Final NLU project: Joint intent classification and slot filling

Nicola Farina (229296)

University of Trento

nicola.farina@studenti.unitn.it

1. Introduction

This report presents three neural network architectures that I have trained to jointly solve the tasks of intent classification and slot filling on the *ATIS* and *SNIPS* datasets. The first model acts as a baseline. The second model is an implementation of the architecture introduced in [1], which applies a NLU mechanism on top of BERT to also use slot information to classify intent (and vice-versa). The third model is an extension of the first model, with some changes in the architecture, an alternative version of the NLU mechanism of the second model, and CRF for slot filling. The second and third models both successfully outperform the baseline.

2. Task Formalisation

The project consists in solving two tasks: *intent classification* and *slot filling*. **Intent classification** is a text classification problem where the goal is to estimate the intent of an input sequence. More formally:

- given a sequence of tokens $w = w_1, w_2, ..., w_n$,
- and a set of intent labels I where $i \in I$,
- estimate the label $\hat{i}$ such that $\hat{i} = \underset{i}{\operatorname{argmax}}\, P(i|w)$.

Slot filling is a sequence labelling task that assigns a label to each token in the input query in order to find meaningful *slots* that help in explaining the query. More formally:

- given a sequence of tokens $w = w_1, w_2, ..., w_n$,
- defining a sequence of slot labels as $l = l_1, l_2, ..., l_n$,
- compute the sequence $\hat{l}$ such that $\hat{l} = \underset{l}{\operatorname{argmax}}\, P(l|w)$.

An example of user query and relative outputs, in the format used in this project, can be seen in Table 1. It also shows that, in the context of this project, slot filling is done together with segmentation using IOB tags. The main objective is to solve both of the aforementioned tasks in a **multi-task learning** setting, that is, when multiple related tasks are solved at the same time in order to exploit their similarities and gain better generalization [2].

3. Data Description Analysis

The datasets used in this project are *ATIS* and *SNIPS*, both frequently used benchmarks in the state-of-the-art.

Query	ground transportation in san jose	
Output	**Intent**	ground_service
	Slots	O O O B-city_name I-city_name

Table 1: *Example from ATIS dataset*

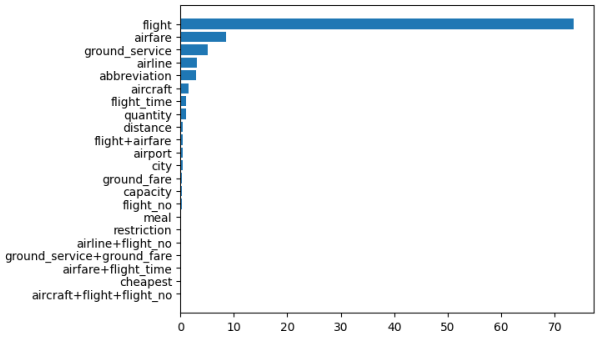


Figure 1: *Intent distribution (%) of the ATIS training set*

3.1. ATIS

The ATIS (*Airline Travel Information Systems*) dataset consists of transcripts of humans asking for information on automated airline travel inquiry systems. It is a small dataset, containing 5871 sentences in total, split into training (4978) and test (893). For development purposes, I split the training set into a smaller training set (4381) and a validation set (597). Its vocabulary consists of 861 words, 26 intents and 129 slots. It is not a balanced dataset: the intent *flight* takes about 75% of the whole intents distribution, while the slot *O* takes about 63% of the slots distribution. Furthermore, given its small size, there are several labels with very little support (as low as 1). This also results in the fact that some labels are present in the test split but not in the training split (*day_name*, *flight+airline*, *flight_no+airline*, *airfare+flight* for the intents; *I-state_name*, *B-booking_class*, *I-flight_number*, *B-flight*, *B-compartment*, *B-stoploc.airport_code* for the slots). Figure 1 shows the intent distribution of the training set, which is very similar to that of the other splits. I considered building a more balanced version of this dataset by performing a new train/validation/test split, but I decided against it because I wanted to improve the baseline over the same dataset.

