Final NLU project

Joy Battocchio (229367)

University of Trento

joy.battocchio@studenti.unitn.it

In this paper is explained the development and the evaluation of two models with respect to a given baseline. The purpose of these models is to perform jointly slot filling + intent classification. The evaluation is done on two different datasets, ATIS and SNIPS.

1. Introduction

Intent classification and slot filling are two sentence level tasks that aim respectively to:

- assign an intent for a given sentence or utterance
- map the sentence to a sequence of domain-slot labels

This situation is an excellent candidate for multitask learning, since the two tasks are correlated, a model build to predict both these classifications can use information learned from one task to better perform on the other.[0]

The baseline model proposed consists of a simple LSTM that encode the sentence in order to classify the slots using the hidden layer, and the intents through the output layer. I decided to improve this model by computing a better loss that push the model to improve on both sides in a more balanced way and using the patience for the early stop in a smarter way. Moreover the encoder is a bidirectional LSTM (BLSTM).

The last model is a bi-model composed of two encoders and two decoders. It aims to split the two tasks among the two models but sharing the hidden layer so to keep the two related. This model is the implementation of the paper *A Bi-model based RNN Semantic Frame Parsing Model for Intent Detection and Slot Filling* [1]

2. Task Formalisation

The tasks for this project consists in building a model capable of learning and inference on two task simultaneously, in such a way that the final performances for both classifications are better than using a model each. This comes from the advantage of learning related tasks from which a system can abstract high level 'task agnostic' features to use for the classification. Such a strategy is most similar to the one used by humans. In particular we need to solve these two problems:

- intent classification: a text classification task in which the objective is to assign an intent for a given sentence or utterance. (e.g. 'Can you help me find out about flights?' - Info request)
- slot filling (or concept tagging): a sequence labelling task where the objective is to map a given sentence or utterance to a sequence of domain-slot labels. This can be seen as the natural consequence task after intent classification, after having filtered or classified the sentences, I would want to retrieve the information useful to understand them, and take the proper actions.

  (e.g.
  'I want to travel from nashville to tacoma'
  O O O O O B-fromloc O B-toloc)

The given baseline results are the following:

- ATIS → Slot f1 score: 92%, Intent accuracy: 94%
- SNIPS → Slot f1 score: 80%, Intent accuracy: 96%

The goal of this work is to implement and evaluate two model that perform better than the baseline

3. Data Description & Analysis

For this project I used ATIS and SNIPS datasets as specified in the instructions. For both of them I used the JSON format.

3.1. ATIS

The ATIS dataset (Airline Travel Information Systems) is composed of 4978 samples in the train set, and 893 samples in the test set. Since there is no dev set, I used the same method used in the lab 10 to split the train set, doing so the final samples count for the sets is the following : 4381 test set, 597 dev set, 893 test set. The total number of intents is 26, while the slot labels are 129. Each sample is composed by the utterance, the sequence of slot labels, and the intent. [2]

```
{'intent': 'flight',
 'slots': 'O O O O O B-fromloc.city_name O
           B-depart_time.time I-depart_time.time
           O O O B-toloc.city_name O
           B-arrive_time.time O O
           B-arrive_time.period_of_day',
 'utterance': 'i want to fly from boston at
               838 am and arrive in denver at
               1110 in the morning'}
```

3.2. SNIPS

The SNIPS dataset is composed of 13084 samples in the train set, 700 samples in the dev set and 700 samples in the test set. The total number of intents is 7, while the slot labels are 72. Each sample is composed by the utterance, the sequence of slot labels, and the intent. [3]

```
{'intent': 'PlayMusic',
 'slots': 'O O B-artist O B-album O B-service
           I-service',
 'utterance': 'listen to westbam alumb allergic
               on google music'}
```

4. Model

4.1. Common part

All the models I used share the same criterion for computing the task-specific loss, which is the *Cross entropy loss* that push

the model to be confident on its predictions. Then I used Adam optimizer for all three models, since it has shown to be the best for this tasks.

4.2. Baseline model

The baseline model is taken directly from lab 10, and its structure is composed as following:

- embedding layer
- LSTM
- linear layer for intent
- linear layer for slot

The intent classification is the output of a linear layer that has as input the output of the LSTM, while for the slot classification is used another linear layer whose output is the last hidden state. Both the losses are computed through *Cross Entropy Loss* The overall loss is computed as the sum between the losses of the two task: $L = loss\_intent + loss\_slot$ During training a *patience* variable is used with a starting value of 3, and get decreased every time the f1-score gets worse, when the *patience* reach 0 the training stops.

