

## **Assignment – 4**

**CS 585**

**Vatsal Ketan Patel**

**A20450861**

### **Task: Using pre-trained BERT vectors for text classification**

#### **Introduction**

BERT (Bidirectional Encoder Representations from Transformers) is developed by researchers at Google AI Language. It has various technical applications and one of the key areas where it is used is in applying the bidirectional training of Transformer, a popular attention model, to language modelling. In the paper, the researchers detail a novel technique named Masked LM (MLM) which allows bidirectional training in models in which it was previously impossible.

#### **Working of BERT**

BERT makes use of Transformer, an attention mechanism that learns contextual relations between words (or sub-words) in a text. As opposed to directional models, which read the text input sequentially (left-to-right or right-to-left), the Transformer encoder reads the entire sequence of words at once. Therefore, it is considered bidirectional. This characteristic allows the model to learn the context of a word based on all of its surroundings (left and right of the word).

#### **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**

**Precision** =  $\text{True Positives} / (\text{True Positives} + \text{False Positives})$

**Recall** =  $\text{True Positives} / (\text{True Positives} + \text{False Negatives})$

**F-measure** =  $(2 * \text{Precision} * \text{Recall}) / (\text{Precision} + \text{Recall})$

## Use of BERT for the given task

Since BERT is quite resource intensive, we will only be using it for feature generation, rather than retraining the model itself. The task at hand is to build a text classification model to address the task of **native language identification**. The dataset contains data of English texts authored by writers whose native language is some other language than English, and we will build a model to predict their native language based on the text.

Steps followed for the task:

**Step 1:** Cloning the BERT repository to local directory and referring as BERT\_BASE\_DIR.

```
git clone https://github.com/google-research/bert.git
```

**Step 2:** Downloading pre-trained BERT model and referring the uncompressed file as BERT\_DATA\_DIR.

[https://storage.googleapis.com/bert\\_models/2018\\_10\\_18/uncased\\_L-12\\_H-768\\_A-12.zip](https://storage.googleapis.com/bert_models/2018_10_18/uncased_L-12_H-768_A-12.zip)

**Step 3:** Since the BERT extract\_features.py script only accepts .txt files. I converted the CSV files to txt files such that each input file contains each text on a single line, with no other information i.e. dropping the native language column as well as the header line named "text".

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Thu Apr 21 15:54:55 2020
4
5 @author: vatsal
6 """
7
8 import pandas as pd
9 import os
10
11 df_train = pd.read_csv('lang_id_train.csv')
12 file= "train.txt"
13 path= r"C:\Users\vatsa\OneDrive\Desktop\hw4-handout\handout\bert_input_data"
14 for i in df_train["text"]:
15     with open(os.path.join(path, file), 'a',encoding='utf-8') as fp:
16         fp.write("%s\n" %i)
17
18 df_test = pd.read_csv('lang_id_test.csv')
19 file= "test.txt"
20 path= r"C:\Users\vatsa\OneDrive\Desktop\hw4-handout\handout\bert_input_data"
21 for i in df_test["text"]:
22     with open(os.path.join(path, file), 'a',encoding='utf-8') as fp:
23         fp.write("%s\n" %i)
24
25 df_eval = pd.read_csv('lang_id_eval.csv')
26 file= "eval.txt"
27 path= r"C:\Users\vatsa\OneDrive\Desktop\hw4-handout\handout\bert_input_data"
28 for i in df_eval["text"]:
29     with open(os.path.join(path, file), 'a',encoding='utf-8') as fp:
30         fp.write("%s\n" %i)
```

**Step 4:** Downgrading the TensorFlow to version 1.14 since BERT is not compatible with TensorFlow version 2.0. And then, use the text files generated from the above step and run them through BERT using `run_bert_fv.sh` script by giving the making changes to some of the directories in order to generate feature vectors to represent each document.

Command for install git bash on conda: `conda install -c conda-forge git-bash`

Command for downgrading TensorFlow version 1.14: `conda install -c conda-forge tensorflow=1.14`

Adjustments to run `bert_fv.sh` script: Assigning the right paths to BERT Base and Data Directories.

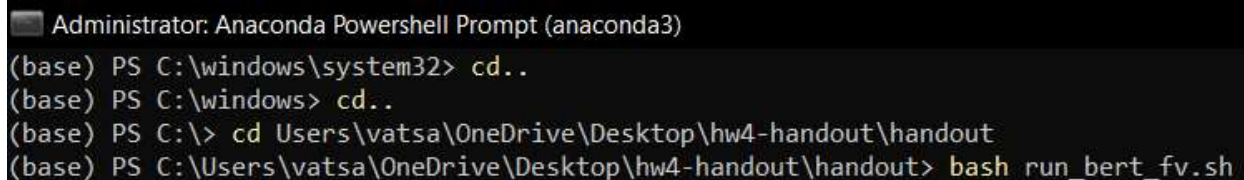
```
#!/bin/bash

export BERT_BASE_DIR=C:/Users/vatsa/OneDrive/Desktop/hw4-handout/bert
export BERT_DATA_DIR=C:/Users/vatsa/OneDrive/Desktop/hw4-handout/uncased_L-12_H-768_A-12

