Final NLU project

Pierluca Faccin 233424

University of Trento
`pierluca.faccin@studenti.unitn.it`

Abstract

This report presents the findings of the final project undertaken as part of the Natural Language Understanding course.
The objective of the project was to develop and implement a neural network model for intent classification and slot filling utilizing a multitask learning approach.

1. Introduction

Slot filling and intent detection are crucial components in Natural Language Understanding (NLU) systems. These two tasks are often interdependent, leading to the frequent utilization of multitask learning approaches for jointly performing intent classification and slot filling.
This report presents the development and evaluation of three models, all of which are based on the encoder-decoder paradigm and utilize hard parameter sharing on the encoder.
The first model serves as the baseline, incorporating a simple LSTM encoder and separate classifiers for the slot and intent tasks.
The second model is an improvement on the baseline, utilizing a bidirectional two-layer LSTM encoder.
The third model changes the LSTM with a GRU and introduce the attention mechanism.
All the code is available in my GitHub profile. [1]

2. Task Formalisation

The objective of this project is to effectively perform joint slot filling and intent classification utilizing a multitask learning approach. The use of multitask learning allows for the model to learn shared features between the two similar tasks of slot filling and intent classification. Intent classification refers to the identification of the customer's intent by analyzing the language they use, while slot filling involves identifying specific parameters of the user's query through the analysis of a running dialog. The proposed approach involves utilizing a shared encoder to map the input representation of words and sentences to a hidden representation that captures the relevant features of the input, followed by task-specific decoding to predict the intent or slot based on the hidden representation.
In the accompanying notebook, 5 examples of intent classification and slot filling for user queries taken from the ATIS and SNIPS datasets can be found, such as the **query**: *'how many flights does delta have with a class of service code f'* which has an **intent** of *'quantity'* and **slots**: *'O O O O B-airline_name O O O O O O O B-fare_basis_code'*.
The initial performance of the baseline models was as follows:

- ATIS: Slot F1 Score of 92% and Intent Accuracy of 94%;
- SNIPS: Slot F1 Score of 80% and Intent Accuracy of 96%.

The objective of this study is to develop and evaluate two models that surpass the performance of the baseline models.

3. Data Description & Analysis

The proposed models were trained and evaluated on two publicly available benchmark datasets: ATIS and SNIPS. [2]

ATIS: The Airline Travel Information System (ATIS) dataset consists of audio recordings and corresponding manual transcripts of human inquiries related to flight information on automated airline travel inquiry systems. The dataset is comprised of 4978 (`len(ATIS_tmp_train_raw)`) samples in the training set and 893 (`len(ATIS_test_raw)`) samples in the test set. The validation set was created using the method outlined in the course, resulting in a final distribution of 4381 (`len(ATIS_train_raw)`) samples for training, 597 (`len(ATIS_dev_raw)`) samples for validation, and 893 (`len(ATIS_test_raw)`) samples for testing.
The dataset contains 129 distinct slot labels and 26 intent labels.

```
'intent': 'flight',
'slots': 'O O O O O B-fromloc.city_name O
          B-toloc.city_name'
         'B-toloc.state_code O
          B-depart_date.day_name '
         'B-depart_time.period_of_day O
          B-depart_date.day_name'
         'B-depart_time.period_of_day',
'utterance': 'show me the flights from
              atlanta to washington dc on
              wednesday'
             'night and thursday morning'
```

SNIPS: The SNIPS dataset is more complex and larger than ATIS, with 13084 (`len(SNIPS_train_raw)`) samples in the training set, 700 (`len(SNIPS_test_raw)`) samples in the test set, and 700 (`len(SNIPS_dev_raw)`) samples in the validation set. The dataset contains a total of 7 intent labels and 72 slot labels, and a vocabulary length of 10621.

```
'intent': 'SearchScreeningEvent',
'slots': 'O O B-movie_name I-movie_name
          I-movie_name I-movie_name O O'
         'B-location_name I-location_name
          I-location_name',
'utterance': 'when is eye of the spider
              playing at regal entertainment
              group'
```

4. Models

4.1. ModelIAS

The *ModelIAS* (First Model) is the baseline model of my project. It has an embedding layer which is used to convert the input utterances into a numerical representation. The model also has an LSTM encoder which is used to encode the input utterances. The model also has two linear layers, one for intent classification and one for slot filling. These layers are used to convert the encoded input into the output intent and slot labels. The forward pass of the model takes in an input utterance and a sequence of lengths and it returns the slot and intent predictions. During the forward pass, the input utterance is first passed through the embedding layer, then the LSTM encoder, and finally through the linear layers for intent and slot predictions. The forward pass also uses `packed_sequence` and `pad_packed_sequence` to handle variable-length sequences.
So the structure is:

- Embedding layer
- LSTM
- Output layers for slot filling and intent classification

4.2. ImprovedModelIAS

The *ImprovedModelIAS* (Second Model) is an improved version of the baseline model, ModelIAS. It has the same architecture as the baseline model but with some changes to improve its performance.
Firstly, the ImprovedModelIAS uses a bidirectional LSTM in the encoding layer. This means that it processes the input sequence in both forward and backward directions and concatenates the last hidden state of both directions to produce the final hidden state. This allows the model to take into account the context before and after a given word in the input sequence, which can improve the performance.
Secondly, the ImprovedModelIAS adds a dropout layer after the embedding layer. This helps to prevent overfitting by randomly dropping out some of the embeddings during training. It is a regularization technique that can improve the model's generalization performance.
Finally, the ImprovedModelIAS has an additional train_loop_second function. This function is similar to the train_loop_first function, but it allows for more flexibility in the weighting of the slot and intent losses and for the introduction of L2 regularization. It allows to weight the different losses (slots and intent) differently and to add a regularization term to the loss. It can help to prevent overfitting by adding a penalty to the loss for large weights.
So the structure is:

- Embedding layer
- BiLSTM
- Dropout
- Output layers for slot filling and intent classification

4.3. AttentionModelIAS

The *AttentionModelIAS* (Third Model) is a type of neural network that performs both intent classification and slot filling in a multitask learning setting.
The model consists of several parts, including an embedding layer, a GRU (Gated Recurrent Unit) encoder, an attention mechanism [4], and two classifiers (one for intents and one for slots). The embedding layer maps the input sequences of token IDs to dense vector representations, which are then fed into the GRU encoder. The GRU encoder takes these embedded representations and encodes the sequence into a hidden representation, capturing its important features.
The attention mechanism computes attention scores for each time step in the encoded sequence, which are used to weigh the importance of each time step for the intent classification and slot filling tasks. The intent classifier outputs the predicted intent based on the weighted context representation generated by the attention mechanism. The slot classifier outputs the predicted slots for each time step in the encoded sequence, also weighted by the attention mechanism.
Attention is used for this type of problem because it allows the model to focus on the most relevant parts of the input sequence when making predictions. By assigning higher attention weights to the most important time steps, the model can effectively capture the most relevant information for each task, without being overwhelmed by irrelevant or noisy input. Additionally, attention mechanisms can also handle input sequences of varying lengths.
One of the key features of this model is the ability to run in both a bidirectional and unidirectional mode, by changing the value of the boolean variable *bidirectional* from False to True. This allows for the study of different scenarios within the same code. Additionally, like the AttentionModelIAS, this model also has a different train_loop, which will be explained in more detail in the following section.
So the structure is:

- Embedding layer
- Encoder (GRU) with Dropout
- Attention layer
- Output layers for slot filling and intent classification

5. Evaluation

5.1. Metrics

The evaluation of the various models was conducted with the aim of determining their performance on the task of intent classification and slot filling. The project specifications outlined two main metrics to be used for the evaluation, namely the F1 score for slot filling and accuracy for intent classification. [3]

- The **F1 score** is a well-established measure for evaluating the accuracy of a model. It provides a comprehensive assessment of the performance of a model by considering both precision and recall. Precision refers to the fraction of true positive predictions made by the model, whereas recall refers to the fraction of actual positive cases that are correctly identified by the model. The F1 score is calculated using the conll.py script and takes into account the true positives, false positives, and false negatives, while ignoring the individual classes, by applying a microaverage approach.

$$F1_{score} = \frac{2 \times (Precision \times Recall)}{Precision + Recall} \quad (1)$$

- **Accuracy**, on the other hand, is a simple and straightforward metric that measures the proportion of correct predictions made by a model compared to the total number of predictions. This metric provides a direct and intuitive assessment of the performance of a model, making it an important measure of model performance.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2}$$

Given the limited size of the datasets used in this project, it was deemed necessary to perform multiple runs of each model to ensure a robust evaluation of the models. The number of runs was chosen to be five, and the average performance and standard deviation were calculated for each model and dataset. This approach helps to account for any random variations that may occur during the evaluation and provides a more comprehensive comparison between models.

In conclusion, the evaluation of the models was conducted in a rigorous and systematic manner, using two important performance metrics, the F1 score for slot filling and accuracy for intent classification. The multiple runs of each model and the calculation of the average performance and standard deviation helped to ensure a comprehensive and robust evaluation of the models.

