NLU course project - NLU assignment

Pietro Bologna (248715)

University of Trento

pietro.bologna@studenti.unitn.it

1. Introduction

The NLU assignment focuses on enhancing the baseline architecture of the ModelIAS. For the first task I modified the architecture by incorporating a dropout layer with probability *p=0.1* and adding bidirectionality. As we see, these modifications have made improvements to our model. The second task requires fine-tuning a pre-trained BERT model. Here, one of the key challenges being the handling of sub-tokenization issues. To solve this problem, adaptation strategies were implemented to maintain correct and accurate slot predictions.

Performance will be evaluated using accuracy for intent classification and F1 score with the conll [1] metric for slot filling.

2. Implementation details

For the first part of the assignment, I modified the ModelIAS by adding **bidirectionality**. This technique allows the model to capture much more information from a sequence. To do this, the hidden size is doubled and, in the forward step, the last hidden states of the two directions are concatenated. After that I implemented **dropout layer**, which is a regularisation technique used to prevent overfitting.

For the second part, the challenge is to adapt the code to fine-tune a pre-trained **BERT model** [2] for the same tasks as before (slot filling and intent classification). Here, the most challenging part involved the problem of the sub-tokenization issue of BERT. Specifically, BERT uses a WordPiece tokenizer, which differs from the simple word-based tokenization used in ModelIAS. The WordPiece tokenizer can split words into smaller subword units, or sub-tokens. This creates a challenge for slot filling because a single word might correspond to multiple sub-tokens. Note that, two important tokens of BERT are:

- *[CLS]*: a token added at the beginning of the phrase;
- *[SEP]*: a token added at the ending of the phrase.

Another character used by BERT is *##*, which indicates that a word has been tokenized into multiple sub-tokens. Since BERT requires these special tokens, I adapted the $IntentsAndSlots$ function to obtain a result similar to the one presented below:

	Word-based	BERT
Utterance	['playing', 'drum']	['play', '##ing', 'drum']
Slot	['O', 'B-INST']	['O', 'pad', 'B-INST']

Table 1: *Comparison of word-based and BERT tokenization*

Table 1 illustrates how tokenization differs between word-based methods and BERT, highlighting the need for adjustments in the slot labeling process. As we can see, the first sub-token should retain the original slot label, while the subsequent sub-tokens should be labeled as padding (*pad*). Furthermore, for the *[CLS]* and *[SEP]* tokens, I've added them to the utterance (one at the beginning and one at the end) and then replace them with a *'O'* in the slots.

3. Results

The implementation of bidirectionality has improved the model's ability to capture contextual dependencies within the sentence and the dropout to generalise better and prevent overfitting. The Table 2 shows the results obtained for the first part of the assignment. All these results were obtained using the ModelIAS and the Adam optimizer with a learning rate of 0.0001 [1]:

Experiment type	Slot F1	Intent Accuracy
Only ModelIAS	0.898 ± 0.019	0.921 ± 0.012
Bidir	0.938 ± 0.002	0.943 ± 0.002
Bidir + Drop	0.935 ± 0.002	0.937 ± 0.002

Table 2: *Results with ModelIAS*

As we can see, the difference between using the standard ModelIAS and implementing bidirectionality increases both slot F1 and intent accuracy. In particular, we can see a large increase in the slot F1 (about 3.5%). Moreover, dropout appears to reduce accuracy slightly compared to bidirectionality alone. This might suggest that the model is not significantly overfitting, in fact if we look at Figure 1, we see that the model is not overfitting. During the various tests, I also tried replacing the Adam optimizer with AdamW. I noticed that, due to the different handling of the weight decay, AdamW performs slightly better.

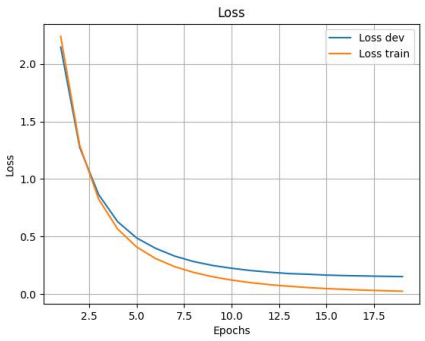

Figure 1: *Loss with ModelIAS*

In Table 3, we can see that the results obtained with BERT are superior to those obtained previously. As it was intended to prove, this confirms that BERT, due to its pre-training on large amounts of data and its transformer-based architecture, is much

[1] The values show a variation of $\pm$ as several tests were performed by varying the values of the hidden size and embedding size.

more effective in capturing context and solving intent classification and slot fillin tasks than a traditional LSTM model.

This confirms, as intended, that BERT is much more effective in capturing context and solving intent classification and slot filling tasks compared to a traditional LSTM model, due to its pre-training on large amounts of data and its transformer-based architecture.

Experiment type	Slot F1	Intent Accuracy
BERT	0.954 ± 0.002	0.977 ± 0.002

Table 3: *Results with BERT*

4. References

[1] BrownFortress, "Nlu-2024-labs: Labs for natural language understanding 2024," https://github.com/BrownFortress/NLU-2024-Labs/blob/main/labs/conll.py, 2024, modified version of https://pypi.org/project/conlleval/.

[2] Hugging Face, "Bert-base, uncased." [Online]. Available: https://huggingface.co/google-bert/bert-base-uncased