Assignment 4
Movies Reviews Classification Using BERT

1 Objectives

1. Applying state of the art language model BERT to solve NLP classification problem.
2. Using the whole knowledge you got throughout the course to tackle a real life AI problem.

2 Problem Statement

IMDB is the most globally famous movie reviews website where you can publish a review for any film you watched. Classifying the positive reviews and the negative ones can be useful for several purposes such as giving an overall rating for the film or making statistical analysis about the preferences of people from different countries, age levels, etc... So IMDB dataset is released which composed of 50k reviews labeled as positive or negative to enable training movie reviews classifiers. Moreover, NLP tasks are currently solved based on pretrained language models such as BERT. These models provide a deep understanding of both semantic and contextual aspects of language words, sentences or even large paragraphs due to their training on huge corpus for very long time. In this assignment you will download the IMDB dataset from kaggle using this Link. Then, you will train BERT based classifier for movie reviews.

3 Lab session

1. **Data Split**
   Split your dataset randomly so that the training set would form 70% of the dataset, the validation set would form 10% and the testing set would form 20% of it. You should keep all the splits balanced.

2. **Text Pre-processing**
   Text pre-processing is essential for NLP tasks. So, you will apply the following steps on our data before used for classification:

   (a) Remove punctuation.
   (b) Remove stop words.
   (c) Lowercase all characters.
   (d) Lemmatization of words.

   You can use any NLP library such as nltk to pre-process the data.

3. **Classification using BERT**
   You need to build a classifier model based on BERT. You can use transformers library supplied by hugging face to get a pretrained and ready version of BERT model. It will also help you to tokenize the input sentence in the BERT required form and to pad the short sentences or trim the long ones. We will use the CLS token embedding outputs of BERT as input to the hidden dense classification layers we need to add after BERT. This embedding is of size 768.
   You need to add 4 hidden layers of 512, 256, 128, 64 units respectively before the output layer. You will use binary cross entropy loss and adam optimizer.

4. **Validation and Hyperparameter Tuning**
   Use the validation split to evaluate the model performance after each training epoch then save the model checkpoint to choose the one with the best performance as the final model. You can use dropout between dense layers to avoid overfitting if it arises.
   Also, you need to tune the learning rate hyperparameter of Adam optimizer using the performance on the validation set.

5. **Checking Pre-processing Importance**
   BERT is assumed to capture the semantic and contextual aspects of the language. So, sometimes it is better to input the text to it without pre-processing. To check the pre-processing importance on our task we will train the model twice one using the pre-processed version of data and the other using the original version then test both models using the testing set and compare between the results.
   Note that you need to repeat the validation and hyperparameter tuning steps in both cases. Also, note that the model trained on pre-processed data must be validated and tested using pre-processed data and vice versa.

6. **Report Requirements**

   - You should report graphs representing the change of training and validation accuracies with the number of training epochs for your experiments.
   - You should report a graph comparing between the best validation accuracies for the different values of learning rate.
   - You should report the model accuracy, precision, recall, specificity and F-score as well as the resultant confusion matrix using the testing set for the best model with pre-processing and without.
   - Your comments on all results and comparisons.

7. **Bonus**
   Finetune the number of hidden dense layers we need to add for classification and the number of units in each layer using the validation set. Then test the best model using the testing set and report all the above required metrics.

8. **Notes**

   - You should write your code in python.
   - You will need nltk, transformers and pytorch libraries.
   - You should work in groups of 3 or 4.

9. **References**

   - Classification using BERT
   - BERT technical details

Good Luck