Final NLU Project

Gizem Mert (229716)
University of Trento
gizem.mert@studenti.unitn.it

***Abstract*—This document represents the report of the project developed for the M.Sc. course Natural Language Understanding (NLU) at the University of Trento. The main goal was to develop a pipeline for joint intent detection and slot filling.**

I. Introduction

Multitask learning (MTL) is a field of machine learning where the main goal is to solve multiple problems at the same time. MTL is similar to transfer learning where the information gained by solving one problem is applied to another.

In this work, by employing MTL, I tackle two natural language processing problems: intent detection and slot filling, which can be considered as classification and named-entity detection problems respectively. Detecting the intent of a user's query helps prioritize the prompts and find an accurate response. Finding the key slots of a prompt also helps the responding system to retrieve the correct information. I constructed three deep learning models and trained them on two datasets to discover the best model for this joint task.

II. Task Formalization

Here, I formally define intent classification and slot filling. Given a prompt p from a user containing the tokens $w_1, \ldots, w_l$, the objective of intent classification is to associate a word from the set of intent labels to p as the user's intentions and needs. E.g., the intent of "How can I get from Milan to Trento?" may be "*navigation*". Moreover, the goal of slot filling is detecting the key parameters of the user's query. This corresponds to extracting target entities that will help finding a response for the prompt. E.g., in the same query for intent detection, the slots are *from_city="Milan"* and *to_city="Trento"*.

In other words, for an input utterance, intent detection can be simply defined as a classification task, where the utterance has to be placed into one of the C categories. Each category represents the user's intent. On the other hand, slot filling is slightly trickier. The model has to perform classification on each of the input tokens and also pay attention to the slot spans. Sometimes, slots can span along multiple words, e.g. *from_city* can be *"San Francisco"*. Partial slot filling should be penalized in this case.

III. Data Description Analysis

I used two datasets to train and evaluate a deep learning model capable of intent detection and slot filling: ATIS (Airline Travel Information Systems) [1], which consists of prompts from users of a automated airline travel inquiry system; and SNIPS [2], that is another set of queries from various intents. The datasets are described in the following paragraphs[1]:

a) ATIS: The version of ATIS I used in this project consists of 4978 training and 893 test samples. Each sample is represented by an utterance, i.e. the user's prompt, an intent, and a list of slots corresponding to each word in the query. The vocab size is 891 words and there are a total of 26 intents in ATIS. Slots span between multiple words in an utterance and are marked using the IOB2 tagging scheme, i.e. "B" for a slot's beginning, "I" for a word inside a slot but not its first, and "O" for outside a slot. There are 64 slots in ATIS, each having a "B" and "I" tag. Added with a single "O" slot, the dataset contains 129 total slots.

b) SNIPS: This dataset contains 13084 train, 700 test, and 700 validation samples. The training dataset has a vocabulary size of 11420 words, and there are 7 intents and 72 slots to be targeted by the models.

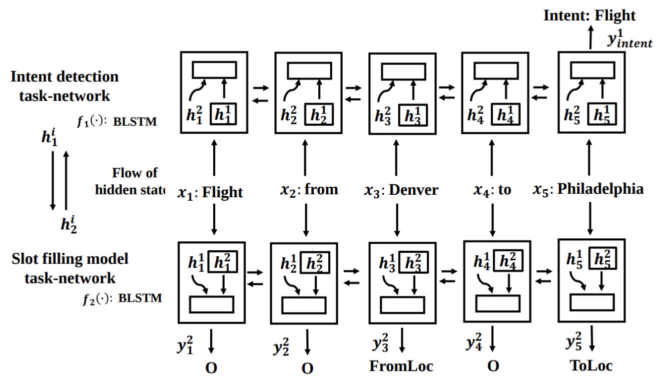


Fig. 1: Bi-Model RNN architecture without a decoder [3]

IV. Models

I trained three different network architectures which are described in the following sections.

