# Final NLU project

*Sebastiano Dissegna(232090)*

University of Trento
sebastiano.dissegna@studenti.unitn.it

## 1. Introduction

In this report, i will explain my approach in solving a Multi-task Learning problem. More specifically the one of joint Intent Classification and Slot Filling. The objective is to evaluate the performance of a baseline model and find some way to improve upon it. In this case I tried using different final layer given a model, changing the encoder and using a completely different architecture. The Datasets used to train and test all the models are the ATIS and the SNIPS, which are common for this tasks.

All the experiments were performed on a machine with the following specification and software:

- CPU: Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz,
- RAM: 16GB at 2400MHz,
- GPU: Nvidia GTX 1060(6GB) laptop version,
- CUDA version 11.6,
- Python version 3.7

The code alongside the log from each run can be found here NLU_Project_232090

## 2. Task Formalisation

What i had to do for this project is to try different models that should be able to tackle simultaneously this two tasks:

- Intent classification,
- Slot Filling;

Both of this two tasks are very important work in the field of Natural Language Understanding(NLU). Let's start with Intent Classification. Here we want to, given a sentence or utterance, find the belonging class from a specified set of other classes. An example taken directly from the ATIS dataset is this one, where the sentence "what is the arrival time in san francisco for the 755 am flight leaving washington" will need to be classified with the intent "flight_time". For the other task, the Slot Filling one, we need to attach to each token in a sentence a slot label, in order to be able to give a semantic meaning to each of them, allowing us to better understand the general meaning of an utterance and helping in choosing the right response to it. An example, taking as input the same sentence used for intent classification, we will need to map it to "O O O B-flight_time I-flight_time O B-fromloc.city_name I-fromloc.city_name O O B-depart_time.time I-depart_time.time O O B-fromloc.city_name ". As you can see these tasks are extremely correlated, so much so that it's easier to train a model to learn them at the same time, than training one specifically for each of them. The baseline given to us that we have to improve on the given datasets are the following:

- ATIS: Intent Accuracy 92%, Slot F1 92%;
- SNIPS: Intent Accuracy 96%, Slot F1 80%;

As you will see in the following sections this was achieved.

```
{
    "utterance": "what 's the airport at orlando",
    "slots": "O O O O O B-city_name",
    "intent": "airport"
},
```

Listing 1: *Dataset JSON structure example*

## 3. Data Description and Analysis

The datasets used for training and evaluating the models are two common datasets for this tasks, ATIS and SNIPS. Both of them were provided to us by the lab professor during the tenth laboratory. They are in the JSON format and structured like this 1. We have three fields for each example:

- "utterance": the raw sentence,
- "slots" : the ground truth labels for the slot filling task,
- "intent" : the ground truth label for the intent classification task;

Now let's take a more detailed look at the Datasets.

### 3.1. SNIPS

The data for this dataset where collected from the Snips voice assistants. The dataset is already splitted in three subsets:

- Training: with 13084 samples,
- Valid: with 700 samples,
- Test: with 700 samples,

For a total of 14484 samples, with a vocabulary length of 10621 words. The total number of intents is 7, while slot labels are 72. Both of them are equally distributed in the dataset. The structure of the file is the aforementioned one1. As we will see later the complexity of this dataset will make the gain in performance with a more complete model more significant.

### 3.2. ATIS

ATIS stands for Airline Travel Information system, and its a dataset composed of audio recordings of people asking information about flights booking to an airport. The dataset is splitted in two subsets:

- Train: with 4978 samples,
- Test: with 893 samples;

For a total of 5871 samples, with a vocabulary of 863 words. Since in this case a given Validation set is not provided, I extracted it from the training one with the methods illustrated in laboratory 10 which take into account the label distribution, allowing us to have a balanced training and validation set, without having some label missing from one or the other. So around 10% of the training set was used to create a validation set of

597 samples, reducing its size to 4381 of the original 4978 examples. Here the number of intents is 26, and the slots labels are 129. One important thing to say is that in the training set only 21 intents and 120 slot labels are present, resulting in an unrecoverable difference in performance between the evaluation set and the test set, which from the experiment amounts to a 2-3% loss in accuracy. One last thing to add is the very low support, also known as frequence, with which some intents appears in the training set, making them harder to learn. All of this makes this simple dataset harder to train on compared to SNIPS even if the overall complexity is lower.

