Assignment 2: Implement your own BERT
CS678 Advanced Natural Processing Fall 2022

**Introduction:** To Implement some core components of the BERT model to gain a better understanding of its architecture. This implementation is based on PyTorch and generic Python tools.

**BERT Model** : BERT stands for "Bidirectional Encoder Representation with Transformers" that extracts patterns or representations from the data or word embeddings by passing it through an encoder. The encoder itself is a transformer architecture that is stacked together. It is a bidirectional transformer which means that during training it considers the context from both left and right of the vocabulary to extract patterns or representations.

**Execution process:** Due to time constraints on CPU, the program is executed in Google Collab with runtype as GPU**.** Installed conda environment and ran the setup.sh file that contains all the CUDA developer kit for enabling GPU in pytorch and other respective packages and dependencies.

**1.The final dev and test set performance of the Pre-trained classifier on SST dataset:**

```
!python3 classifier.py --option pretrain  --epochs 10 --lr 1e-3 --train
sst-train.txt --dev sst-dev.txt --test sst-test.txt --use_gpu
```

```
args: {'train': 'sst-train.txt', 'dev': 'sst-dev.txt', 'test': 'sst-test.txt', 'seed'
load 8544 data from sst-train.txt
load 1101 data from sst-dev.txt
save the model to pretrain-10-0.001.pt
epoch 0: train loss :: -0.286, train acc :: 0.278, dev acc :: 0.261
save the model to pretrain-10-0.001.pt
epoch 1: train loss :: -0.298, train acc :: 0.320, dev acc :: 0.312
save the model to pretrain-10-0.001.pt
epoch 2: train loss :: -0.310, train acc :: 0.338, dev acc :: 0.327
save the model to pretrain-10-0.001.pt
epoch 3: train loss :: -0.326, train acc :: 0.365, dev acc :: 0.370
epoch 4: train loss :: -0.332, train acc :: 0.315, dev acc :: 0.302
epoch 5: train loss :: -0.337, train acc :: 0.362, dev acc :: 0.348
epoch 6: train loss :: -0.339, train acc :: 0.340, dev acc :: 0.324
epoch 7: train loss :: -0.346, train acc :: 0.365, dev acc :: 0.351
epoch 8: train loss :: -0.345, train acc :: 0.371, dev acc :: 0.363
save the model to pretrain-10-0.001.pt
epoch 9: train loss :: -0.347, train acc :: 0.364, dev acc :: 0.373
load model from pretrain-10-0.001.pt
load 1101 data from sst-dev.txt
load 2210 data from sst-test.txt
dev acc :: 0.373
test acc :: 0.370
```

**2.The final dev and test set performance of the fine-tuned classifier on SST dataset**

```
!python3 classifier.py --option finetune  --epochs 10 --lr 1e-5 --train
sst-train.txt --dev sst-dev.txt --test sst-test.txt --use_gpu
```

```
args: {'train': 'sst-train.txt', 'dev': 'sst-dev.txt', 'test': 'sst-test.txt
load 8544 data from sst-train.txt
load 1101 data from sst-dev.txt
save the model to finetune-10-1e-05.pt
epoch 0: train loss :: -0.396, train acc :: 0.471, dev acc :: 0.442
epoch 1: train loss :: -0.452, train acc :: 0.435, dev acc :: 0.410
epoch 2: train loss :: -0.456, train acc :: 0.476, dev acc :: 0.435
save the model to finetune-10-1e-05.pt
epoch 3: train loss :: -0.469, train acc :: 0.506, dev acc :: 0.458
save the model to finetune-10-1e-05.pt
epoch 4: train loss :: -0.482, train acc :: 0.501, dev acc :: 0.467
epoch 5: train loss :: -0.486, train acc :: 0.501, dev acc :: 0.458
epoch 6: train loss :: -0.483, train acc :: 0.496, dev acc :: 0.450
save the model to finetune-10-1e-05.pt
epoch 7: train loss :: -0.492, train acc :: 0.519, dev acc :: 0.470
epoch 8: train loss :: -0.499, train acc :: 0.501, dev acc :: 0.455
epoch 9: train loss :: -0.503, train acc :: 0.517, dev acc :: 0.445
load model from finetune-10-1e-05.pt
load 1101 data from sst-dev.txt
load 2210 data from sst-test.txt
dev acc :: 0.470
test acc :: 0.448
```

**3.The final dev set performance  fine-tuned classifier on the CFIMDB dataset**

```
!python3 classifier.py --option finetune  --epochs 10 --lr 1e-5 --train
cfimdb-train.txt --dev cfimdb-dev.txt --test cfimdb-test.txt --use_gpu
```

```
args: {'train': 'cfimdb-train.txt', 'dev': 'cfimdb-dev.txt', 'test': 'cfimdb-test.txt', 'seed':
load 1707 data from cfimdb-train.txt
load 245 data from cfimdb-dev.txt
save the model to finetune-10-1e-05.pt
epoch 0: train loss :: -0.748, train acc :: 0.982, dev acc :: 0.955
epoch 1: train loss :: -0.958, train acc :: 0.973, dev acc :: 0.951
epoch 2: train loss :: -0.968, train acc :: 0.979, dev acc :: 0.955
save the model to finetune-10-1e-05.pt
epoch 3: train loss :: -0.973, train acc :: 0.991, dev acc :: 0.967
epoch 4: train loss :: -0.967, train acc :: 0.988, dev acc :: 0.955
epoch 5: train loss :: -0.978, train acc :: 0.988, dev acc :: 0.963
epoch 6: train loss :: -0.981, train acc :: 0.987, dev acc :: 0.947
epoch 7: train loss :: -0.981, train acc :: 0.991, dev acc :: 0.951
epoch 8: train loss :: -0.969, train acc :: 0.990, dev acc :: 0.963
epoch 9: train loss :: -0.985, train acc :: 0.991, dev acc :: 0.959
load model from finetune-10-1e-05.pt
load 245 data from cfimdb-dev.txt
load 488 data from cfimdb-test.txt
dev acc :: 0.967
test acc :: 0.512
```

**4.Accuracy analysis for SST and CF-IMDB:**

| | SST | SST | CFIMDB |
|---|---|---|---|
| Pretraining | Dev - 0.373 | Test - 0.370 | —---------- |
| Fine-tuning | Dev - 0.47 | Test - 0.448 | Dev - 0.967<br>Test - 0.512 |

During pre-training, the model is trained on unlabeled data over different pre-training tasks. For fine-tuning, the BERT model is first initialized with the pre-trained parameters, and all of the parameters are fine-tuned using labeled data from the downstream tasks. Each downstream task has separate fine-tuned models, even though they are initialized with the same pre-trained parameters.

Here, when the pre-training is performed SST-dataset ,the model achieved same 0.37 accuracy in both dev and test datasets, after fine-tuning, the model performance increased around 0.47 accuracy. For the CIMDB , the fine-tuned model has shown highest accuracy of 0.967 for Dev while 0.512 for Test. This states that if the model is pre-trained and fine-tuned with same dataset, then that model performs well even if that fine-tuned model is applied for different dataset of same domain without pre-training.

**References:**


https://nlp.seas.harvard.edu/2018/04/03/attention.html

https://jalammar.github.io/illustrated-transformer/

https://neptune.ai/blog/how-to-code-bert-using-pytorch-tutorial