Natural Language Processing

Assignment 4

Text Classification Using BERT

Yogasree Segar

CWID: A20448385

Table of Contents

Abstract:

The aim of the assignment is to classify the texts of different authors whose native is not English into their native languages using BERT and Logistic Regression and to evaluate the efficiency of the system. The BERT model developed by the researchers in Google is a pre trained model that is used to determine the word embedding for the given input file. Finally the Logistic Regression model is trained with these vectors and used to classify the native language of the authors.

Prior Materials:

BERT Repository - BERT repository cloned from the website

Data to be trained and tested - Data used to train BERT and fit in Logistic Regression model

Pre trained BERT data model – Previously trained model

Models Used:

The two main models used in this assignment are

1. BERT
2. Logistic Regression

1. BERT

BERT stands for Bidirectional Encoder Representations from Transformers. It is an open source pre training model for language classification developed by researchers in Google. It has outperformed many other language processing models with its high efficiency.

Working of BERT

BERT acquires its high efficiency because of its bidirectional and unsupervised language representations. It creates word embedding for each word in the input based on the word before and after it. This makes the pre trained model more efficient as it is capable of handling homonyms.

BERT Strategies

The pre trained model is made better with two different strategies. They are

Mask Language Model: 15% of the words in the given input is masked with some token words and the BERT model attempts to determine the words from the non-masked words beside it.

Next Sentence Prediction: BERT then is made to learn the next sentence once a sentence is given. This is done by giving a pair of sentence during training and the model is made to learn the relation between the sentences.

2. Logistic Regression

Logistic regression is a statistical model to determine the outcome of a dataset that has one or more independent variable. It is a supervised Machine Learning model used to determine the outcome which is mostly binary.

Evaluation Metrics:

Evaluation metrics are used to determine the efficiency of the system. This is done with the help of three different metrics namely

- Precision
- Recall
- F-measure

These three metrics require four different terms to evaluate the model. They are

- True Positives – Items present both in predicted and true class
- False Positives – Items in predicted class but not in true class
- True Negatives – Items not in true and predicted class
- False Negatives – Items in true class but not in predicted class

Thus Precision and Recall is given as

- Precision = True Positives / (True Positives + False Positives)
- Recall = True Positives / (True Positives + False Negatives)

F-measure is given as

- F-measure = (2 * Precision * Recall) / (Precision + Recall)

Confusion Matrix

The confusion matrix is a table of the evaluation metrics. It is a table that has four different markings marked across the predicted values and the actual values.

Procedure:

All the prior documents are downloaded. The procedure following the download to obtain the results are as follows:-

1. The csv files are converted to text files and are modified to get one text in one line. For this modification the first column and the first row of the csv files are removed. The code is as follows:

   Unix code to remove first column and to convert to text files:-

   ```
   sed -r 's/^[^,]+,//' lang_id_train.csv > out_train1.txt

   sed -r 's/^[^,]+,//' lang_id_train.csv > out_train1.txt

   sed -r 's/^[^,]+,//' lang_id_train.csv > out_train1.txt
   ```

   Unix code to remove first row:-

   ```
   sed '1d' out_test1.txt > out_test1.txt

   sed '1d' out_test1.txt > out_test1.txt

   sed '1d' out_test1.txt > out_test1.txt
   ```

2. The text files are then stored in the respective repository

   This PC\Documents\repos\bert\bert_input_data

3. The pre trained BERT model is also placed in the same location and the shell file run_bert_fv.sh is run. This file calls the python code extract_features.py which requires the installation of the package tensorflow (version 1.14). Thus the package is downloaded in anaconda prompt using the commands:

   - Creating new environment - tensorenviron for the tensorflow
     conda create –name tensorenviron
   - Tensorflow is then installed using the command
     pip install tensorflow==1.14.0
   - The shell file is finally ran by calling it
     run_bert_fv.sh

4. The shell file ran for about an hour resulting in three JSON files in the location

   This PC\Documents\repos\bert\bert_output_data

5. In a new jupyter notebook created all these files are uploaded and the Example Model Training BERT vectors.ipnyb is ran. The train file is fitted into the logistic regression model.

6. Finally the efficiency of the model is predicted using the code

   ```
   lr_model.score(X, y) (for the train file)

   lr_model.score(X_test, y_test) (for the test file)

   lr_model.score(X_eval, y_eval) (for the eval file)
   ```

7. The evaluation metrics for the test dataset are calculated as follows

   - Confusion matrix:
     ```
     from sklearn.metrics import confusion_matrix
     cm = confusion_matrix(y_test, y_test_pred)
     print(cm)
     ```

   - Precision
     ```
     from sklearn.metrics import precision_score
     precision_score(y_test, y_test_pred, average = None)
     ```

   - Recall
     ```
     from sklearn.metrics import recall_score
     recall_score(y_test, y_test_pred, average = None)
     ```

   - Precision-Recall-Fscore
     ```
     from sklearn.metrics import precision_recall_fscore_support
     precision_recall_fscore_support(y_test, y_test_pred, average = None)
     ```

Outcome:

The outcome of the language model is as follows

Train.csv

The following are the evaluation metrics calculated for the test dataset. It consisted ten different languages thus yielding in ten different values for each metric.

