NLU Report (Assignment 2)

Silvano Vento Maddonni (247370)

University of Trento

silvano.maddonni@studenti.unitn.it

1. Introduction

The topic of this project is the task of Slot filling and Intent classification. The dataset used is ATIS, which contains airlines travel information. The work is divided in two parts. In the first one, starting from a IAS (Intent And Slot) Model, we modify the baseline architecture by adding bidirectionality and dropout layers. In the second part, the goal is to modify our architecture to fine-tune a pre-trained BERT model using a multi-task learning setting on intent classification and slot filling.

2. Implementation details

Part 1

1.1 - Bidirectionality

The ModelIAS architecture is initially modified by adding bidirectionality. This is done by setting the bidirectional option of the LSTM to *True*, but in order to make it work, we also have to double the hidden layers matrix size. This happens because in a bidirectional LSTM, the hidden state output size effectively doubles because it concatenates the hidden states from both the forward and backward passes.

1.2 - Dropout

Dropout layers could be added in different positions:

- After the Embedding layer to reduce the chance of overfitting on the raw input data.
- After the LSTM layer, preventing the network from relying too heavily on specific LSTM outputs.
- Before the Output layer, ensuring that the final representations used for slot and intent prediction are also regularized.

After some experiments, we come to the conclusion that the dropout on the embeddings behave positively, but adding it also on the LSTM or output doesn't improve (see Table 1).

Part 2

Fine-tune BERT

To fine-tune BERT for this task, we start by getting the model (*bert-base-uncased*) and the tokenizer from Hugging Face, and we use them to build our new architecture. The implementation consisted of different steps:

- As first thing, we modified the data loading procedure:
  - In the Lang class, we removed the w2id method as we will use the tokenizer's vocabulary.
  - One of the main difference with using BERT and his tokenizer is that this model uses a word-piece tokenizer, where the words are further sub-tokenized. For the slots, the approach is to assign the slot label to the first token that represents a word. The tokenization mapping is performed on the words instead than on sequences.
  - A new *IntentAndSlotsBert* class is created based on the previous one. Here, using the tokenizer, we will get the ids of the tokens and the attention mask. Inside that, the *mapping_seq* is updated to handle special tokens. We only map the first token and add padding for the rest. We exclude [CLS] and [SEP] from mapping, adding them to the sequence later.
  - Changed the *collate_fn* function to match (ids, mask, slots, intent) instead of (utt, slots, intent).
- A BERT_1 model is created. The structure is very simple: It is composed of a pre-trained bert-uncased that outputs the tuple: (*last_hidden_state, pooler_output*) and on top of it two additional linear layers: *slot_classifier* (130 slots) and *intent_classifier* (26 classes). On the sequence and pooled outputs, before calling the classifier, a dropout (p=0.3) is applied, but a version without dropout is also tried (showing that the difference is not very high. See Results table).
- Train and Eval loops are also adjusted: In the first, we make a different call to the model and ensure that the slots are permuted. The second is slightly different from the original, because when getting *utt_ids* we want to get the sequence without the padding *0*s.

3. Results

We present now the results of the experiments conducted. The evaluation metrics used are: *Accuracy* for Intent classification and *F1-score* with *conll* for Slot filling. All the proposed modifications improved the performance of both F1 and Accuracy, which were already high.

Table 1: *Summary of Experimental Results.*

	Model	**Slot F1**	**Intent Acc.**
–	Base IAS Model	0.924	0.943
1.1	Bidirectionality	0.934	0.948
1.2	Dropout on Embeddings	0.941	0.946
1.2	Dropout on Embeddings and on LSTM	0.943	0.945
1.2	Dropout on LSTM and output	0.940	0.944
2.0	Fine-tuned BERT	0.957	0.968
2.0	Fine-tuned BERT with dropout	0.958	0.976