and the evaluation or validation is done on the test set. The classification report is then presented and the precision, recall, f1-score and accuracy are recorded.

After the cross validation is completed, the average precision, recall, f1-score and accuracy are generated into the cv_results dictionary.

The average precision, recall, f1-score and accuracy is generated:

```
{'precision': 0.5562068297615173,
 'recall': 0.5518072289156626,
 'f1_score': 0.5520140623191052,
 'accuracy': 0.5518072289156626}
```

Q4.
Using the first fold train and test set, training done on train set and evaluation done on test set generating the predicted list. The confusion_matrix_heatmap is then called to draw the true positives and negatives, and also the false positive and negative.

False negative occurs when prediction results in FAKE when sentiment is actually REAL. True positive occurs when prediction correctly predicts the positive class, i.e., model predicts sentiment as REAL when it is indeed actually REAL. From figure 4.1 below, when the sentiment is indeed actually REAL, the model can predict the sentiment correctly, showing high value on true positive with 32. And it also shows that the false negative has a low value of 17.

False positive occurs when prediction results in REAL when sentiment is actually FAKE. True negative occurs when prediction correctly predicts the negative class, i.e., model predicts sentiment as FAKE when it is indeed actually FAKE. Also, from figure 4.1 below, when the sentiment is indeed actually FAKE, the model is not able to appropriately predict the sentiment correctly, showing high false positive of 20, compared to the true negative of 14.

It can also be seen that there is overfitting on the REAL (positive) outcome, where the total number of REAL predictions is 52, whilst the total outcome of FAKE (negative) predicted outcome is only 31.

