Final NLU Project

Gabriele Padovani (229207)

University of Trento

gabriele.padovani@studenti.unitn.it

1. Introduction

This paper offers an explanation of the development process, as well as the evaluation, of three models, with respect to a common baseline.

The architecture used as baseline is a classifier based on an Long Short Term Memory cell, which uses as loss function the sum of two different cross entropy losses, one for each task.

The first model developed, is an improvement of the initial baseline, where the LSTM layer was substituted by a Gated Recurrent Unit, as well as tweaking the parameters to allow bidirectionality in the recurrent layer. Several variations of this architecture were also experimented with, by swapping, for example, the GRU unit for RNN layers, implementing a different loss function, or testing the performance with more then one recursive layer.

An additional model implemented is an Encoder-Decoder structure. In the first component, the input is passed through embedding and GRU layers. In the latter, a second recursive layer is used to decode the embedded information. Since the project is based around multitask learning, two different decoders have to be used in this model, one for intent classification, the other for slot filling.

2. Task Formalisation

The Natural Language Understanding task assigned for the project is joint intent classification and slot filling, which consists in classifying the intent of a specific phrase, and detecting all relevant entities, or slots, with the same architecture.

For intent classification (text classification task), the aim is to, for each sentence, assign the respective intent which may be, for example, asking for information about a flight, time of departure, cost of the ticket etc. On the other hand, slot filling (sequence labelling task) consists in, for all words in an utterance, convert to the respective slot label, for example the city of departure and arrival, or the departure time.

This type of joined tasks can be grouped under multi-task learning[1].

The reason this technique may be advantageous is that, since the same model is operating on two tasks, it may be able to link meanings that would otherwise have no connection. Also multi-task learning is based on inductive transfer, an example of which are L1 and L2 regularization, where a model is taught to prefer sparse solutions. In this case, the inductive bias is learned, which allows for better generalization.

The baseline's performance, trained on each respective dataset, is:

ATIS:

- Slot F1 Score: 92%
- Intent Accuracy: 94%

SNIPS:

- Slot F1 Score: 80%
- Intent Accuracy: 96%

3. Data Description Analysis (approx. 200-500 words)

The information used for training and testing of the different models are:

- ATIS dataset, consisting in flight information from airline systems
- SNIPS dataset, which contains queries from seven different intents, complemented with the respective slot filling information

In both cases, the files used were in JSON format.

3.1. ATIS Dataset

The ATIS dataset, which stands for Airline Travel Information System, is composed of a train and test set, while the validation set was extracted from the training one, using the same approach showed in the Sequence Labelling and Text classification laboratory. The train and test sets contain respectively 4381 and 893 samples, while the validation set has length 597. Each item is a dictionary with three entries: utterance, which is the plain-text phrase; intent, which consists in a label representing the correct solution out of the 26 possible intents; and slots, a sequence that maps each token to the corresponding concept tag (one of 129).

```
{
'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'
}
```

