Final NLU project

Edoardo Barba (231592)

University of Trento

edoardo.barba@studenti.unitn.it

Abstract

This document is the report of the final project of the Natural Language Understanding course. The purpose of this project is to implement a neural network model to perform intent classification and slot filling in a multitask learning setting.

1. Introduction

Slot filling and intent detection play important roles in Natural Language Understanding (NLU) systems. These 2 tasks are usually dependent one from the other, that's why a multitask learning setting is used a lot in order to do joint intent classification and slot filling. In this report I present 3 models, all based on encoder/decoder paradigm, with hard parameter sharing on encoder. The first model is the baseline model, in which the encoder consists on a simple LSTM and 2 different classifiers for slot and intent tasks. The second model is an improved version of the first one, in which the encoder is a bidirectional 2 layer LSTM. The last model is based on BERT (Bidirectional Encoder Representations from Transformers), a multilayer bidirectional Transformer encoder based on the original Transformer model [1]

2. Task Formalisation

The goal of this project is to perform joint *slot filling* and *intent classification* using a multitask learning setting. The advantage of multi-task learning is the fact that the model can learn common features for 2 similar tasks. Intent classification is the process of classifying the customer's intent by analyzing the language they use. The goal of slot filling, instead, is to identify from a running dialog different slots, which correspond to different parameters of the user's query.
Table 1 shows an example of intent classification and slot filling for a user query.

The goal is to first find a mapping from the input representation of words and sentences to a hidden representation that capture the relevant features of the input through the shared encoder. The task specific decoder then predicts the intent or slot based on this hidden representation.

3. Data Description and Analysis

The models were trained and tested on 2 public benchmark datasets, ATIS and SNIPS

3.1. ATIS

ATIS (Airline Travel Information System) is a dataset consisting of audio recordings and corresponding manual transcripts about humans asking for flight information on automated airline travel inquiry systems. The dataset is composed of 4978 samples in the training set and 893 samples in the test set. I used the method seen at lesson to create the validation set. The final distribution of samples is 4381 for training, 597 for validation and 893 for test set. It contains 22 intent labels and 129 slot labels, with a vocabulary length of 863. In Fig. 1 we can see the intent labels distribution of ATIS training set. We can notice that the most frequent intent is 'flight', with a frequency of around 75% on the whole dataset; all other intents instead have a much lower support. The slot labels distribution, instead, is much more uniform.

Table 1: *An example from user query to intent classification and slot filling*

Query	listen to westbam album allergic on google music
Intent	PlayMusic
Slots	O O B-artist O B-album O B-service I-service

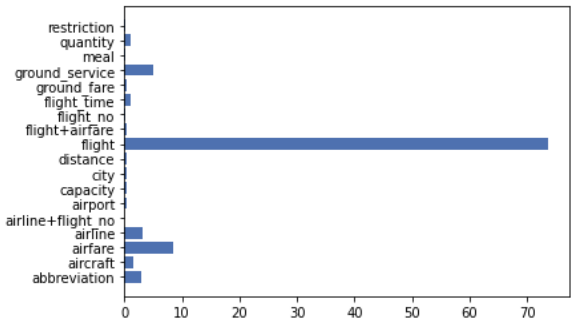

Figure 1: *ATIS intent labels distribution*

3.2. SNIPS

The SNIPS dataset is composed of 13084 samples in the training set and 700 samples in the test set. After the creation of the validation set we have a final distribution of 11644 samples in training set, 1440 samples in validation set and 700 samples in test set. The dataset contains a total of 7 different intent labels and 72 slot labels, and a vocabulary length of 10621. In Fig. 2 we can see the intent label distribution of SNIPS training set. We can notice that we have a uniform distribution of intent labels, they have in fact almost the same frequency on the dataset.

4. Models

4.1. Common settings

For all the model I used the mapping seen at lesson to convert words, slots and intents to ids. I used the Cross Entropy Loss to compute the loss for both tasks and Adam optimizer to train the models.

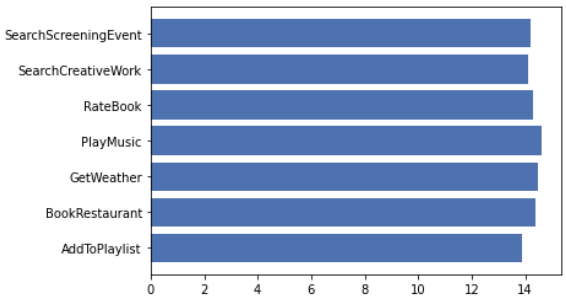

Figure 2: *SNIPS intent labels distribution*

4.2. Baseline Model

As baseline model I took the LSTM model saw in class during Lab 10. It is composed of an Embedding layer, a single layer LSTM encoder, a dropout layer and 2 different linear classifiers. The LSTM encoder is used to capture the features of the sentence, the hidden representation, is then used by the 2 task specific classifiers. The loss for the multitasking learning is computed as the sum between the losses of the two tasks.