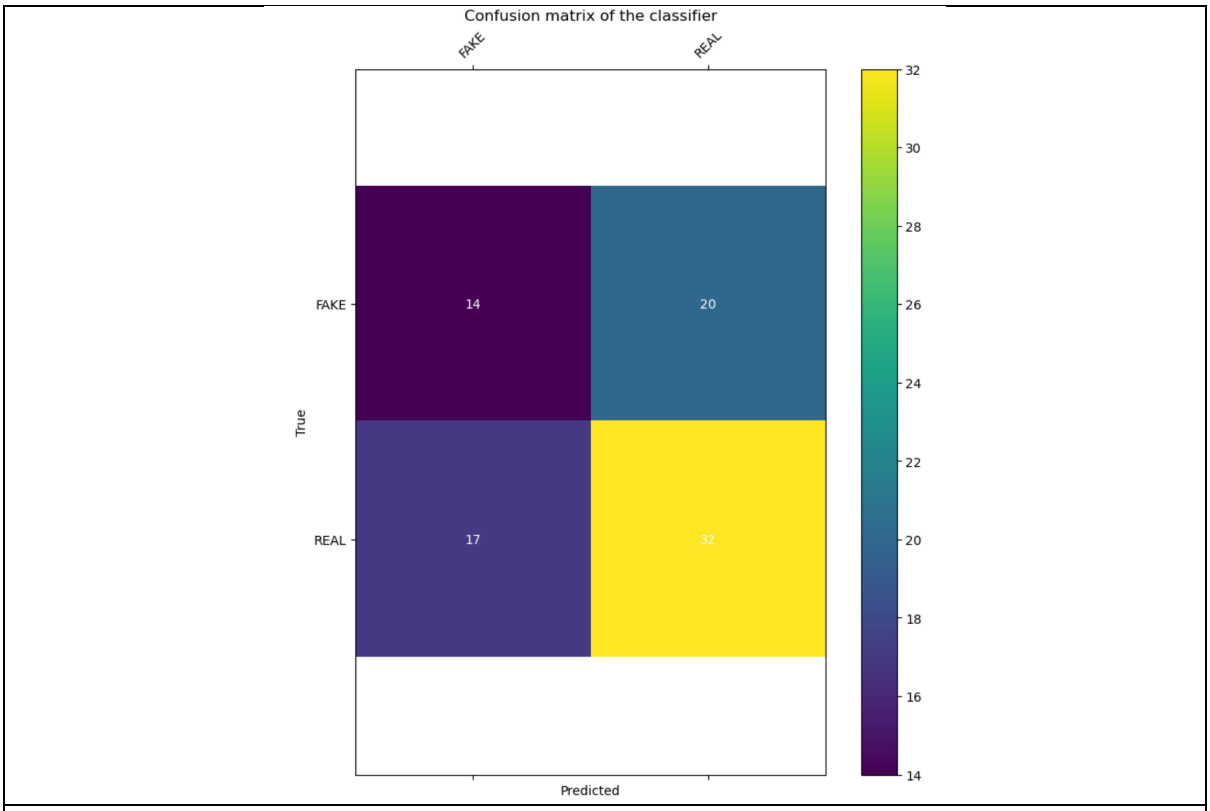


Figure 4.1 Confusion Heat Matrix

The classifier produces false negative and false positive for it doesn't have knowledge of the world. For example, index 7 in Figure 4.2a below, predicts that the sentiment is FAKE, for it has more negative words such as 'undocumented' and 'didn't' in it, confusing the model to classify the sentiment to a negative class. This created a False negative.

```
7:  says president barack obama promised a pathway to citizenship undocumented immigrants
and didnt deliver jack squat on any of it
Actual -> REAL | Predicted -> FAKE (False Negative)
```

Figure 4.2a False Negative

The model does not have any statistical knowledge if the sentiment in index 51 of Figure 4.2b is real. And only relies on the negative connotation of the sentiment such as ‘dont’ and ‘not’. Thus, generating a False negative.

```
51:  salaries of austin residents who dont work for city government have not been going up
3 percent a year the last several years
Actual -> REAL | Predicted -> FAKE (False Negative)
```

Figure 4.2b False Negative

Similarly, in index 19 of Figure 4.3a, the model does not have contextual knowledge of what is being talked about. The model does not know what the contents of the email practices are, but only relies on the positive word such as ‘allowed’, and thus, classifies the sentiment to a positive class. This created a False Positive.

```
19:  it was allowed referring to her email practices
Actual -> FAKE | Predicted -> REAL (False Positive)
```

Figure 4.3a False Positive

Another example of the model being confused due to not having background knowledge of the world is on index 74 of Figure 4.3b. The model classifies it to a positive class with the words ‘up’ and ‘soared’. Confusing the model and thus, generating a False positive.

```
74:  under new york mayor bill de blasio homicides are up by 20 percent and subway delays
have soared 45
Actual -> FAKE | Predicted -> REAL (False Positive)
```

Figure 4.3b False Positive

Overall classification for Q1 – Q4:

```
({2: 2, 42: 1, 265: 1, 699: 1, 614: 1, 25: 1, 847: 1, 136: 1, 7930: 1, 1837: 1, 15: 1, 484
: 1}, 'REAL')
Training Classifier...
Done training!
Precision: 0.554570
Recall: 0.554905
F Score:0.554729
```

Q5.
.ipynb file to be used for Q5 is named: NLP_Assignment_1_Q5.ipynb
Please note the negative-words.txt should be placed in the same location as the .ipynb file

In addition to the pre-processing created in Q2, the spacing for numbers representing percent, million, billion, am and pm are made as into one word example 90 percent, 90 billion, 80 million and 10:20 am are converted to 90percent, $90billion, 80 million and 10:20am. These numbers are kept. Also, words with @ and # at its prefix are also kept. Emails are kept, and words with punctuation in the middle are separated at the punctuation.

Punctuations is removed similar to what is done at Q2. Apostrophe-s representing ownership is removed. Plurals are assumed to have an s at the postfix of the word and thus the s word is removed.

Stop word removal is also applied at pre-processing to capture just the root words, decreasing the number of features.

To increase accuracy, for feature extraction, instead of counting the word frequency for the weight, the weight becomes frequency of word divided by the number of words in the token.