Score for the train dataset: 0.7348

Confusion matrix:

```
[[ 99  11   9   7  12  14  10  16   9  13]
 [ 11  68  14  12  46  12  10   4   9  14]
 [  8  13  99  23  11  17  10   5   5   9]
 [  7  17  23  91  13   8  12   9  12   8]
 [ 12  36  19  14  63   9  10  14   6  17]
 [ 11  14   8   3   5 102  29  10   9   9]
 [ 10   8  14   5   6  23 114  14   0   6]
 [ 17   2  11   5  14  17  14 104   5  11]
 [ 12  10   7  15   9   2   1  13 120  11]
 [ 12  29  11  13  19  13  12  11   8  72]]
```

Precision score:

```
array([0.49748744, 0.32692308, 0.46046512, 0.48404255, 0.31818182,
       0.47004608, 0.51351351, 0.52      , 0.6557377 , 0.42352941])
```

Recall score:

```
array([0.495, 0.34 , 0.495, 0.455, 0.315, 0.51 , 0.57 , 0.52 , 0.6  ,
       0.36 ])
```

F-measure:

```
(array([0.49748744, 0.32692308, 0.46046512, 0.48404255, 0.31818182,
        0.47004608, 0.51351351, 0.52      , 0.6557377 , 0.42352941]),
 array([0.495, 0.34 , 0.495, 0.455, 0.315, 0.51 , 0.57 , 0.52 , 0.6  ,
        0.36 ]),
 array([0.4962406 , 0.33333333, 0.47710843, 0.46907216, 0.31658291,
        0.48920863, 0.54028436, 0.52      , 0.62663185, 0.38918919]),
 array([200, 200, 200, 200, 200, 200, 200, 200, 200, 200], dtype=int64))
```

Test.csv

Score for the test dataset: 0.466

The following are the evaluation metrics calculated for the test dataset. It consisted ten different languages thus yielding in ten different values for each metric.

Confusion matrix:

```
[[ 91  11  10  10   7   7   9  26  15  14]
 [ 11  79  13  16  35  11   4  10  12   9]
 [ 11  10 119  15  11   6  15   3   9   1]
 [  6  15  26  95   8   7  11   6  16  10]
 [ 14  43  22  17  62   7   5   8   3  19]
 [ 11   8   7   6   5 105  22  19   5  12]
 [  6   6  13   5   5  27 108  14   7   9]
 [ 16   5  10   5  11  19   8 113   3  10]
 [  4  10   8  15  10   9   3   3 122  16]
 [ 17  15  13  16  19  14   6   8  15  77]]
```

Precision score:

```
array([0.48663102, 0.39108911, 0.49377593, 0.475     , 0.3583815 ,
       0.49528302, 0.56544503, 0.53809524, 0.58937198, 0.43502825])
```

Recall score:

```
array([0.455, 0.395, 0.595, 0.475, 0.31 , 0.525, 0.54 , 0.565, 0.61 ,
       0.385])
```

F-measure:

```
(array([0.48663102, 0.39108911, 0.49377593, 0.475     , 0.3583815 ,
        0.49528302, 0.56544503, 0.53809524, 0.58937198, 0.43502825]),
 array([0.455, 0.395, 0.595, 0.475, 0.31 , 0.525, 0.54 , 0.565, 0.61 ,
        0.385]),
 array([0.47028424, 0.39303483, 0.53968254, 0.475     , 0.33243968,
        0.50970874, 0.55242967, 0.55121951, 0.5995086 , 0.40848806]),
 array([200, 200, 200, 200, 200, 200, 200, 200, 200, 200], dtype=int64))
```

Eval.csv

The following are the evaluation metrics calculated for the test dataset. It consisted ten different languages thus yielding in ten different values for each metric.

Score for the eval dataset: 0.4855

Confusion matrix:

```
[[ 91  11  10  10   7   7   9  26  15  14]
 [ 11  79  13  16  35  11   4  10  12   9]
 [ 11  10 119  15  11   6  15   3   9   1]
 [  6  15  26  95   8   7  11   6  16  10]
 [ 14  43  22  17  62   7   5   8   3  19]
 [ 11   8   7   6   5 105  22  19   5  12]
 [  6   6  13   5   5  27 108  14   7   9]
 [ 16   5  10   5  11  19   8 113   3  10]
 [  4  10   8  15  10   9   3   3 122  16]
 [ 17  15  13  16  19  14   6   8  15  77]]
```

Precision score:

```
array([0.48663102, 0.39108911, 0.49377593, 0.475     , 0.3583815 ,
       0.49528302, 0.56544503, 0.53809524, 0.58937198, 0.43502825])
```

Recall score:

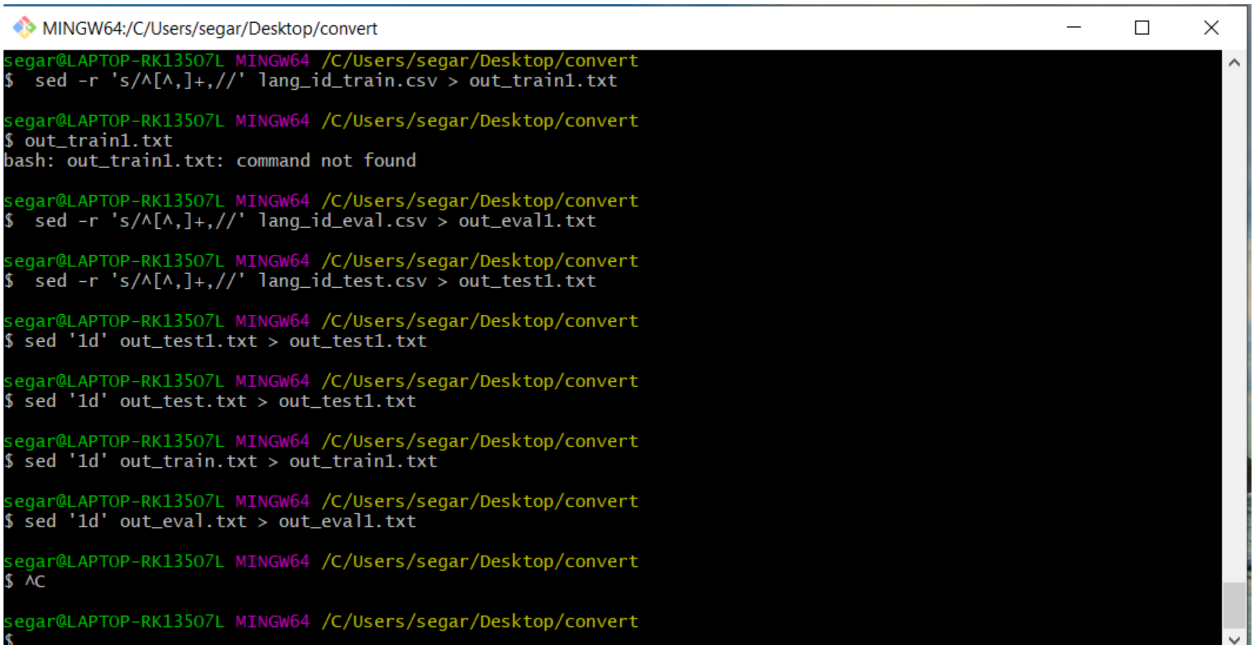
```
array([0.455, 0.395, 0.595, 0.475, 0.31 , 0.525, 0.54 , 0.565, 0.61 ,
       0.385])
```

F-measure:

```
(array([0.48663102, 0.39108911, 0.49377593, 0.475     , 0.3583815 ,
        0.49528302, 0.56544503, 0.53809524, 0.58937198, 0.43502825]),
 array([0.455, 0.395, 0.595, 0.475, 0.31 , 0.525, 0.54 , 0.565, 0.61 ,
        0.385]),
 array([0.47028424, 0.39303483, 0.53968254, 0.475     , 0.33243968,
        0.50970874, 0.55242967, 0.55121951, 0.5995086 , 0.40848806]),
 array([200, 200, 200, 200, 200, 200, 200, 200, 200, 200], dtype=int64))
```

Output:

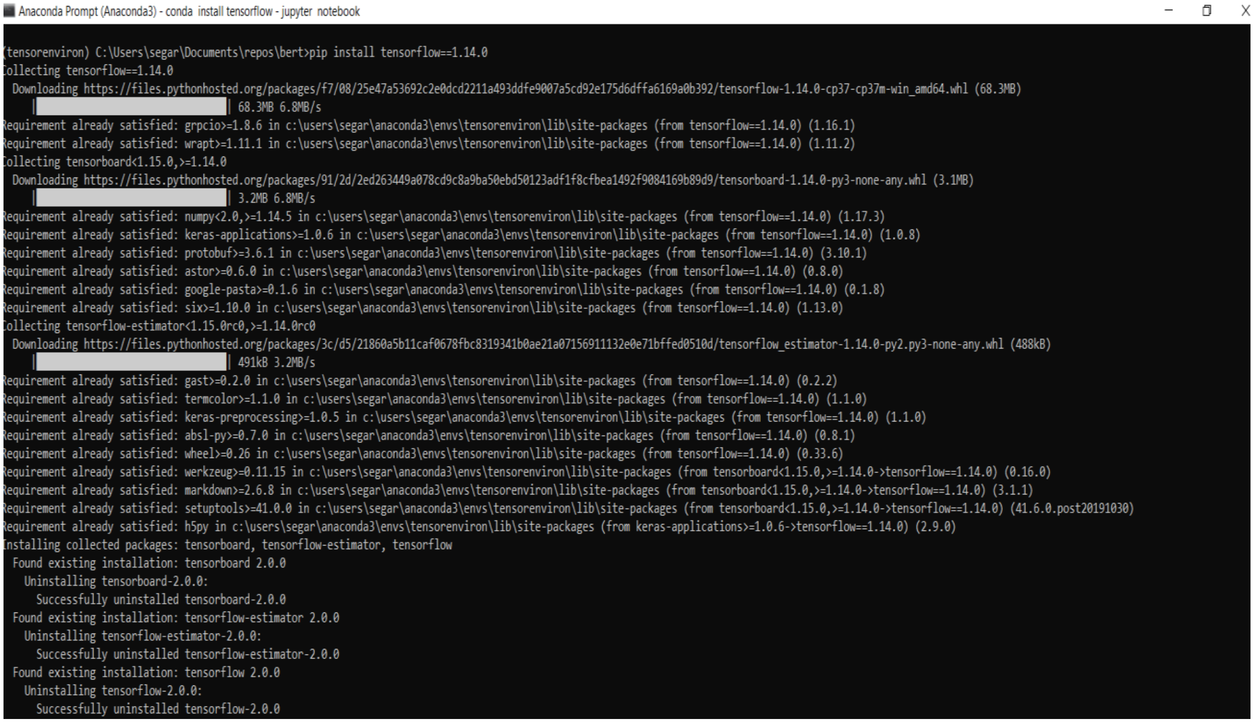
1. Modification of csv files



MINGW64:/C/Users/segar/Desktop/convert

```
segar@LAPTOP-RK13507L MINGW64 /C/Users/segar/Desktop/convert
$  sed -r 's/^[^,]+,//' lang_id_train.csv > out_train1.txt

segar@LAPTOP-RK13507L MINGW64 /C/Users/segar/Desktop/convert
$ out_train1.txt
bash: out_train1.txt: command not found

segar@LAPTOP-RK13507L MINGW64 /C/Users/segar/Desktop/convert
$  sed -r 's/^[^,]+,//' lang_id_eval.csv > out_eval1.txt

segar@LAPTOP-RK13507L MINGW64 /C/Users/segar/Desktop/convert
$  sed -r 's/^[^,]+,//' lang_id_test.csv > out_test1.txt

segar@LAPTOP-RK13507L MINGW64 /C/Users/segar/Desktop/convert
$ sed '1d' out_test1.txt > out_test1.txt

segar@LAPTOP-RK13507L MINGW64 /C/Users/segar/Desktop/convert
$ sed '1d' out_test.txt > out_test1.txt

segar@LAPTOP-RK13507L MINGW64 /C/Users/segar/Desktop/convert
$ sed '1d' out_train.txt > out_train1.txt

segar@LAPTOP-RK13507L MINGW64 /C/Users/segar/Desktop/convert
$ sed '1d' out_eval.txt > out_eval1.txt

segar@LAPTOP-RK13507L MINGW64 /C/Users/segar/Desktop/convert
$ ^C

segar@LAPTOP-RK13507L MINGW64 /C/Users/segar/Desktop/convert
$
```

