Intent Classification and Slot filling

Davide De Martini

University of Trento

davide.demartini@studenti.unitn.it

1. Introduction

This report outlines the development of a system for intent classification and slot filing. The first task was to enhance the existent system based on LSTM, adapting it to be bidirectional and adding some dropout layers in order to regularize the model.

For the second task, the LSTM architecture has to be substituted in favor of BERT architecture. Then it has to be fine-tuned for the slot filling and intent classification tasks. The major challenge was to handle the sub-tokens that the BERT Tokenizer generate.

The dataset used is ATIS and the metrics for the evaluation are accuracy for the intent classification task and F1 score for the slot filling one.

2. Implementation details

For the first part, adding bidirectionality and dropouts are pretty straightforward and standard for the NLP community, so the details of this implementation process are omitted.

These two modifications have to be made sequentially, monitoring the performance of the model. As reported in table 3, the two changes together will increase the overall performance of the model.

For the second part, the LSTM has to be replaced in favor of a transformer architecture based on BERT model (Devlin et al. [1]). This model was pretrained for masked language modeling (MLM) and next sentence prediction (NSP) objectives. Slot filling and intent classification task are not covered, so it has to be fine-tuned in order to success for these jobs. A guideline for performing this was proposed by Chen et al. [2], this work was used as a reference. A high view of the proposed model can be seen in figure 1.

The challenge for this part was to preprocess the data in the right way requested from BERT. The Bert Tokenizer is based on byte pair encoding algorithm, so the sentence is not separated *word by word*, but it is possible to have sub tokens inside the same word. An example of this behavior is obtained by tokenizing the word ”StarLord”: normal tokenizer output would be ”starlord” but BERT tokenizer output is:”’Star’, ’##Lord’”. This will cause a desalination on the length of the utterances and the one of the slots. A solution for this problem is to tokenize the utterances and track whenever the token is splitted in subtokens. If this happens, the slot corresponding to the token has to be expanded in order to match the length of the subtokens. The first subtoken will have associated the right slot id, the others will have associated the id corresponding to the *padding* tag. Done this, an extra padding has to be added to the slot at the start and the end in order to cover the two special tokens: [CLS] and [SEP] (see table 1 for a better view of the process).

Let’s now focus on the model. It is composed by a BERT layer, then the output of BERT is forwarded to the two classification layer: the one for the intents and the one for the slot filling task. Before these two linear layers, there are two dropouts

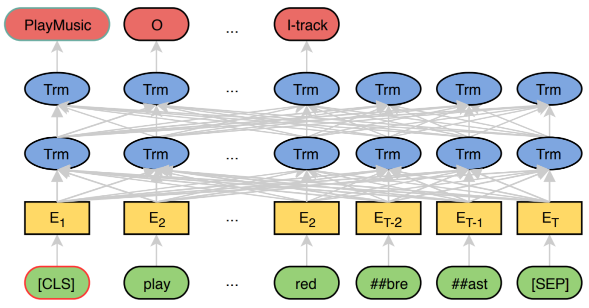

Figure 1: *High level of the BERT architecture from the original paper of Chen et al. [2]*

added in order to regularize the model.

For the intent classification task, the *pooled output* is used, it corresponds to the embedding of the [CLS] token, a special token used for text classification. For slot filling, instead is used the *last hidden state*.

The loss function is obtained by summing up the two different cross entropy losses produced by the tasks, and the model is trained on top of it.

For the evaluation part, the data is preprocessed in order to clear the ground truth of the extra pad tokens added in the preprocessing phase. In parallel, also our hypothesis are cleared, not by deleting the padding tokens, but deleting the slot with the index corresponding to the one in the ground truth having the padding token. This is done to be consistent with the two representations and for evaluating the model (see table 2). Then the reference and the hypothesis are passed to the evaluation function in order to obtain the F1 score. Instead, for the intents, a simple classification is performed and the accuracy is calculated.

The implementation is inspired by the code of Joint-BERT repository, the full repository can be found at this link: https://github.com/monologg/JointBERT.

3. Results

For the first part, the regularization techniques have been incrementally added and tested in order to recognize if the model was performing better or not. All the changes made to the model leads it to better performances. The experiment is composed by 4 runs, each run has 200 epochs, final results are averaged. Results are reported in table 3

For the second part, the experiment was run in a single run, for 65 epochs. The results were pretty similar to the one of Chet et al. [2]. The resulted slot f1 is: 0.97 and the intent accuracy is: 0.96. The batches size used are: 128 for the training set, 64 for the test set and validation set. The BERT model used is *bert-base-uncased* and the learning rate used is 10^{-4}. The training terminated after 70 epochs.

utterance	[CLS]	what	classes	of	service	does	t	##wa	have	[SEP]
slots	[PAD]	O	O	O	O	O	B-airline_code	[PAD]	O	[PAD]

Table 1: *In this example, the B-airline_code is splitted in two subtokens (t ##wa), so its corresponding slot is splitted in the real tag and the pad tag.*

reference	O	O	T-POS	~~[PAD]~~	O
hypothesis	O	T-POS	[PAD]	~~O~~	T-NEG

Table 2: *The strikeout text is the removed item in the hypothesis and reference*

	Slot F1	Intent Accuracy
Bidirectional	0.95	0.94
Bidirectional + Dropout	0.97	0.94

Table 3: *Results from the first part. The hyperparameter used are: hidden size 200, embedding size 300, lr* 10^{-4}. *The batches size used for the experiment are: 128 for the training set, 64 for the validation and the test set.*

4. References

[1] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019.

[2] Q. Chen, Z. Zhuo, and W. Wang, "Bert for joint intent classification and slot filling," 2019.