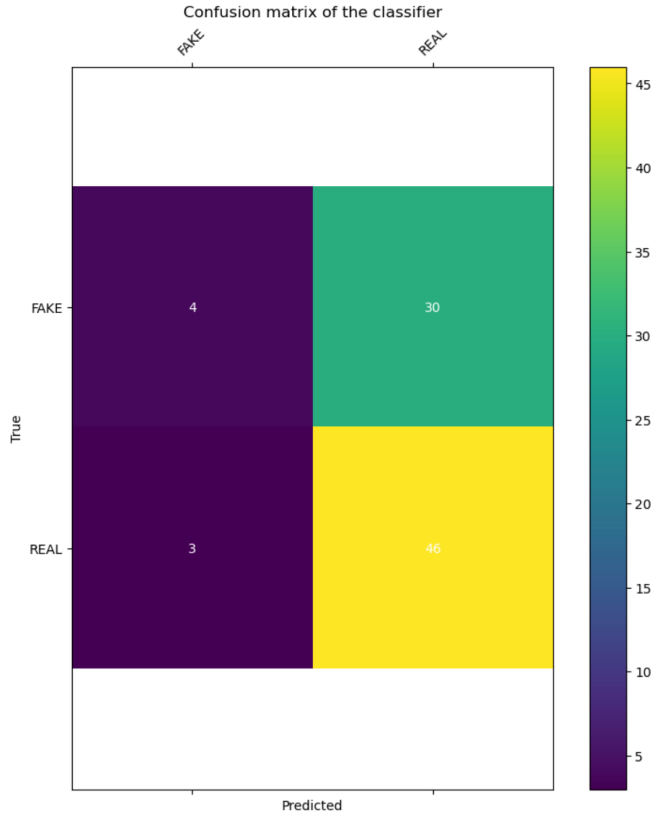
With this changes, 12545 features is generated.

```
Now 0 rawData, 0 trainData, 0 testData
Preparing the dataset...
Now 10241 rawData, 0 trainData, 0 testData
Preparing training and test data...
After split, 10241 rawData, 8192 trainData, 2049 testData
Training Samples:
8192
Features:
12545
```

For cross validation, the average precision, recall, f1-score and accuracy decreased in comparison from above Q3:

```
{'precision': 0.5372891961011101,
 'recall': 0.5469879518072289,
 'f1_score': 0.4960216418401833,
 'accuracy': 0.5469879518072289}
```

Overfitting to the REAL (positive) classes is more pronounced:



Overall, these changes are working as it can be seen that the overall training generates an overall better classification than Q1 – Q4:

```
({21: 0.125, 184: 0.125, 121: 0.125, 458: 0.125, 649: 0.125, 7208: 0.125, 1540: 0.125, 237
: 0.125}, 'REAL')
Training Classifier...
Done training!
Precision: 0.603480
Recall: 0.605173
F Score:0.587932
```

To improve on this, lemmatisation is added. Lemmatisation advantage is to increase the speed of the task by decreasing the number of features as it groups the words of the same meaning to a base word.

And in order not to lose the meaning and sequence of the sentiment context, generation of bigram and trigram is added into the preprocessing.

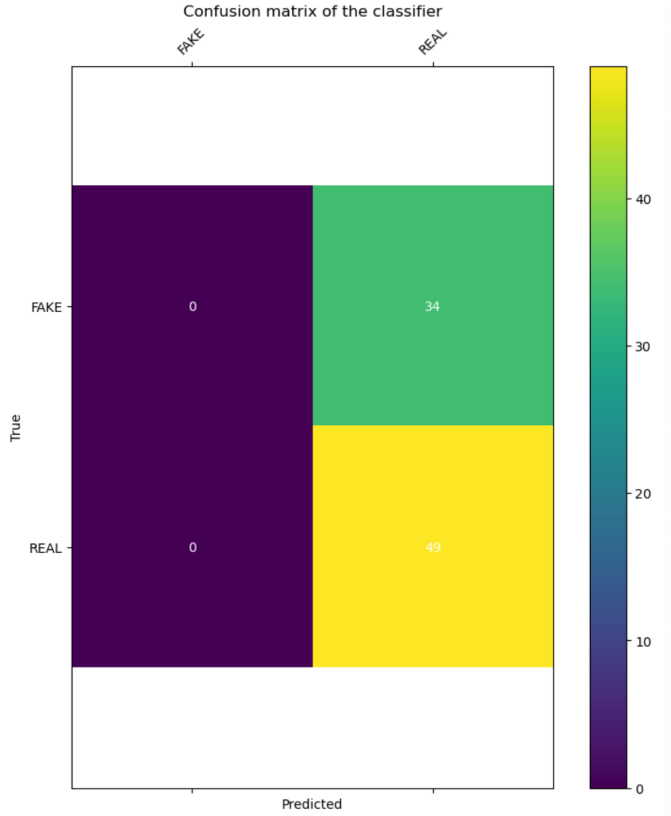
With this, the number of features increased:

```
Now 0 rawData, 0 trainData, 0 testData
Preparing the dataset...
Now 10241 rawData, 0 trainData, 0 testData
Preparing training and test data...
After split, 10241 rawData, 8192 trainData, 2049 testData
Training Samples:
8192
Features:
104410
```

