Indian Institute of Technology Gandhinagar

Project A
Assignment-3 Report
CS 613: NLP

Group Members

Dhakad Bhagat Singh: 20110055

Tejas Parmar: 20110125

Prey Patel: 20110132

Prakram Rathore: 20110141

Rahul Kumar: 20110153

Sandeep Patel: 20110183

Venkata Sriman Narayana Malli: 20110224

Zeeshan Snehil Bhagat: 20110242

Solutions:

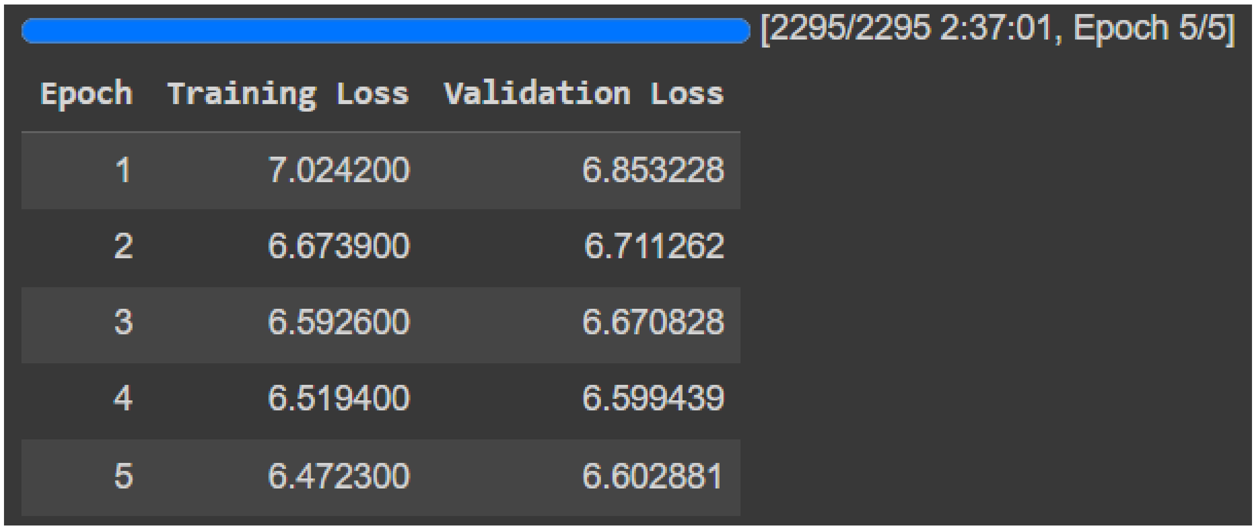
Q.1 BERT-BASE-UNCASED was pre-trained on wikipedia and Bookcorpus. Instead of using it by randomizing weights, we used a fresh model from BertMaskLM using config (vocab_size=30_522, max_position_embeddings=256). The parameters, vocab size, and positional embedding lengths are all similar.

Q.2 Calculate the **number of parameters** of the selected model from the code. Do your calculated parameters match with the parameters reported in the respective paper? **[05 pts]**
Ans: Parameters of:

- Bert Base, as per the paper, parameters = 110M;
- Bert Model as calculated in Sir's Transformer Parameters Calculation = 108,432,384
- BertForMaskedLM(BertConfig(vocab_size=30_522, max_position_embeddings=256)) = 109317690

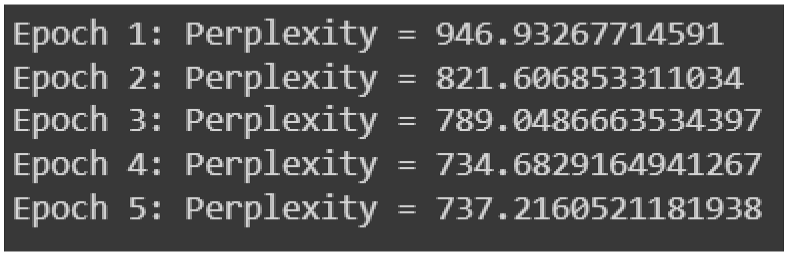
Q.3 Pretrain the selected model on the train split of 'wikitext-2-raw-v1'. **[10 pts]** For 5 epochs. Use the hyperparameters as per your choice.
Ans: Shared collab link in readme file

Q.4 Compute and report the Perplexity scores using the inbuilt function on the test split of 'wikitext-2-raw-v1' for each epoch. Do scores decrease after every epoch? Why and why not? **[10 pts]**
Ans:



[2295/2295 2:37:01, Epoch 5/5]

