# Final NLU project

*Alberto Casagrande (229362)*

University of Trento

alberto.casagrande@studenti.unitn.it

## Abstract

This document is the report for the final project of the Natural Language Understanding course. The project consists of developing a neural network that predicts intents and slots in a multi-task learning setting. The goal was to improve the performances of a given baseline model on two different datasets: ATIS and SNIPS.

## 1. Introduction

Slot filling and intent detection play important roles in Natural Language Understanding (NLU) systems, in which the main goal is to identify the intent and extract semantic constituents from the utterance.

As a baseline, the model used is simply the one used during Lab 10, which consists in the implementation of an LSTM that encodes the input sentence and uses two linear layers to predict the slots and the intents. The second model is just a slightly modified version of the baseline model, in which instead of using a unidirectional LSTM it is made use of a bidirectional GRU (Gated Recurrent Unit) which is able to understand the context better. Moreover, I played with the network parameters, finding 256 as the optimal hidden size value.

The third and final model is the implementation of the paper *A Bi-model based RNN Semantic Frame Parsing Model for Intent Detection and Slot Filling* [1]. It consists of a bi-model designed to perform the intent detection and slot filling tasks jointly, by considering their cross-impact to each other using two correlated bidirectional LSTMs (BLSTM). The model is implemented using two encoders and two decoders, each specific to a task.

## 2. Task Formalisation

The goal of the project is to implement a neural network that, given a sentence, predicts intents and slots simultaneously, so that the performance of the resulting model benefits from the Multi-Task Learning framework. In this way the model obtains shared parameters and features between two tasks and implements joint optimization using a weighted loss function. The problem is therefore treated as a multitask learning problem, where these two tasks are tackled simultaneously trying to exploit the interactions and information sharing between them [2]. In particular, **intent classification** aims to identify speaker's intent from a given utterance (works on the utterance-level) and it is modeled as a classification problem, while **slot filling** annotates the utterance on a word-level, indicating the slot type mentioned by a certain word. Slot filling is modeled as a sequence labeling problem. The two tasks are used to obtain a structured representation of the meaning of the utterance, so that it can be processed by a computer.

> Example
> *Utterance*: "show me flights from new york to miami"
> *Slots*: O O O O B-fromloc I-fromloc O B-toloc
> *Intent*: flight

The ultimate aim of the project is to implement and evaluate two models that perform better than a given baseline. The baseline performances are as follows:

| Dataset | Slot F1 score | Intent accuracy |
|---|---|---|
| ATIS | 92% | 94% |
| SNIPS | 80% | 96% |

## 3. Data Description & Analysis

For the purposes of the project, two different datasets were used, ATIS and SNIPS. Both datasets present data in JSON format and the data scheme is the following:

```
[
  {
    "utterance": "on april first i need a flight going from
        phoenix to san diego",
    "slots": "O B-depart_date.month_name B-depart_date.
        day_number O O O O O O B-fromloc.city_name O
        B-toloc.city_name I-toloc.city_name",
    "intent": "flight"
   },
   "..."
]
```

Basically, the data scheme is an array of dictionaries, where each dictionary is an element of the dataset.

### 3.1. ATIS

The ATIS dataset (Airline Travel Information Systems) is composed of 4978 samples in the training set and 893 samples in the test set. The dataset contains audio recordings of people making flight reservations. In the original split the development set is missing and therefore I have used a specific function to create a dev set starting from the training set. The size of the dev set is the 10% of the dataset. In our case, we do a stratified sampling from the training set by stratifying on intents. In this way we end up having 4381 samples in the training set, 597 in the dev set and 893 in the test set. The total number of intents is 26, while the number of slot labels is 129. The vocabulary length is 941.

As for the distribution of intents, the dataset is highly unbalanced, with most queries labeled as "flight". Looking at the intent labels, we can see that some sentences have multiple intents associated with them, such as `airline+flight_no`. Slots are highly unbalanced as well, with most of the slots labeled as `fromloc.city_name` and `toloc.city_name`.