## 4. Model

For the choice of which model to implement i decided to use 3 different architecture, with the first two very similar with each other. Both of them are a different version from the one used in laboratory 10, and the third one is an implementation of the JointBert Model illustrated in its paper[1]; of which no official code was available, but I found an unofficial version avaialble hereJointBert implementation.

### 4.1. LSTM and GRU

Let's start with the first two models, whose architecture is visible here.

```
LSTM_MODEL(
  (embedding): Embedding(863, 300, padding_idx=0)
  (utt_encoder): LSTM(300, 100, bidirectional=True)
  (intent_classifier): Sequential(
    (0): LayerNorm((100,), eps=1e-05, elementwise_affine=True)
    (1): Linear(in_features=100, out_features=26, bias=True)
  )
  (slot_classifier): Sequential(
    (0): LayerNorm((200,), eps=1e-05, elementwise_affine=True)
    (1): Linear(in_features=200, out_features=130, bias=True)
  )
)

GRU_Model(
  (embedding): Embedding(863, 300, padding_idx=0)
  (utt_encoder): GRU(300, 100, bidirectional=True)
  (intent_classifier): Sequential(
    (0): LayerNorm((100,), eps=1e-05, elementwise_affine=True)
    (1): Linear(in_features=100, out_features=26, bias=True)
  )
  (slot_classifier): Sequential(
    (0): LayerNorm((200,), eps=1e-05, elementwise_affine=True)
    (1): Linear(in_features=200, out_features=130, bias=True)
  )
)
```

For this first two models i decided to go for the KISS design and the Occam razor principle. For the baseline there is no need to create something complicated, so the answer is a simple one; even a simple model can perform quite well. Moreover this is proven to be true from the results showed in section 5. Both of this two model share the same type of embedding layer and the two classification layer, one for the intent classification task and the other one for the slot filling one, where each of them has a Layer Norm layer followed by a final fully connected Linear layer. What has changed from the implementation provided in laboratory 10 is the Encoder which now use of a multi-layer bidirectional long short-term memory (LSTM) RNN instead of a single layer one. The second one uses a multi-layer gated recurrent unit (GRU) RNN in the place of the LSTM one. In theory the difference between a LSTM and a GRU is that the second one uses only two gates, an update and a reset one, resulting in less parameter and a faster training time. This is very important since it allow to train for more epochs, resulting in better results. On top of this to compare different regularization techniques the Layer Norm layer can be substituted with a Dropout layer. One final addition is a CRF layer which is put on top of everything. For the optimizer Adam is used in both models, along with the cross entropy loss as for the loss function, which is converted to a negative log likelihood if the CRF layer is enabled.

### 4.2. JointBert

Now for the third and final model i decided to use this model illustrated in the paper [1]. For the encoder part a pre trained uncased BERT model, also known as Bidirectional Encored Representations from Transformers is used 1. The full model architecture is available in the "models.txt" file in the github repository of this project NLU_Project_232090. In order to use BERT the tokenizer needed to be changed in favor of the WordPiece one, which is capable of slitting complex words into smaller tokens and in order to work at every sample a new token "[CLS]" needs to be added at the start of the sentence along with a token "[SEP]" at the end, and they will serve as a token for the sentence classification. The intent Classification takes as input the hidden state of the first special token "[CLS]", and for Slot filling the last hidden state if other tokens are used. In this case a more efficient version of the Adam optimizer is used called AdamW, which changes the way the weight decay is computed making it faster, furthermore as we will see in the next section 5, faster computation are essential due to the complexity of this model.