The averages values in recall and accuracy increased in cross validation, however the precision and f1-score decreased.

```
{'precision': 0.4894479345994487,
 'recall': 0.5638554216867468,
 'f1_score': 0.42932826023514653,
 'accuracy': 0.5638554216867468}
```

Overfitting to REAL values becomes more prominent with no FAKE values being generated:



Overall, the precision has increased, however the recall and F score has decreased:

```
({61: 0.05555555555555555, 7825: 0.05555555555555555, 496: 0.05555555555555555, 48008: 0.0
5555555555555555, 319: 0.05555555555555555, 4446: 0.05555555555555555, 1296: 0.05555555555
555555, 87204: 0.05555555555555555, 13411: 0.05555555555555555, 1944: 0.05555555555555555,
14347: 0.05555555555555555, 42634: 0.05555555555555555, 87205: 0.05555555555555555, 5357:
0.05555555555555555, 87206: 0.05555555555555555, 28842: 0.05555555555555555, 85: 0.0555555
5555555555, 646: 0.05555555555555555}, 'REAL')
Training Classifier...
Done training!
Precision: 0.613499
Recall: 0.594924
F Score:0.540194
```

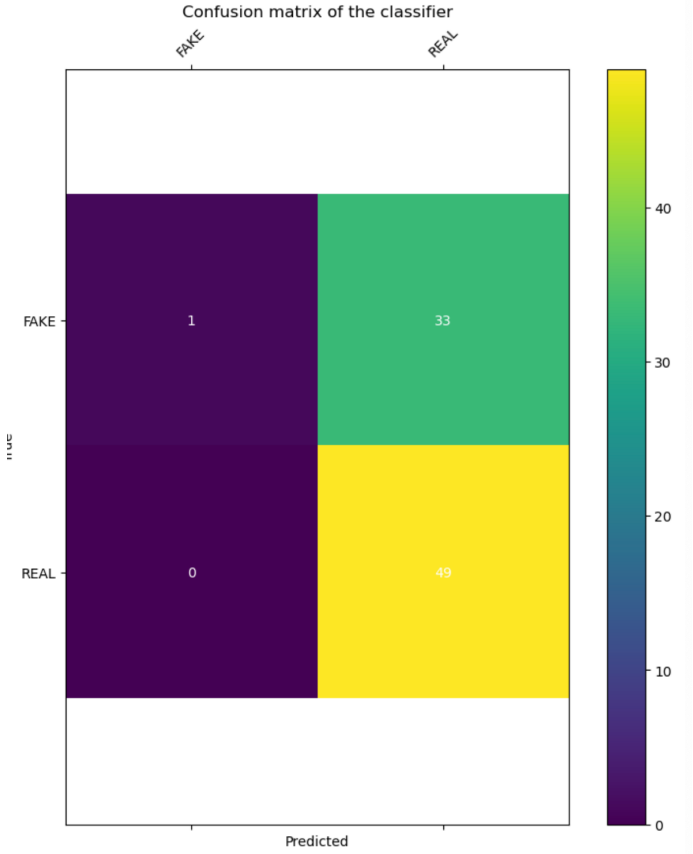
Removing Lemmatisation and stop words removal, since dataset is not very big, and lemmatisation may cause the sentiment meaning to change. And thus, the number of features increases:

```
Now 0 rawData, 0 trainData, 0 testData
Preparing the dataset...
Now 10241 rawData, 0 trainData, 0 testData
Preparing training and test data...
After split, 10241 rawData, 8192 trainData, 2049 testData
Training Samples:
8192
Features:
106779
```