4.3. Bi-LSTM Model

The second model is an upgraded version of the first one. There are some changes both on the structure and in the training phase. The embedding layer remains the same, while the encoder is a Bidirectional LSTM with 2 layers. The dropout probability is increased from 0.1 to 0.6 and a new dropout layer is introduced after the embedding layer. The embedding size and hidden size are increased respectively from 300 to 400 and from 200 to 400. The loss is computed as follow:

$$Loss = \alpha \times Loss\_Intent + \beta \times Loss\_Slot \qquad (1)$$

where *alfa* is equal to either 0.3 with 80% probability or else a random value between 0 and 1 and beta is simply (1-*alfa*). I used a weighted sum between the 2 task specific losses to give more importance to the slot loss becuase slot filling is a more difficult task. In order to also add some stochasticity, we can see that with some probability the weights could also be chosen randomly. I decided to apply this mechanism because as stated in [2], training a multi-task model with random weights sampled from a distribution can improve the performance.

4.4. BERT Model

The third model is implemented following the idea described in the paper *BERT for Joint Intent Classification and Slot Filling* [3]. The structure of the model is similar to the previous ones. In order to produce the hidden state, instead of an LSTM network, a BERT network is used. BERT is a multi-layer bidirectional Transformer encoder based on the original Transformer model [1]. Then dropout is applied before passing the produced hidden states to 2 task specific classifiers that will output the labels for intent and slots. In order to map words to ids, I used the same mapping used for the first 2 models.

4.5. Experiments

I tried a lot of different techniques and architectures during the realization of this project, here are briefly explained some of them. By using these techniques although I didn't obtained an improvement of the performance.

4.5.1. Total loss computation

I tried different techniques for computing the total loss. One idea was to apply an decay on *alfa* parameter which increased exponentially in number of epochs. It started from 0.5 and then decrease with an exponential increasing in velocity. In this way, during the training, the intent task is first learned and then more importance is given to the slot task. However with the use of this technique the model underperformed the actual values of F1-score and accuracy in both tasks.

4.5.2. Improved hidden representation

Another thing that I tried is to aggregate the output slot hypotesis with the hidden state to predict the utterance-level intent of the utterance. I added to the hidden representation produced by the encoder, the slot filling hypotesis, given in output by the slot classifier, in order to feed the intent classifier. The intent classification depends a lot on the slots in the sentence; so I though to exploit more this information and to exploit also the slot's linear classifier in order to recognise intents. This method resembles, in a very simplified way, the idea described in this paper [4], in which they tried to exploit the semantic hierarchy for effective modeling with a a capsule based neural network model which accomplishes slot filling and intent detection via a dynamic routing-by-agreement schema. However with the addiction of this techniques I didn't noticed an increasing of the performances, but only a slower training phase.

4.5.3. 1D Convolution

Another thing that I tried is to add a 1D-convolution layer after the encoder network in order to exploit more the correlation between different parts of the sentence and to also try to reduce the dimension of the hidden representation.

4.5.4. Task specific LSTM

I tried to add other 2 task specific LSTM networks after the first one and use each one them only for one task. I trained the first common LSTM network using both intent and slot losses as before and then trained the second one specifically for intent classification or slot filling, so using only one loss. This idea comes from the paper *A Bi-model based RNN Semantic Frame Parsing Model for Intent Detection and Slot Filling* [5].

4.6. Training setup

All the models are trained 5 different times on each dataset, with a maximum of 200 epochs. A early stopping mechanism is applied which take into consideration the slot F1-score.

5. Evaluation

The performance of the models were measured mainly with F1-score for slot filling and accuracy for intent classification. I trained the models 5 times on each datasets in order to obtain more trusworthy results and I reported the average and standard deviation of the measures. The results of the 3 models on ATIS and SNIPS datasets are shown in Table 2

Table 2: *Intent and slot performance*

Dataset	SNIPS		ATIS	
	Slot F1	Intent Accuracy	Slot F1	Intent Accuracy
Baseline	78.8 +- 0.15 %	96.6 +- 0.15%	91.4 +- 0.03%	93.7 +- 0.03%
Bi-LSTM	89.7 +- 0.04%	97.7 +- 0.04%	93.5 +- 0.04%	95.6 +- 0.04%
BERT	87.3 %	97.2%	93.1%	95.8%