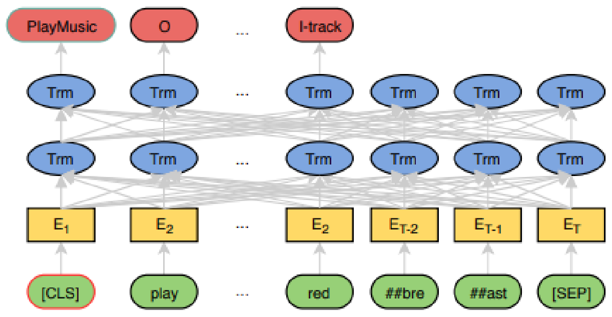

Figure 1: *BERT model with input the query: "play the song little robin redbreast"*

### 4.3. Experiment Settings

To recap briefly what can be changed in all three models:

- Add a CRF layer on top of everything,
- Substitute the Layer Norm with a Dropout layer

## 5. Evaluation

As stated in the beginning of this report every experiment run was executed on my laptop, which have an old GPU, so all the training process was very slow, especially for JointBert, were it would take more than one hour for a single epoch. For this reason the number of epochs for a single run of JointBert was set at 3, and no multiple consecutive runs were performed with it. Another thing i noticed is that adding a CRF layer slows down the training process by a lot, making it 10 times slower than a version without it. As expected the GRU version is faster to train than LSTM.

| Dataset | CRF layer | Reg | Epochs | Intent Acc. | Slot F1 score |
|---|---|---|---|---|---|
| ATIS | No | LNorm | 70 | 91.6 | 87.4 |
| | Yes | LNorm | 70 | 90.8 | **93.1** |
| | No | Dropout | 200 | **94.6** | 92 |
| | Yes | Dropout | 50 | 86.8 | 88.1 |
| SNIPS | No | LNorm | 50 | 95.5 | 81.4 |
| | Yes | LNorm | 50 | 95.5 | **88.1** |
| | No | Dropout | 50 | **96.2** | 76.76 |
| | Yes | Dropout | 50 | 96 | 83.3 |

Table 1: *BiLSTM Single Run Results*

| Dataset | CRF layer | Reg | Epochs | Intent Acc. | Slot F1 score |
|---|---|---|---|---|---|
| ATIS | No | LNorm | 200 | **95.3** | 93.7 |
| | Yes | LNorm | 150 | 94.2 | **94.6** |
| | No | Dropout | 150 | 93.4 | 89.1 |
| | Yes | Dropout | 50 | 76 | 84.8 |
| SNIPS | No | LNorm | 45 | 95.6 | 81.8 |
| | Yes | LNorm | 36 | 94.7 | 83.7 |
| | No | Dropout | 50 | 95.4 | 80.6 |
| | Yes | Dropout | 50 | **96.8** | **88** |

Table 2: *GRU Single Run Results*

| Dataset | CRF layer | Reg | Epochs | Intent Acc. | Slot F1 score |
|---|---|---|---|---|---|
| ATIS | No | LNorm | 3 | **96.9** | 94.5 |
| | Yes | LNorm | 3 | 95.2 | **95.1** |
| | No | Dropout | 3 | 96.2 | 93 |
| | Yes | Dropout | 3 | 88.3 | 94.2 |
| SNIPS | No | LNorm | 5 | 98.1 | 95.5 |
| | Yes | LNorm | 5 | 98 | 96.4 |
| | No | Dropout | 5 | 98 | 94.6 |
| | Yes | Dropout | 5 | **98.3** | **97.4** |