5.2. ModelIAS results

In the baseline training phase, the results obtained were found to be comparable to the ones reported in the assignment. The performance was evaluated using two datasets, the Airline Travel Information System (ATIS) and the SNIPS.

For the ATIS dataset, the Slot F1 score was 0.925 with a standard deviation of 0.005, and the Intent Accuracy was 0.935 with a standard deviation of 0.001. Similarly, for the SNIPS dataset, the Slot F1 score was 0.807 with a standard deviation of 0.008 and the Intent Accuracy was 0.961 with a standard deviation of 0.005.

Upon conducting five consecutive runs, the variance in the results was found to be on average 0.005, and it was observed that the variance was consistent regardless of the dataset used.

The baseline utilizes a loss function that is defined as the sum of the intent loss and slot loss, mathematically represented as:

$$Loss = Intent\ Loss + Slot\ Loss \tag{3}$$

5.3. ImprovedModelIAS results

In the ImprovedModelIAS the improvement comes from the use of a bidirectional LSTM for encoding the utterance, and adding a dropout layer to prevent overfitting. The results of this model are better compared to ModelIAS.

For the ATIS dataset, the Slot F1 score was 0.944 with a standard deviation of 0.002, and the Intent Accuracy was 0.956 with a standard deviation of 0.003. Similarly, for the SNIPS dataset, the Slot F1 score was 0.874 with a standard deviation of 0.003 and the Intent Accuracy was 0.963 with a standard deviation of 0.003.

Here I tried to improve the model including the options to control the weight given to the intent and slot loss, as well as the option to add a regularization term to the loss. This allows for more control over the optimization process and can lead to improved performance by adjusting the weighting of the different losses, or by avoiding overfitting with regularization.

The weighted sum of the intent loss and slot loss is calculated as the total loss in the following formula:

$$\begin{aligned} Loss = intent_{weight} \times Intent\ Loss +\\ slot_{weight} \times Slot\ Loss +\\ regularization \times L2_{loss} \end{aligned} \tag{4}$$

5.4. AttentionModelIAS results

The final model, AttentionModelIAS, is split into two, monoidirectional and bidirectional, and each was tested with all the loss functions used for the first two models as well as a new loss function, the one in the train loop of the bidirectional version of the third model.

The new loss function takes the maximum value of either the loss for intents or the loss for slots, weighted by the maximum of two random weights, and adds it to the minimum value of either the loss for intents or the loss for slots, weighted by the minimum of two random weights (w_1,w_2). This loss function was used because it allows to balance the two losses and gives more weight to the task with larger loss [5].

$$\begin{aligned} Loss = max(Intent\ Loss, Slot\ Loss) \times max(w_1, w_2)+\\ min(Intent\ Loss, Slot\ Loss) \times min(w_1, w_2) \end{aligned} \tag{5}$$

For the case monoidirectional version, for the ATIS dataset, the Slot F1 score was 0.926 with a standard deviation of 0.003, and the Intent Accuracy was 0.955 with a standard deviation of 0.005. Similarly, for the SNIPS dataset, the Slot F1 score was 0.822 with a standard deviation of 0.01 and the Intent Accuracy was 0.96 with a standard deviation of 0.003. On the other hand, for the bidirectional model, the Slot F1 score was 0.944 with a standard deviation of 0.002 and the Intent Accuracy was 0.952 with a standard deviation of 0.003 for the ATIS dataset, and the Slot F1 score was 0.88 with a standard deviation of 0.003 and the Intent Accuracy was 0.965 with a standard deviation of 0.003 for the SNIPS dataset.

5.5. Confusion matrices

Confusion Matrix is a tool used in machine learning to evaluate the performance of a classification model. It is a N x N matrix, where N is the number of classes, and it is used to compare actual and predicted values. By analyzing the values in the matrix, one can gain insight into the performance of the model, including its accuracy, precision, recall and more.

The Confusion Matrix is constructed by calculating four key terms: True Positive (TP), False Positive (FP), True Negative (TN) and False Negative (FN). These values provide a summary of how well the model is performing in terms of classifying the data.

In the report, three different confusion matrices are shown, each representing the performance of a different model. Each matrix shows the results of both slot filling and intent classification for both ATIS and SNIPS datasets. The use of colors in the matrices indicates the frequency of prediction. The more the color tends towards yellow, the more frequently that particular class was predicted.

One important aspect to consider when examining these matrices is the vertical line corresponding to the label *flight* in the ATIS matrix and a vertical line corresponding to the label *O* in both the slot filling and intent classification matrices. This is because *O* is the most frequent label and as a result, all models tend to overpredict this label.