Epoch	Training Loss	Validation Loss
1	7.024200	6.853228
2	6.673900	6.711262
3	6.592600	6.670828
4	6.519400	6.599439
5	6.472300	6.602881



```
Epoch 1: Perplexity = 946.93267714591
Epoch 2: Perplexity = 821.606853311034
Epoch 3: Perplexity = 789.0486663534397
Epoch 4: Perplexity = 734.6829164941267
Epoch 5: Perplexity = 737.2160521181938
```

The scores decrease after every epoch, though not so much (from the 900s to 700s, where it could be somewhat saturating). Bert being an encoder-based model, it learns to predict masked tokens(Masked Language Modelling) by taking context bidirectionally. As a result, it might also learn next-word prediction by only looking at the previous tokens. This results in perplexity decrement. Later, it could be saturating around 700s due to diminishing returns for a certain configuration (vocab size, length of positional embeddings, and so on) and data objective (masked language modeling).

Q.5 Push the pre-trained model to 🤗. **[2.5 pts]**

Model: "rishirathore/RishiBERTWiki"
https://huggingface.co/rishirathore/RishiBERTWiki

Q.6 Fine-tune the final pre-trained model on the following tasks:

a. Classification: SST-2 **[10 pts]**

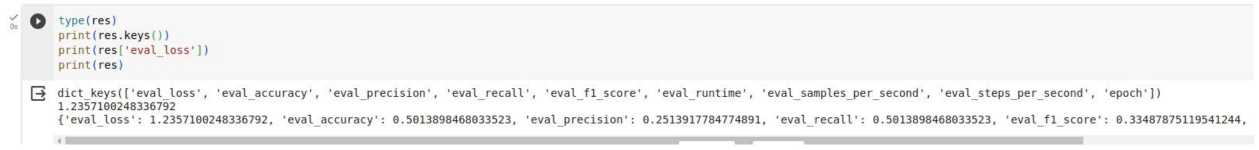
Model: "rishirathore/RishiBERTWikiClassification"

b. Question-Answering: SQuAD **[10 pts]**

Model: "rishirathore/RishiBERTWikiQA"

Q.7 Calculate the scores for the following metrics on the test splits. Note that metrics depend on the selected task:

a. Classification: Accuracy, Precision, Recall, F1 **[15 pts]**

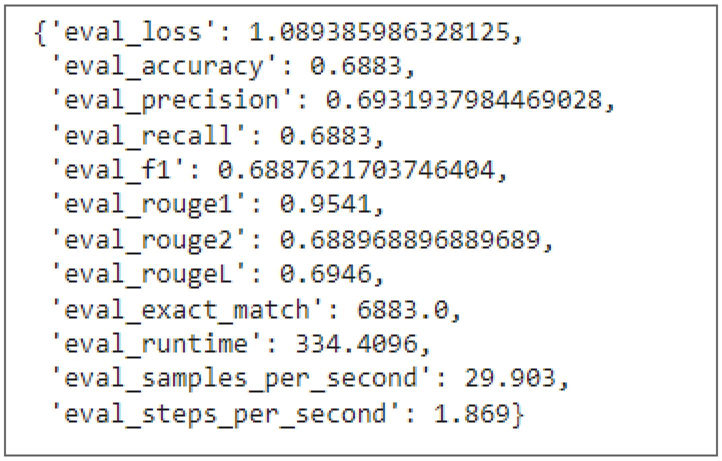


```
type(res)
print(res.keys())
print(res['eval_loss'])
print(res)
```

```
dict_keys(['eval_loss', 'eval_accuracy', 'eval_precision', 'eval_recall', 'eval_f1_score', 'eval_runtime', 'eval_samples_per_second', 'eval_steps_per_second', 'epoch'])
1.2357100248336792
{'eval_loss': 1.2357100248336792, 'eval_accuracy': 0.5013898468033523, 'eval_precision': 0.2513917784774891, 'eval_recall': 0.5013898468033523, 'eval_f1_score': 0.33487875119541244,
```

Accuracy	**0.5013898468033523**
Precision	**0.2513917784774891**
Recall	**0.5013898468033523**
F1 Score	**0.33487875119541244**

b. Question-Answering: squad_v2, F1, METEOR, BLEU, ROUGE, exact-match [Read this!, and this too!] **[15 pts]**



```
{'eval_loss': 1.089385986328125,
 'eval_accuracy': 0.6883,
 'eval_precision': 0.6931937984469028,
 'eval_recall': 0.6883,
 'eval_f1': 0.6887621703746404,
 'eval_rouge1': 0.9541,
 'eval_rouge2': 0.688968896889689,
 'eval_rougeL': 0.6946,
 'eval_exact_match': 6883.0,
 'eval_runtime': 334.4096,
 'eval_samples_per_second': 29.903,
 'eval_steps_per_second': 1.869}
```

