HW2
Section: 001
Adel Alkhamisy

All the ToDo lists were done according to the requirements of Homework 2. Google Colaboratory Pro with access to GPU was used to implement Homework 2.

Part 1: bert.py

1. The attention is calculated according to the following formula:

$$Z = Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad [1]$$

2. The add_norm layer is implemented by making a residual connection to skip the previous layer and then followed by normalization to stabilize the neural network [2].
3. Multi-head attention is calculated by the following code:

```
z = self.attention(key_layer, query_layer, value_layer, attention_mask)
```

4. The embed for each token is implemented by the following code:

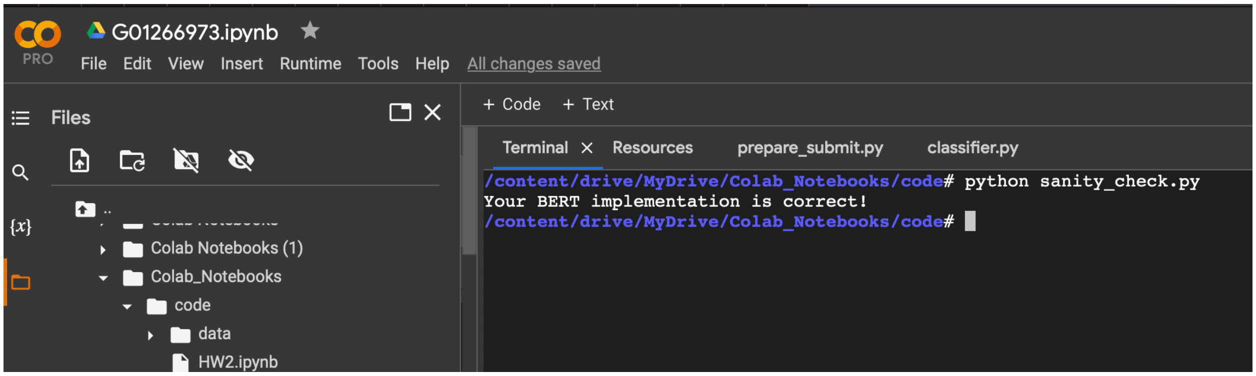
```
        embedding_output = self.embed(input_ids=input_ids)
```

Part 2: classifier.py

1. To obtain the pooled output [CLS] token and then prediction, a dense layer and a dropout layer is created to get the contextualized embedding, then using the logsoft of the contextualized embedding, I get the prediction [3].

Part 3: experiment

1. Pass Sanity check:


G01266973.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Files
+ Code + Text
Terminal Resources prepare_submit.py classifier.py
Colab Notebooks (1)
Colab_Notebooks
code
data
HW2.ipynb

```
/content/drive/MyDrive/Colab_Notebooks/code# python sanity_check.py
Your BERT implementation is correct!
/content/drive/MyDrive/Colab_Notebooks/code#
```

2. Pretrain and finetune on SST and CFIMDB

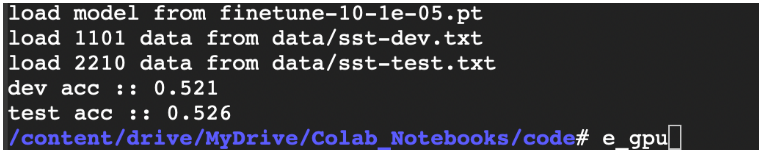
Table 1

Run	1	2	3	4	5	6	7
epochs	5	10	15	10	15	10	15
option	pretrain	pretrain	pretrain	pretrain	pretrain	pretrain	Pretrain
batch_size	5	8	20	8	8	12	8
hidden_dropout_prob	0.2	0.3	0.3	0.3	0.3	0.3	0.3
Learing rate	0.00001	0.00001	0.00001	0.001	0.0001	0.001	0.001
SST Pretrain (Dev acc/Test acc)	0.303/ 0.297	0.311/ 0.295	0.305/ 0.294	0.406/ 0.419	0.379/ 0.383	0.402/ 0.414	0.406/ 0.419

Best scores are highlighted with yellow.

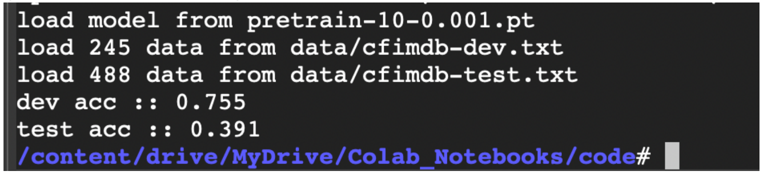
From Table 1, the best performance was obtained with a learning rate of 0.001. It can be said that picking a small rate, which is the optimizer's step size, is preferable to a large learning rate in general. However, too small a learning rate such as 0.00001 prevents the optimizer from converging; hence reducing the accuracy. In run 4 and 6, all the hyperparameters were the same except batch size which was 8 in column 4 and 12 in column 6. Observe that the accuracy decreased from 0.406/0.419 to 0.402/0.414 which indicates that the optimal batch size is 8. The pre-trained model was used to finetune the SST. Also, the Bert Model used a lot of GPU RAM and power in general (See Picture 1 in page 3).