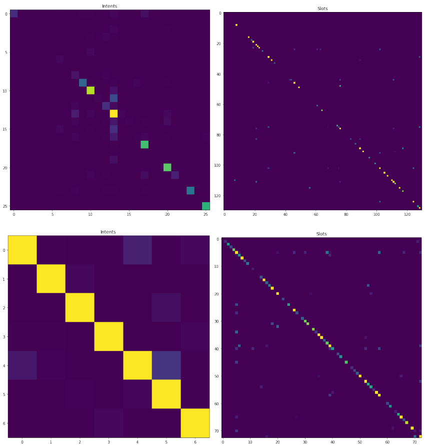

Figure 1: *Intents and slots cm for ATIS and SNIPS in ModelIAS*

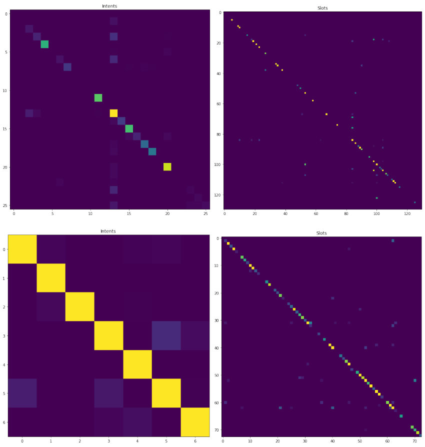

Figure 2: *Intents and slots cm for ATIS and SNIPS in ImprovedModelIAS*

6. Conclusion

The study presented in this report aimed at investigating and improving the performance of the joint slot filling and intent classification task. The baseline model was already been improved by 2/3% with the implementation of the ImprovedModelIAS.
The use of bidirectionality in the model architecture was identified as having the most significant impact on performance. Subsequently, other values were carefully selected to optimize the model's performance, such as the dropout value which was tested from 0.1 to 0.9 before being set at 0.2 as it showed the best performance. Furthermore, the choice of loss function was also tested before being assigned to a specific model.
In addition, one potential strategy to enhance the performance of the model would be to substitute the LSTM layer with a GRU layer. The advantage of this change is that GRUs have fewer parameters compared to LSTMs, making them computationally more efficient and faster to train and predict. Additionally, GRUs have been shown to have a reduced risk of overfitting due to the number of parameters, which can potentially lead to improved accuracy in the model [6].
It is important to note that other modifications, such as changing the seed or patience, could also impact performance and were not considered in this study due to the computational limitations of testing both ATIS and SNIPS datasets.
Future work could include the implementation of new solutions and improvements to the presented models, as well as further analysis into how the model transfers knowledge between slot filling and intent classification tasks. This could lead to the adoption of new strategies that could increase the overall performance of the model.

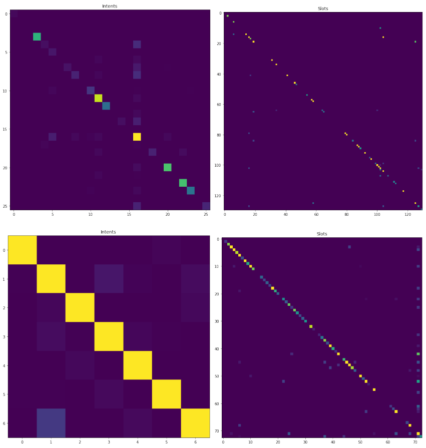

Figure 3: *Intents and slots cm for ATIS and SNIPS in AttentionModelIAS (bidirectional)*

7. References

[1] Faccin Pierluca, "https://github.com/pierlucafaccin/NLU-Slots-Intents" *My GitHub*.

[2] Brown Fortress, "https://github.com/BrownFortress/IntentSlotDatasets" *IntentSlotDatasets*.

[3] Purva Huilgol, "https://medium.com/analytics-vidhya/accuracy-vs-f1-score-6258237beca2" *Accuracy vs. F1-Score*, 2019 August 24.

[4] Michel Cana, "https://towardsdatascience.com/practical-guide-to-attention-mechanism-for-nlu-tasks-ccc47be8d500" *Practical guide to Attention mechanism for NLU tasks*, 2019 Septembre 08.

[5] Lin, Baijiong and Ye, Feiyang and Zhang, Yu and Tsang, Ivor W., "https://arxiv.org/abs/2111.10603" *Reasonable Effectiveness of Random Weighting: A Litmus Test for Multi-Task Learning*, arXiv 2021.

[6] Chung, Junyoung and Gulcehre, Caglar and Cho, KyungHyun and Bengio, Yoshua, "https://arxiv.org/abs/1412.3555" *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*, arXiv 2014.