Natural Language Processing: HW4 report

Taewook Kang

Department of Computer Science, Hanyang university (Student ID: 2021025205).

1. Introduction

Everyday human language has the potential to be inherently multi-intentional. From this perspective, language models should reflect this nature and be able to clearly identify the multiple intentions expressed in a natural language utterance. To equip large language models with this ability, I fine-tuned the pre-trained RoBERTa [1] model to classify multiple intentions using BlendX [7] dataset. In detail, LoRA was mainly used to enable efficient training in a resource-constrained training environment by updating a relatively small number of parameters to the full set of parameters in a large language model. Through various experiments to determine the optimal hyperparameters, the fine-tuned model achieved an accuracy of 87% and an F1 score of 93%. The model can be downloaded at `https://huggingface.co/twkang43/lora-roberta-cse4057`

2. Experiments

2.1. Task explanation

In this report, I solved the task of multi-intent detection for natural language utterances. Intent detection is crucial for understanding the main concept of natural language sentences, as they are usually lengthy and inherently ambiguous. For example, the model should interpret the sentence “Book a table at 7.” as a “book a restaurant” by resolving the ambiguity in the phrase “table” and understanding the context conveyed by the key components of the sentence. Moreover, we often convey multiple intents within a single sentence, such as “What’s my credit limit, and can I increase it?”. Therefore, to effectively process natural utterances and grasp the core meaning of the words, the model must identify and classify multiple intents within a sentence accurately.

2.2. Model information

I used RoBERTa [1], specifically `roberta-base` model from HuggingFace, as a pretrained large language model to address this task. RoBERTa is a variation of BERT [2], trained on a significantly larger dataset compared to the original BERT with the addition of optimized training techniques. Since large languages models typically demonstrate better performance when trained on larger datasets and a greater number of tokens, I assumed that RoBERTa would deliver more robust and rigorous performance than BERT.

rank	8
lora_aplha	32
target_modules	[“key”,”query”]
lora_dropout	0.1
task_type	“SEQ_CLS”

[Table 1] LoRA hyperparameters

epochs	10
learning rate	1e-3
optimizer	AdamW
lr scheduler type	cosine
warmup ration	0.1
precision	fp16

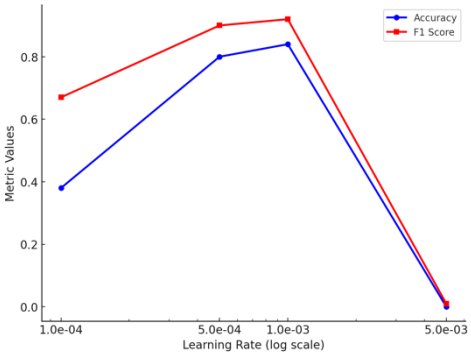
[Table 2] Trainer hyperparameters

Epoch	Training Loss	Validation Loss	Accuracy	Precision	Recall	F1
1	0.033200	0.021224	0.498264	0.731771	0.731771	0.731771
2	0.016400	0.011168	0.761806	0.877083	0.877083	0.877083
3	0.012500	0.008621	0.820139	0.907986	0.907986	0.907986
4	0.012600	0.007281	0.859722	0.928125	0.928125	0.928125
5	0.008100	0.005925	0.888542	0.943056	0.943056	0.943056
6	0.006500	0.004777	0.909028	0.953472	0.953472	0.953472
7	0.005000	0.003977	0.924306	0.961806	0.961806	0.961806
8	0.004100	0.003687	0.932292	0.965799	0.965799	0.965799
9	0.003500	0.003415	0.938889	0.968750	0.968750	0.968750
10	0.003200	0.003381	0.937500	0.967882	0.967882	0.967882

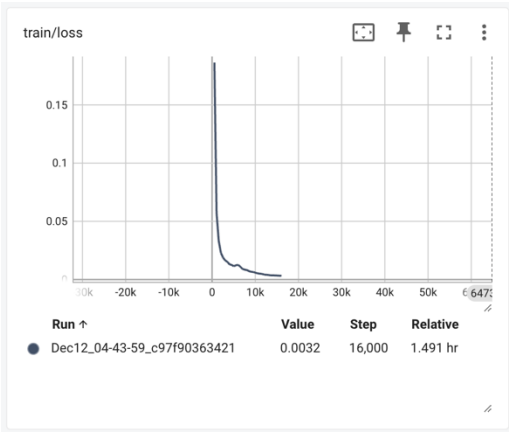
[Figure 1] Metrics observed during model fine-tuning

2.3. Training tactics

To train and evaluate model efficiently, HuggingFace trainer was used for training process. For the dataset, the BlendX [7] dataset with two intents per utterance was adopted for the multi-intent classification task. To prepare the data for the model, the intent field of the dataset was converted into multi-hot vectors, allowing for effective classification. Regarding the dataset split, the “train” data was used for training and validation, and “dev” data was used for evaluation. I split the “train” data into training and validation datasets with a 9:1 ratio and used them to monitor whether the model was overfitted during training. A batch size of 16 was used for both training and evaluation, as it demonstrated the best performance compared to other potential



[Figure 2] Impact of learning rate on evaluation metrics



[Figure 3] Loss trajectory of training

batch sizes (e.g. 32, 64).

