Individual Final Report

Natural Language Processing

Final Project

December 7, 2021

Yuan Meng

1. Introduction.

Our project utilizes the transfer learning method to apply the transformers models from the Hugging Face [1] website to solve the GLUE benchmark [2] tasks. First, we loaded the BERT and the RoBERTa pre-trained models, added an LSTM layer, and fine-tuned the model. Then, we followed the pipeline of the lecture codes and the Hugging Face transformers guide [3] to load and process the data and train and evaluate models. So the main coding part of our project is to build a class of custom model, and I customized the BERT model, which is discussed in the next section. As a result, we compared two models and found the RoBERTa worked better.

2. Description of individual work

I researched the BERT source code to learn how to modify it. And here is the structure of the model after adding the LSTM layer.

```
CustomBERTModel(
  (bert): BertModel(
    (embeddings): BertEmbeddings(
      (word_embeddings): Embedding(30522, 768, padding_idx=0)
      (position_embeddings): Embedding(512, 768)
      (token_type_embeddings): Embedding(2, 768)
      (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (encoder): BertEncoder(
      (layer): ModuleList(
        (0): BertLayer(
... ...
        (11): BertLayer(
          (attention): BertAttention(
            (self): BertSelfAttention(
              (query): Linear(in_features=768, out_features=768, bias=True)
              (key): Linear(in_features=768, out_features=768, bias=True)
              (value): Linear(in_features=768, out_features=768, bias=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
            (output): BertSelfOutput(
              (dense): Linear(in_features=768, out_features=768, bias=True)
              (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
          )
          (intermediate): BertIntermediate(
            (dense): Linear(in_features=768, out_features=3072, bias=True)
          )
          (output): BertOutput(
            (dense): Linear(in_features=3072, out_features=768, bias=True)
            (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
      )
    )
    (pooler): BertPooler(
      (dense): Linear(in_features=768, out_features=768, bias=True)
      (activation): Tanh()
    )
  )
  (lstm): LSTM(768, 768, batch_first=True, bidirectional=True)
  (clf): Linear(in_features=1536, out_features=2, bias=True)
)
```

It has the word, position, token type embeddings, and the layer norm for the embedding part. And the model has the encoder with 12 bert layers, containing the attention, output, intermediate parts. Then, the pooler layer is attached under them. Last, I added the LSTM layer and the linear classifier layer.

3. Description of the portion of my work

My contribution to the project includes codes, fine-tuning and training, reports and slides, and the GitHub repository. First, I combined the lecture codes and the hugging face tutorial to organize a processing pipeline for our project for the coding part. Then inspired by Ashwin Geet D'Sa's answer on Stackoverflow [4], I copied and modified the codes to build a customized BERT model. The specific percentages are shown in the fifth section. In addition, I modified the loss function and unsequenced the labels for the STS-B task because it's a regression task, not a classification task. Second, I trained the nine tasks with the customized BERT model for fine-tuning and training and fine-tuning them separately with Yuchen. The MNLI task's dataset is too large, so I must upgrade my GCP from NVIDIA T4 GPU to the NVIDIA A100 GPU to have enough Cuda memory. Third, I wrote for the Introduction, dataset description, and the experimental setup parts for the reports and slides. I also created a table to show our project evaluation results and filled in the BERT model results. In addition, we wrote the group proposal together, so we didn't split it. Last, I created a GitHub repository for our project and structured it following the requirements, and wrote the readme files.

4. Results

Here are my training and fine-tuning results of the BERT model.

Task	Batch size	Epoch	Learning rate	BERT
CoLA	64	30	5e^5	Matthew's Corr: 0.5931
SST-2	64	5		Accuracy: 0.9243
MRPC	16	5		F1: 0.9091 Accuracy: 0.8701
STS-B	8	5		Pearson: 0.8780 Spearman: 0.8757
QQP	32	5		F1: 0.8789 Accuracy: 0.9095
MNLI	32	5		Accuracy: 0.8338
QNLI	32	5		Accuracy: 0.9028
RTE	16	5		Accuracy: 0.6065
WNLI	4	30		Accuracy: 0.5493

This table represents the best results I got after fine-tuning; however, as we have other group members training the tasks together, it is not our group's best result. The WNLI dataset is small, so I defined a small batch size of data and a large number of epoch to get more training steps.

On the contrary, the MNLI' dataset is too large, so to get enough cuda memory, I couldn't have a batch size larger than 32, and I could only train for a small number of epochs. Then, for the learning rate, I found all the tasks fit 5e^5, so I didn't adjust it. Last, we can see that the model worked well for the SST-2 and MRPC, but not suitable for CoLA and WNLI tasks. Such differences could be caused by the size of datasets and task types.

5. Summary and conclusions

In conclusion, I contributed with codes, fine-tuning and training, proposal, reports and slides, and the GitHub repository to the project. Completing this project, I got a clear pipeline for dealing with different kinds of natural language processing tasks. Also, I learned how to apply transformers models and how to adjust them. In the future, I would like to try different models of architecture to solve the GLUE tasks to see any improvements.

6. Percentage of the code found from the internet.

Including the codes from lecture and transformers guides:
(75-8)/(75+2)*100 = 67/77*100 = 87.0%

Only for custom BERT model:
(12-6)/(12+2)*100 = 6/14*100 = 42.9%

7. References

1. Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz M., Davison J., Shleifer, S., Platen, P.V., Ma, C., Jernite Y., Plu, J., Xu, C., Scao, T.L., Gugger, S., Drame, M., Lhoest, Q., Rush, A.M. (2020, October). Transformers: State-of-the-Art Natural Language Processing, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations* (pp. 38-45). Association for Computational Linguistics. Retrieved December 7, 2021, from https://www.aclweb.org/anthology/2020.emnlp-demos.6.
2. Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., & Bowman, S. R. (2019, February 22). *Glue: A multi-task benchmark and analysis platform for natural language understanding*. arXiv:1804.07461. Retrieved December 7, 2021, from https://arxiv.org/abs/1804.07461.
3. Hugging Face Hub. (n.d.). *Text Classification on GLUE*. Retrieved December 8, 2021, from https://colab.research.google.com/github/huggingface/notebooks/blob/master/examples/text_classification.ipynb#scrollTo=7k8ge1L1IrJk.
4. Grable, A., & Geet D'Sa, A. (n.d.). *How can I add a Bi-LSTM layer on top of Bert Model?* Stack Overflow. Retrieved December 7, 2021, from https://stackoverflow.com/questions/65205582/how-can-i-add-a-bi-lstm-layer-on-top-of-bert-model?rq=1.