A. Bi-Model RNN

Originally designed by Wang et al. [3], Bi-Model RNN[2] consists of two identical bi-directional LSTM [4] models, one for predicting the intent and one for the slots. As shown in Fig. 1, the flow of hidden states between the two LSTM encoders connects the information learned by the model about intents and slots.

In my setup, I used a Bi-Model RNN without a decoder for faster computation. Each encoder contains two LSTM layers

[1]Datasets are taken from https://github.com/BrownFortress/IntentSlotDatasets.
[2]Code taken from https://github.com/ray075hl/Bi-Model-Intent-And-Slot.

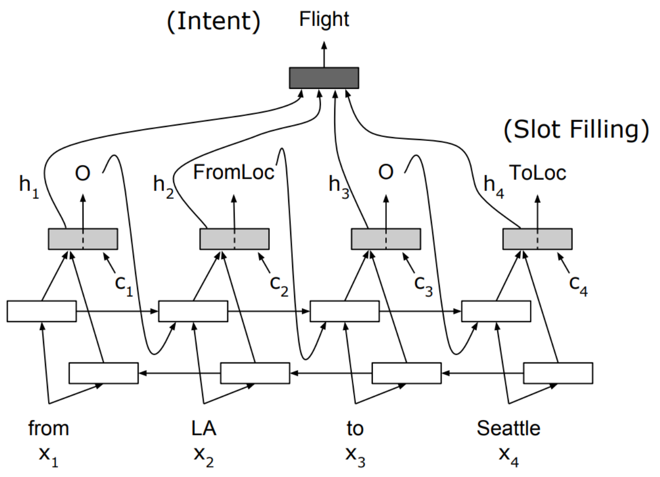


Fig. 2: Attention Bi-RNN architecture [5].

with an hidden dimensionality of 200. The intent detection LSTM takes the hidden states of the slot filling LSTM and vice versa. Beforehand, the utterance tokens are fed into an embedding layer that produces a 300 dimension vector for each word. The LSTM outputs are finally used as input for two linear layers to find logits for the output intent and slots. In each time-step, the slot for the input token is predicted; whereas the output intent is only taken from the model at the last token.

B. Bi-LSTM + Self-attention

With the rise of the attention mechanism's popularity, many models, such as the one developed by Liu et al. (Attention Bi-RNN, Fig. 2) [5], have been applied to the joint intent classification and slot filling task. Attention Bi-RNN is built upon an bidirectional LSTM encoder which receives the input token embeddings and feeds output and hidden states to another LSTM layer. The second layer takes context information as a weighted average of the first layer's output, as per the attention mechanism. Finally, intent detection and slot filling is achieved using the outputs of the second layer.

I designed a network based on the work of Liu et al. [5], as illustrated in Fig. 3. This architecture consists of a Bi-LSTM encoder with a hidden size of 100 and a multi-head self-attention layer. The Bi-LSTM encodes the word embeddings (300 embedding dim.) to vectors and the multihead attention layer discovers semantic dependencies between the encodings. This setup improves the training speed due to the superior performance of the attention mechanism compared to LSTM layers. I show this further in Section V. The intent and slot output layers are linear, where the intent takes the sum of the attention layer's output as input.

C. BERT

BERT [6] is a transformer-based network [7] pre-trained on vast datasets for masked language modeling and next sentence prediction. It can be fine-tuned for various tasks, including intent prediction and slot filling. Chen et al. [8] have performed

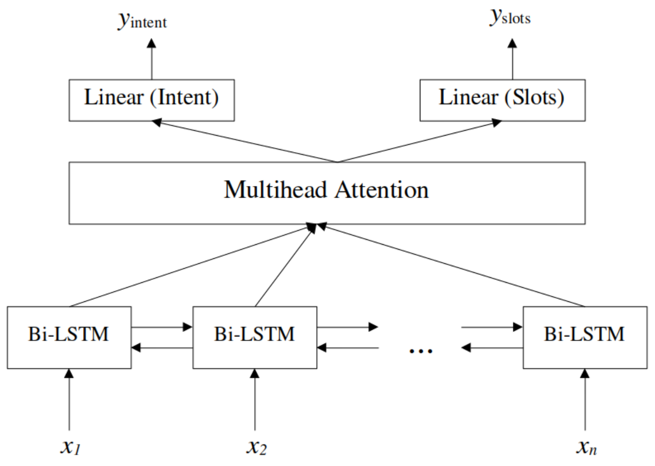


Fig. 3: Bi-LSTM + self-attention architecture.

