NLU course projects - Lab 5 (NLU)

Matteo Mascherin (247183)

University of Trento

matteo.mascherin@studenti.unitn.it

1. Introduction

In this lab, slot filling and intent classification over a sequence are performed. The segmentation of the input sequence is represented with IOB tags and the labeling part are the concepts defined in the annotation schema of the ATIS corpus. While for intents, there is one label for each sequence which has to be correctly classified. To face this challenge a pretrained Bert model is used to reach the best possible results. The main challenge consists in adapting the system to include Bert tokenization process inside the whole pipeline.

2. Implementation details

The neural architecture consists in three main layer, the first one to perform the embedding of a certain token, the second one to compute the dependencies between tokens through a LSTM and the third one is composed of two linear layer, one for each task. The architecture is then improved using a Bert model substituting both the embedding and LSTM layer.

2.1. Part 1

In order to address slot filling task, the whole output of the LSTM over the sequence is used. While in intent classification we just consider the last hidden state which contains the output after the entire sequence is examined. For the first part of the assignment two modification were applied:

- dropout layers are added
- bidirectional LSTM is used

All the modifications are implemented using standard torch library. The dropout layers are positioned after the embedding layer. For the experiments a dropout with probability of 0.1 is applied. Once bidirectionality is activated, to join the two hidden states for each token, both concatenation and sum has been tested.

2.2. Part 2

In the second part the paper of Qian Chen et al [1] was a reference to further improve the performance of the model. A pretrained Bert model is used to substitute the embedding and the LSTM layer. Bert is inserted in the architecture and the input size of the two output layers are changed to meet the Bert hidden size. The class BertConfig is used to set the dropout inside Bert model. Code from part 1 is refined to insert BertTokenizer to tokenize the sequences in particular:

- slot2id and intent2id dictionaries are kept unchanged since they do not require tokenization
- word2id dictionary is replaced by BertTokenizer

In case a word is tokenized in multiple tokens, the corresponding slots are padded accordingly to reach equal sizes. Tokenization is done word by word following an unofficial implementation of the reference paper [2].

Model	**ATIS**		
	Params	**Intent**	**Slot**
LSTM	1.1M	94.4	92.2
+ dropout (concat)	1.1M	94.8	94.4
+ dropout (sum)	1.1M	95.0	94.0
Frozen Bert	0.12M	78.5	77.9
Bert	109M	**97.87**	**95.34**

Table 1: *Results in term of accuracy and f1 score for intent and slots respectively*

3. Results

The intent classification performance is measured in term of accuracy, while for slot filling is measured in term of F1 score. The result for the different experiments are shown in 1.

3.1. Part 1

Analyzing the model based using LSTM cells, turns out that multiple techniques can be used to concatenate the bidirectional output we have for each token. Even if the differences are difficult to catch in such a small dataset (ATIS), we have comparable results for both experiments. While using sum as form of join between the two hidden states reduce the model complexity, concatenation is the safest way to keep all the information extracted by the LSTM. With regard to training, default parameters are maintained and good results are achieved in 200 epochs, a learning rate of 1e-4 and AdamW optimizer.

3.2. Part 2

In part 2, performance are further boosted using a pretrained Bert model. In order to perform the two subtask, from Bert output the last hidden state is extracted for slot filling and then is again filtered to keep just the information referred to the [CLS] token for intent classification. This particular token has the purpose to capture the overall meaning of a sequence and is automatically inserted by the tokenizer at the beginning of every sequence. In the first experiment Bert is kept frozen and only output layers are trained. This approach shows poor results compared to the previous model as listed in 1. Once the whole model is unlocked for training, there is a consistent boost in the evaluation metrics. Since the model is bigger and for its major part already trained, convergence is achieved in $\approx$ 30 epochs.

4. References

[1] W. W. Qian Chen, Zhu Zhuo, “Bert for joint intent classification and slot filling,” 2019.

[2] J. Park, “Unofficial pytorch implementation of jointbert,” 2020. [Online]. Available: http://aiweb.techfak.uni-bielefeld.de/content/bworld-robot-control-software/