5.1. Baseline Model

In both datasets the model converges in a very fast way in intent task and more slowly in the slot classification. The model almost every run started to overfit on intent classification task while in the slot filling didn't converged yet. This problem can be seen by plotting the intent and slot losses. On SNIPS dataset, the performance of the baseline model is characterized by a quite high standard deviation. This difference in performance is probably caused by the early stopping mechanism that sometimes stop the training too late, causing a decreasing of performance due overfitting in intent classification task.
In intent classification task, we can see that the model perform better in SNIPS dataset, this is probably also caused by the fact that in ATIS we can have intent with very few samples, while in SNIPS dataset the intent distribution is almost uniform, as can be seen in Fig. 2. Looking at worst and best classified intent labels, we can see that the model is not able to recognise composed intent, for example 'airfare+flight' or 'flight+airline'; this is mainly caused by the fact that these labels have a very low support. An example of intent with a major support that is still not recognised by the model is 'flight_no' label. Looking at worst and best classified slot labels, we can notice that the model is able to recognise very well slots that contains codes, like 'toloc.state_code' or 'depart_time.period_mod'

5.2. Bi-LSTM Model

The second model overperforms the baseline on both datasets and both tasks. On SNIPS we have a gain of more than 10% in Slot F1 and a gain of around 1% in Intent accuracy. In ATIS we have a 2% improvement in slot F1 and 2% improvement in intent accuracy. We have a more stable training, with a much lower variance in SNIPS dataset. The great improvement in slot filling in SNIPS dataset is also yield by the fact that the loss is computed by a weighted sum between the slot and intent losses, giving more important to slot loss. This allow the training to proceed with a more simultaneous convergence of the 2 tasks. The idea of giving more importance to slot loss comes from the fact that, in general, slot filling task is more difficult than intent classification. Without this mechanism the model converges very fast on intent task and slow on slot one. However this problem is not completely solved yet since we have still a faster convergence on intent task and slower one in slot task, as we can see in Fig. 3 and Fig. 4

Another important factor that allowed to increase the performance is the dropout. The introduction of a dropout layer before the Bi-LSTM network and the increasing of the dropout probability with respect to the baseline model allowed to avoid overfitting, expecially on intent classification task. It helped expecially with SNIPS dataset, where the intent classification task is quite simple (only 7 classes) and the model tends to converge faster.
Looking at best and worst classified labels, we have quite the same results for ATIS dataset, with only a general improvement

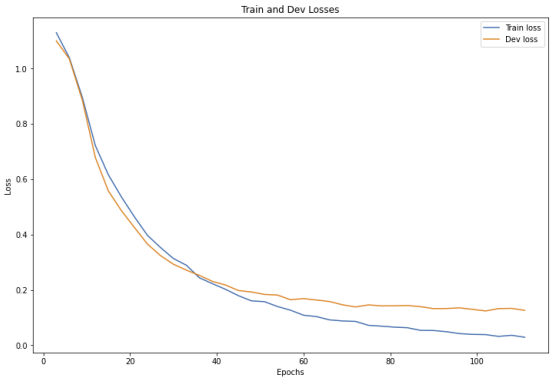

Figure 3: *ATIS intent loss*

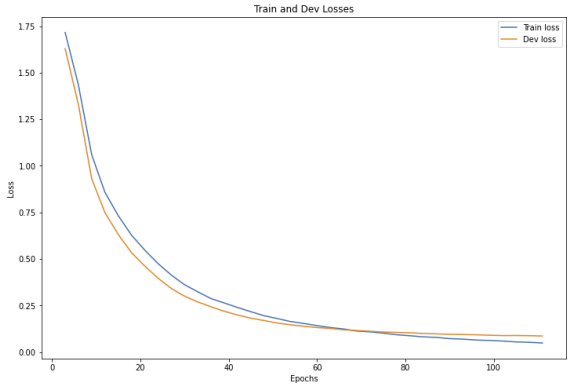

Figure 4: *ATIS slot loss*

in F1-score and accuracy. However we can notice a massive improvement in recognising some labels, like 'flight_no' in ATIS dataset which pass from an F1-score of 0% to 76.9%. We can also notice an improvement in recognising with a very high F1-score more labels of slots in which we find different things than codes, like 'condition_description' label in SNIPS dataset.

