NLU course projects

Ettore Saggiorato (247178)

University of Trento

ettore.saggiorato@studenti.unitn.it

1. Introduction

This report presents the implementation and evaluation of slot filling and intent classification for a natural language understanding task, as the second assignment for the Natural Language Understanding course at the University of Trento. The assignment consists of two parts: the first involves training a Long Short-Term Memory (LSTM) model, progressively enhancing it with bidirectionality and dropout layers to observe their impact on performance. The second part focuses on fine-tuning BERT[1] on the same dataset. The challenge in the second part arises due to the complexities associated with BERT's tokenizer.

The objective in both parts is to train models for intent classification and slot filling. Performance is evaluated using accuracy for intent classification and F1 score for slot filling.

2. Implementation Details

The project is implemented entirely in Python using the PyTorch deep learning framework. Early stopping is implemented only in the first part.

2.1. Part 1 - LSTM Model

The first part of the assignment explores how bidirectionality and dropout layers affect the LSTM model's performance.

2.1.1. Architecture

Bidirectionality is configured through a parameter within the PyTorch LSTM module, it doubles the hidden size as the model processes the input sequence in both forward and backward directions. This configuration should improve the model's ability to capture contextual information, making it more robust when handling phrases with complex contexts, especially those with crucial information located towards the end of a sentence.

2.1.2. Regularization

Dropout is applied after the embedding layer to improve the model's robustness. The expectation is that by adding dropout, the model will better handle poorly structured or noisy input sequences, as it forces the network to learn more generalized features rather than relying on specific patterns.

2.2. Part 2 - BERT Fine-tuning

The second part of the assignment require fine-tuning a pre-trained BERT model [1] for the same task. This process follows guidelines from Bert's paper [1], with an eye to being able to compare the results with JointBERT framework [2][3]. The training is set for six epochs, as opposed to the ten epochs used in JointBERT, with each experiment being run five times to average the results for consistency.

The used model and tokenizer is `bert-base-uncased`. BERT's tokenizer does not natively support the tags required for intent classification and slot filling. To address this, two approaches were explored:

1. **Extending the Tokenizer Vocabulary** by adding the necessary tags directly to the tokenizer's vocabulary. This approach required substantial modifications to the model's input structure, such as adding a linear layer before the embedding or altering the embedding layer shape. These modifications proved ineffective (more training could have been beneficial) and were thus abandoned.

2. **Using Separate Tag Mappings**: by mapping the tags to unique, sequential IDs using separate dictionaries for slot filling and intent classification. The input of the model is Bert's tokenizer output, but the model's output consists of two linear layers with reduced expression capabilities to match the mapping shape. This method simplifies the process by allowing the model to extract logits and identify the maximum values directly, without complex modifications to the tokenizer or model input architecture.

2.2.1. Training Regularization

To enhance training efficiency, gradient clipping and Bert's learning rate scheduler were employed. Gradient clipping helps prevent exploding gradients, while the learning rate scheduler adjusts the learning rate during training to improve convergence.

3. Results

3.1. LSTM Model

The results for the LSTM model experiments are summarized in Table 1. It is evident that bidirectionality significantly improves the model's performance, and the addition of dropout further enhances its capabilities.

3.2. BERT Fine-tuning

The BERT fine-tuning results are directly comparable with those from the JointBERT implementation [2][3]. The learning rate used is $5e-5$, as suggested in the BERT paper's appendix [1], with the AdamW optimizer and batch sizes of 32 and 64. The results, shown in Table 2 and in Table 3, clearly demonstrate the positive impact of gradient clipping on training. However, the learning rate scheduler appears to negatively affect performance. Additionally, as said in Appendix A of BERT's paper, the model struggles with larger batch sizes, leading to less optimal convergence.

4. References

[1] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[2] P. Jangwon, "Jointbert," https://github.com/monologg/JointBERT/tree/master.

[3] C.-k. Lee and S. Ting, "Robust multi-task learning for natural language understanding," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. ACL, 2019, pp. 1247–1254.

Model	Emb. Drop.	Bidirectionality	lr	F1 (%)	Accuracy (%)
baseline1	-	False	0.0001	2.07%	70.77%
baseline2	-	False	0.001	92.25%	92.65%
baseline3	-	False	0.01	91.81%	91.44%
drop01	0.1	False	0.001	90.72%	92.79%
drop02	0.2	False	0.001	89.95%	93.39%
bid	-	True	0.001	93.71%	94.47%
bidrop01	0.1	True	0.001	**93.91%**	**94.60%**

Table 1: *LSTM Experiment Results*

Model name	Batch size	Gradient clip.	Train. sch.	F1 (%)	Accuracy (%)
32_simple	32	False	False	93.63%	**97.42%**
grad_32	32	True	False	**94.45%**	97.31%
sch_32	32	False	True	91.58%	95.97%
sch_grad_32	32	True	True	94.10%	96.98%
64_simple	64	False	False	92.53%	95.74%
grad_64	64	True	False	93.87%	97.09%
sch_64	64	False	True	80.61%	89.47%
sch_grad_64	64	True	True	90.48%	93.84%

Table 2: *Bert Experiment Results. More info in the TensorBoard log directory.*

Model name	F1 (%)	Accuracy (%)
32_simple	93.63%	97.42%
grad_32	94.45%	97.31%
BERT	*95.59%*	*97.87%*
best	***96.32%***	***97.98%***

Table 3: *Bert results comparison. Italics: results from [2].*