Figure 1: *Intents distribution ATIS*

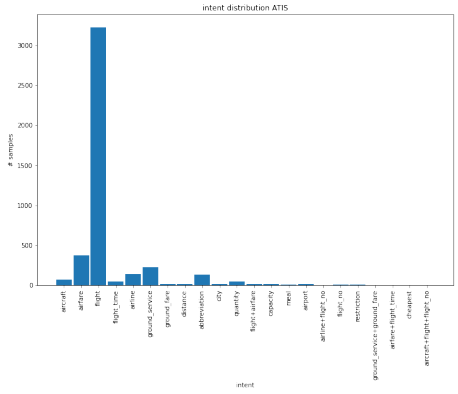
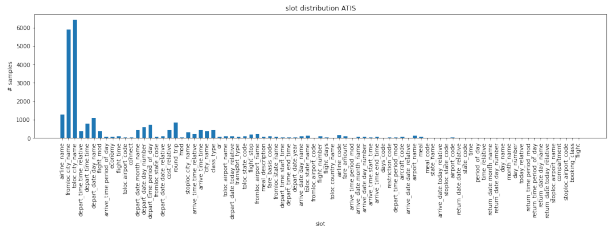


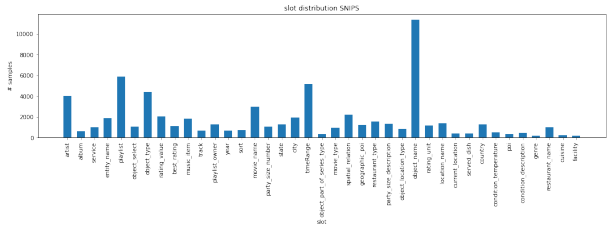

Figure 2: *Slots distribution ATIS*




### 3.2. SNIPS

SNIPS dataset is composed of 13084 samples in the training set, 700 samples in the dev set and 700 samples in the test set. The number of intents is 7, while the slot labels are 72. In terms of intents, the training data are quite balanced, as there are almost equal data points per class. Regarding the slots, apart from the 'O' tag which is the one that occurred most frequently (not considered in the graph), the most frequent tags are `object_name`, `playlist` and `timeRange`. Compared with the ATIS dataset, the SNIPS dataset is more complex due to its large vocabulary and cross-domain intents. The vocabulary length is 12134.

Figure 3: *Slots distribution SNIPS*




## 4. Model

As mentioned earlier, the baseline and the second model follow the same philosophy. The main change made to the baseline lies in the use of bidirectional GRU and a different value of the hidden size. The third model, on the other hand, is totally different in that the network consists of two models, each of which deals with one task and consists of an encoder and a decoder. In all three models, *Adam* was used as the optimizer, and *Cross entropy loss* was used as the loss function for both tasks.

### 4.1. Baseline model

The baseline model is very simple in that it consists of an encoder implemented using a unidirectional LSTM. It takes as input the embedding of the utterance and outputs the encoded utterance and the final hidden state for each element in the sequence. Finally, two linear layers are used to perform intent classification and slot filling. In the case of intent classification, the corresponding linear layer takes the last hidden state as input, while for slot filling the output of the LSTM (`utt_encoded`) is provided as input to the related linear layer. Both loss functions are calculated using the *Cross entropy loss*, and the total loss is defined as the sum of the two losses.

$$\mathcal{L} = \mathcal{L}_{intent} + \mathcal{L}_{slot}$$

### 4.2. Second model

The second model is simply a slightly improved version of the baseline model in which the encoder is implemented as a **bidirectional GRU** (Gated Recurrent Unit). The GRU is a type of Recurrent Neural Network (RNN) that uses the same philosophy as LSTM. It controls the flow of information like the LSTM unit, but without having to use a memory unit. Since there are only two gates (the *update gate* and the *reset gate*), it is computationally more efficient as it uses less training parameters. The overall structure of the model is composed of an embedding layer, a bidirectional GRU as an encoder and finally two linear layers for slot filling and intent classification respectively.