INPUT_DIR=bert_input_data
OUTPUT_DIR=bert_output_data
rm -rf $OUTPUT_DIR
mkdir -p $OUTPUT_DIR

for i in train eval test
do
    python $BERT_BASE_DIR/extract_features.py \
        --vocab_file=$BERT_DATA_DIR/vocab.txt \
        --bert_config_file=$BERT_DATA_DIR/bert_config.json \
        --init_checkpoint=$BERT_DATA_DIR/bert_model.ckpt \
        --max_seq_length=128 \
        --batch_size=8 \
        --layers=-1 \
        --input_file=$INPUT_DIR/$i.txt \
        --output_file=$OUTPUT_DIR/$i.jsonlines
done
```

Command for executing the shell script: `bash run_bert_fv.sh`



```
Administrator: Anaconda Powershell Prompt (anaconda3)
(base) PS C:\windows\system32> cd..
(base) PS C:\windows> cd..
(base) PS C:\> cd Users\vatsa\OneDrive\Desktop\hw4-handout\handout
(base) PS C:\Users\vatsa\OneDrive\Desktop\hw4-handout\handout> bash run_bert_fv.sh
```

**Step 5:** Training the text categorization model using features derived from BERT in order to predict native\_language attribute. It is done by getting the `bert_vectors` from `train.jsonlines` and training it with logistic regression model using `fit` method.

**Step 6:** Getting the test\_vectors from test.jsonlines and testing using the test set by calling the predict method.

## **Results:**

Overall Accuracy for test case:

```
test_labels = lr_model.predict(x_test)
accuracy = lr_model.score(x_test, y_test)
print("Accuracy% : " + str(accuracy*100))
```

Accuracy% : 45.6

Metrics by Class:

- 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)
- Fscore  
from sklearn.metrics import precision\_recall\_fscore\_support  
precision\_recall\_fscore\_support(y\_test, y\_test\_pred, average = None)

The method I used for getting the below results is:

```
precision,recall,fscore,support =  
sklearn.metrics.precision_recall_fscore_support(test_df['native_language'],  
test_df['predicted_language'])
```

Class	Misclassification rate	Precision	Recall	Fscore
Arabic	0.515	0.480	0.485	0.482
Cantonese	0.71	0.308	0.29	0.298
Japanese	0.475	0.472	0.525	0.497
Korean	0.580	0.461	0.42	0.439
Mandarin	0.685	0.308	0.315	0.311
Polish	0.52	0.475	0.48	0.477
Russian	0.430	0.508	0.57	0.537

Spanish	0.5	0.515	0.5	0.507
Thai	0.410	0.59	0.59	0.59
Vietnamese	0.403	0.423	0.385	0.403

Frequency of errors and misclassification rate between classes:

- 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 thus helps in evaluating the frequency of errors between different classes.

```
#Confusion matrix determines the frequency of missclassifications between different classes
confusion_matrix = sklearn.metrics.confusion_matrix(test_df['native_language'], test_df['predicted_language'])
print("Confusion Matrix: \n", confusion_matrix)
```

**Confusion Matrix:**

```
[[ 97   8   8  12  12  17  13  16   5  12]
 [ 10  58  14  15  48  10   8   6  14  17]
 [ 15  12 105  16   9  13   5   7  10   8]
 [  4  17  25  84  16   6  13   7  16  12]
 [ 13  40  14  14  63   7  10  12  10  17]
 [ 11   9   6   6   7  96  33  15   8   9]
 [  8  10  13   6   8  21 114  11   1   8]
 [ 18   7  14   5  12  11  18 100   6   9]
 [ 16  10   9  10   8   4   2  10 118  13]
 [ 10  17  14  14  21  17   8  10  12  77]]
```

```
misclassification_rate = confusion_matrix/200
print("Misclassifications rate between each pair of classes: \n", misclassification_rate)
```

Misclassifications rate between each pair of classes:

```
[[0.485 0.04  0.04  0.06  0.06  0.085 0.065 0.08  0.025 0.06 ]
[0.05  0.29  0.07  0.075 0.24  0.05  0.04  0.03  0.07  0.085]
[0.075 0.06  0.525 0.08  0.045 0.065 0.025 0.035 0.05  0.04 ]
[0.02  0.085 0.125 0.42  0.08  0.03  0.065 0.035 0.08  0.06 ]
[0.065 0.2  0.07  0.07  0.315 0.035 0.05  0.06  0.05  0.085]
[0.055 0.045 0.03  0.03  0.035 0.48  0.165 0.075 0.04  0.045]
[0.04  0.05  0.065 0.03  0.04  0.105 0.57  0.055 0.005 0.04 ]
[0.09  0.035 0.07  0.025 0.06  0.055 0.09  0.5  0.03  0.045]
[0.08  0.05  0.045 0.05  0.04  0.02  0.01  0.05  0.59  0.065]
[0.05  0.085 0.07  0.07  0.105 0.085 0.04  0.05  0.06  0.385]]
```

## **Difficulties**

- Tensorflow 2.0 was not compatible and thus downgrading to compatible version i.e. Tensorflow 1.14
- Running the shell script was time consuming.

## **Conclusion**

Despite using a highly efficient pre trained model like BERT and the logistic regression module, the accuracy of the model is low because it was only used for feature generation rather than retraining the BERT model, if we retrain the train set using BERT model it would increase the accuracy of the model.