Bert-based SLU Exploration

Hefei Chen Yuyang Xie Jiarui Ge

January 12, 2025

1 Summary

In the task of speaking language understanding (SLU), we managed to implement different algorithms to get higher scores in the classification and understanding of triples of act slot value. Specifically, we implemented the pre-trained Bert model and did some fine-tune. And we applied LoRA on the pre-trained Bert to lower the space cost of it. And we also implemented our FNN to suit the Bert and get a rather good result. We also analyze the result of the SLU-Baseline and our algorithms. And when finding the best parameters, we use optuna to help us.

2 Contribution

- Hefei Chen 522030910184 : Responsible for the fine-tune of Bert , the implementation of LoRA on Bert and optuna for parameter adjustment.
- Yuyang Xie 522030910015: Realize the replacement of Bert's last layer.
- Jiarui Ge 522030910209: Implement multi-head output of Bert's last layer

3 Implementation of algorithms

3.1 Bert fine-tune

First , we realized that the slu-baseline model is not good enough because it lacks a powerful model to process the language itself. So naturally we try to fine-tune a bert model to fix the problem. So from hugging face we downloaded bert-base-uncased and bert-base-chinese for simple test. Then we found that bert-base-uncased acted so poor that it was even worse than the baseline model. So we used bert-base-chinese for all our tasks then. We attempted to implemented the code of training and found that our f-score is higher than natural(comparing with our low dev accuracy). So we realized that we had to convert the ID label to act-slot-value to compute the f-score. So after modifying the code, we got our first bert fine-tune model.