An attempt was also made to improve the loss function, since in the previous approach, to simultaneously learn multiple tasks it was used a naive weighted sum of losses, where the loss weights are uniform. The new loss function is represented by the following formula:

$$\mathcal{L} = \frac{1}{2\sigma_1^2}\mathcal{L}_{intent}(W) + \frac{1}{2\sigma_2^2}\mathcal{L}_{slot}(W) + log(\sigma_1) + log(\sigma_2)$$

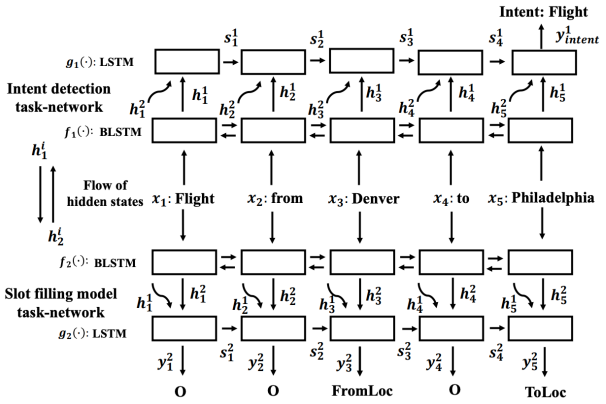
It implements a sort of *dynamic loss weight learning* [4], where the idea is that the performance of multi-task learning systems is strongly dependent on the relative weighting between each task's loss. This means that weightings are task-dependent and this multi-task loss function can learn to balance these weightings optimally. Using the new loss function, the performances were very similar compared to the simple sum of the losses, so I decided to keep the latter.

Finally, I played with the network parameters, finding 256 as the optimal hidden size value.

### 4.3. Third model

The third model is nothing more than the implementation of the paper *A Bi-model based RNN Semantic Frame Parsing Model for Intent Detection and Slot Filling* [1]. It is based on an unofficial implementation on GitHub [3]. The overall structure is shown in figure 4. It is quite complex, as it is composed by two models - one for intent detection and the other for slot filling. Each task-network includes one BLSTM with an LSTM decoder. The basic idea of the network is to take the cross-impact between the two tasks into account. For this purpose, the two bidirectional LSTMs (BLSTMs) are

Figure 4: *bi-model structure*




inter-connected. Each BLSTM $f_i(\cdot)$ reads the input utterance sequences forward and backward and generates a sequence of hidden states $(h_1^i, h_2^i, \cdots, h_n^i)$ where $i = 1$ corresponds to the network for intent classification task and $i = 2$ is for the slot filling task. The peculiarity lies in the fact that, in order to detect intent, the hidden state $h_t^1$ is combined together with $h_t^2$ from the other BLSTM $f_2(\cdot)$ in slot filling task-network to generate the state $s_t^1$.
For the slot filling task, a similar network structure is considered. The biggest difference is that we have an output at each time step *t*.

The two task-networks are trained based on their own cost function in an asynchronous manner. Both loss functions $\mathcal{L}_{intent}$ and $\mathcal{L}_{slot}$ are defined using cross entropy. This has the advantage of filtering the negative effects between the two tasks compared with using only one joint model. Furthermore, each of the two models is specialized for a single task and the cross-impact between the two tasks can be learned by sharing hidden states of the two models.

The learning rate adopted is different for the two datasets. In the case of ATIS the learning rate was set to 0.001 while in the case of SNIPS it was set to 0.00025. This diversification was necessary judging by the behavior of the loss on the dev set. In particular, the learning rate used for ATIS, in case of SNIPS dataset, caused the model to converge too quickly to a suboptimal solution.

# 5. Evaluation

The main metrics that have been used for performance evaluation and analysis are F1 score for slot filling task and classification accuracy for intent detection task. The F1 score is calculated from the precision and recall of the test, where the precision is the number of true positive results divided by the number of all positive results, including those not identified correctly, and the recall is the number of true positive results divided by the number of all samples that should have been identified as positive. It is computed as:

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

The accuracy describes how the model performs across all classes. It is calculated as the ratio between the number of correct predictions to the total number of predictions.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

In addition to these metrics, for a more accurate analysis, confusion matrices and classification reports were used. The confusion matrix is a tabular summary of the number of correct and incorrect predictions made by a classifier, while the classification report displays the precision, recall, F1, and support scores for the model.