This also resulted in increase in average of cross validation values, where precision have increased the most:

```
{'precision': 0.5963149089255015,
 'recall': 0.5843373493975903,
 'f1_score': 0.4803458773250928,
 'accuracy': 0.5843373493975903}
```

However, overfitting is still predominant:



Although, overall, the precision and recall are high. F-score is lower than precision and recall:

```
({2: 0.08695652173913043, 79: 0.043478260869565216, 8231: 0.043478260869565216, 565: 0.043
478260869565216, 49259: 0.043478260869565216, 1698: 0.043478260869565216, 21403: 0.0434782
60869565216, 1440: 0.043478260869565216, 47: 0.043478260869565216, 89222: 0.04347826086956
5216, 13972: 0.043478260869565216, 2131: 0.043478260869565216, 275: 0.043478260869565216,
14935: 0.043478260869565216, 43787: 0.043478260869565216, 89223: 0.043478260869565216, 568
2: 0.043478260869565216, 89224: 0.043478260869565216, 27: 0.043478260869565216, 29735: 0.0
43478260869565216, 85: 0.043478260869565216, 1096: 0.043478260869565216}, 'REAL')
Training Classifier...
Done training!
Precision: 0.623739
Recall: 0.612006
F Score:0.576820
```

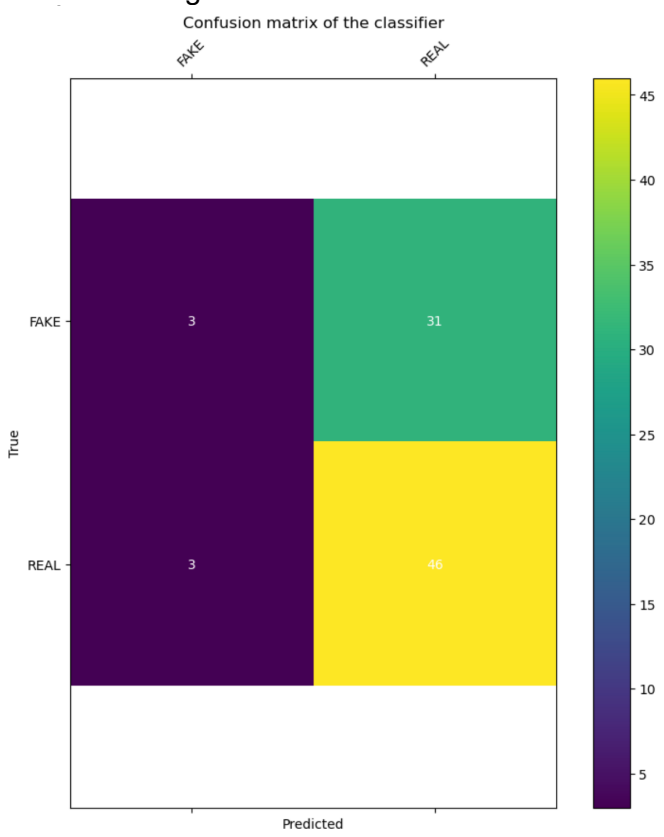
An external resource of negative words is retrieved, and weight of the negative words is increased. The weight for the negative words is increased by having the frequency of the negative words divided by the number of tokens.

Since there is no change at pre-processing, the number of features remain the same at 106779.

At cross validation, the average precision, recall and accuracy decreased, but the f1-score shows a bit of improvement.

```
{'precision': 0.5128875612661243,
 'recall': 0.5530120481927712,
 'f1_score': 0.46941563739437714,
 'accuracy': 0.5530120481927712}
```

The first fold of cross validation also shows some improvement in the overfitting, adding more values to the negative class.