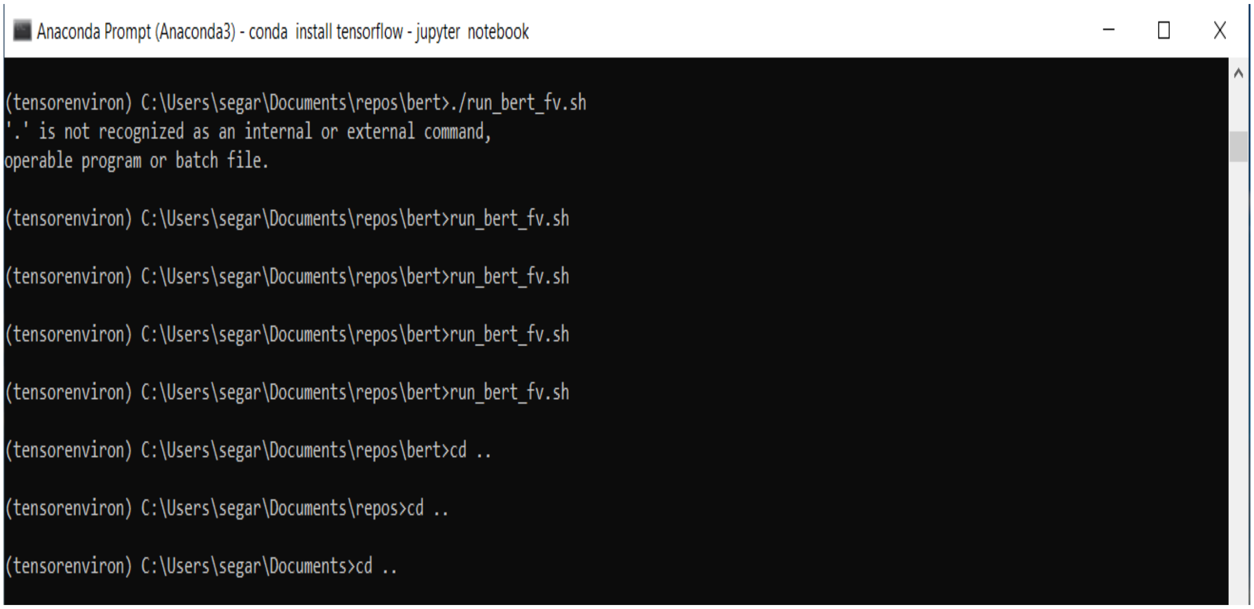
2. Installing tensorflow



Anaconda Prompt (Anaconda3) - conda install tensorflow - jupyter notebook

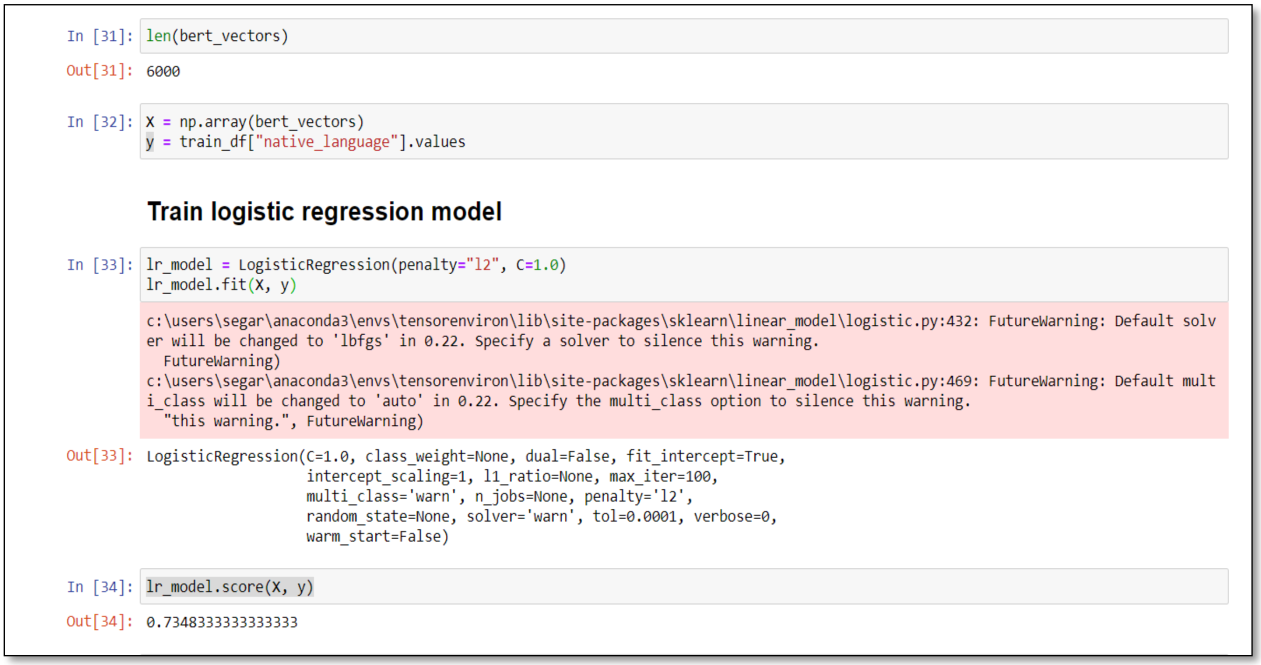
```
(tensorenviron) C:\Users\segar\Documents\repos\bert>pip install tensorflow==1.14.0
Collecting tensorflow==1.14.0
  Downloading https://files.pythonhosted.org/packages/f7/08/25e47a53692c2e0dcd2211a493ddfe9007a5cd92e175d6dffa6169a0b392/tensorflow-1.14.0-cp37-cp37m-win_amd64.whl (68.3MB)
     |████████████████████████████████| 68.3MB 6.8MB/s
Requirement already satisfied: grpcio>=1.8.6 in c:\users\segar\anaconda3\envs\tensorenviron\lib\site-packages (from tensorflow==1.14.0) (1.16.1)
Requirement already satisfied: wrapt>=1.11.1 in c:\users\segar\anaconda3\envs\tensorenviron\lib\site-packages (from tensorflow==1.14.0) (1.11.2)
Collecting tensorboard<1.15.0,>=1.14.0
  Downloading https://files.pythonhosted.org/packages/91/2d/2ed263449a078cd9c8a9ba50ebd50123adf1f8cfbea1492f9084169b89d9/tensorboard-1.14.0-py3-none-any.whl (3.1MB)
     |████████████████████████████████| 3.2MB 6.8MB/s
Requirement already satisfied: numpy<2.0,>=1.14.5 in c:\users\segar\anaconda3\envs\tensorenviron\lib\site-packages (from tensorflow==1.14.0) (1.17.3)
Requirement already satisfied: keras-applications>=1.0.6 in c:\users\segar\anaconda3\envs\tensorenviron\lib\site-packages (from tensorflow==1.14.0) (1.0.8)
Requirement already satisfied: protobuf>=3.6.1 in c:\users\segar\anaconda3\envs\tensorenviron\lib\site-packages (from tensorflow==1.14.0) (3.10.1)
Requirement already satisfied: astor>=0.6.0 in c:\users\segar\anaconda3\envs\tensorenviron\lib\site-packages (from tensorflow==1.14.0) (0.8.0)
Requirement already satisfied: google-pasta>=0.1.6 in c:\users\segar\anaconda3\envs\tensorenviron\lib\site-packages (from tensorflow==1.14.0) (0.1.8)
Requirement already satisfied: six>=1.10.0 in c:\users\segar\anaconda3\envs\tensorenviron\lib\site-packages (from tensorflow==1.14.0) (1.13.0)
Collecting tensorflow-estimator<1.15.0rc0,>=1.14.0rc0