4.3. Second model

The second model is just an upgraded version of the baseline model, the only change in the structure is that the LSTM encoder is bi-directional:

- embedding layer
- BLSTM
- linear layer for intent
- linear layer for slot

The overall loss is computed as the weighted sum between the losses of the two task, the weights are random value s.t. $0 < random\_1 < 1$; $0 < random\_2 < 1$; $random\_1 + random\_2 = 1$. The loss is then calculated as:

$$L = max(loss\_intent, loss\_slot) * max(random\_1, random\_2) + min(loss\_intent, loss\_slot) * min(random\_1, random\_2)$$

In this way the task that carries the more loss is weighted more and therefore push the training to be more balanced between the two tasks.

The weights are random because as written in the paper *A closer look at loss weighting in multitask learning* [4] it has shown to be very effective in empirical studies. Also the early stopping mechanism has been changed: I used two different values, a *patience_f1* and a *patience_accuracy*, their initial value is 5, they decrease by one every time their respective performance gets worse and the training stops only when both of them reach 0 or below. In addition their values increase by one every time the performance breaks the previous best score, until a ceiling of 5 (their initial value)

I tried different initial values for the patience and different ways to calculate the loss before finding what seems to be the best. Another interesting loss I tried is the one in which the two components were weighted by their *improving degree*, calculated as $loss_t - loss_{t-2}$, however this has shown to be not so effective.

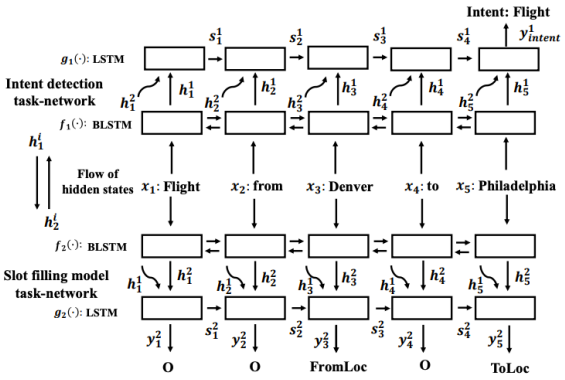

Figure 1: *bi-model structure*

4.4. bi-model

The third model is an implementation of the paper *A Bi-model based RNN Semantic Frame Parsing Model for Intent Detection and Slot Filling*. The structure has been taken from an unofficial implementation on github [1]. The network is rather complex, it is composed by two different models, one for each task, and every one of this model is composed by an encoder and a decoder. [figure 1]

- Slot filling model
  - Encoder
    - embedding layer
    - BLSTM
  - Decoder
    - LSTM
    - linear layer
- Intent classification model
  - Encoder
    - embedding layer
    - BLSTM
  - Decoder
    - LSTM
    - linear layer

Since the two losses are calculated from two different models there is no need to combine them for back-propagation. This is an advantage because it helps keeping the models specialized in their own task while transferring the information from the hidden layer. Also the early stopping mechanism is the same as the one used in the second model.

After many tries I decided to combine to the word features after the decoding also two more features taken from the **spacy parsing**, those are the head of the word, and its dependency tag. For this purpose I had to extend the datasets, adding these two components two each sample, that are completely ignored by the first two models. The resulting example looks like this:

```
{'deps': 'det nsubj prep pobj aux nsubj
ROOT prep pobj prep pobj
prep nummod pobj',
'heads': 'type fly type of fly fly fly
```

```
fly from fly to fly pm before',
'intent': 'aircraft',
'slots': 'O O O O O B-airline_name O O
B-fromloc.city_name O B-toloc.city_name
B-depart_time.time_relative
B-depart_time.time I-depart_time.time',
'utterance': 'what type of aircraft does
eastern fly from atlanta to denver before
6 pm'}
```

5. Evaluation

The main evaluation has been done on two main metrics, that were specified in the instruction of the project.

- F1 score for slot filling: the F1 score is a metric of accuracy that take into consideration both the precision and the recall of the predictions.

  The formula is $2 * \frac{p*r}{p+r}$.

  The F1 score is usually computed for each class, and the overall F1 is calculated averaging these scores with different techniques. I used connl evaluation, which uses the *micro average*, that does not consider the classes separately, but computes the total F1 directly from true positive, false positive and false negative.

- Accuracy for intent classification: mathematically, the accuracy represents the ratio of the sum of true positive and true negatives out of all the predictions. We can see it as how much the confusion matrix is similar to a perfect diagonal matrix.