3.2. SNIPS

The SNIPS dataset consists of 14484 queries, split into training (13084), validation (700) and test (700). It spans 7 unique intents related to various domains (like restaurants, music and weather), it has 72 unique slots, and has a vocabulary size of 11418. Its intent distribution is very balanced, and all intents and slots appear at least once in every split. Figure 2 shows the intent distribution of the training set.

4. Model

In this section, I firstly introduce the settings that are shared by all three models that I implemented. Then, I present each model in detail: the **Baseline**, the *BERT + Bi-directional NLU mechanism* (**BERT+BiNLU**), and the *Bi-directional GRU + Bi-*

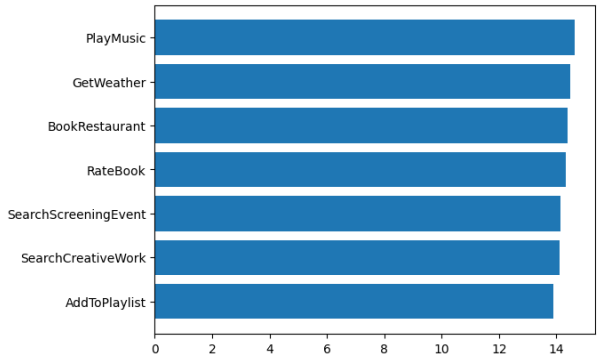


Figure 2: *Intent distribution (%) of the SNIPS training set*

directional NLU mechanism + CRF (**BiGRU+BiNLU+CRF**).

For all models except *BiGRU+BiNLU+CRF*, I use the Cross-Entropy Loss for both intent classification and slot filling, one of the most widely used loss functions for classification tasks. The losses are combined with a weighted sum:

$$\text{Total loss} = \alpha \times \text{Intent loss} + (1 - \alpha) \times \text{Slot loss} \quad (1)$$

In all models except *BiGRU+BiNLU+CRF*, I set $\alpha = 0.5$, giving equal importance to both tasks.

The pre-processing pipeline is limited by the datasets, which have target slots aligned to white-space tokenization of the corresponding word sequence. This simplified the preparation of the input for *Baseline* and *BiGRU+BiNLU+CRF*, but resulted in additional steps when classifying slots for *BERT+BiNLU*.

4.1. Baseline

The baseline model consists in an embedding layer, followed by a uni-directional, single-layer LSTM network, which feeds into two separate linear classifiers, one for each task. The input of the slot classifier are the hidden states of each token as generated by the LSTM network, while the input of the intent classifier is the final hidden state.

4.2. BERT+BiNLU

This is an implementation of the architecture introduced in [1]. The first module of this model consists in a pretrained BERT model (specifically *bert-base-uncased* from Hugging Face). The bi-directional NLU mechanism uses the output of BERT and consists in two models:

- **Intent2Slot**: it first computes intent logits from the hidden representation of the *[CLS]* token output by BERT (a special token introduced for classifying the input sequence), using a linear classifier. It then concatenates these logits to the hidden state of each token output by BERT (minus the *[CLS]* token), forming the input for N different linear classifiers that predict N slots (where N is the number of tokens for which we are predicting a slot).
- **Slot2Intent**: it first uses N different linear classifiers to get slot logits for each token for which we are predicting a slot. It then concatenates the output of each of these layers to the hidden state of the *[CLS]* token, forming the input to another linear classifier that predicts the intent of the input sequence.

BERT was trained using a special kind of tokenizer, which may introduce sub-word tokens and also introduces the [CLS] and [SEP] tokens. This affects the slot filling task, since the ATIS and SNIPS datasets use a different kind of tokenization. For this reason, for slot classification, I ignore the [CLS] and [SEP] tokens, and only use the first sub-word token in case a word is split into multiple tokens.

