1 Project contents

1.1 Clarifying the Issue: Papers and Code

BERT (Devlin et al., 2019), a large pre-trained model, has revolutionized many topics in NLP. However, Zhang et al. (2021) pointed out an implementation issue with the AdamW optimizer for BERT. The major open-source packages, Hugging Face and Pytorch, have resolved this issue. To clarify the issue, we would like you to:

- Read the paper and describe the issue.
- Confirm the optimization issue in the original implementation of BERT[1].
- Point out the AdamW implementation in Hugging Face[2]. Specifically, you should indicate where the AdamW implementations are in the packages and what the differences are.

1.2 Experimenting with LibMultiLabel

Now you have a basic understanding of the issue. Let's investigate how differences in implementation impact multi-label text classification tasks. We will work with LibMultiLabel, an ongoing project in our lab.

- Install LibMultiLabel on the provided workstation.
- Download the data set **EUR-Lex** from `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multilabel.html`.
- Set up the configuration file. The file `example_config/EUR-Lex-57k/bert.yml` is an example for running BERT for EUR-Lex-57k data set, not for the EUR-Lex data set. Please modify the data-related arguments (e.g., `training_file`). For the following training arguments, please keep it the same except for `batch_size` and `patience`. You can modify `batch_size` and `patience` by situation.

```
seed: 1337
epochs: 100
batch_size: 32
optimizer: # TODO
learning_rate: 0.00005
weight_decay: 0.01
patience: 10
```

[1] `https://github.com/google-research/bert/`
[2] `https://huggingface.co/docs/transformers/main_classes/optimizer_schedules`

- Run BERT on EUR-Lex using three optimizers implemented in Pytorch and Hugging Face.
  - **PyTorch AdamW**. The functionality is available in LibMultiLabel.
  - **Hugging Face AdamW** with bias correction. To do this, you must figure out how to replace the PyTorch AdamW used in LibMultiLabel with Hugging Face AdamW.
  - **Hugging Face AdamW** without bias correction.
- Monitor the following metrics in your experiments. For each metric, generate a performance versus training epochs plot. Compare the rate of convergence and the performance of each optimizer.
  - **Training loss**: Modify the code to record the training losses of each epoch. Check out the `training_step` function in the `nn/model.py`. (Hint: Each loss obtained here is the average loss of one batch, so you need to consider the batch size when accumulating the total loss.)
  - **Validation Micro-F1, RP@5, and loss**: The functionality is available in LibMultiLabel.

1.3 Report

Write a 2-page progress report and send it to us before the last interview. Describe other findings in your investigation. If you have any unclear documentation or issues with LibMultiLabel, please let us know. We appreciate your feedback!

2 Project schedule

- day 1: a meeting to describe the project
- day 4: a Q/A session and a progress check
- day 8: project presentation

References

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In J. Burstein, C. Doran, and T. Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT*, pages 4171–4186, 2019. doi: 10.18653/v1/n19-1423.

T. Zhang, F. Wu, A. Katiyar, K. Q. Weinberger, and Y. Artzi. Revisiting few-sample BERT fine-tuning. In *Proceedings of International Conference on Learning Representations*, 2021.