  The formula is $\frac{TP+TN}{TP+TN+FP+FN}$

5.1. Results

- Baseline:
  - F1-score on ATIS: 92,6%
  - Accuracy on ATIS: 93,6%
  - F1-score on SNIPS: 81,9%
  - Accuracy on SNIPS: 96,1%

  Training the model 5 times shows that thank to the fact that the momentum of the optimizer is not initialized every time the performance on the slot filling task slightly increases over time, probably due to the fact that it takes longer to converge with respect to the intent classification task. The accuracy instead tends to oscillate but without general improving, probably because since the early stopping takes into consideration only the F1-score, it tends to be already overfit when the other converges.

  In the ATIS dataset the intents that are better classified are the ones in which we find codes, especially numerical, while the poorly classified are the composed ones. The slots that are better classified are the *from location* and *to location*, because they can be easily recognized by their relative position, while the poorly classified are all the ones represented by a single number, which can be a year, a code, or something else, and its difficult to tell apart.

  In the SNIPS dataset all the intents are well classified, the ones the performed the worst are *search screening events* and *search creative work*, these two in fact don't have many key-words in their sentence and are often confused for *book restaurant* or *get weather* especially when there are city names in the utterances.

  The better recognized slots are the ones that are most domain specific, like *movie genre* or *weather*, while the worst are those concerning the title of art works or companies, like *movie title* or *restaurant name* which often refer to daily life and are therefore hard to classify.

- Second model:
  - F1-score on ATIS: 94,4%
  - Accuracy on ATIS: 95,6%
  - F1-score on SNIPS: 87,7%
  - Accuracy on SNIPS: 96,1%

  The new strategy used for the early stopping that considers both the F1-score and the accuracy stabilised the training, therefore the standard deviation of the measurements along the 5 training is lower than the baseline.

  In both datasets the F1-score for the slot filling task increases over the trials for the same reasons as for the baseline. The ranking of the most classified labels is almost the same, but the overall performance is higher due mostly to the fact that the bi-directional LSTM is able to capture more meaningful feature from the sentence.

  We can see that while in the ATIS dataset the different in the performance with respect to the baseline is almost the same for both the metrics, in the SNIPS dataset we have the same accuracy, but a massive improvement in the slots filling F1-score. I think this is due among the other things to the way the loss is calculated, pushing more on the task that is behaving worse. Also the confusion matrices confirm these assumptions, they have almost the same distribution of points as in the baseline, but brighter in the main diagonal.

- Bi-model:
  - F1-score on ATIS: 93,1%
  - Accuracy on ATIS: 93,5%
  - F1-score on SNIPS: 88,8%
  - Accuracy on SNIPS: 83,4%

  The last model was the most difficult to implement and also to evaluate. Its performance did not meet the expectations, since the paper claims F1-score around 96% and 98% on ATIS dataset. I was not able to train it 5 times because it took too long even on colab. I believe that training it more times like I did with the other models, slightly better results could be achieved, but still far from the numbers claimed in the paper.

  On ATIS the performance are slightly better than the baseline in slot filling, while are basically the same in intent classification. The ranking of the best classified slot is different from the other two models and it's difficult to find a pattern in it. On the top we can find both

names and codes, and labels with strong support and not, and so it is on the bottom. The intent are classified very similarly, the overall accuracy is lower but the model is able to classify a lot better the label *airport*.

On SNIPS we have an interesting result, the model perform terribly in intent classification, with more than 10% gap with the baseline. All the labels are classified slightly worse, but the worst classification is on the *book restaurant* label. This label is easy to classify for the previous model because there is a strict correlation between this label and the slot labels concerning restaurant, like *restaurant type*, but since this model carries the two tasks more independently it is more difficult for it to get it right. On the other hand the highly representative feature representation allows the model to predict very well the slot labels. Also in this case the best and worst labels remain the same, with a poor performance on the *song name*, *movie name* and *restaurant name*, but the overall score gets an important improvement. I think that weighting the classes at training time pushing on *book restaurant* could make the model save some points in accuracy on SNIPS, however this would not lead to a better general performance.

	ATIS		SNIPS	
	F1-score	Accuracy	F1-score	Accuracy
Baseline	92,6%	93,6%	81,9%	96,1%
Second model (upgraded)	94,4%	95,6%	87,7%	96,1%
Third model (bi-model)	93,1%	93,5%	88,8%	83,4%

6. Conclusion

The baseline reached a slightly better score than it did during the lab 10, this is due to the *adam optimizer* that is not initialized every time as I already pointed out.

