Final NLU project

Michele Yin (229359)

University of Trento

michele.yin@studenti.unitn.it

1. Introduction

Here are presented two neural network based models on the joint task of slot filling and intent classification. The dataset tested are ATIS and SNIPS.

The first model is based on a bidirectional LSTM. The second model consists of fine tuning a pretrained BERT model as backbone. Both models use random loss weighting and focal loss to get slightly better results.

Although both are an improvement with respect to a simple baseline model, neither is able to achieve results close to the current state of the art on ATIS and SNIPS for joint slot filling and intent classification.

2. Task Formalisation

The task consists of two separate, but related tasks, *Slot filling* and *Intent classification*:

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

  An example from SNIPS dataset is the following:

Input sequence:	find	fish	story
Output sequence:	O	B-movie_name	I-movie_name

- *Intent classification* defined as follows:
  - given a sequence of tokens $w = w_1, w_2, ..., w_n$,
  - defined a set of labels L where $l \in L$
  - compute the label $\hat{l}$ such as $\hat{l} = \underset{l}{\operatorname{argmax}}\, P(l|w_1, ..., w_n)$

  For the same example from SNIPS dataset is the following:

Input sequence:	find	fish	story
Output label:	SearchScreeningEvent		

It is known that joint training on both tasks at the same time yields better results than separate task specific training. This is probably because the two tasks are related and joint multi-task learning leverages this relation, allowing models to generalize more and thus performing better.

3. Data Description Analysis

The datasets used for training and testing are ATIS and SNIPS. Both are small datasets that are specifically designed for this purpose. Both share the same general structure and are made of examples as shown in Listing 1.

Both dataset were initially divided in only training and test sets. The dev set was obtained from a part of the training set using the function *train_test_split* from sklearn, This is to enforce statistical consistency of the different labels in training and development sets.

```
[
 {
  "utterance": "find movie times",
  "slots": "O B-object_type
     I-object_type",
  "intent": "SearchScreeningEvent"
 },...
]
```

Listing 1: *JSON example on SNIPS*

3.1. ATIS

ATIS (Airline Travel Information System) is specifically designed for airline related tasks, such as flight requests and ticket purchases. For this reason, the dataset is very task specific. As a side effect the vocabulary used is very small, including only 853 unique tokens.

The dataset is composed by a total of 5871 elements, that are partitioned in sizes of 4381, 597, 893 for training, development and test sets respectively. There are 130 different slots labels, including the filler 'O', and 26 intents labels.

These labels are unevenly distributed. Around 73% of the intents are 'flight' and 63% of the slots are the tag 'O' whereas all other intents and slots occur in very few instances.

For example the intent 'cheapest' and the slot 'I-time' occur only once in the training set. This means that models are very unlikely to predict these labels at the test phase.

There are also some labels that do not appear at all in either train or dev set but do occur in the test set, such as 'flight+airline', 'day_name', 'flight_no+airline', 'airfare+flight'. Similarly for slot labels, 'B-booking_class', 'B-compartment', 'B-flight', 'B-stoploc.airport_code', 'I-flight_number', 'I-state_name' all appear in the test set but not in the training set or dev set.

This creates an upper limit to the performances of any model with this partitioning, unless a data augmentation or re-partitioning of the train, dev and test sets is done. A new partitioning of the dataset that considers statistical consistency of the labels for train, dev and test sets was also tested, with a significant improvement across all models with respect to the default partitioning.

3.2. SNIPS

SNIPS is a slightly bigger and more balanced dataset. Its entries are related to tasks such as playing music or booking a hotel, typical for voice assistants. It has 13784 entries, divided in 11644, 1440, 700 for training, dev and test set respectively.

The vocabulary used is much wider, being composed of 10638 unique words. At the same time, there are fewer labels, 73 for slots, including 'O', and 7 for intents.

Intents are almost equally distributed and while the slot label 'O' composes 49% of the slots. Unlike ATIS, there are no slot or intent labels that appear in the test set but not in train or dev set.

Overall SNIPS should make for a better dataset for training, given the bigger size, larger vocabulary and fewer number of labels. It has to be noted that because ATIS is so small and unbalanced, a model may perform decently by just predicting 'flight' for every sentence. SNIPS does not present the same problem.

Lastly, both datasets have a disproportionate amount of 'O' for slot filling. As such, all models tested have a tendency to overly predict 'O' for slot labels.