  Downloading https://files.pythonhosted.org/packages/3c/d5/21860a5b11caf0678fbc8319341b0ae21a07156911132e0e71bffed0510d/tensorflow_estimator-1.14.0-py2.py3-none-any.whl (488kB)
     |████████████████████████████████| 491kB 3.2MB/s
Requirement already satisfied: gast>=0.2.0 in c:\users\segar\anaconda3\envs\tensorenviron\lib\site-packages (from tensorflow==1.14.0) (0.2.2)
Requirement already satisfied: termcolor>=1.1.0 in c:\users\segar\anaconda3\envs\tensorenviron\lib\site-packages (from tensorflow==1.14.0) (1.1.0)
Requirement already satisfied: keras-preprocessing>=1.0.5 in c:\users\segar\anaconda3\envs\tensorenviron\lib\site-packages (from tensorflow==1.14.0) (1.1.0)
Requirement already satisfied: absl-py>=0.7.0 in c:\users\segar\anaconda3\envs\tensorenviron\lib\site-packages (from tensorflow==1.14.0) (0.8.1)
Requirement already satisfied: wheel>=0.26 in c:\users\segar\anaconda3\envs\tensorenviron\lib\site-packages (from tensorflow==1.14.0) (0.33.6)
Requirement already satisfied: werkzeug>=0.11.15 in c:\users\segar\anaconda3\envs\tensorenviron\lib\site-packages (from tensorboard<1.15.0,>=1.14.0->tensorflow==1.14.0) (0.16.0)
Requirement already satisfied: markdown>=2.6.8 in c:\users\segar\anaconda3\envs\tensorenviron\lib\site-packages (from tensorboard<1.15.0,>=1.14.0->tensorflow==1.14.0) (3.1.1)
Requirement already satisfied: setuptools>=41.0.0 in c:\users\segar\anaconda3\envs\tensorenviron\lib\site-packages (from tensorboard<1.15.0,>=1.14.0->tensorflow==1.14.0) (41.6.0.post20191030)
Requirement already satisfied: h5py in c:\users\segar\anaconda3\envs\tensorenviron\lib\site-packages (from keras-applications>=1.0.6->tensorflow==1.14.0) (2.9.0)
Installing collected packages: tensorboard, tensorflow-estimator, tensorflow
  Found existing installation: tensorboard 2.0.0
    Uninstalling tensorboard-2.0.0:
      Successfully uninstalled tensorboard-2.0.0
  Found existing installation: tensorflow-estimator 2.0.0
    Uninstalling tensorflow-estimator-2.0.0:
      Successfully uninstalled tensorflow-estimator-2.0.0
  Found existing installation: tensorflow 2.0.0
    Uninstalling tensorflow-2.0.0:
      Successfully uninstalled tensorflow-2.0.0
```