Table 3: *Joint Bert Single Run Results*

| Dataset | CRF layer | Reg | Epochs | Intent Acc. | Slot F1 score |
|---|---|---|---|---|---|
| ATIS | No | LNorm | 100 | 94.8±0.4 | 91.9±0.4 |
| | Yes | LNorm | 100 | 93.9 ± 0.4 | 94 ± 0.4 |
| | No | Dropout | 100 | 89.3±0.6 | 83.5±0.6 |
| | Yes | Dropout | 100 | 88.2±1.1 | 92±1.1 |
| SNIPS | No | LNorm | 100 | 96±0.7 | 83.7±0.7 |
| | Yes | LNorm | 100 | 95.9±0.6 | 87.9±0.6 |
| | No | Dropout | 100 | 96.2±0.4 | 80.7±0.4 |
| | Yes | Dropout | 100 | 95.6±0.3 | 86.6±0.3 |

Table 4: *BiLSTM Multiple Runs Results*

| Dataset | CRF layer | Reg | Epochs | Intent Acc. | Slot F1 score |
|---|---|---|---|---|---|
| ATIS | No | LNorm | 100 | 94.9±0.7 | 91.6±0.7 |
| | Yes | LNorm | 100 | 94.2±0.3 | 94.4±0.3 |
| | No | Dropout | 100 | 89.1±0.4 | 85.7±0.4 |
| | Yes | Dropout | 100 | 86.8±0.6 | 91.5±0.6 |
| SNIPS | No | LNorm | 100 | 95.9±1.4 | 84.7±1.4 |
| | Yes | LNorm | 100 | 94.7±1.2 | 87.6±1.2 |
| | No | Dropout | 100 | 94.8±1 | 84.1±1 |
| | Yes | Dropout | 100 | 95.5±1.5 | 89.6±1.5 |

Table 5: *GRU Multiple Runs Results*

### 5.1. Metrics used

Following the project instruction two types of metrics are used:

- F1 Score for slot filling,
- Accuracy for Intent Classification;

Accuracy is computed as the sum of True positive and True negative, all divided by the number of samples. For computing the F1 score, the already implemented function inside the provided "conll.py" was used; this is a metric that consider both recall and precision scores.

### 5.2. Single Run Results

The results of each test are visible here:

- Table11 shows the results of the single run using the BiLSTM model with all possible configuration,
- Table22 shows the results of the single run using the BiGRU model with all possible configuration,
- Table33 shows the results of the single run using the JointBert model with all possible configuration;

All this results are computed on the test sets. The first important thing to say about all the runs is the importance in the number of epochs. For the GRU and LSTM models this number vary from 36 to 200, this is due to some reason regarding the training settings, which will be explained later. Due to hardware limitation only in some occasion where the training was faster, visible where CRF layer was not enable, I increased the number of epochs and this resulted in an higher accuracy of 1% for both Intent and Slot. We can safely say that in this kind of domain, increasing the number of epochs is a safe and straightforward way to achieve better results. There are two things to notice in table 2, when using both Dropout and CRF on the ATIS dataset we can see a very low result, but this can be simply considered an unlucky run because if we take a look at the same configuration run multiple times in sequence, the results are on par to the ones obtained with different settings. The second thing concerns the two cases on the SNIPS dataset, were the number of epochs is less than 50. In all the single runs experiment a form of early stopping was introduced, where if for more epochs than a "patience" parameter, there were no form of improvements the training would stop. The JointBert model is the one that achieves better results for both Intent and Slot tasks, when compared to the other two, even if the number of epochs is only 3 or 5. We can see the effectiveness of a deep model that uses the self attention technique, the most important characteristic of transformers, which allow to better comprehend the context inside an utterance. The given target performance were surpassed even when using the simpler models with no CRF layer and Dropout. From the results we can see that especially when looking at the runs where CRF layer is enabled the F1 score increases by a lot, proving the effectiveness of adding such layer on top of everything. The same thing can be said for dropout, even if to a lower magnitude. Is important to also remember that for the ATIS Dataset there is an unavoidable loss in performance when evaluating the model on the test set compared to the evaluation one, for the missing label not present in the training set, which will be impossible to learn. This difference is estimated to be around 2-3 % on average. The best configuration for the SNIPS dataset is the one with JointBert with both CRF and dropout enabled, with an accuracy of 98.3 for the Intent, and a F1 score of 97.4 for Slot. For the ATIS dataset, JointBert with no CRF and dropout is the