4.3. BiGRU+BiNLU+CRF

I implemented this model to improve the baseline without relying on a pre-trained backbone such as BERT. This allowed me to make faster experiments (since BERT is a big and complex model), thus I could fine-tune hyperparameters more easily.

First of all, I increased the embedding size from 300 to 400 and the hidden size from 200 to 600, after trying different values. I then used a bi-directional GRU network with two layers to encode the embeddings. I chose GRU over LSTM because GRU networks are simpler and have less parameters, which means they are more efficient and possibly more suited to the smaller datasets I am working with. The number of layers is a result of trade-off between model performance and complexity. On top of the GRU module, I added the *Intent2Slot* and *Slot2Intent* models described in Section 4.2, with the change that I do not train N different linear classifiers for each token, but only one, to reduce complexity. I chose $\alpha = 0.4$ in Equation 1, giving more importance to the task of slot filling, which is harder than intent classification. Finally, I added a ready-made CRF layer for the slot filling task. CRF is commonly used, for example, for *Named Entity Recognition*, which poses some similarities to the slot filling task (classification of each token). Due to this CRF layer, the loss function I used for the slot filling task is the negative log likelihood.

5. Evaluation

In this section, I introduce the metrics used to evaluate the models, followed by the training procedure. I then present the results of the evaluation of each model, comparing them and analyzing their errors.

5.1. Metrics

Like in most state-of-the-art benchmarks for intent classification and slot filling, I used a different metric for each task. For intent classification, the chosen metric is *accuracy*, which is defined as follows:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} \quad (2)$$

For slot filling, the metric used is the *F1 score*, which combines the *precision* and *recall* scores of a model. *Precision* is defined as the ratio between the number of true positives and the sum of true positives and false positives:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \quad (3)$$

while *recall* is defined as the ration between the number of true positives and the sum of true positives and false negatives:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (4)$$

F1 score is then defined as follows:

$$\text{F1 score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (5)$$

	ATIS		SNIPS	
	Intent Accuracy	Slot F1 score	Intent Accuracy	Slot F1 score
Baseline	$94.1 \pm 0.2\%$	$92.6 \pm 0.3\%$	$96.4 \pm 0.5\%$	$80.1 \pm 0.6\%$
BERT+BiNLU	96.8%	92.8%	98.4%	95%
BiGRU+BiNLU+CRF	$95.3 \pm 0.6\%$	$94.1 \pm 0.9\%$	$96.3 \pm 0.9\%$	$87.4 \pm 2.9\%$

Table 2: *Evaluation results*

5.2. Training procedure

For all models, I used an initial learning rate of 10^{-4}, adapted during training with the Adam optimizer. I used early stopping based on the slot F1 score, with a patience of 5, for further regularization. Since the datasets are small in size, for each model except *BERT+BiNLU*, I ran some individual experiments (3 for *BiGRU+BiNLU+CRF* due to long training times, 5 for the rest) for a maximum of 200 epochs (100 for *BiGRU+BiNLU+CRF*), averaging the results of each one. I trained the BERT-based model only once and for only a maximum of 50 epochs due to its complex architecture, which resulted in very long training times.

5.3. Results

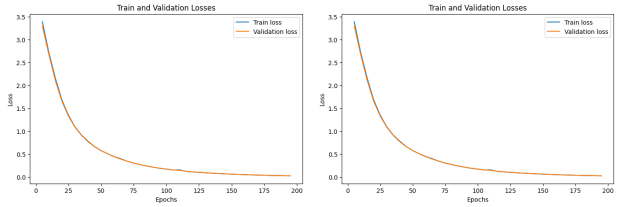
Intent accuracy and slot F1 score for each dataset and model can be seen in Table 2. For the models that have been trained more than once, I report the mean and the standard deviation. I will now cover the results of each model in detail.