3. Running run_bert_fv.sh


Anaconda Prompt (Anaconda3) - conda install tensorflow - jupyter notebook

```
(tensorenviron) C:\Users\segar\Documents\repos\bert>./run_bert_fv.sh
'.' is not recognized as an internal or external command,
operable program or batch file.

(tensorenviron) C:\Users\segar\Documents\repos\bert>run_bert_fv.sh

(tensorenviron) C:\Users\segar\Documents\repos\bert>run_bert_fv.sh

(tensorenviron) C:\Users\segar\Documents\repos\bert>run_bert_fv.sh

(tensorenviron) C:\Users\segar\Documents\repos\bert>run_bert_fv.sh

(tensorenviron) C:\Users\segar\Documents\repos\bert>cd ..

(tensorenviron) C:\Users\segar\Documents\repos>cd ..

(tensorenviron) C:\Users\segar\Documents>cd ..
```

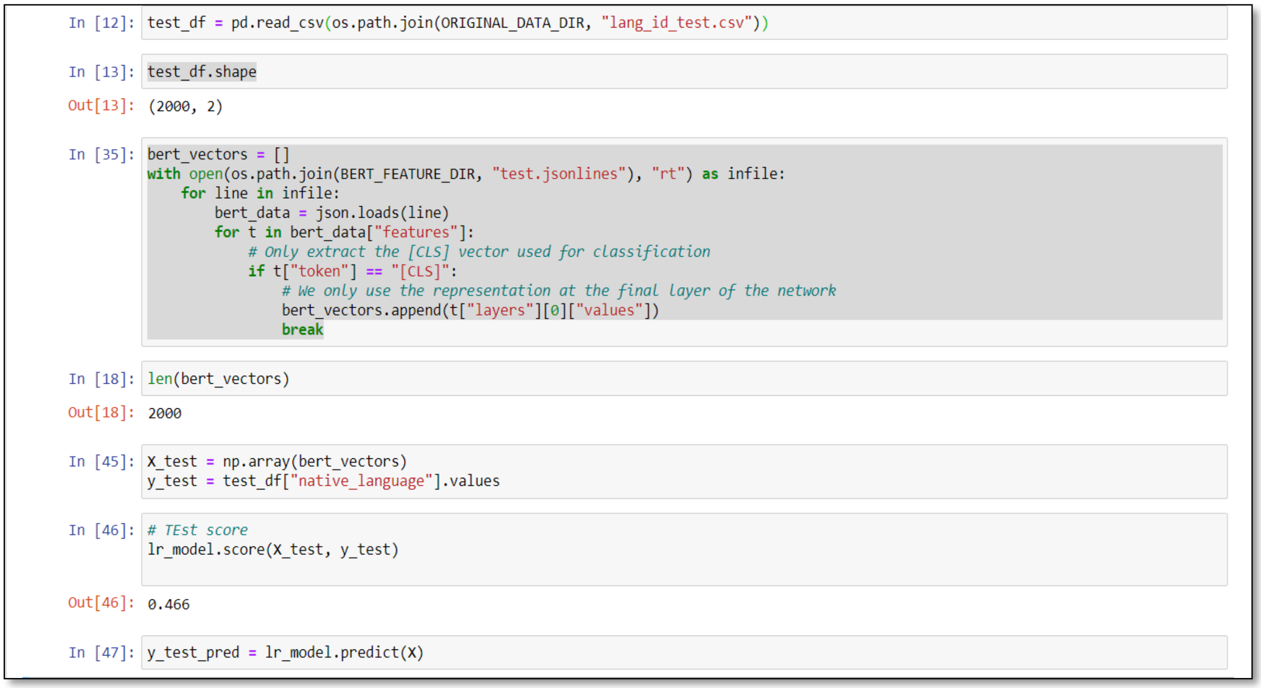
4. Running the jupyter notebook



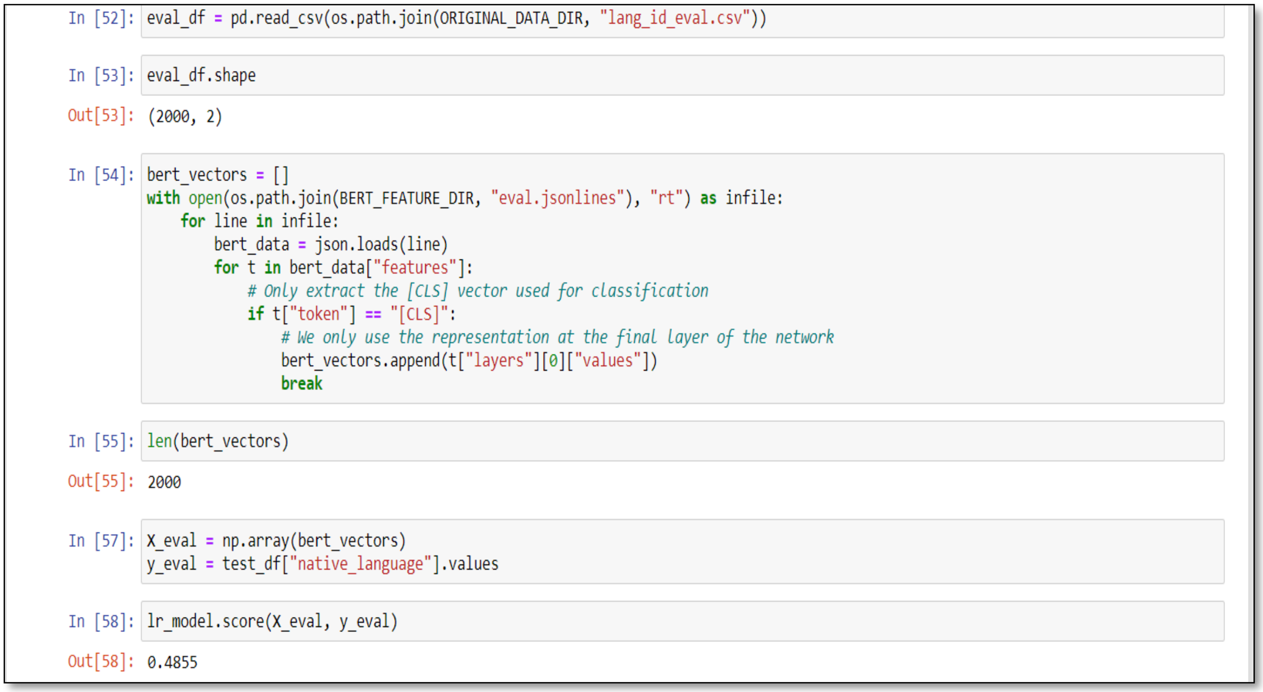
```
In [31]: len(bert_vectors)

Out[31]: 6000

In [32]: X = np.array(bert_vectors)
         y = train_df["native_language"].values
```

