Final NLU project
Intent Classification and Slot Filling

Pietro Demurtas (229707)

University of Trento

pietro.demurtas@studenti.unitn.it

1. Introduction

In the current work the final submission of the NLU course is presented. In order to address the intent classification and slot filling task 3 different models have been implemented. The first and baseline model is an adaptation of the LSTM model seen in class, the second model is an extension of the baseline model with some improvements to overcome the its shortcomings. The final model is an implementation of S-o-A techniques that have been proven outstanding in solving this task. The 3 models share some components, this is done not only because one is the improvement of another but also to get a common ground to test the influence of the differences on performance.

2. Task Formalisation

The NLP task of intent classification and slot filling can be divided into two subtasks. The first task is the intent classification one. The intent classification task is a text classification task where the goal is to classify the general intent expressed by the utterance. (i.e. PlayMusic or GetWheter) The second task, slot filling, involves assigning a domain specific label to each world of the utterance. (i.e. B-artist, B-track) The domain specific labels are usually properties and/or information related to the overall intent of the utterance. It is important to note that most words of the utterance do not represent valuable domain-specific knowledge and as such they don't have a specific label assigned and get labeled as `O`. Since we are interested in **joint** intent classification and slot filling our models should perform both tasks at the same time. In order to do so all of the models architectures will be composed of an utterance encoder and two task specific classifiers which will fulfill the tasks using the utterance representation provided by the encoder.

3. Data Description & Analysis

For intent classification and slot filling usually two bechmark datasets are used, namely ATIS and SNIPS. The two datasets have different peculiarities which will be described in details.

3.1. General description and origin

3.1.1. ATIS-Airline Travel Information Systems

It is a dataset made up of manual transcripts of audio recordings of people querying automated airline travel inquiry systems about flights. The first version of this dataset was published in 1990 [1] and has undergone several improvements and extensions [2], the dataset version used in this project has been taken from Microsoft CNTK. It has been used for a variety of different NLU tasks related to intent classification and slot filling like text-to-SQL conversion.

3.1.2. SNIPS

The SNIPS dataset is composed of data collected from public or commercial sources by the Snips corporation which develops home assistants and voice control support for IoT devices. The first version of the dataset with audio recordings and transcripts was introduced in 2018 [3].

3.2. Datasets structure

Both datasets are provided in a handy `.json` format. Each example of both datasets has three entries:

- "Utterance" which contains the phrase itself
- "Slots" which contains the slots labels, one for each word in the utterance
- "Intent" which contains the intent label for the whole example

ATIS dataset is composed of **5871** examples in total, **4978** in the training set and **893** in the test one. Since the original dataset does not provide a dev set it is obtained by splitting the training set with final lengths of training and dev set of **4380** and **598** respectively.
The SNIPS dataset on the other hand already comes with training, dev and test sets composed of **13084**, **700** and **700** examples respectively.

3.2.1. Utterance composition

The ATIS dataset has a vocabulary length of 863, in contrast SNIPS has a much larger vocabulary with a length of 10621. As can be seen in figure 1 ATIS utterances are generally longer.

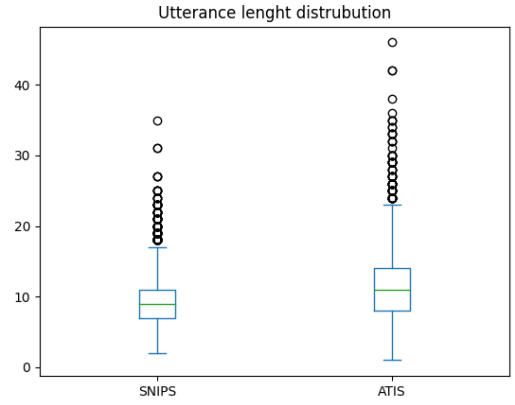


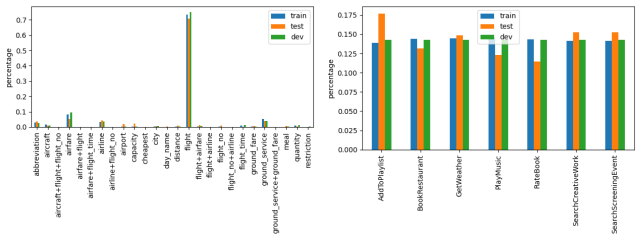
Figure 1: *Utterance length distribution*