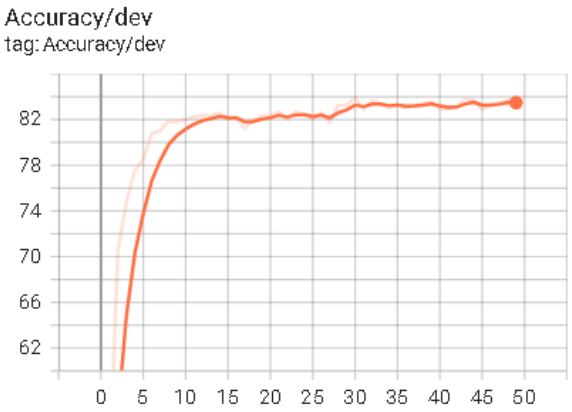

Figure 1: Curve of accuracy

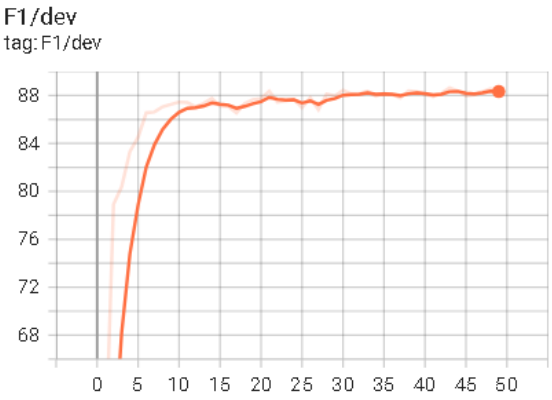

Figure 2: Curve of f1

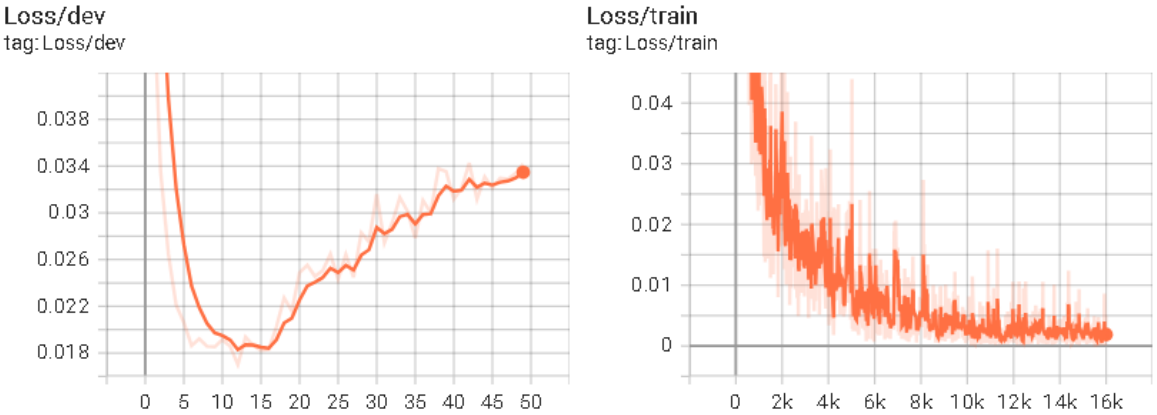

Figure 3: Curve of loss

3.2 Analysis of result

In the process of training, we found that the loss on development droped to the lowest and then rised. But the accuracy on development rised continuously, which is a little bit weird for the training process. But it is still explainable because the loss is calculated on a batch, and one mistake can cause the loss to rise tremendously.

And through the graph we can tell that the model converged quickly.

We will only show some of our graphs because of the similarity between different graphs of different model and limited space of our report.

3.3 Size reduction of bert

3.3.1 Layer frozen

Then we realized that the bert model is so huge. It costs space about 1.14G. So we wanted to freeze some layers of bert to get a smaller model. And we trained a model with 8 layers frozen, and the f-score is still satisfying.

3.3.2 Implementation of LoRA

Then we thought that we can use LoRA to reduce the size of bert. We imported peft and used its integrated function to implement LoRA on our bert model. And it turned out that the new model is even better than the previous one , and it only uses 1/4 space. And that's our best model and is the model submitted in the supplementary.

3.4 Parameter adjustment

We found that it's not an easy job to find the best hyper parameter for the model. So we tried some different ways to adjust our parameters.

3.4.1 Manually adjustment

First we try to adjust the parameters on our own. We tried to modify the learning rate and tried to use different optimizers. And we tried to add weight decay of Adam to make the training more efficient. And we tried to use different schedulers, including stepLR and warmup scheduler.

Finally the result is quite satisfying because we get the best model of dev accuracy of 83.91.

3.4.2 Implementation of Optuna

We also tried to adjust our parameters using Optuna.

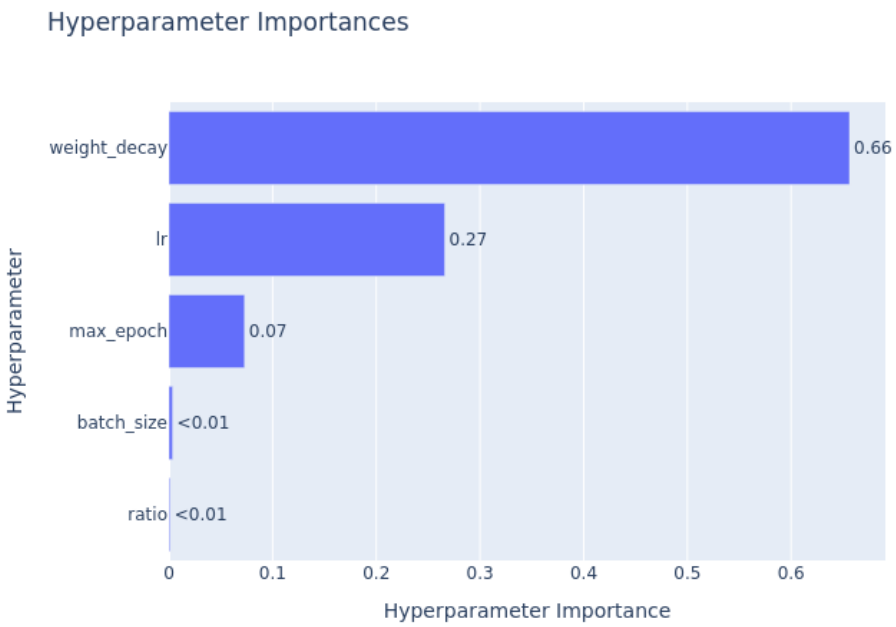

Figure 4: Importance of parameters

We can tell from the graph that weight decay matters most for our model, then is learning rate, which is quite surprising to us.