Train logistic regression model

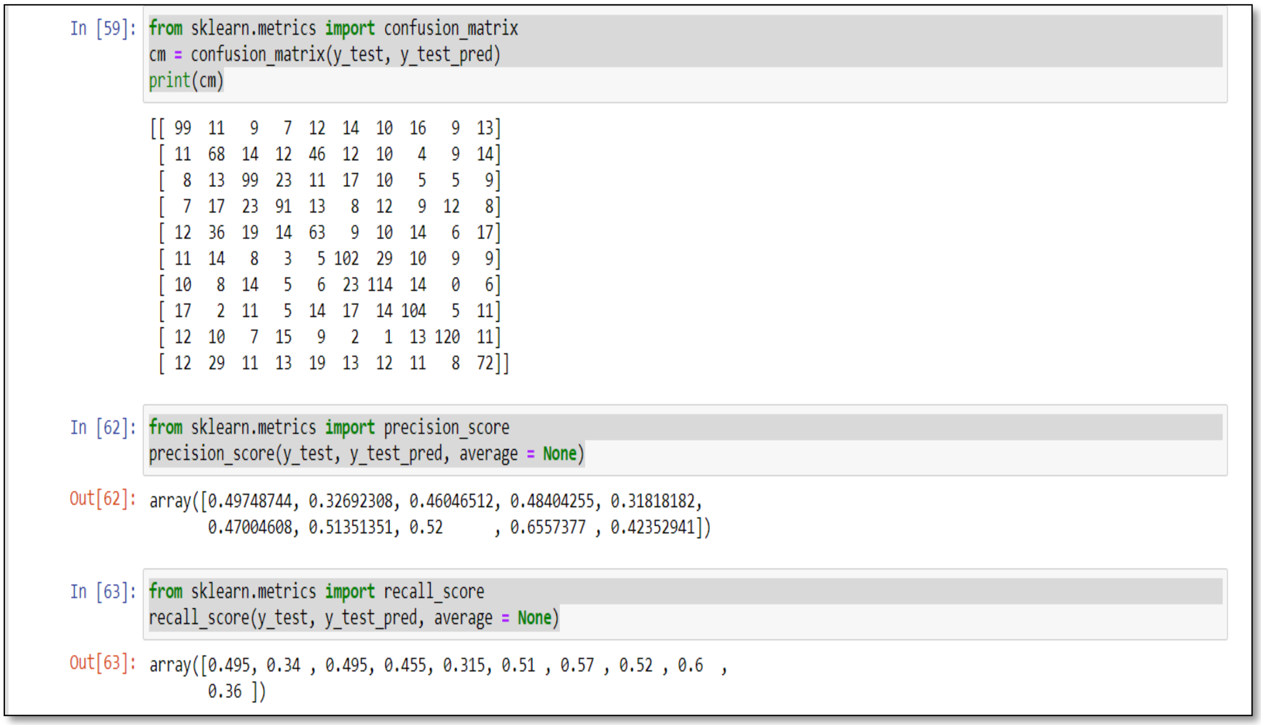
```
In [33]: lr_model = LogisticRegression(penalty="l2", C=1.0)
         lr_model.fit(X, y)

         c:\users\segar\anaconda3\envs\tensorenviron\lib\site-packages\sklearn\linear_model\logistic.py:432: FutureWarning: Default solv
         er will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
           FutureWarning)
         c:\users\segar\anaconda3\envs\tensorenviron\lib\site-packages\sklearn\linear_model\logistic.py:469: FutureWarning: Default mult
         i_class will be changed to 'auto' in 0.22. Specify the multi_class option to silence this warning.
           "this warning.", FutureWarning)

Out[33]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                            intercept_scaling=1, l1_ratio=None, max_iter=100,
                            multi_class='warn', n_jobs=None, penalty='l2',
                            random_state=None, solver='warn', tol=0.0001, verbose=0,
                            warm_start=False)

In [34]: lr_model.score(X, y)

Out[34]: 0.7348333333333333
```



```
In [12]: test_df = pd.read_csv(os.path.join(ORIGINAL_DATA_DIR, "lang_id_test.csv"))

In [13]: test_df.shape

Out[13]: (2000, 2)

In [35]: bert_vectors = []
         with open(os.path.join(BERT_FEATURE_DIR, "test.jsonlines"), "rt") as infile:
             for line in infile:
                 bert_data = json.loads(line)
                 for t in bert_data["features"]:
                     # Only extract the [CLS] vector used for classification
                     if t["token"] == "[CLS]":
                         # We only use the representation at the final layer of the network
                         bert_vectors.append(t["layers"][0]["values"])
                         break

In [18]: len(bert_vectors)

Out[18]: 2000

In [45]: X_test = np.array(bert_vectors)
         y_test = test_df["native_language"].values

In [46]: # TEst score
         lr_model.score(X_test, y_test)

Out[46]: 0.466

In [47]: y_test_pred = lr_model.predict(X)
```



```
In [52]: eval_df = pd.read_csv(os.path.join(ORIGINAL_DATA_DIR, "lang_id_eval.csv"))

In [53]: eval_df.shape

Out[53]: (2000, 2)

In [54]: bert_vectors = []
         with open(os.path.join(BERT_FEATURE_DIR, "eval.jsonlines"), "rt") as infile:
             for line in infile:
                 bert_data = json.loads(line)
                 for t in bert_data["features"]:
                     # Only extract the [CLS] vector used for classification
                     if t["token"] == "[CLS]":
                         # We only use the representation at the final layer of the network
                         bert_vectors.append(t["layers"][0]["values"])
                         break

In [55]: len(bert_vectors)

Out[55]: 2000

In [57]: X_eval = np.array(bert_vectors)
         y_eval = test_df["native_language"].values

In [58]: lr_model.score(X_eval, y_eval)

Out[58]: 0.4855
```



```
In [59]: from sklearn.metrics import confusion_matrix
         cm = confusion_matrix(y_test, y_test_pred)
         print(cm)

         [[ 99  11   9   7  12  14  10  16   9  13]
          [ 11  68  14  12  46  12  10   4   9  14]
          [  8  13  99  23  11  17  10   5   5   9]
          [  7  17  23  91  13   8  12   9  12   8]
          [ 12  36  19  14  63   9  10  14   6  17]
          [ 11  14   8   3   5 102  29  10   9   9]
          [ 10   8  14   5   6  23 114  14   0   6]
          [ 17   2  11   5  14  17  14 104   5  11]
          [ 12  10   7  15   9   2   1  13 120  11]
          [ 12  29  11  13  19  13  12  11   8  72]]

In [62]: from sklearn.metrics import precision_score
         precision_score(y_test, y_test_pred, average = None)

Out[62]: array([0.49748744, 0.32692308, 0.46046512, 0.48404255, 0.31818182,
                0.47004608, 0.51351351, 0.52      , 0.6557377 , 0.42352941])

In [63]: from sklearn.metrics import recall_score
         recall_score(y_test, y_test_pred, average = None)

Out[63]: array([0.495, 0.34 , 0.495, 0.455, 0.315, 0.51 , 0.57 , 0.52 , 0.6  ,
                0.36 ])
```

Difficulties:

There were certain difficulties encountered during running the assignments such as

- Installation of tensorflow was time consuming
- Not all versions of tensorflow was compatible
- Running the shell file was time consuming
- All the input and output files were huge thereby leading to loss of memory and time management.

Conclusion:

Despite using a highly efficient pre trained model like BERT and the logistic regression module, the accuracy of the model is low. This can be overcome by training the BERT model with more of training sets.

Result:

Thus the given assignment was executed as instructed and the language processing model was successfully built and the evaluation metrics were found.