5.3.1. Baseline

Looking at the loss graph in Figure 3, it is clear that the model converges quickly on both datasets, although a bit slower on SNIPS. The fact that the losses are close to zero but the performance is still not very good indicates a problem of overfitting. This is possible even if the training and validation losses are both small, in particular when the training set is not particularly representative of the test set, which is the case (especially for ATIS).

For intent classification, both ATIS and SNIPS have relatively high scores. In ATIS, the accuracy is brought down by some slots that are in the test set but not in the training set, as explained in Section 3.1. The most common errors in ATIS are clearly associated to the most common intents (i.e. *flight*), while for SNIPS, as can be seen in Figure 4, the errors come mostly from the model predicting *PlayMusic* instead of *SearchCreativeWork* (music is creative work) and predicting *SearchCreativeWork* instead of *SearchScreeningEvent* (both involve searching).

For slot filling, the discrepancy between ATIS and SNIPS F1 score is relevant. I believe this is, again, due to the imbalance



(a) *ATIS* (b) *SNIPS*

Figure 3: *Loss over time for Baseline model*

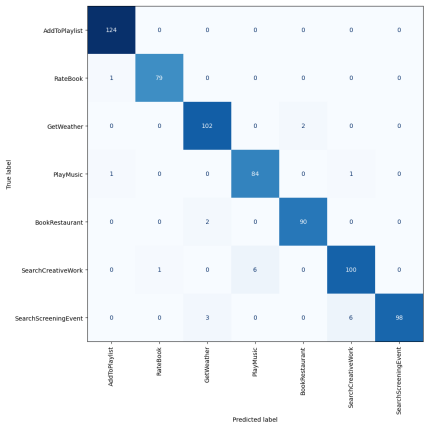


Figure 4: *Baseline intent confusion matrix on SNIPS*

of the label distribution in the ATIS dataset.

In both datasets, the most common prediction errors belong to the *O* slot, which is also the most prevalent slot. Other sources of errors are, for example, related to similar slots, like predicting *B-city* instead *B-geographic_poi* or mixing *B-album* with *B-track* (SNIPS).

5.3.2. BERT+BiNLU

This model has a clear improvement of 2% on intent classification on both ATIS and SNIPS. This is not surprising, since using BERT as a pre-trained backbone for sentence-level classification works really well (the special [CLS] token has this exact purpose). For ATIS, the model is not confused anymore between *flight* and *flight_no* or *city*, but it still struggles to classify "composite" intents, such as 9 examples of *flight+airfare* being classified as *flight*. For SNIPS, it halves the errors between *SearchCreativeWork* and *PlayMusic*, but it still struggles with *SearchCreativeWork* and *SearchScreeningEvent*.

The slot F1 score remains almost unchanged from the baseline model on ATIS, but significantly improves by 15% on SNIPS. To explain the lack of improvement on ATIS, on which the model makes very similar errors as the baseline model, I would say that the intent offers much less information than in the SNIPS dataset, since all ATIS sentences come from a single domain, while SNIPS covers varying domains. This would render the NLU mechanism less useful for slot filling. In fact, ATIS has more slots than SNIPS, and by having only one domain, most of these slots will overlap in some ways. Most likely, though, these errors also come from a sub-optimal training. Figure 5 shows an overfitting problem, since the validation loss is much higher than the training loss. This could be confirmed by tuning some hyperparameters, like increasing the weight for the slot filling loss or applying more regularization (e.g. increase dropout probabilities). Due to the time-consuming training of this model, I did not have time to fine-tune the training pro-

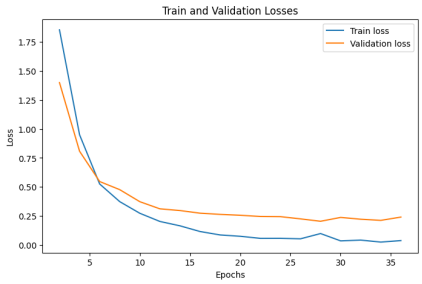


Figure 5: *Loss for BERT model on ATIS*

cess. As for the massive improvement in slot filling on SNIPS, a probable cause would be the much better embeddings and hidden states of each token, which have a bigger impact on SNIPS due to its larger vocabulary size over ATIS. Looking into the results more in detail, there is much less confusion between *B-object_name* and *O*, even though there still are some errors between *B-object_name* and *I-object_name*, showing that the IOB tagging further complicates this task.

