NLU project exercise lab: 10

Thomas Trevisan (240458)

University of Trento

thomas.trevisan@studenti.unitn.it

1. Introduction

In the first part:

- the bidirectional flag of the LSTM was set to *True*, concatenating the outputs before the respective classification layers
- A dropout layer was added to the embedding layer, to the output layer of the LSTM and to every hidden layer of the LSTM

In the second part:

- the word id's in the Lang class were changed with the id's of BertTokenizer's vocabulary. Bert special tokens were also added to the vocabulary
- In the collate function, the attention masks were computed
- Dropout and linear layers were added to the model in order to accomplish the classification task

For both parts, the Lang class was created in order to make the same vocabulary at every run.

2. Implementation details

In order to use Bert for joint slot filling and intent classification as in [1], I had to figure out a way to use its tokenizer in order to create the input ids and attention masks. Here, a criticality arises because Bert tokenizer is based on WordPiece and tends to generate more tokens compared to whitespace tokenization. We then would end up with having more words than slots in the dataset. To avoid this, I used only the *convert_tokens_to_ids* function of the BERT tokenizer, and passed it the whitespace tokenized text. This could result in having more tokens not recognized by BERT tokenizer, and therefore in more [UNK] ids, but simplified a lot the implementation. Special tokens [CLS] and [SEP] are added in the *get item* function of the dataset class. Since I don't use the Bert tokenization, i had to create manually the mask in the collate function of the dataloader and returned it along with the rest of the data.

For what concerns the model, I built on top of the last hidden state a linear classifier in order to perform intent classification, and on top of the output layer another linear layer to perform slot classification. Before being passed to these linear classifiers, a dropout layer was applied to the output.

The used loss is the CrossEntropyLoss and as optimizer Adam was used. The settings and chosen hyperparameters are:

- Dropout probability = 0.2
- Learning rate = 5e-5
- Epsilon = 1e-8
- Epochs = 10

A total of 10 runs were made resulting in a total of 10 different models, and the best between all runs was saved. The test performances are done on this best model.

3. Results

	Intent Accuracy	**Slot F1**
LSTM	0.962	0.951
JointBert	0.972	0.947

For the intent classification, i used accuracy as metric. On the other hand, for slot classification i used f1. Table 3 shows the results achieved in part 1 and part 2 performing the task with an LSTM and Bert respectively. As we can see, there is a clear improvement for what concerns intents. Slots f1 instead suffers a bit more with Bert, probably due to the unknown words in BertTokenizer's vocabulary.

4. References

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