# NLU course project lab5

*Mattia Rigon (mat. 217182)*

University of Trento

mattia.rigondz@studenti.unitn.it

## 1. Introduction

This report aims to discuss the implementation of a model that performs intent classification and slot filling. The assignment is divided into two main parts:

1. Improvement of an existing model based on an LSTM by adding bidirectionality and dropout.
2. Substitution of the LSTM model with a pre-trained BERT, which has to be fine-tuned for the tasks of intent classification and slot filling, handling in the correct way the sub-token problem introduced by the BERT tokenizer.

The project was tested on the ATIS dataset and accuracy for intent classification and F1 score for slot filling was used in order to evaluate the model.

## 2. Implementation details

The first task consisted in adding bidirectionality and the dropout layer to the LSTM layer by using the standard implementation. These two changes were implemented incrementally in order to observe the improvement offered by each one.

The second task consisted in substituting the LSTM architecture with the BERT one (Devin et. al.) which is a pre-trained transformer model that can be fine-tuned in order to be used in several tasks, such as intent classification and slot filling, as suggested by Cheng et. al.

The problem that needed to be solved during this task was the desalination between the tokens generated by the Bert tokenizer and the labels of the slots provided by the ATIS dataset. The desalination was solved in order to provide the Bert model with correctly pre-processed data. The BERT tokenizer does not tokenize on a word-by-word basis; instead, a word can be tokenized into multiple subtokens. For example, the word "tokenization" is tokenized into two subtokens: "token","##ization", but it has just one slot value. The solution that was implemented solves this problem by keeping track of the tokens that had been split into subtokens during the tokenization step. For each token that had been split into subtokens, the slot id relative to the token was assigned to the first subtoken, and the special token '[PAD]' was assigned to the other subtokens. In this way, the length of the tokenized utterance is the same as the length of the slot ids, solving the mismatch of length introduced by the Bert tokenizer.

The architecture of the model that was implemented has a Bert layer, two dropout layers, and two linear layers. The data were placed as inputs to the Bert layer which produces two outputs: the last hidden state, which was used for the slot filling task, and the polled output, which was used for the intent classification. In order to regularize the model, before these two linear layers, two dropout layers were respectively included. The dimensions of these two layers are, respectively, the number of slot classes and intent classes that are in the dataset.

In order to train the model, the loss is defined as the sum of two losses, one for each task. For both the tasks the cross entropy loss was used.

For the evaluation of the model of the slot filling task, it was necessary to process both the ground truth data and the hypothesis data. For the ground truth, it was necessary to delete all the pad tokens that had been inserted in the preprocessing of our data. For the hypothesis generated by our model, it was necessary to delete all the slots that had the same index as the ones that had been deleted in the ground truth. Finally, both the ground truth and the hypothesis were used in order to calculate the F1 score. The evaluation of the intent has been carried out by using the classical classification and by computing the accuracy value.

The implementation has been inspired by: https://github.com/monologg/JointBERT

## 3. Results

In the first part, the bidirectionality and the dropout layer improved the performance by adding to the LSTM. In order to get more significant results, 4 runs by 200 epochs each were run and the results were then averaged.

Table 1: *Results of the first part. Using hidden size 350, embedding size 350, lr* $10^{-4}$*. The batches size used for the experiment are: 128 for the training set, 64 for the validation and the test set.*

| | **Slot F1** | **Intent Accuracy** |
|---|---|---|
| **Bidirectional** | 0.937 | 0.94 |
| **Bidirectional + Dropout** | 0.939 | 0.95 |

For the second part the experiment is carried out by using a single run of 50 epochs.

Table 2: *Results of the second part. lr* $= 5 * 10^{-5}$*. The batches size used for the experiment are: 128 for the training set, 64 for the validation and the test set.*

| | **Slot F1** | **Intent Accuracy** |
|---|---|---|
| **Bert** | 0.97 | 0.96 |

## 4. References

[1] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019.

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

Table 3: *In this example, the B-airline_code is split into two subtokens (t ##wa), so its corresponding slot is split in the real tag and the pad tag.*

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| utterance | [CLS] | what | classes | of | service | does | t | ##wa | have | [SEP] |
| slots | [PAD] | 0 | 0 | 0 | 0 | 0 | B-airline_code | [PAD] | 0 | [PAD] |

Table 4: *Deleting process in the evaluation of the pad token from the ground truth, and the slot predicted in the same position in the hypothesis.*

| | | | | | |
|---|---|---|---|---|---|
| reference | O | T-POS | ~~[PAD]~~ | O | O |
| hypothesis | O | T-POS | ~~O~~ | T-NEG | O |

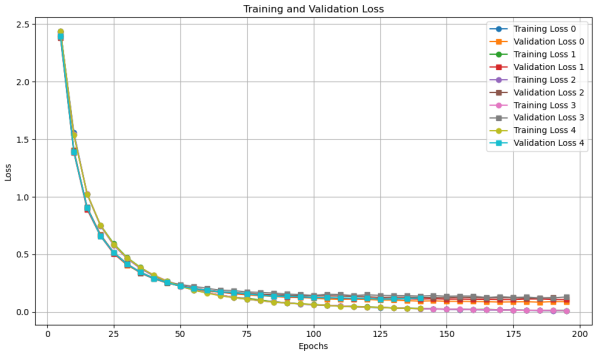


Figure 1: *Plot of the training and validation losses of the first part.*