## 5.1. Results

Regarding the baseline model and the second model, they were trained and tested 5 times from scratch and the mean and variance for Slot F1 score and Intent accuracy were calculated. The third model, on the other hand, was trained only once, as a single training can take up to 30 minutes, being a much more complex model.

### *5.1.1. Performances on the ATIS dataset*

| | **Baseline** | **Second model** | **Bi-model** |
|---|---|---|---|
| **Slot F1** | 92% | 94% | 94% |
| **Intent accuracy** | 94% | 96% | 96% |

Table 1: *Performances of the models on the ATIS dataset*

The baseline model was able to obtain a Slot F1 score of 92% and an intent accuracy of 94%.
In terms of intents, as can be seen from the classification report on the jupyter notebook, the biggest issues are related to labels made up of two intents (such as `flight+airfare`), as well as `meal`, `city` and `flight_no`. In particular, 6 examples belonging to `flight_no` are incorrectly classified as `flight`, as well as 3 examples belonging to `city` and 7 examples belonging to `flight+airfare`. The best classified slots are `fromloc` and `toloc`, which are the dominant ones in the dataset. The worst classified are those that include a numeric value.

The second model allows to improve the performance of the baseline by 2 percentage points in both tasks, achieving a Slot F1 of 94% and intent accuracy of 96%. Specifically, taking a look at the classification report and the confusion matrix of intents, we can clearly see how the second model improves the classification of some intents, such as `city` and `flight_no`. The latter now includes 7 examples correctly classified out of 8. However, it is also interesting to note that `distance` intent is now classified worse, as 4 out of 10 examples are classified as `flight`. The Slot F1 score also had a 2% improvement, which is due to the use of the bidirectional GRU that can produce a more meaningful output, combining GRU layers from both directions and thus taking into account information from both the past and present.

The third model shows absolutely comparable performance to the second model on the ATIS dataset. It achieved a Slot F1 score of 94% and an intent accuracy of 96%. Compared with the baseline, fewer misclassifications are made on some

intents, including `flight_no`, `meal` and `ground_fare`.

Despite these improvements, some classification errors are due to the structure of the dataset itself, in whose test set there are some intents that are not present in the training set (*airfare+flight, flight_no+airline, day_name, flight+airline*).

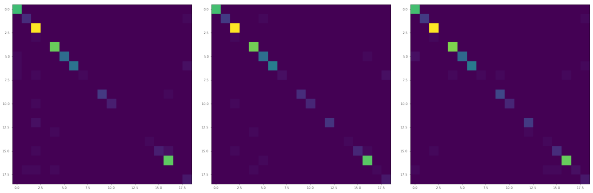


Figure 5: *from left to right: ATIS - intent classification with baseline model, second model and bi-model*

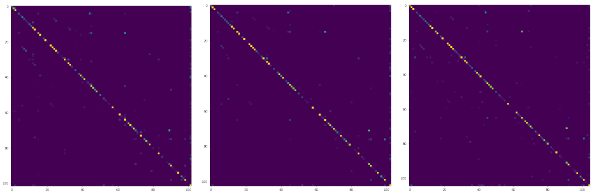


Figure 6: *from left to right: ATIS - slot filling with baseline model, second model and bi-model*

### *5.1.2. Performances on the SNIPS dataset*

| | **Baseline** | **Second model** | **Bi-model** |
|---|---|---|---|
| **Slot F1** | 80% | 86% | 88% |
| **Intent accuracy** | 96% | 97% | 97% |

Table 2: *Performances of the models on the SNIPS dataset*

The baseline model achieves poor performance on Slot F1 (80%), while it manages to achieve an intent accuracy of 96%. All intents are classified well, the only ones showing worse performance are `SearchScreeningEvent` and `SearchCreativeWork`. The latter is often confused for `PlayMusic` and both are sometimes confused for each other. As for slots, the worst performances are with those that include the name of objects, restaurants, entities as well as `city`, `track` and `album`.