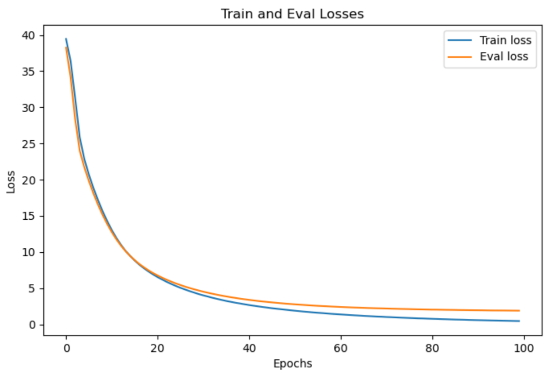

Figure 2: *BiGRU model with both CRF and Dropout enabled*

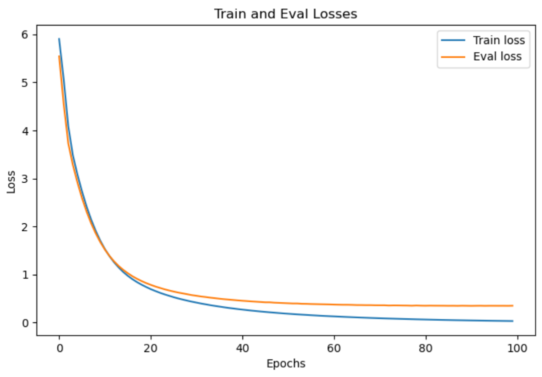

Figure 3: *BiGRU model with both CRF and Dropout disabled*

best for Intent Accuracy with 95.3, while for the slot it is 94.6 still with JointBert but with the CRF layer enabled.

### 5.3. Multi Runs results

The results of each test are visible here:

- Table44 shows the results of the multiple runs using the BiLSTM model with all possible configuration,
- Table55 shows the results of the multiple runs using the BiGRU model with all possible configuration,

As stated before the multiple runs where not computed for the JointBert model, since just a single run with 3 to 5 epochs would take more than 5 hours of training.

For this tests I changed two parameters to make all runs more similiar to each other to better compute the standard deviation. Early stopping was removed entirely and the number of epochs for each run was set at 100. The results here confirmed what already discussed in the single run results, and they explain definitely that the low accuracy for that one configuration of GRU was simply an outlier. We can reaffirm that adding the CRF layer improves by a good marging the performance on the Slot task and dropout helps to avoid overfit.

### 5.4. Extra considerations

When looking at the evolution of the difference in the loss of Validation and Training Set across epochs we can see that there is no particular case of overfit in any case. The examples in figure 2 and figure 3 are taken from the BiGRU model with the two polar opposite configuration.

## 6. Conclusion

The objective of this project was achieved even when using rather simple models on both the ATIS and SNIPS Datasets. The best results were obtained with a more complex model witch make use of the ability to better understand the context of a sentence of transformers. The most effective addition to the aforementioned models was the introduction of a CRF layer on top of them, which improved by a good margin the performance in the Slot task. The results obtained can easily be further improved by increasing the number of epochs during the training phase. Regarding the performance on the ATIS dataset they can also be further improved with adding examples of the missing labels. The same thing can be said for the SNIPS dataset, by increasing the number of examples for the less represented labels. The code alongside the logs of each run can be found on my github project folder NLU_Project_232090.

## 7. References

[1] Q. Chen, Z. Zhuo, and W. Wang, "BERT for joint intent classification and slot filling," *CoRR*, vol. abs/1902.10909, 2019. [Online]. Available: http://arxiv.org/abs/1902.10909