3.2. SNIPS Dataset

The SNIPS dataset is a collection of queries, separated on the basis of 7 distinct intents. It is composed of a train, test and validation set, respectively of size 13084, 700, 1440. The items consist in a dictionary with the same three entries as the ATIS dataset: utterance, intent and slots. As for concept tagging, the number of total slots is 72.

```
{
```

```
'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. Models

4.1. Baseline model

The baseline model was taken directly from the Sequence Labelling and Text classification laboratory and is composed by an Embedding layer, a recursive LSTM layer, a Dropout layer, and two Linear layers, one for slot filling, the other for the intent classification task. The loss function used for the baseline is a simple addition of the respective intent and slot loss.

$$\begin{gathered} total\_loss = intent\_loss + slot\_loss \\ \text{where } \; intent\_loss = CrossEntropyLoss() \\ \text{and } \; slot\_loss = CrossEntropyLoss() \end{gathered}$$

To prevent overfitting, the model also presents an early-stopping mechanism with patience, which stops the training if the f1 slot score for the validation set decreases for three epochs in a row. In some of the following architectures, the number of epochs used in early stopping may increase, to better test the accuracy, or decrease to improve the train time of the model, since five consecutive iterations are necessary.

4.2. Bi-directional GRU model

The first model implemented is a variation of the baseline, where the LSTM layer was replaced by a GRU one, which should allow for similar performance while reducing computational complexity. The rest of the architecture was tweaked to allow bi-directionality in the recursive layer.

The loss function used in the model was modified too, randomly weighting the two intermediate losses (intent classification and slot filling loss), in such a way to give more relevance to the task with higher miss ratio [2].

$$\begin{gathered} r1 = random(), \quad r2 = 1 - r1 \\ min\_loss = min(intent\_loss, slot\_loss) * min(r1, r2) \\ max\_loss = max(intent\_loss, slot\_loss) * max(r1, r2) \\ total\_loss = min\_loss + max\_loss \end{gathered}$$

Since this architecture turned out to be the most performant, many tests were executed by variating the number of layers, type of loss function and mono or bi-directionality.

4.3. Bi-directional RNN model

An alternative architecture to the fist one has been tested, replacing the GRU cell for two RNN layers, and keeping bi-directionality in the recursive units.

The loss function used in the model is the same as for the last one, so randomly re-weighting the two intermediate losses, to balance training.

RNN layers are not really used anymore, since better alternatives exist, such as GRU and LSTM cells, the idea of this model is to test if there is noticeable accuracy loss, and trying to make up for this loss with several layers, although hurting performance.

4.4. Encoder-Decoder model

For the final model, a simple encoder-decoder architecture was implemented. The former includes an Embedding layer, a mono-directional GRU layer and Dropout. The latter, on the other hand, is composed by another embedding and GRU block, followed in addition by a fully connected linear layer.

Since the model is supposed to work in the multitask domain, two decoders are needed, one to output the slots' hypothesis, the other for intent classification.

TODO: add stuff

5. Evaluation

5.1. Metrics

Each model was evaluated using the two metrics proposed in the Sequence Labelling and Text classification laboratory, F1 score and accuracy.

- Accuracy: the simple ratio between the correct hypothesis (true positives and true negatives), divided by the total number of tests.

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

  One problem with the accuracy metric occurs if the dataset used for training has imbalanced classes. The result of this may be a very high accuracy score, but a model that still performs poorly (for example an entire category may be misclassified). This issue can be obviously solved by balancing the classes in the dataset, or by using a metric that takes imbalance into account, such as F1 score.

- Slot Filling F1 Score: is calculated using the `conll.py` implementation, which averages the total f1 score for each class, calculated with the formula:

  $$f1 = \frac{2*p*r}{p+r}$$

  p is the precision metric, which takes into account the correct predictions only within the subset of samples that have been evaluated as positive. It is calculated as:

  $$p = \frac{TP}{TP+FP}$$

  A precise model may not find all positives, but the ones found are often correct.

  r is the recall metric, which counts how many, of the total number of positives, the model identified correctly. It is calculated as:

  $$r = \frac{TP}{TP+FN}$$

  A model with high recall will find all positive cases, although including some negatives in the count.

  To have an architecture that identifies all and only positive cases, it a trade-off is needed between the two previous metrics, in such a way to have a high result only if both precision and recall are high [3].

5.2. Baseline results

The baseline training, both for the individual iteration and the five runs required by the assignment, gave comparable results to the ones reported:

ATIS:

- Slot F1: 0.918 +- 0.005

- Intent Acc: 0.938 +- 0.005

SNIPS:

- Slot F1: 0.794 +- 0.005
- Intent Acc: 0.966 +- 0.005

As for the five consecutive runs, the variance in the calculated results is around 0.005, and it doesn't vary depending on the dataset used.

5.3. Bi-directional GRU model results

Since several changes have been implemented between this second architecture and the initial baseline, many training phases have been executed. All the results have been reported in the table later on in this chapter. First of all, the full model was trained for five iterations with the intended loss, so the randomly weighted version, yielding the following results:

ATIS:

- Slot F1: 0.931 +- 0.00001
- Intent Acc: 0.955 +- 0.0001

SNIPS:

- Slot F1: 0.859 +- 0.0003
- Intent Acc: 0.961 +- 0.0001

Since training took too long, each one of the five iterations has been done individually, calculating then mean and covariance by hand.

During a second phase, the performance of the model was compared, first using the original loss, then without bi-directionality in the recursive layers, and finally incrementing the number of recursive layers. To speed up the process, these tests have been executed only once, and not five times, just on the ATIS dataset.

Model	Slot F1	Intent Acc.
Baseline	0.918	0.938
Re-weighted Loss	0.931	0.952
Plain Loss	0.939	0.956
Mono-Directional	0.861	0.854
Two Layers	0.938	0.951
Three Layers	0.931	0.946

Interestingly, the change in loss function actually yields worst, although comparable, performance. In both cases, however, it seems to stay above the threshold of accuracy required by the assignment, which is around 2-3% higher then the baseline.
It's clear that what is yielding the real improvement in accuracy is the bi-directionality in the recursive layers, as without it results are significantly worst.
Similarly, adding several recursive layers to the architecture does not seem to improve accuracy in a very noticeable way, and even removing the early stopping mechanism, to allow training for the complete number of epochs, did not change the results.
It is interesting to note that, even after several training iterations, the addition of a third recursive layer seems to consistently lower the accuracy.

5.4. Bi-directional RNN model results

Another change that was tested on a model similar to the baseline, was the use of RNN layers instead of LSTM or GRU cells. This small variation in the recursive layer, achieved comparable performance, but considering the presence of a secondary recursive layer, the computational complexity of the architecture was considerably increased without a significant, although present, gain over the baseline.

ATIS:

- Slot F1: 0.934 +- 0.001
- Intent Acc: 0.944 +- 0.001

SNIPS:

- Slot F1: 0.853 +- 0.0013
- Intent Acc: 0.964 +- 0.0002

Although expecting this particular model to much slower in training, the use of early stopping consistently cut short the iterative process, around the 30% mark, so after 60-70 epochs, making the obtainment of results actually faster for this architecture.

5.5. Encoder-Decoder model results

The final model implemented is an encoder-decoder architecture, taken partially from [4]. The model used was simplified, ignoring the shared memory section, while keeping one encoder for each task.

Two variations of this model were also tested, the difference in the two lies in the forward function. In the first model, the simpler one, the input is passed through a single encoder, and the hidden representation of the recursive layer is used to predict intents, while the output is passed to the slot-filling decoder. In the latter one, the more similar architecture to the one in [4], the input is passed through two different encoders, producing output *hi* and *hs* for intent and slots respectively. These representations are then concatenated to be passed through each decoder. As for the decoding process for the slot filling task, each sequence is passed through the recursive layer iteratively, which slows down the training efficiency.