In the end, as already hinted, I suspect that most of the heavy-lifting is done by the encoding of the input sequence via BERT, which gives much more informative hidden representations than an embedding learned from scratch over a tiny dataset.

5.3.3. BiGRU+BiNLU+CRF

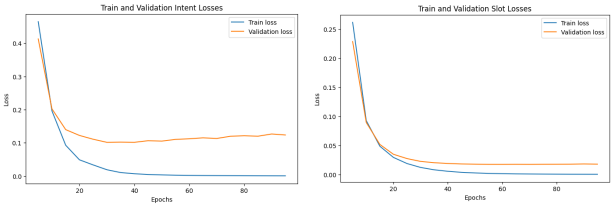
On ATIS, this model makes a good improvement both on intent accuracy and slot F1 score, actually performing the best out of all models on the latter. Figure 6 shows the two separate losses on ATIS, one for each task, before they are combined together to train the model. It is clear that there is a problem of overfit on the slot classification, which is less pronounced (but still present) in the intent classification. This might be due to the CRF loss combined with the weight $\alpha = 0.6$ assigning more importance to the slot loss. The kind of errors made by the model are, again, similar to those of the baseline, just in fewer quantities.

On SNIPS, this model improves on the baseline only for the slot F1 score, and not by a lot. More than on ATIS, this model clearly has an overfitting problem, as shown in Figure 7 by the validation losses not converging. This instability also explains the high standard deviation.

I was expecting the intent accuracy on SNIPS to increase with the *Slot2Intent* module: since the slots are spread across multiple domains, and they are more separated than in ATIS (meaning that they overlap less between different intents), I thought that augmenting intent prediction with slots information would bring a bigger benefit. I assume this is a problem with how I trained or implemented the model, as the authors in [1] also share this observation in their results.

6. Conclusion

I have successfully reached the goal of implementing two neural network models that are able to outperform a simple baseline for joint intent classification and slot filling. The implementation of [1] makes use of a very performant backbone such as BERT and introduces a bi-directional NLU mechanism that uses slots to inform intent prediction and vice-versa. It allowed me to better understand the role of tokenization and how to han-



(a) *Intent* (b) *Slot*

Figure 6: *Losses for BiGRU+BiNLU model*

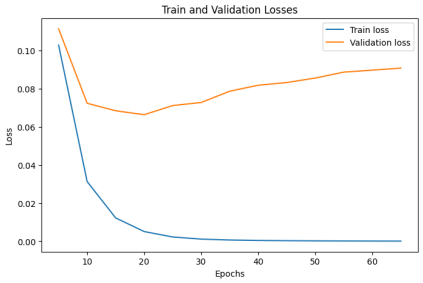


Figure 7: *Loss for BiGRU+BiNLU model on SNIPS*

dle situations where labels are tokenized in a different way than the sequence encoder expects. I then borrowed their idea and used it together with a GRU network and a CRF layer in order to learn a smaller model that I would be able to fine-tune for experimentation. While my model did not reach the same performance of the BERT-based model, it managed to improve on the slot filling score for ATIS and allowed me to experiment with different hyperparameters and see their effect. The results also clearly show the importance of having a properly balanced and big dataset, otherwise you can expect several problems, like overfitting and poor performance over certain classes.

7. References

[1] S. C. Han, S. Long, H. Li, H. Weld, and J. Poon, "Bi-directional joint neural networks for intent classification and slot filling," *arXiv preprint arXiv:2202.13079*, 2022.

[2] S. Ruder, "An overview of multi-task learning in deep neural networks," 2017, https://www.ruder.io/multi-task/ [Accessed: 22/09/2023].