In training procedure, I used a Google Colab's free tier environment which provides access to GPUs with a limited runtime. However, updating all model parameters during fine-tuning to achieve meaningful performance took a significant amount of time (approximately two hours for three epochs), which limited the number of experiments I could perform for proper hyperparameter tuning. Therefore, I adopted Low-rank adaptation of large language model (LoRA) [3] to optimize the fine-tuning process. LoRA makes a fine-tuning process efficient by learning low-rank updates to the pre-trained model's weight matrices, enabling the model to adapt to new tasks with minimal computational and memory overhead while achieving performance comparable to full fine-tuning. In experiment, the training time was dramatically reduced compared to updating all model hyperparameters, as LoRA required only about 24 minutes for three epochs. Table 1 presents the hyperparameters of LoRA used for training. Those values were chosen as they represent the common default settings for the method.

For hyperparameters, I used the values listed in Table 2. Specifically, I selected 10 epochs for training, as it achieved the best accuracy and F1 scores on the validation dataset. I did not use more than 10 epochs because the model trained with 10 epochs already showed slightly better performance than the one trained with 5 epochs. Continuing beyond 10 epochs raised concerns about overfitting and would have resulted in an unreasonable training time (10 epochs took approximately one and half hours). As shown in Figure 2, I selected a learning rate of 1e-3, as it achieved the best performance with 5 training epochs. Since the HuggingFace Trainer uses the AdamW [4] optimizer by default, I used it without any modifications. Additionally, a cosine learning rate scheduler with a warmup ratio of 0.1 [5] was used to stabilize the training process. This scheduler helps optimize the process by starting with a relatively large learning rate and gradually decreasing it over time, following a cosine scheduler, to facilitate smooth and stable model convergence by the end of the training. Also, FP16 precision [6] was adopted to optimize resource utilization. This method enables the model to achieve faster computations and reduced GPU memory consumption, while maintaining performance comparable to that of full precision. With these efforts, the training loss stabilized and converged (as shown in Figure 3) ensuring that the model was appropriately trained for multi-intent classification.

2.4. Results

	Vanilla	Fine-tuned
Accuracy	0.00	0.87
F1 score	0.01	0.93

[Table 3] Fine-tuning results

For the evaluation, accuracy and F1 score were used as the performance metrics, both calculated using the micro method to provide a comprehensive measure of the model's overall performance in multi-label classification tasks. To interpret the model's prediction results, the `compute_metrics` function first calculates the probability of each label using the sigmoid function. The reason why I used the sigmoid function instead of the softmax function, which is commonly used for classification tasks, is that the BlendX dataset is originally designed for multi-intent classification across various classes. In such cases, converting logits into independent probabilities using the sigmoid function is more appropriate than selecting labels with the softmax function, which induces dependency among labels. It then selects the top 2 labels with the highest likelihoods, thereby constructing a two-hot vector. Finally, the function computes the metrics by comparing the inferred label with the gold label.

The accuracy and F1 scores of the vanilla RoBERTa model and the fine-tuned RoBERTa model are presented in Table 3. The vanilla RoBERTa model, having never been exposed to the labels of the BlendX dataset, naturally exhibited very low accuracy and F1 scores. On the other hand, the LoRA-assisted fine-tuned RoBERTa model showed remarkable performance in multi-intent classification. These results demonstrate that the fine-tuning process can significantly enhance model performance for specific tasks.

2.5. Trial and error

As described in Section 2.2., it was crucial to train the model as efficiently as possible due to the strict limitations on available computing resources. Initially, full fine-tuning was attempted, but it was infeasible due to excessive training time and memory usage. Therefore, LoRA was chosen for fine-tuning the model, as it significantly reduces training time by decreasing the number of parameters that need to be updated.

For the data, the labels were initially used as-is; however, the intent field was not automatically separated into distinct intents. To resolve this issue, the external Python library `ast` was used to parse the intent field into individual components, ensuring proper labeling. Furthermore, to prevent the order of the labels from impacting evaluation results, they were converted into multi-hot (two-hot) vectors with a length equal to the total number of unique labels.

3. Conclusion

In this report, I fine-tuned a pre-trained large language model to tailor it for the specific task the resulting model is expected to perform. By adopting techniques to train the model with a lightweight additional structure and ensure stable convergence, it has been demonstrated that these methods effectively enhance the performance of the fine-tuned model. Since foundation large language models grow larger following scaling laws, their adoption becomes challenging in resource-constrained environments. Consequently, practical approaches are crucial to overcoming these limitations and enabling effective use of such models. This, in turn, will foster more active experimentation and research within the community. Furthermore, identifying essential tasks, such as understanding multi-intention expressions in human language as demonstrated in this assignment, and addressing these challenges with creative solutions is expected to greatly contribute to the field of natural language processing.

References

[1] Liu, Yinhan. "Roberta: A robustly optimized bert pretraining approach." *arXiv preprint arXiv:1907.11692* 364 (2019).

[2] Devlin, Jacob. "Bert: Pre-training of deep bidirectional transformers for language understanding." *arXiv preprint arXiv:1810.04805* (2018).

[3] Hu, Edward J., et al. "Lora: Low-rank adaptation of large language models." *arXiv preprint arXiv:2106.09685* (2021).

[4] Loshchilov, I. "Decoupled weight decay regularization." *arXiv preprint arXiv:1711.05101* (2017).

[5] Loshchilov, Ilya, and Frank Hutter. "Sgdr: Stochastic gradient descent with warm restarts." *arXiv preprint arXiv:1608.03983*(2016).

[6] Micikevicius, Paulius, et al. "Mixed precision training." *arXiv preprint arXiv:1710.03740* (2017).

[7] Yoon, Yejin, et al. "BlendX: Complex Multi-Intent Detection with Blended Patterns." *arXiv preprint arXiv:2403.18277* (2024)