4. Model

I present two models, BiLSTM and BERT based respectively.

Their results are compared against a simple yet effective baseline model shown in lab 10 of the course, in Fig **??**:

4.1. Baseline

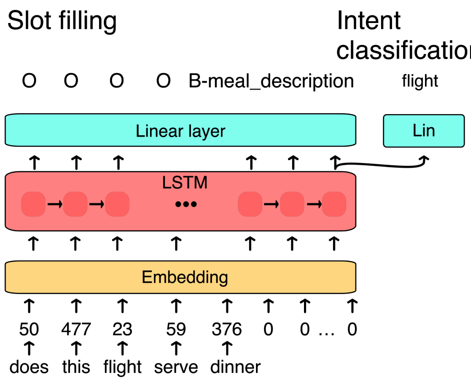


Figure 1: *baseline*

- embedding layer
- LSTM with 1 layer
- output layers for slot filling and intent classification

4.2. BiLSTM

This model is based on a bidirectional Long Short Term Memory network. The architecture is show in Fig 2

More specifically it consists of:

- embedding layer
- bidirectional LSTM with 4 layers
- layer normalization

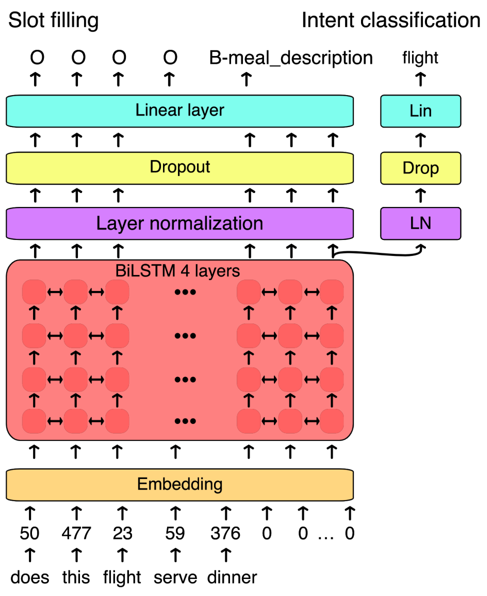


Figure 2: *BiLSTM*

- dropout layer
- output layers for slot filling and intent classification

The model is composed by a BiLSTM as backbone, followed by two heads, one for slot filling and one for intent classification. Slot filling considers the whole encoded sentence by utilizing the hidden outputs from the last layer of the BiLSTM backbone. Intent classification only considers the last hidden state of the last layer to make the prediction.

4.3. BERT

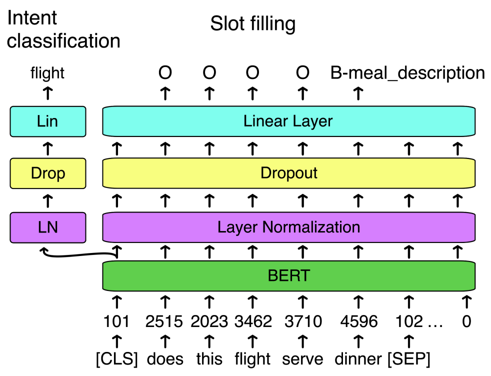


Figure 3: *BERT based*

The second model proposed uses a similar architecture, but it replaces the embedding and BiLSTM with a BERT model as backbone. This was pre-trained from Huggingface, specifically *bert-base-uncased*, which was trained using book-corpus and wikipedia datasets. Due to lack of computational resources, the model was tested only once. Larger versions of BERT were not tried for similar reasons.

Because BERT is pre-trained on a different corpus, with a different task, it also expects a specific word to id encoding.

BertTokenizer is able parse the input sequence according to BERT needs, adding '[CLS]' and '[SEP]', two special tokens required by BERT. However the tokenizer does also split some tokens in base word and suffix. This creates an inconsistency with the number of slots. Therefore only the base word is considered and suffixes are ignored. Similarly '[CLS]' and '[SEP]' are removed in the slot filling.

utterance	how many people fit on a 72s airplane
BERT tokenization	[CLS] how many people fit on a 72 ##s airplane [SEP]

Table 1: *Example of tokenization differences*

The architecture is shown in Fig 3 and it consists of:

- BERT backbone
- layer normalization
- dropout
- output layers for slot filling and intent classification

Intent classification takes as input the pooler_output of the BERT backbone, which can be considered as the encoded token '[CLS]'.