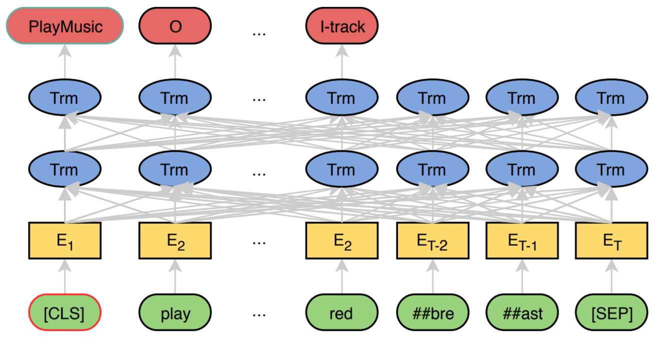


Fig. 4: BERT for joint intent classification and slot filling [8].

this experiment by feeding a prompt's utterance to BERT and taking the first token's output for intent prediction and all the resulting hidden states for slot filling (Fig. 4).

To prepare the input for BERT, a [CLS] token must be added to the beginning of the token list. This special token is used for the next sentence prediction task in the pre-training stage and represents the whole sentence as a single vector. I use this vector for intent prediction to fine-tune BERT using a linear layer. The remaining outputs are fed to another linear layer for slot filling. Moreover, BERT employs a WordPiece tokenizer which splits complex words into smaller tokens (e.g. *learning* into *learn* and *##ing*). For slot detection, we take the slot predicted for the first token in such splits and rejoin the rest of the tokens, discarding any output computed for the slot.

V. Evaluation

In this section, I compare the performance of the three models described above.

A. Model Loss

I define each model's prediction error as the sum of the cross-entropy between the ground truth intents and slots and the predicted intent and slot probabilities. Formally, if there are I possible intents and S slots, given a sample of T tokens, the total loss of a model is computed as (1). Ground truth intents and slots are represented as $y^{(i)}$ and $y^{(s)}$, where the model's output is $\hat{y}^{(i)}$ and $\hat{y}^{(s)}$.

$$E = -\sum_{k=1}^{I} \hat{y}_k^{(i)} \log(y_k^{(i)}) - \sum_{i=1}^{T}\sum_{j=1}^{S} \hat{y}_{i,j}^{(s)} \log(y_{i,j}^{(s)}) \tag{1}$$

In addition, I perform L2 regularization with a factor of $\lambda = 0.001$ to prevent over-fitting. Therefore, the total loss of a model with the parameters W is defined as (2).

$$\mathcal{L} = E + \lambda \sum_{w \in W} w^2 \tag{2}$$

B. Metrics

I used two metrics for measuring the quality of a model's intent classification and slot filling results: accuracy and F1 score.

a) Intent accuracy: The percentage of correct guesses for the intents of a batch of utterances.

b) Slot filling F1 score: The harmonic mean of the prediction's precision and recall. Output slots are aligned to the ground truth using the CoNLL script.

C. Experiment Setup

I ran the experiments in a Google Colab notebook with GPU enabled. Model sizes and configurations were presented in Section IV. I trained each model for a different number of epochs using early stopping. The SNIPS dataset already had a validation set, whereas for ATIS, I used 10% of the training data for validation. Each experiment is run 5 times and the average and standard deviation of the metrics are reported. I used an Adam [9] optimizer to minimize the loss.