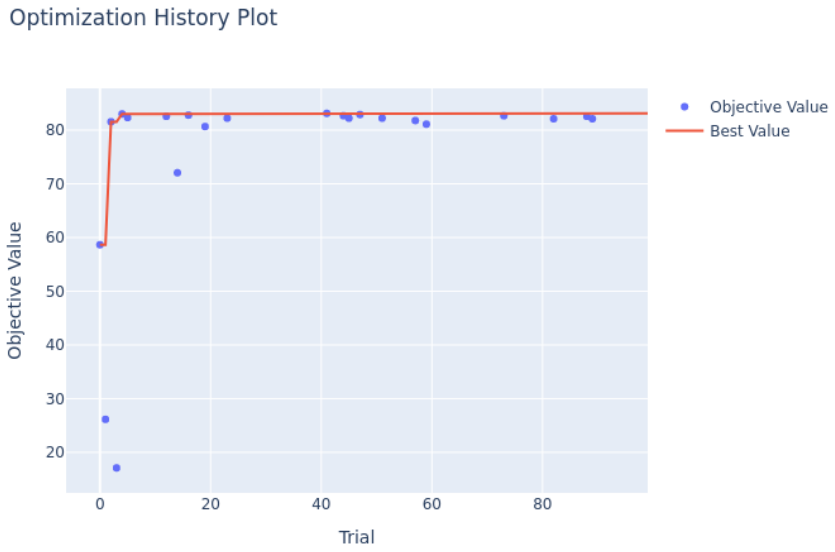

Figure 5: Slice plot

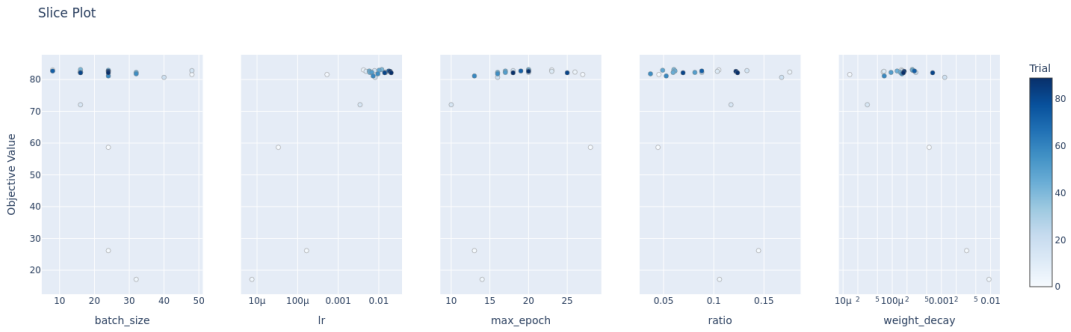

Figure 6: Optimization history

But we don't have much time for optuna training, so only slu-bert-small.bin is trained by optuna. Whose result is quite satisfying.

3.5 Replacement of the final layer of Bert

In this part we try to only replace the fully-connected layer at the end of the Bert model, and totally try three different Recurrent Neural Networks, includes the origin RNN, LSTM and GRU.

At first, I try to reconstruct the layer on my partner's codes, but I meet with some problems when modifying the layer in a BertTokenizer, so I use the inherit characteristic of python to build my "new" Bert model. And that's quite efficient, so I start my tuning of the network's arguments quickly. Although I have carefully adjusted the arguments, the performance of my new network was even worse than the baseline, which is quite strange. My new model occupies about 400MBs, which is ten times larger than the baseline model. After checking the codes, I found the mistake – I didn't use the bidirectional option in the RNNs config, which matters a lot. After using this option, my models' accuracy immediately rised to about 77 percents. I will show the detail outputs in the following table.