Very small learning rates ($1e^{-5}$) were used, with high number of epochs. Using higher learning rates with fewer epochs yielded baffling results. This is probably because backpropagation with larger learning rates completely destroys the pretrained information within the BERT backbone in very few iterations. Small learning rates paired with high epochs allow to find the correct tradeoff between adapting the backbone to the downstream tasks and overfitting.

Both simple architectures are complemented by a few considerations in order to improve performances.

- early stopping
- random loss weighting
- focal loss

4.4. Early stopping

Early stopping is used to avoid overfitting. If both losses on slot filling and intent classification increase while training, the training is halted. Losses for early stopping are computed on the dev set. A hyperparameter for patience is set to 5 for both, which allows the model to increase the losses on the dev set up to 5 times before terminating.

4.5. Randomized loss weighting

In multi task problems some tasks may converge earlier than others. Therefore weighting appropriately the losses for each task is not trivial and if done correctly it can improve performances. In *A Closer Look at Loss Weighting in Multi-Task Learning* [1], the authors propose to chose weights of the losses randomly at each iteration. This simple strategy allows them to achieve results similar to more complex state of the art multi task loss weighting techniques.

I use a very similar approach but with one key difference. I discovered that for both ATIS and SNIPS both models perform slightly better if among the two random weights, the larger is assigned to the task with higher loss. This is probably due to the fact that placing the larger weight to the task with higher loss increases the focus on that specific tasks. If one of the two tasks converges earlier, the loss will be lower and the smaller weight is assigned to the loss, as there is no need to learn more on that task.

4.6. Focal loss

Finally, both models use a specific loss function. Instead of cross entropy, focal loss is used. It tackles the specific need for a loss that is better suited for classification on uneven datasets, with some labels that are easier or harder. It can be considered as a properly weighted cross-entropy loss, with highest loss to hard examples.

$$FL = -\alpha(1 - P_t)^\gamma log(P_t) \tag{1}$$

with α and γ two hyperparameters that regulate the weights placed on the labels.

Optimizer used for both are Adam. Batch sizes are 128 for training set, 64 for validation and test set.

5. Evaluation

The main metrics used for evaluation are F1 score for slot filling and accuracy for intent classification. This is in line with the state of the art metrics on both ATIS and SNIPS for joint slot filling intent classification.

The final table here is shown in Tab 4

Both models are far from the state of the art, but are a slight improvement over the baseline.

Confusion matrixes are reported in figures 2 and 3 The colors are row normalized to prevent very frequent labels to dominate all the rest. Blank lines are where the models did not make any prediction for that specific label.

It is possible to see that for slot filling in ATIS, balanced ATIS and SNIPS, there is a vertical line on the label corresponding to 'O'. This is because 'O' is the most frequent label and therefore all models have a tendency to overpredict this label. Hard examples, that occur with few instances are also likely to be predicted 'O'.

A similar consideration can be made for the label 'flight' in intent classification in ATIS. Intent classification on SNIPS does not show the same issue.

In theory, balanced ATIS is an easier dataset to train, as it is more evenly distributed than ATIS. Although there are significant improvements on F1 score and accuracy for slot filling and intent classification, these advantages do not directly translate to a better confusion matrix. Because most of the blank lines correspond to labels that are not present in the train or dev sets, it would be expected that with a better partitioning, the number of blank lines would diminish. However this does not happen. Probably the reason lays in the very limited number of examples with those labels. It is also possible to notice that both in slot filling and intent classification, the models are less inclined to predict 'O' and 'flights' on balanced ATIS than default ATIS.

Although SNIPS is arguably a better dataset than ATIS, being larger, evenly distributed, with fewer labels and a bigger vocabulary, all models perform better on ATIS rather than SNIPS for intent classification and significantly worse on slot filling. The better performances for intent classification can be easily explained by the limited (7) number of labels. The worse results on slot filling may be caused by the scope of the tasks. On one hand, it is true that on ATIS there are more slot label, with fewer examples, but on the other hard all labels are under the wider scope of airport related requests. On SNIPS the requests cover a wider spectrum. This and the limited vocabulary on ATIS, probably lead to more examples for each token on ATIS than on SNIPS.