The overall precision, recall and f-score remains the same at estimated 0.60, although a slight decrease in precision and recall can be seen.

```
({2: 0.08695652173913043, 79: 0.043478260869565216, 8231: 0.043478260869565216, 565: 0.043
478260869565216, 49259: 0.043478260869565216, 1698: 0.043478260869565216, 21403: 0.0434782
60869565216, 1440: 0.043478260869565216, 47: 0.043478260869565216, 89222: 0.04347826086956
5216, 13972: 0.043478260869565216, 2131: 0.043478260869565216, 275: 0.043478260869565216,
14935: 0.043478260869565216, 43787: 0.043478260869565216, 89223: 0.043478260869565216, 568
2: 0.043478260869565216, 89224: 0.043478260869565216, 27: 0.043478260869565216, 29735: 0.0
43478260869565216, 85: 0.043478260869565216, 1096: 0.043478260869565216}, 'REAL')
Training Classifier...
Done training!
Precision: 0.603380
Recall: 0.602245
F Score:0.576964
```

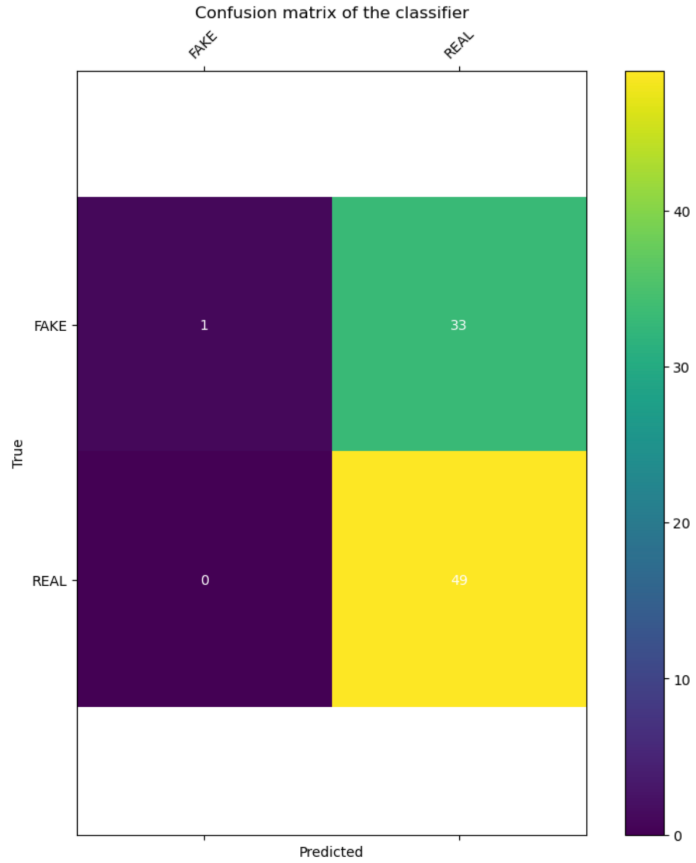
In order to be able to match the words to the external source, lemmatisation is again placed in the pre-processing, thus reducing the number of features as lemmas are generated.

```
Now 0 rawData, 0 trainData, 0 testData
Preparing the dataset...
Now 10241 rawData, 0 trainData, 0 testData
Preparing training and test data...
After split, 10241 rawData, 8192 trainData, 2049 testData
Training Samples:
8192
Features:
104940
```

In cross validation, average values have increased

```
{'precision': 0.6541348088786059,
 'recall': 0.5903614457831325,
 'f1_score': 0.4931667749385896,
 'accuracy': 0.5903614457831325}
```

Since the f1-score is low, it can be seen that there is some overfitting towards the positive class.