3.2.2. *Slots labels*

ATIS dataset contains 129 slot labels, with the most represented label being 'B-toloc.city_name'. Most labels are equally present in all 3 datset exept for 8 labels which are only present in the test set. SNIPS dataset has 72 slots labels with the most represented one being "I-object_name". The slot labels are almost equally represented in all three datasets with no category missing in any dataset.

3.2.3. *Intent labels*

ATIS contains 22 different intent labels with the "flight" class being by far the most represented one being present in the 73% of the example as can be seen in figure 5a. Intent labels are distributed almost equally among the three datasets except for "airfare+flight", "day_name", "flight+airline" and "flight_no+airline" which miss form the training set. In order to address this problem a stratified split has been tried but since some of these labels where only present in a single copy the stratified split was not possible. SNIPS on the other hand has only 7 intent categories which are almost equally distributed among the 3 sets as can be seen from figure 5b .



(a) *ATIS* (b) *SNIPS*

Figure 2: *Intent class distribution*

4. Models

In general all three models are composed of a part responsible for extracting or encoding the information of each utterance and a part responsible to perform classification for the two specific tasks using as input the representation provided by the encoder. A general comparison chart of the 3 implemented models can be seen in figure 3

4.1. Common features

I first proceed in describing the common features of all 3 models.

4.1.1. *Tokenizer and custom dataset constructor*

From the start of the project I decided to implement a model making use of the BERT encoder. Due to this reason I decided include the BERT pretrained tokenizer in all 3 models in such a way to ensure a uniform processing of data. The BERT tokenizer differs from rule-based tokenizers, it is in fact a WordPiece tokenizer, first introduced in 2016 by Wu et al [4]. WordPiece tokenizers split some word into sub-words allowing the models to learn the meaning of several sub-fixes and prefixes. In addition to the tokenization process we need to encode the features as numerical labels, for the intent labels the progressive numeral labeling as seen in class is straight forward, for slots label the process requires more attention. Since I used a WordPiece tokenizer the tokenized utterance has no more the same length as the slot label string, in order to address this problem we assign the original slot label to the first token of a given word and we add padding labels to the consecutive tokens belonging to the same word. Finally, the tokenized dataset together with the generated attention mask is cached on the hard-drive in form of torch tensors. I did this rather than customizing the torch class because the preprocessing is computationally expensive and thus it saves time during training.

4.1.2. *Optimizer*

All 3 models have been optimized with the AdamW optimizer.

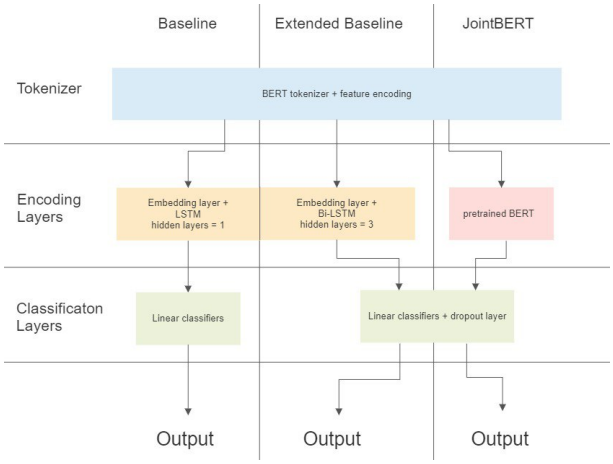


Figure 3: *Comparison chart showing the general architecture of the implemented models*

4.2. Baseline model

The baseline model is a simple LSTM model as seen during lectures which has been adapted to use the differently tokenized dataset.