Although BERT performs as well as BiLSTM and in some cases better for intent classification, it significantly worse on

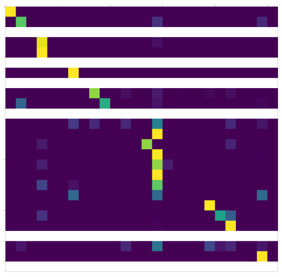
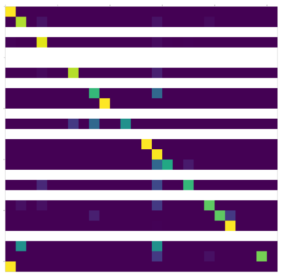
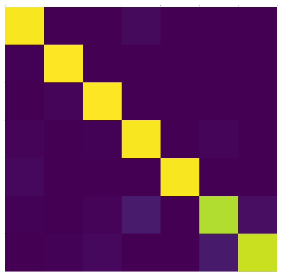
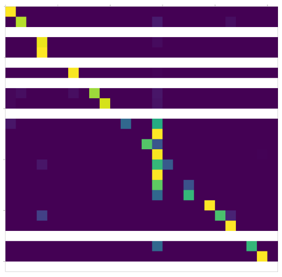
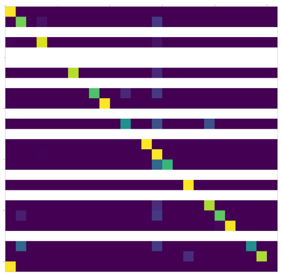
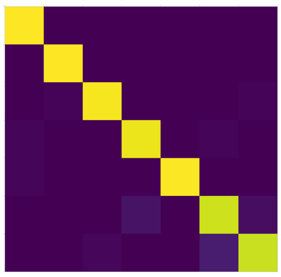
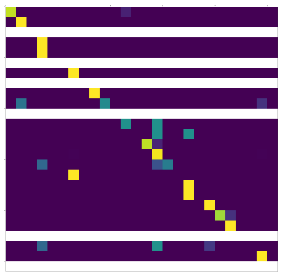
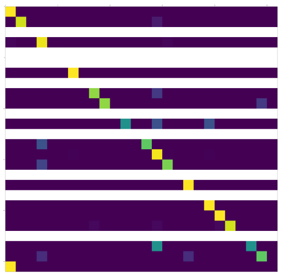
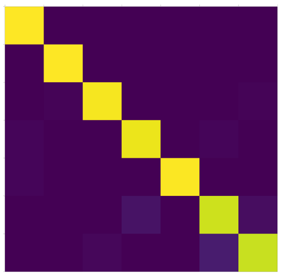
Intent Classification	ATIS	balanced ATIS	SNIPS
Baseline			
BiLSTM			
BERT			

Table 2: *Confusion matrixes for intent classification*

slot filling, being surpassed by the much simpler baseline in all datasets. Because this models it is based on a fine tuning of a pretrained model, in theory this should solve the issue of having a small and biased dataset. These results show that this is not the case. In 'BERT for Joint Intent Classification and Slot Filling' [2] the authors use a similar architecture and are able to perform much better on slot filling. I was unable to replicate the results.

Finally, looking at the training losses on BiLSTM in Fig 4 a final remark.

Even using early stopping, the loss on slot filling does increase on the dev set. This is because the training stops only if both dev losses start to increase. I was unable to change the hyperparameters regulating early stopping to remove this issue without compromising results. If this issue is solved, it is very likely possible to increase the slot filling performances further.

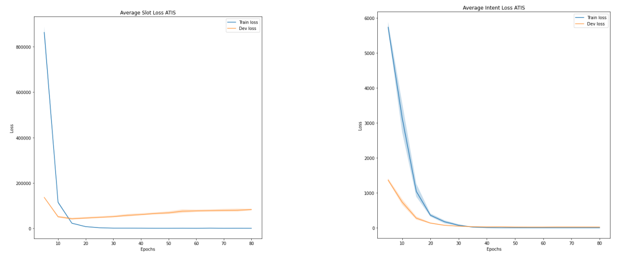


Figure 4: *Training losses on BiLSTM*

6. Conclusion

I presented two models for the joint task of slot filling and intent classification. While neither is able to achieve results com-