The overall precision, recall and f-score had increased.

```
({2: 0.08695652173913043, 58: 0.043478260869565216, 8109: 0.043478260869565216, 561: 0.043
478260869565216, 48484: 0.043478260869565216, 360: 0.043478260869565216, 4684: 0.043478260
869565216, 1425: 0.043478260869565216, 47: 0.043478260869565216, 87719: 0.0434782608695652
16, 13760: 0.043478260869565216, 2109: 0.043478260869565216, 273: 0.043478260869565216, 14
700: 0.043478260869565216, 43102: 0.043478260869565216, 87720: 0.043478260869565216, 5605:
0.043478260869565216, 87721: 0.043478260869565216, 27: 0.043478260869565216, 29277: 0.0434
78260869565216, 84: 0.043478260869565216, 730: 0.043478260869565216}, 'REAL')
Training Classifier...
Done training!
Precision: 0.619285
Recall: 0.611030
F Score:0.579441
```

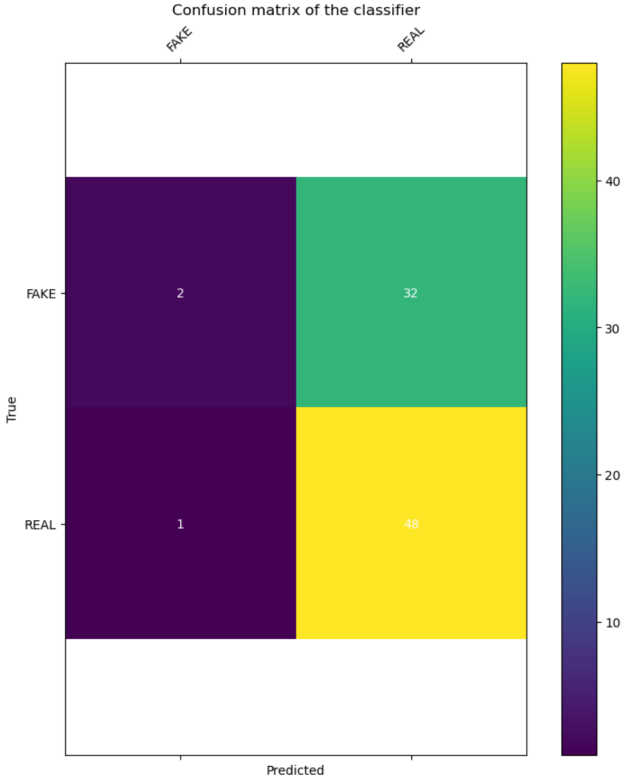
Stop words is added back in to reduce the number of features.

```
Now 0 rawData, 0 trainData, 0 testData
Preparing the dataset...
Now 10241 rawData, 0 trainData, 0 testData
Preparing training and test data...
After split, 10241 rawData, 8192 trainData, 2049 testData
Training Samples:
8192
Features:
70702
```

Average results in cross validation, shows some improvement on the f1-score as precision, recall, and accuracy decreases.

```
{'precision': 0.6067739638274976,
 'recall': 0.5698795180722892,
 'f1_score': 0.46826150745582407,
 'accuracy': 0.5698795180722892}
```

It can be seen that overfitting has improved for the first fold.



Overall, the precision, recall and f-score has decreased.

```
({33: 0.07142857142857142, 340: 0.07142857142857142, 3837: 0.07142857142857142, 217: 0.071
42857142857142, 26187: 0.07142857142857142, 891: 0.07142857142857142, 58747: 0.07142857142
857142, 1310: 0.07142857142857142, 28089: 0.07142857142857142, 58748: 0.07142857142857142,
58749: 0.07142857142857142, 3532: 0.07142857142857142, 444: 0.07142857142857142, 58750: 0.
07142857142857142}, 'REAL')
Training Classifier...
Done training!
Precision: 0.601804
Recall: 0.598341
F Score:0.566244
```