4.3. Extended Baseline model

After training the baseline model, experiments where made trying to extend it's architecture and overcome the baseline model' shortcomings.
In the first place experiments where made with the encoder part of the network. The LSTM layer was replaced by a Bi-LSTM layer and next experiments where made on the number of Bi-LSTM layers to use in the encoder. In the end, 3 layers was found to be a good compromise between performance and computational cost. For the classification part a drop-out layer was added just before each linear classifier to prevent overfitting sice we are workig with small datasets. I also experimented with different loss coefficients which can modify the impact of the single losses on the total one, however since no significant improvement was found I decided to omit it in the final model.

4.4. JointBERT

As the final model I decided to implement the JointBERT model which has shown outstanding performances [5]. I implemented a model which makes use of the **bert-base-uncased** pretrained transformer model from the Huggingface' transformers library

[6] as the feature extractor. The classification part is carried out by two dedicated liner classifiers preceded by a dropout layer as in the extended baseline model.

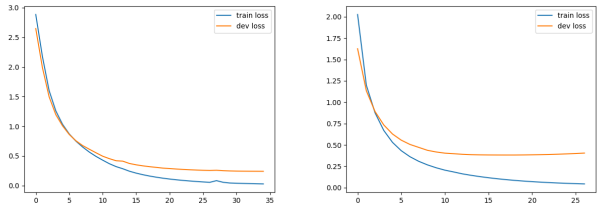
5. Evaluation

5.1. Metrics

In order to asses the performances of the 3 modes I have used two different metrics; for the slot filling task I used the F-1 score and for intent classification I used accuracy. The first two models were trained from scratch using the patience mechanism over multiple epoch focused on slot F-1; if the slot F-1 score does not increase in a given number of dev set evaluations, the training is stopped early. Given the relatively small size of the models and of the training set, both baseline models are rather quick to train. On the other hand, JointBERT employs a very large pretrained BERT encoder with 110M parameters. JointBERT during training time has to train from scratch only the two task specific classifiers while fine-tuning the transformer's encoding layers. Due to the sheer size of the JointBERT model training takes much more time when compared with the LSTM-based models but requires overall less training steps to reach much better performance as I will discuss shortly.

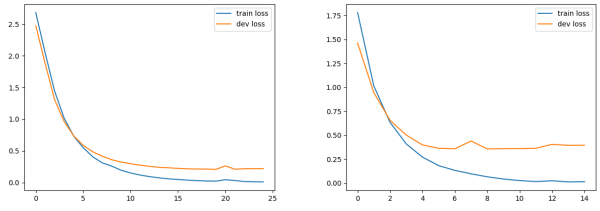
5.2. Losses

All the losses function have been checked and the plots are in accordance with the results obtained and with the composition of the datasets. Moreover, looking at individual plots for intent and slot loss we can see that the intent loss converges much faster as slot filling is the more complex task.



(a) *ATIS* (b) *SNIPS*

Figure 4: *Baseline train and dev loss function*



(a) *ATIS* (b) *SNIPS*

Figure 5: *Extended baseline train and dev loss function*

5.3. Evaluation results

To fully evaluate the models I trained and evaluated on test set each one 5 times from scratch. Metrics' mean and significant variations are reported in tables 1 and 2.

	Baseline	Extended Baseline	JointBERT
Slot F-1	0.963 ± 0.004	0.962 ± 0.008	0.98
Intent acc.	0.934 ± 0.004	0.955 ± 0.006	0.98

Table 1: *ATIS results*

	Baseline	Extended Baseline	JointBERT
Slot F-1	0.892 ± 0.05	0.941 ± 0.02	0.96
Intent acc.	0.962 ± 0.015	0.972 ± 0.023	0.97

Table 2: *SNIPS results*

5.3.1. Baseline

On the both datasets the baseline model has average performances. As expected, the model obtained worst performances on labels with low support but managed to achieve high scores on labels with sufficiently large support. In fact, the model obtains high accuracy on all intent labels of SNIPS dataset which have all almost the same support. It is very important to note that on both datasets the baseline model managed to surpass the baseline slot F-1 score provided in the project report which was obtained with the same model seen in class with the use of a different tokenizer.