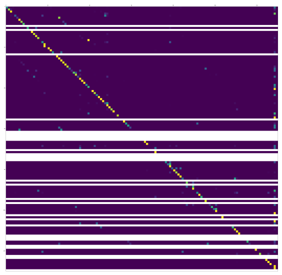
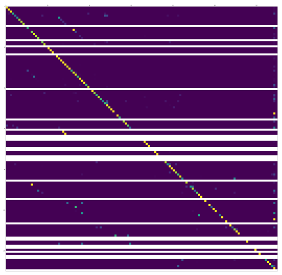
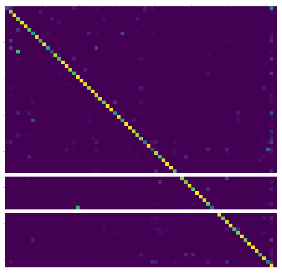
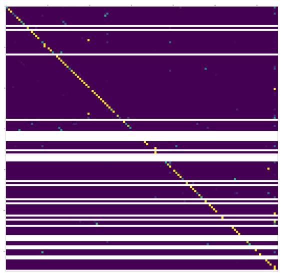
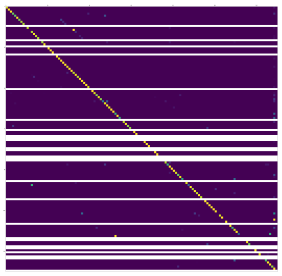
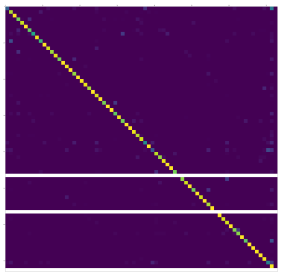
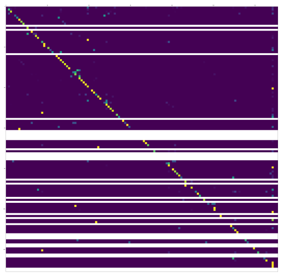
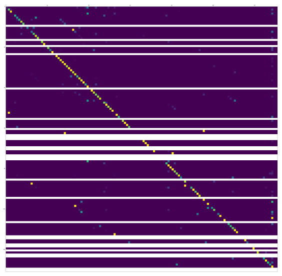
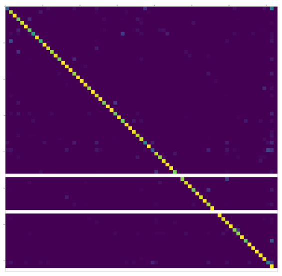
Intent Classification	ATIS	balanced ATIS	SNIPS
Baseline			
BiLSTM			
BERT			

Table 3: *Confusion matrixes for slot filling*

Slot filling

Model:	ATIS		ATIS balanced		SNIPS	
	Slot F1	Intent Acc	Slot F1	Intent Acc	Slot F1	Intent Acc
baseline	92.10%	93.14%	94.37%	96.79%	80.36%	95.19%
1° proposal BiLSTM	94.70%	95.30%	97.33%	97.20%	88.10%	96.29%
2° proposal BERT	86.67%	94.73%	89.84%	96.86%	69.20%	96.42%
Some state of the art						
Joint BERT [2]	96.1%	97.5%	-	-	97.0%	98.6%
Bi-model with decoder[3]	98.99%	96.89%	-	-	-	-
Stack-Propagation (+BERT) [4]	96.1%	97.5%	-	-	97.0%	99.0%

Table 4: *results*

parable to the state of the art, they are not too far, especially for intent classification. As future work, it would be possible to explore more complex architectures for both models presented. Furthermore, on BERT, one key aspect that remains to be explained is understanding the reasons behind the significant worse results on slot filling than intent classification.

7. References

[1] B. Lin, F. YE, and Y. Zhang, “A closer look at loss weighting in multi-task learning,” 2022. [Online]. Available: https://openreview.net/forum?id=OdnNBNIdFul

[2] Q. Chen, Z. Zhuo, and W. Wang, “Bert for joint intent classification and slot filling,” 2019. [Online]. Available: https://arxiv.org/abs/1902.10909

[3] Y. Wang, Y. Shen, and H. Jin, “A bi-model based RNN semantic frame parsing model for intent detection and slot filling,” in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 309–314. [Online]. Available: https://aclanthology.org/N18-2050

[4] L. Qin, W. Che, Y. Li, H. Wen, and T. Liu, “A stack-propagation framework with token-level intent detection for spoken language understanding,” in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 2078–2087. [Online]. Available: https://aclanthology.org/D19-1214