SST Finetuning
(Dev acc/Test acc)



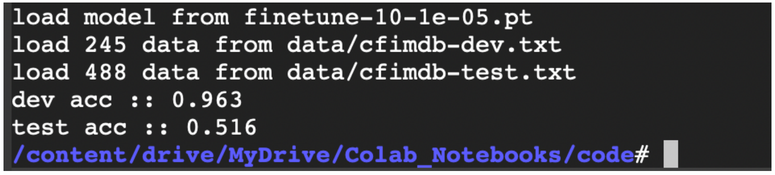
```
load model from finetune-10-1e-05.pt
load 1101 data from data/sst-dev.txt
load 2210 data from data/sst-test.txt
dev acc :: 0.521
test acc :: 0.526
/content/drive/MyDrive/Colab_Notebooks/code# e_gpu
```

CFIMDB Pretrain
(Dev acc/Test acc)



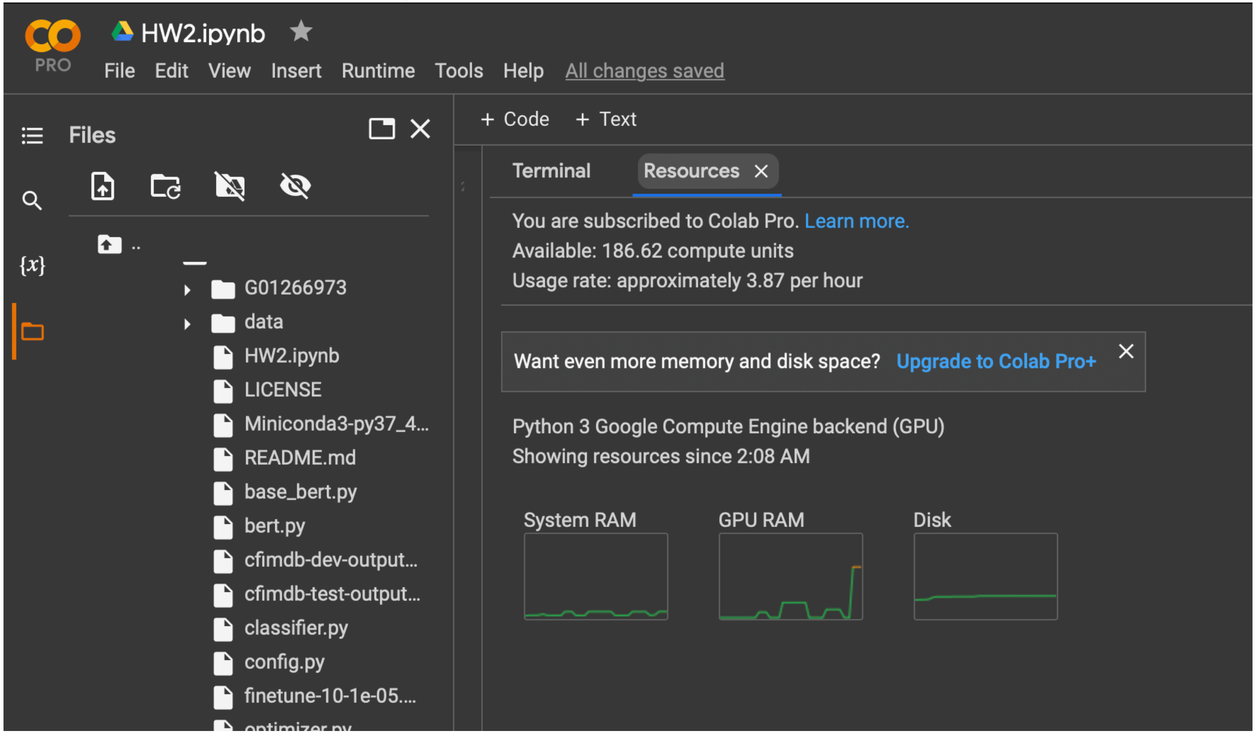
```
load model from pretrain-10-0.001.pt
load 245 data from data/cfimdb-dev.txt
load 488 data from data/cfimdb-test.txt
dev acc :: 0.755
test acc :: 0.391
/content/drive/MyDrive/Colab_Notebooks/code#
```

CFIMDB Finetuning
(Dev acc/Test acc)



```
load model from finetune-10-1e-05.pt
load 245 data from data/cfimdb-dev.txt
load 488 data from data/cfimdb-test.txt
dev acc :: 0.963
test acc :: 0.516
/content/drive/MyDrive/Colab_Notebooks/code#
```

Picture 1: show GPU RAM usage



You are subscribed to Colab Pro. Learn more.
Available: 186.62 compute units
Usage rate: approximately 3.87 per hour

Want even more memory and disk space? Upgrade to Colab Pro+

Python 3 Google Compute Engine backend (GPU)
Showing resources since 2:08 AM

References:

[1] https://arxiv.org/pdf/1706.03762.pdf
[2] https://towardsdatascience.com/illustrated-guide-to-transformers-step-by-step-explanation-f74876522bc0
[3] https://jalammar.github.io/illustrated-transformer/
[4] https://uvadlc-notebooks.readthedocs.io/en/latest/tutorial_notebooks/tutorial6/Transformers_and_MHAttention.html
[5]https://huggingface.co/transformers/v3.1.0/_modules/transformers/modeling_distilbert.html