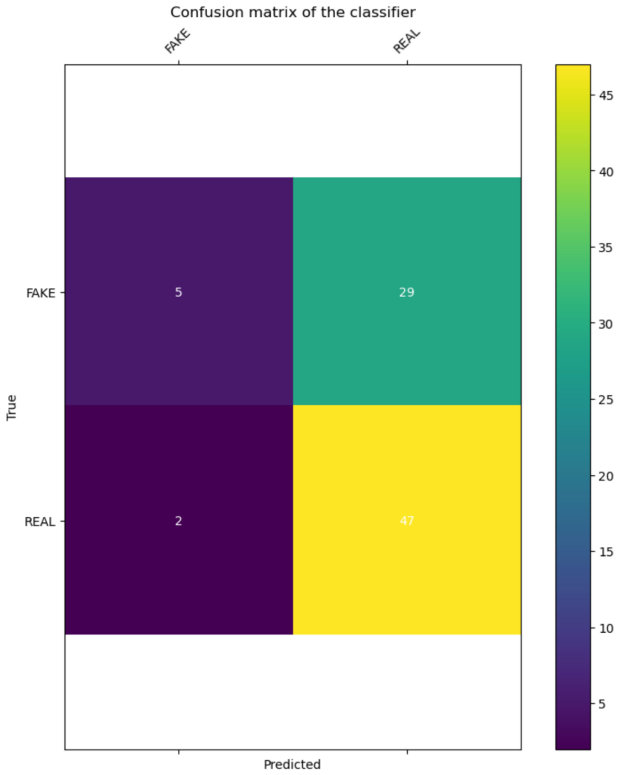
Bigram is removed and trigram is only applied. This reduced the number of features and improved the overall results.

```
Now 0 rawData, 0 trainData, 0 testData
Preparing the dataset...
Now 10241 rawData, 0 trainData, 0 testData
Preparing training and test data...
After split, 10241 rawData, 8192 trainData, 2049 testData
Training Samples:
8192
Features:
37937
```

Cross validation increased f1-score.

```
{'precision': 0.5964190046104922,
 'recall': 0.5734939759036145,
 'f1_score': 0.5061977986999295,
 'accuracy': 0.5734939759036145}
```

Overfitting is also decreased.



Overall, the precision, recall and f-score increases:

```
({25: 0.1, 248: 0.1, 159: 0.1, 15051: 0.1, 638: 0.1, 923: 0.1, 16044: 0.1, 31864: 0.1, 236
2: 0.1, 323: 0.1}, 'REAL')
Training Classifier...
Done training!
Precision: 0.604954
Recall: 0.605173
F Score:0.584313
```

Hence, improving the pre-processing function by applying normalisation, removal of punctuation, stop word removal, lemmatisation and using the trigram model allows improvement in the result of the model.

Adjusting the weight of the tokens, where more weight is added to negative connotated words also improved the overfitting being observed.

Since the overall precision increased in comparison from the model created for Q1 to Q4, this model classification will be used in the subsequent Q6.

Q6.
.ipynb file to be used for Q6 is named: NLP_Assignment_1_Q6.ipynb
Please note the negative-words.txt should be placed in the same location as the .ipynb file

The parse data line is updated to include the 'speaker' feature in the statement. This improved the overall classification of the model where the precision increased from 0.605 to 0.634. Recall ffrom 0.605 to 0.628 and F score from 0.584 to 0.607

```
({26: 0.09090909090909091, 266: 0.09090909090909091, 170: 0.09090909090909091, 16375: 0.09
090909090909091, 696: 0.09090909090909091, 1006: 0.09090909090909091, 17462: 0.09090909090
909091, 34387: 0.09090909090909091, 2576: 0.09090909090909091, 349: 0.09090909090909091, 8
987: 0.09090909090909091}, 'REAL')
Training Classifier...
Done training!
Precision: 0.634018
Recall: 0.628111
F Score:0.606621
```

Adding another feature called 'party affliation' also improves the overall classification of the model as shown.

```
({27: 0.07692307692307693, 275: 0.07692307692307693, 178: 0.07692307692307693, 17194: 0.07
692307692307693, 720: 0.07692307692307693, 1038: 0.07692307692307693, 18344: 0.07692307692
307693, 36473: 0.07692307692307693, 2667: 0.07692307692307693, 361: 0.07692307692307693, 9
381: 0.07692307692307693, 31: 0.07692307692307693, 36474: 0.07692307692307693}, 'REAL')
Training Classifier...
Done training!
Precision: 0.640158
Recall: 0.633968
F Score:0.613972
```

Hence, with more value-adding features added into the model, the overall classification measure improves as the model learns.