Q.8 Calculate the number of parameters in the model after fine-tuning. Is it the same as the pre-trained model? **[05 pts]**
Ans:
Number of parameters of:
Pre Trained Model: (rishirathore/RishiBERTWiki) = 109285632
Fine Tuned Model for Question Answering: (rishirathore/RishiBERTWikiQA) = 109482240
Fine Tuned Model for Classification: (rishirathore/RishiBERTWikiClassification) = 109285632

Q.9 Push the fine-tuned model to 🤗. **[2.5 pts]**

Question Answering: https://huggingface.co/rishirathore/RishiBERTWikiQA
Classification: https://huggingface.co/rishirathore/RishiBERTWikiClassification

Q.10 Write appropriate comments and rationale behind

a. Poor/good performance. **[2.5 pts]**
b. Understanding from the number of parameters between pretraining and fine-tuning of the model. **[2.5 pts]**

Part-a:

The observed performance gap between our BERT model fine-tuned on smaller datasets and models available on Hugging Face pre-trained on extensive corpora can be attributed to several factors. Hugging Face models undergo pre-training on vast and diverse datasets encompassing various linguistic patterns and contexts. These models acquire rich, generalised language representations during pre-training, which might only partially transfer to more minor, domain-specific datasets used for fine-tuning.

When fine-tuning the BERT model on limited data, it might need help to capture the nuances and complexities specific to the smaller dataset, leading to suboptimal performance. In contrast, models pre-trained on extensive corpora possess broader contextual understanding due to exposure to diverse linguistic styles and topics, facilitating better generalisation across various tasks and domains.

Additionally, the discrepancy in performance might arise from the mismatch between the distributions of the pre-training data and the fine-tuning dataset. The Hugging Face models' pre-training data could differ significantly from the smaller fine-tuning dataset, causing a domain shift that hampers the model's adaptability to the new domain, adversely impacting its performance.

Furthermore, fine-tuning on a smaller dataset might need to provide more diverse examples for the model to grasp nuanced patterns effectively, affecting its ability to generalise well. Addressing this performance discrepancy may necessitate domain-specific pre-training strategies or transfer learning techniques to bridge the gap between pre-training on large corpora and fine-tuning on smaller, task-specific datasets, enhancing the model's performance in specific contexts.

Part-b:

Since some extra layers can be added to the pre-trained BERT model in the process of fine-tuning for the given tasks. Thus, the number of parameters in the fine-tuned model can increase slightly as per task specifications.

In our classification task during fine-tuning, we haven't added any extra layers for assigning probabilities to classes. Thus, the model parameter count remains unchanged. However, in the answering model, the parameter count is higher than that of the pre-trained model as some new layers have been added on top of the BERT base model.

As we know, the BERT base model has a predetermined structure comprising several layers, attention heads, and hidden units. During pre-training, the model learns general language representations by adjusting the weights associated with these parameters using unsupervised tasks. Most of these parameters are retained in fine-tuning, and only a small subset is further fine-tuned, or new parameters can be added(as per task-specific) to adapt the learned representations to the specific downstream task. This maintains the model's capacity for language understanding while allowing it to specialise in various NLP tasks by altering(or not) its parameter count.

Contribution

Task	Contributors
Pretraining BERT Model	Prakram Rathore, Zeeshan Snehil Bhagat, Tejas Parmar
Compute Perplexities	Tejas Parmar, Venkata Sriman, Zeeshan Snehil Bhagat
Fine-Tune Pre-trained Model: Classification	Sandeep Patel and Prey Patel
Fine-Tune Pre-trained Model: Question - Answering	Rahul Kumar, Dhakad Bhagat Singh
Calculate the scores for the following metrics on the test splits: Classification	Sandeep Patel, Prey Patel
Calculate the scores for the following metrics on the test splits Question - Answering	Rahul Kumar, Dhakad Bhagat Singh
Calculate the number of parameters in the model after fine-tuning.	Venkata Sriman
Write appropriate comments and rationale behind	Dhakad Bhagat Singh, Zeeshan Snehil Bhagat, Venkata Sriman
Documentation	Zeeshan Snehil Bhagat, Prey Patel, Rahul Kumar
Push the pre-trained/fine-tuned models to 🤗. Add a final section where you pull the pre-train/fine-tuned model (from hugging face) and compute the metrics over the public models.	Prakram Rathore

References:

1. https://huggingface.co/bert-base-uncased
2. https://huggingface.co/datasets/wikitext/viewer/wikitext-2-raw-v1
3. https://www.kaggle.com/datasets/atulanandjha/stanford-sentiment-treebank-v2-sst2
4. https://huggingface.co/datasets/squad_v2
5. https://github.com/skandavivek/transformerQA-finetuning
6. FineTuning BERT for Multi-Class Classification on custom Dataset | Transformer fo…
7. https://huggingface.co/docs/transformers/tasks/question_answering