The second model implemented is just an upgraded version of the baseline, the goal was to demonstrate that with small changes and some smart tricks the performances would have increased by at least 2%.

The last model is the one that is more worth talking about. As I said the paper claimed very high performance, but there was no implementation attached. The implementation I found claimed almost the same results (slightly lower) but many people asserted that they were not able to reach the same score.

They said that the F1-score could have improved by adding slot label dependencies. I tried to do that by applying a CRF layer on top of the slot decoder but it didn't really work. [5] I also found out that they did not considered the composed labels like *flight+airline* and so whatever alternative they used to consider these cases would certainly result in a better accuracy.

Last comment is that the model was evaluated only on ATIS dataset, a poor performance on SNIPS dataset can also mean that the model was tailored for that purpose only, and it has been trained in a very strict environment, so their result cannot be generalized to have a good slot filling + intent classification system that is domain agnostic.

Important note: One big limit to the intent classification score is the fact that in the ATIS test dataset there are some label that are missing from the test set, most of them are composed labels like *ground_service+ground_fare* that are completely missing on simply written the other way around (*ground_fare+ground_service*), or even single labels that are completely missing like *day_name*. This means that there still is improvement margin just by building a better distributed dataset.

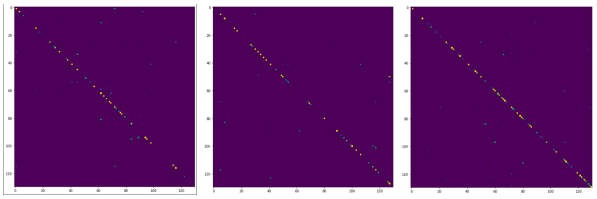

Figure 2: *model 1: slot filling ATIS*

Figure 3: *model 2: slot filling ATIS*

Figure 4: *model 3: slot filling ATIS*

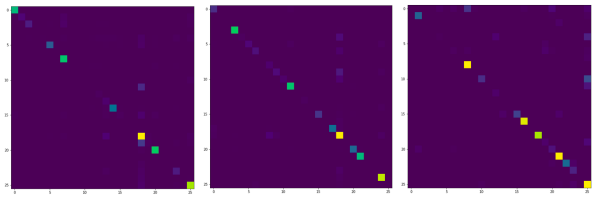

Figure 5: *model 1: intent classification ATIS*

Figure 6: *model 2: intent classification ATIS*

Figure 7: *model 3: intent classification ATIS*

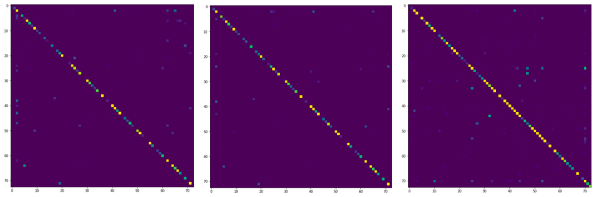

Figure 8: *model 1: slot filling SNIPS*

Figure 9: *model 2: slot filling SNIPS*

Figure 10: *model 3: slot filling SNIPS*

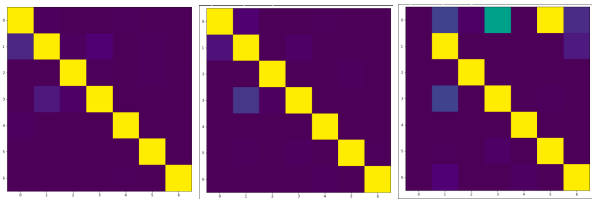

Figure 11: *model 1: intent classification SNIPS*

Figure 12: *model 2: intent classification SNIPS*

Figure 13: *model 3: intent classification SNIPS*

7. References

Sebastian Ruder,
An Overview of Multi-Task Learning in Deep Neural Networks
https://ruder.io/multi-task/

Yu Wang Yilin Shen Hongxia Jin,
A Bi-model based RNN Semantic Frame Parsing Model for Intent Detection and Slot Filling
https://aclanthology.org/N18-2050.pdf

Howl Anderson,
GitHub page of the ATIS dataset
https://github.com/howl-anderson/ATIS_dataset

SNIPS (SNIPS Natural Language Understanding benchmark)
https://paperswithcode.com/dataset/snips

Baijiong Lin Feiyang Ye Yu Zhang,
A closer look at loss weighting in multitask learning
https://arxiv.org/pdf/2111.10603.pdf

MIT licence,
Python CRF
https://pytorch-crf.readthedocs.io/en/stable/

Joy Battocchio,
Source code of the paper
https://github.com/JoBatt96/NLU_proj_2021-22