NLU Assignment 2

Edoardo Cecchinato (247434)

University of Trento

edoardo.cecchinato@studenti.unitn.it

1. Introduction

This assignment is divided into two tasks. The first one consists of adding bidirectionality and a dropout layer to a given model for intent classification and slot filling. The second instead requires the fine-tuning of a pre-trained BERT model using a multi-task learning setting on intent classification and slot filling. What I did in this last part was to create a new model focused on the two requested tasks using BERT as the backbone, handle the tokenization procedure for the input of the model, and modify the training functions to adapt them to my new model.

2. Implementation details

For the first part I made some changes to the module class *ModelIAS*: I set bidirectionality equal to True in the LSTM layer (this imply multiplying for two the input sizes for following intent and slot linear layers) and I created a dropout layer to use in the forward method for the embedding and LSTM output.

The second part was the most challenging and for that I used the *bert-base-uncased* pre-trained model from Hugging Face, and I took inspiration from this paper[1] for the implementation. For fine-tuning BERT I create a PyTorch Module (*BertIAS*), which contains a Bert instantiation, a linear layer for intent classification, a linear layer for slot filling, and a dropout layer as components. In the forward method I took the output of the BERT instance and I passed the first output sequence to the slot layer and the second one to the intent layer, after having applied dropout to both the outputs.

Since BERT requires a specific format for the input, I needed to convert my utterances text into that format using the BERT tokenizer, always from Hugging Face. The problem is that BERT tokenizer adds the special tokens *CLS* and *SEP* and potentially splits one word into more, so I needed to add some slots in the slot array using the *tokenize* method defined in the class *Lang*. What I did was setting the new added slots to -100 because I wanted BERT ignoring them during computation.

A part changing the model call, the train loop function remains the same. What changed was the information retrieval in the evaluation loop, particularly the untokenization of the *input_ids* using the method defined in the *Lang* object.

3. Results

The evaluation of each model was based on the function *eval_loop* that returns the F1 score for the slot filling task and the accuracy for the intent classification one. Following I'm going to show the results for the original model before applying any modification, then the bidirectional LSTM without and with dropout layer, and finally the fine-tuned BERT. You can see every result in table 1. Every new model obtained better scores for both tasks compared with the previous ones, with the fine-tuned BERT showing very high accuracy in intent classification. In table 2 I'm going to show the optimizer and learning rate used for the models during the training phase. For all the models, outside BERT, an embedding size of 200 and an hidden size of 300 were used.

Model	**F1 score (slots)**	**Accuracy (intents)**
Original LSTM	0.9142	0.9384
Bidirectional LSTM	0.9362	0.9417
Bidirectional LSTM + dropout	0.9410	0.9473
Fine-tuned BERT	0.9564	0.9776

Table 1: *Evaluation results for different models.*

Model	**Optim**	**L.R.**
Original LSTM	AdamW	0.0001
Bidirectional LSTM	AdamW	0.0001
Bidirectional LSTM + dropout	AdamW	0.0001
Fine-tuned BERT	AdamW	0.0001

Table 2: *Parameters used for different models.*

4. References

[1] Q. Chen, Z. Zhuo, and W. Wang, "Bert for joint intent classification and slot filling," 2019.