In my opinion, the LSTM may have the strongest ability to classify the categories, but it needs a lot of time to train, so that's why GRU performs better in this situation. RNN's performance is certainly worse than the more complex ones due to its simple structure.

3.6 Multi-head architecture for the final layer of Bert

In this section, we explore modifying the output layer of BERT by transforming it into a multi-head network.

In the original network implementation, there are 74 nodes in the output layer: one for `<pad>`, one for "O", and one for each combination of B/I, act, and slot. Tags that differ in only one component, such as `B-inform-value` and `I-inform-value`, are closely related. However, they correspond to separate output nodes, making it challenging to utilize the relationship between tags during model training.

To address this, we modify the output layer of BERT to include three "heads". The first

head outputs B/I/O, the second head outputs the act, and the third head outputs the slot. This configuration allows related tags to produce related outputs, thereby leveraging these relationships during training. Furthermore, the output layer is reduced to only 25 nodes, which results in a more compact network.

However, the performance of the multi-head network is not satisfactory. This might be because the multi-head architecture necessitates more carefully designed optimizers and loss functions.

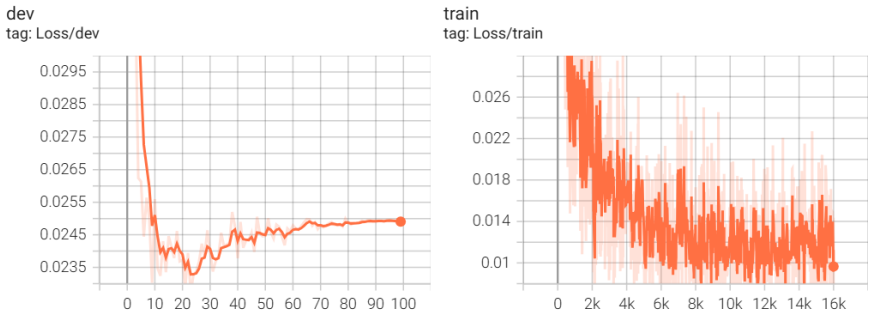

Figure 7: Loss of model with GRU

4 Some results of prediction

Name	dev-acc	dev-fscore	rank	model-size
slu-baseline-GRU	69.94	76.74	8	47.3MB
slu-baseline-LSTM	70.84	75.97	10	57.8MB
slu-baseline-RNN	70.39	76.12	9	26.2MB
slu-bert-GRU	77.02	83.91	4	418MB
slu-bert-LSTM	76.42	83.59	5	424MB
slu-bert-RNN	75.52	82.78	7	406MB
slu-bert-small	82.35	87.08	3	604MB
slu-bert-big	83.35	88.53	2	1.13GB
slu-bert-lora	83.91	88.43	1	392MB
slu-bert-multihead	76.08	82.65	6	414MB

Table 1: Results of different models

We can see from the table that although baseline models are not efficient enough, but they cost less to store. And our model is better in finishing the task, but cost much more to store.

All models are uploaded to jbox. More specific details can be found in the zip uploaded, including predictions.json etc.

Ways to run the code can be found in README.

5 Reference

1. Peters, Matthew, et al. "Semi-supervised sequence tagging with bidirectional language models."*Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vol. 1. 2017.

2. Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "BERT: Pre-training of deep bidirectional transformers for language understanding." *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*. 2019.

3. Vucetic, Danilo, Mohammadreza Tayaranian, Maryam Ziaeefard, James J. Clark, Brett H. Meyer, and Warren J. Gross. "Efficient fine-tuning of BERT models on the edge." *Proceedings of the 2022 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2022, pp. 1838–1842.