The reason for the two different architectures is that the latter one, while performing in a better way compared to the former (reaching baseline level accuracy), has much slower training times, which made it not practical to optimize five different models.

The performance reported below belong to the simpler model, the first one described, and more similar to the baseline rather then to the paper's architecture. They are obtained by training five times the encoder-decoder structure.

ATIS:

- Slot F1: 0.909 +- 0.004
- Intent Acc: 0.912 +- 0.004

SNIPS:

- Slot F1: 0.788 +- 0.001
- Intent Acc: 0.967 +- 0.00001

On the other hand, results for the more complex model are obtained from a single training phase, not five, of 100 training epochs. This is due to the fact that even just training one model took over three hours for this architecture. Early stopping seemed to interrupt the computation due to very low accuracy in the intent classification task, so patience was set to a high value.

ATIS:

- Slot F1: 0.924
- Intent Acc: 0.947

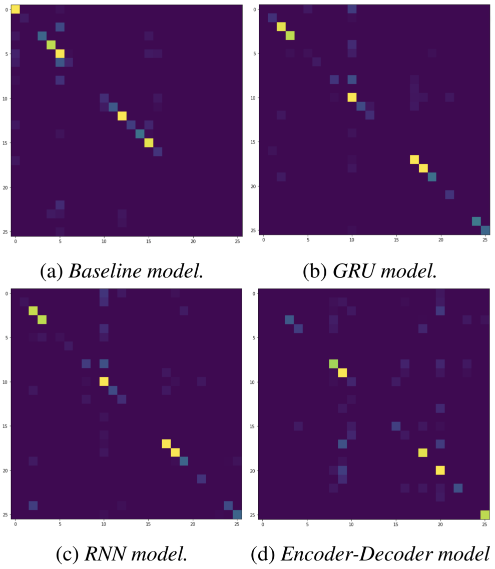
SNIPS:

- Slot F1: 0.816
- Intent Acc: 0.966

5.6. Confusion Matrices for the ATIS Dataset

The results yielded by the four main models trained in this assignment have been plotted in the confusion matrices shown below. The representations include only the results from training on the ATIS dataset, to avoid crowding the report with too many images, and are grouped for task, the first four images show the intent classification results, while the last four show slot filling.

The confusion matrices shown for the encoder-decoder model represent the output for the simpler model, trained five times as requested for the assignment, it was preferred to the better performing architecture since mean and variance information were available.



(a) *Baseline model.* (b) *GRU model.*

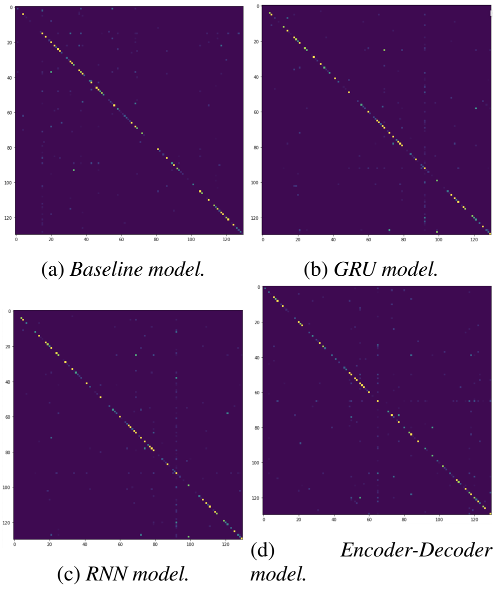
(c) *RNN model.* (d) *Encoder-Decoder model*

Figure 1: *Confusion matrices for intent classification task (ATIS dataset).*

6. Conclusion

The main objective of this paper, and of this assignment, was to implement an architecture that would improve the baseline's performance by around 2%. While not all the models tested achieve this aim, the majority of trial and test steps taken during development have been reported, as they are a significant part of the project results.

It seems as if the change in type of recursive layer doesn't influence the accuracy as much. What makes a very strong impact is enabling bi-directionality and, on the matter of computational complexity, keeping the number of layers as low as possible.



(a) *Baseline model.* (b) *GRU model.*

(c) *RNN model.* (d) *Encoder-Decoder model.*

Figure 2: *Confusion matrices for slot filling task (ATIS dataset).*

As for the encoder-decoder model, accuracy is in line with the baseline and, although the paper claims higher performance, the one trained is a simplified version of the architecture.

Something to note, is the fact that better performance could be achieved, by raising the patience threshold used in early stopping. In the experiments with just one model being trained, this number was kept at three, however, when it came time to train five times consecutively, the patience level was lowered to two, especially in the GRU and RNN models, training with the SNIPS dataset. This small tweak caused training to often stop around the 60^{th} epoch for the GRU model, and around the 70^{th} epoch for the RNN one. This happened even though the amount chosen to guarantee enough fitting of the architecture was 200 iterations. It is safe to say that some percentage points in accuracy may be gained, by incrementing this threshold.

7. References

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

[2] Lin, Baijiong and Ye, Feiyang and Zhang, Yu, "A Closer Look at Loss Weighting in Multi-Task Learning", `https://arxiv.org/abs/2111.10603`.

[3] Joos Korstanje, "All you need to know about the F1 score in machine learning.", `https://towardsdatascience.com/the-f1-score`.

[4] Yu Wang, Yilin Shen, Hongxia Jin, "A Bi-model based RNN Semantic Frame Parsing Model for Intent Detection and Slot Filling.", `https://arxiv.org/abs/1812.10235`.