The second model allowed to have a massive improvement on the Slot F1 score (86%), while the intent accuracy remained substantially unchanged. Considering the intents, the classification of `SearchCreativeWork` has improved slightly at the expense of a worse classification of `SearchScreeningEvent` (7 examples are now classified as belonging to the `SearchCreativeWork` class). Except for the `track` and `album` slots, all slots are predicted considerably better. In particular, we can see that the performance on the `city`, `country`, and `genre` slots has improved significantly in the F1 score.

The third model achieves even better performance in the Slot F1 score, where it improves the baseline by 8%. The intent accuracy, on the other hand, has improved by only 1 percentage point. However, these results are not highly accurate because the model, unlike the baseline and the second model, was trained only once due to time requirements. In terms of intent classification, the differences with the baseline are minimal. Again, the only slots that did not show improvement in performance are `album` and `track`. Almost all other slots are predicted better than the baseline.

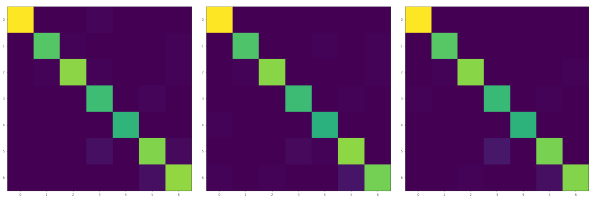


Figure 7: *from left to right: SNIPS - intent classification with baseline model, second model and bi-model*

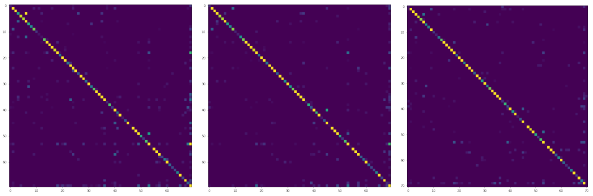


Figure 8: *from left to right: SNIPS - slot filling with baseline model, second model and bi-model*

## 6. Conclusion

This report presents neural network models able to outperform a certain baseline in intent classification and slot filling tasks.

The model taken as a baseline consists simply of a unidirectional LSTM (encoder) and two linear layers for the classification of intents and slots.

The second model improves the baseline model by implementing the encoder by means of a bidirectional GRU. The goal of this model is essentially to demonstrate how baseline performance can be improved simply by making a few changes to the network.

Finally, a different model is proposed in terms of philosophy. It consists of two sub-structures, each of which is composed of an encoder and a decoder. It is more complex than the previous two and requires the update of many more parameters. Encoders are implemented through BLSTMs, while decoders are LSTMs. Despite the fabulous performances on the ATIS dataset claimed in the original paper, the real performances measured in reality are lower.
In addition, as described in the original paper, the embedding size and hidden size were defined at the beginning of the experiment on an empirical basis, while the number of hidden layers is chosen as 2, based on the size of the dataset. This means that the model was somehow designed to fit well with the dataset used (ATIS) and is not guaranteed to generalize to other settings.

# 7. References


[1] W. Yu S. Yilin and J. Hongxia, "A Bi-Model Based RNN Semantic Frame Parsing Model for Intent Detection and Slot Filling," https://arxiv.org/abs/1812.10235

[2] T. Gong, L. Tyler, S. Cory, R. Venkata, P. Suchismita, N. Anthony, K. Gokce and E. Oguz H., "A Comparison of Loss Weighting Strategies for Multi task Learning in Deep Neural Networks," *IEEE Access*, vol. 7, pp. 141627-141632, 2019.

[3] *GitHub page Bi-Model-Intent-And-Slot*. https://github.com/ray075hl/Bi-Model-Intent-And-Slot

[4] A. Kendall, Y. Gal, and R. Cipolla, "Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7482-7491, 2018 https://arxiv.org/pdf/1705.07115v3.pdf