D. Baseline

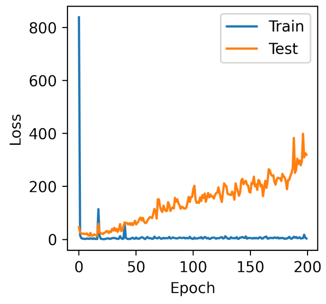
The baseline model consists of an LSTM encoder which feeds input to the two linear layers for intent and slots.

E. Results and Comparison

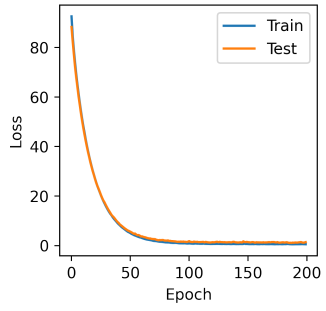
Table I illustrates the results for each model on the datasets. I also show the effect of utilizing L2 regularization on the Bi-LSTM + Self-attention approach in Fig. 5, where it is observed that this method leads to a smoother learning curve. Also, the increase of intent accuracy and slot filling F1 score during the training process is also shown in Fig. 6 and Fig. 7 for Bi-LSTM + Self-attention for both datasets.

VI. Analysis

The superior performance of Bi-Model RNN demonstrates that having multiple encoders for each task instead of a single model improves MTL. The caveat of this approach however is the decreased training speed. Interestingly, the accurate results on ATIS were not reproducible for SNIPS for Bi-Model RNN. Especially the intent accuracy which was significantly low. This is due to the fact that the code of the original paper [3] was not available, and the code we used for our

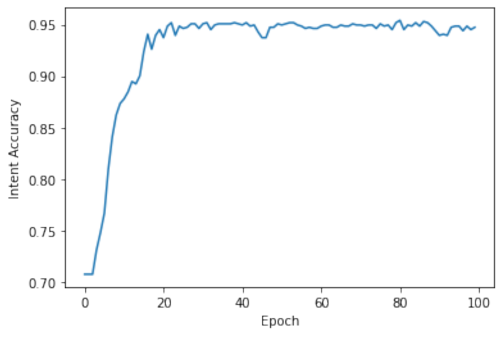


(a) Without regularization

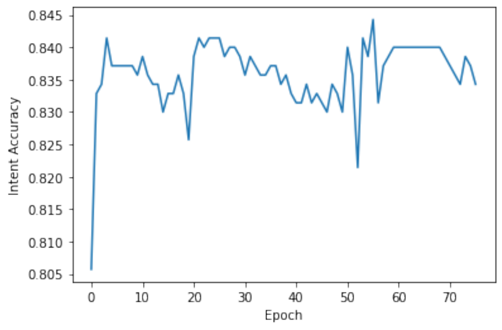


(b) With regularization

Fig. 5: Effect of L2 regularization on Bi-LSTM + Self-attention training.



(a) ATIS

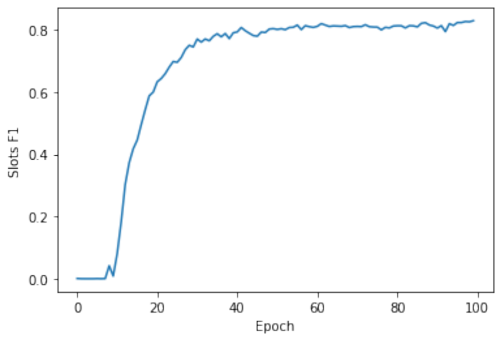


(b) SNIPS

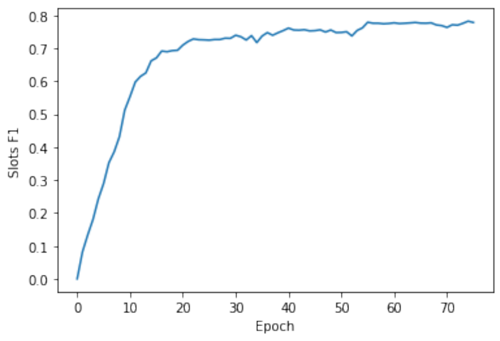
Fig. 6: Model improvement at intent detection during training.