5.3.2. Extended Baseline

The extended baseline model reached higher performances on both datasets for both slots and intents with the exception of ATIS slots' F-1 score. Looking at label specific scores in can be seen that the extended baseline model scores slightly higher for high support classes and sports a significant increase for some of the low support classes, both for slots F-1 and intent accuracy. The improved performances of the model over the simple baseline are as to major extent due to increase in encoder size and capability, this can be seen thanks to the increase in performance on the low-support slot labels which means that the encoder is able to capture better the relationships in the utterance. In addition the increase in performance can be also justified to a smaller extent by the use of the new classifiers, however the only difference in this part of the model w.r.t the baseline is the introduction of a dropout layer per classifier to mitigate overfitting.

5.3.3. JointBERT

As expected, the JoinBERT model outperformed both LSTM based models reaching S-o-A performances. Looking at the label-specif performances we can see that JointBERT sports an increased performance in all aspects. In has better performances on both datasets and for both tasks improving label-specific performances for both low and high support classes. The powerful transformer-based encoder was able to capture more information than the other encoders used; this can be seen particularly looking at the Extended Baseline model which employs the same exact architecture for classification. The BERT pretrained encoder leveraged the knowledge stored in it's architec-

ture to grasp more information, understanding as much as possible the relevant information and relevant relationships among the constituent tokens of each utterance, even if situated far off in the same phrase.

6. Conclusion

To conclude, in the present work I have experimented with several architectures all paired with the WordPiece tokenazier that is used in the BERT architecture. The tokenizer showed to improve also the performance of the baseline model over the same architecture used with a different tokenizer as we have seen in class. This most probably is due to the fact that splitting words into subwords not only reduces the vocabulary size, making it easier to deal with for the models, but it also allow the learning of compound meanings of words made of multiple tokens. Moreover from the experiments made in this project it has emerged that for this task, performance wise, the most important part of the models' architecture is the encoder. This is due to the fact that more complex encoders have an higher capability in extracting, or encoding, the relevant information from the text and its semantic relations. Across the 3 models I have tested 3 different encoders with different architectures and sizes while changing only slightly the task specific classifiers and nonetheless a great improvement in performance has been observed. Finally, the BERT architecture has shown its outstanding properties demonstrating once again the capabilities of a very complex architecture pretrained on general tasks in solving a wide variety specific tasks when included in the proper architecture and fine-tuned.

7. References

[1] P. Price, "Evaluation of spoken language systems: The atis domain," in *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27, 1990*, 1990.

[2] D. A. Dahl, M. Bates, M. K. Brown, W. M. Fisher, K. Hunicke-Smith, D. S. Pallett, C. Pao, A. Rudnicky, and E. Shriberg, "Expanding the scope of the atis task: The atis-3 corpus," in *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*, 1994.

[3] A. Coucke, A. Saade, A. Ball, T. Bluche, A. Caulier, D. Leroy, C. Doumouro, T. Gisselbrecht, F. Caltagirone, T. Lavril *et al.*, "Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces," *arXiv preprint arXiv:1805.10190*, 2018.

[4] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," *arXiv preprint arXiv:1609.08144*, 2016.

[5] Q. Chen, Z. Zhuo, and W. Wang, "Bert for joint intent classification and slot filling," *arXiv preprint arXiv:1902.10909*, 2019.

[6] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz *et al.*, "Huggingface's transformers: State-of-the-art natural language processing," *arXiv preprint arXiv:1910.03771*, 2019.