5.3. BERT Model

I trained the third model only one time on each dataset because of the long time required to train it. The model obtained performance comparable to the second one, outperforming the baseline in both dataset and in both tasks. However the performance are way lower than those declared on the paper [3], which declared on ATIS a F1-score on slot of 96.1% and Intent accuracy of 97.5% and on SNIPS a F1-score on slot of 97.0% and accuracy on intent of 98.6%.
The performance declared on the paper were very high and shows the power of BERT network. I suppose that I was not able to reach these performance for 2 main reasons:

- BERT requires in input sentences with [CLS] token in the beginning and [SEP] token in the end. I tried a lot to use sentences with these tokens but I was not able to solve some alignment problems with the slots ground truth.
- BERT is supposed to work better using his specific word2id mapping. I tried to use it instead of the word2id mapping used for the previous models but the performance were terribly worst. Maybe there was something wrong in the way I implemented the mapping.

The model had a quite stable training phase on both datasets. In Fig. 5, we can see a plot of the total loss in validation and training set on ATIS dataset.

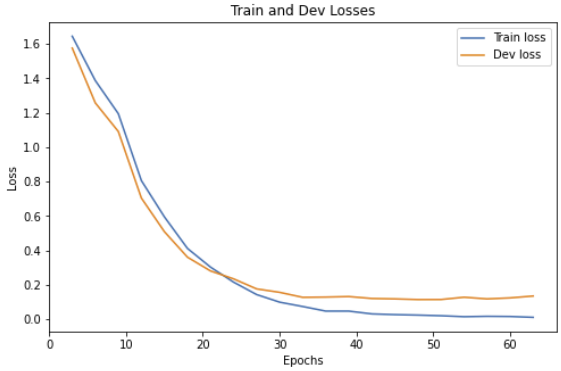

Figure 5: *ATIS loss BERT Model*

On SNIPS dataset, the model had a lot of ovverfitting in intent classification task, as can be seen in Fig. 6. This is probably because of the fact that the BERT network is very powerful and the intent classification task on SNIPS is quite simple. The network learns in a very fast way to classify intents while the convergence in slot filling is slower; the early stop criterion doesn't take into consideration the intent accuracy during training so the training continues even if there is a decreasing of the accuracy on validation set. I tried different early stop techniques in order to take into consideration also the intent loss but with poor results. The training, in fact, stopped while the model hasn't still reached the convergence on slot task. I tried also some regularization techniques, for example L1/L2 regularization but still with poor results on performance.

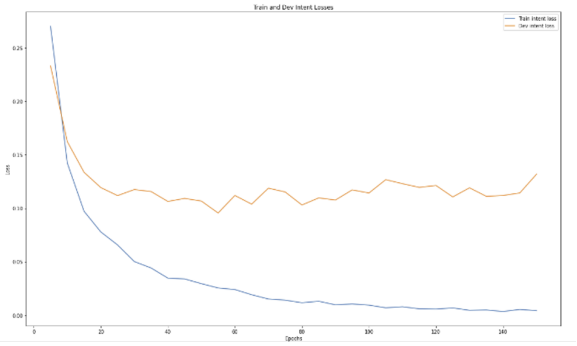

Figure 6: *SNIPS Intent loss BERT Model*

6. Conclusion

The baseline model that I implemented is the network seen in class during Lab 10. it is a model based on a LSTM network shared for the 2 tasks and 2 different linear classifiers for the 2 tasks.
The second model is an upgraded version of the first one with some modifications of both the architecture and the training procedure.
The last model is based on BERT network, a very powerful transformer-based network pre-trained on huge datasets. This model had performance comparable with those of the second model, however with much lower performance than those declared in paper [3]. This is probably caused by the reasons described previously.

The implemented models reached the goal of improving the baseline of at least 2-3% on both datasets. They showed a massive improvement especially in slot filling task on SNIPS dataset, thanks to the high capacity of the Bi-LSMT and BERT encoder to capture highly representative features.

7. References

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017. [Online]. Available: http://arxiv.org/abs/1706.03762

[2] B. Lin, F. Ye, and Y. Zhang, "A closer look at loss weighting in multi-task learning," *CoRR*, vol. abs/2111.10603, 2021. [Online]. Available: https://arxiv.org/abs/2111.10603

[3] 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

[4] C. Zhang, Y. Li, N. Du, W. Fan, and P. S. Yu, "Joint slot filling and intent detection via capsule neural networks," *CoRR*, vol. abs/1812.09471, 2018. [Online]. Available: http://arxiv.org/abs/1812.09471

[5] Y. Wang, Y. Shen, and H. Jin, "A bi-model based RNN semantic frame parsing model for intent detection and slot filling," *CoRR*, vol. abs/1812.10235, 2018. [Online]. Available: http://arxiv.org/abs/1812.10235