TABLE I: Model evaluation results.

Model	ATIS		SNIPS	
	Intent Accuracy (%)	Slots F1 (%)	Intent Accuracy	Slots F1
Bi-Model RNN [3]	**98.76 ±0.18**	**96.65 ±0.14**	17.71 ±0.03	84.75 ±0.12
Bi-LSTM + Self-attention	96.75 ±0.16	86.57 ±0.08	84.85 ±0.07	79.95 ±0.13
BERT [10]	98.10 ±0.32	96.40 ±0.21	**97.30 ±0.16**	**96.03 ±0.12**
Baseline	94.0	92.0	96.0	80.0



(a) ATIS



(b) SNIPS

Fig. 7: Slot filling F1 increase during training.

implementation also lacked experiments on SNIPS. Therefore, the results of Bi-Model RNN on SNIPS may be discarded.

On the other hand, transformers and the attention mechanism are much faster than RNNs at processing input data. The self-attention architecture I used was 43.27% faster than Bi-Model RNN on ATIS (1.73s/epoch vs. 3.05s/epoch) and 29.81% faster on SNIPS (3.86s/epoch vs 5.50s/epoch) on average. However, attention layers usually consist of a large number of learned parameters, which contributes to over-fitting. If the size of the dataset is insufficient (e.g. in ATIS), a bi-directional LSTM coupled with self-attention may not perform better than a bi-model due to an improper fit.

Finally, employing a pre-trained model (e.g. BERT) and fine-tuning it on the ATIS and SNIPS datasets enables us to benefit from prior language information. As opposed to the other two models where the initial weights are assigned randomly, pre-trained models start training on parameters that have already been learned on vast datasets for different tasks. This approach has comparable performance to Bi-LSTM + Self-attention and also results in similar accuracy to Bi-Model RNN.

Deep learning models usually suffer from over-fitting when the available training data is not enough to learn a huge number of parameters. In these cases, methods such as dropout or regularization reduce the complexity of the network and allow the optimization algorithm to follow a smoother trajectory (Fig. 5).

VII. Conclusion

In this project, I trained and tested three different models on the ATIS and SNIPS datasets for joint intent detection and slot filling. The key results are that using two encoder models based on LSTMs is the slowest approach but leads to better intent accuracy and F1 score. While attention-based solutions are faster, employing them on a dataset of insufficient size may lead to over-fitting. Another approach was to fine-tune a pre-trained transformer model such as BERT on the two datasets. This solution lead to a higher performance and comparable accuracy to a bi-model network.

References

[1] C. T. Hemphill, J. J. Godfrey, and G. R. Doddington, "The ATIS spoken language systems pilot corpus," in *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27,1990*, 1990.

[2] A. Coucke, A. Saade, A. Ball, T. Bluche, A. Caulier, D. Leroy, C. Doumouro, T. Gisselbrecht, F. Caltagirone, T. Lavril, M. Primet, and J. Dureau, "Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces," *CoRR*, vol. abs/1805.10190, 2018.

[3] Y. Wang, Y. Shen, and H. Jin, "A bi-model based rnn semantic frame parsing model for intent detection and slot filling," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pp. 309–314, 2018.

[4] M. Schuster and K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[5] B. Liu and I. Lane, "Attention-based recurrent neural network models for joint intent detection and slot filling," *Interspeech 2016*, pp. 685–689, 2016.

[6] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, (Minneapolis, Minnesota), pp. 4171–4186, Association for Computational Linguistics, June 2019.

[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[8] Q. Chen, Z. Zhuo, and W. Wang, "BERT for joint intent classification and slot filling," *CoRR*, vol. abs/1902.10909, 2019.

[9] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR (Poster)*, 2015.

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