AYAYA documentation
- LiRo Hackathon -

Constantin Gabriel-Adrian
Sociu Daniel
Group: 407 (AI)

Preliminary assumptions base on the task

Taking into consideration that we are dealing with a Name Entity Recognition task in the LiRo Hackathon [2], the initial phase entails the representation of tokens (words) from the dataset as (numerical) embeddings. This can be accomplished fairly easy with the help of BERT transformer [3], pretrained on a way bigger Romanian dataset (about 15GB of data) [4]. A transformer is a deep learning model whose goal is solving "sequence to sequence" tasks i.e. encode-decoder. Transformers are similar with recurrent neural networks (RNN), the main difference being that they don't process the data in a specific order. Thus, a given sequence is analyzed in parallel and the influence or importance of each part of the sequence is determined at every step. Given the fact that by parallelizing this task the initial order is forgotten, the transformer adds for each embedding a positional encoding. What makes BERT special compared to other transformers that simply predict each token at pre train is it's ability to train in a bidirectional fashion using masked LM. What this does is basically hiding at random each token that is going to be predicted

Data preprocessing

Considering that the input statements have different lengths (number of tokens), to pass them through a transformer we would have to make all of them a specific length. This was done by splitting the longer statements in multiple parts of max length 128, and the rest that didn't reach the set length, we padded them. The tokenizer used splits the tokens in multiple subtokens, therefore only the first token should have a label, therefore the rest of the subtokens were asigned the special label of -100, which is the default ignored value by optimizers. After token splitting, there were a few problems with some characters that were not processed correctly by the transformer (some non-alphanumeric characters), which were replaced.

Dataset creation and Model implementation

With the tokens good to go, we created a Dataset class to store all our input data which was further passed to our Dataloader. The Dataloader is meant to simplify the workflow of our training process by automatically splits the data in batches. With the data ready, the pre-trained BERT model was not completely prepared for our current task, therefore we had to apply two operations:

1. **Fine Tuning**: this represents training the model starting from the weights of the pre-trained model and also updating the output head to be corresponding for our task.

2. **Feature Extraction**: which in our case implies to freeze some superior layers which contributes to the creation of some embedding features.

Experiments and training methods used

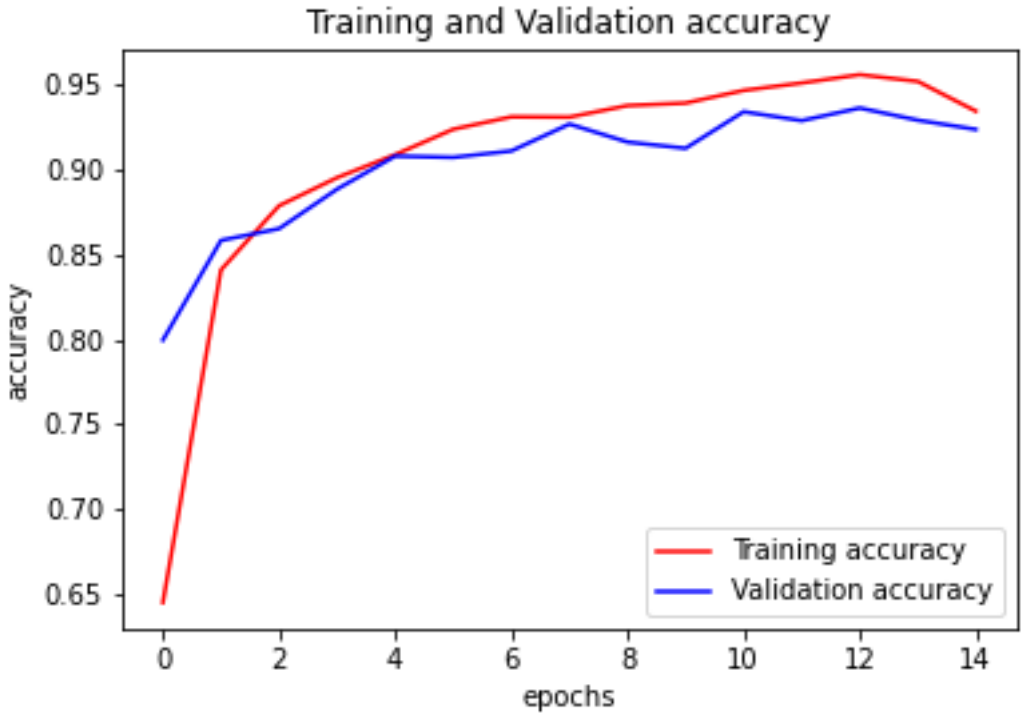
After training the model for 10 epochs, with the batch dimension of 64, this model was having a very bad performance, which was caused by the non-balanced training classes, therefore it had a tendency to predict the dominant classes more which affected the validation score. To solve this problem we computed the weights of the classes with the sklearn function compute_class_weight, which were passed to the optimizer that improved the performance of the model by a lot.

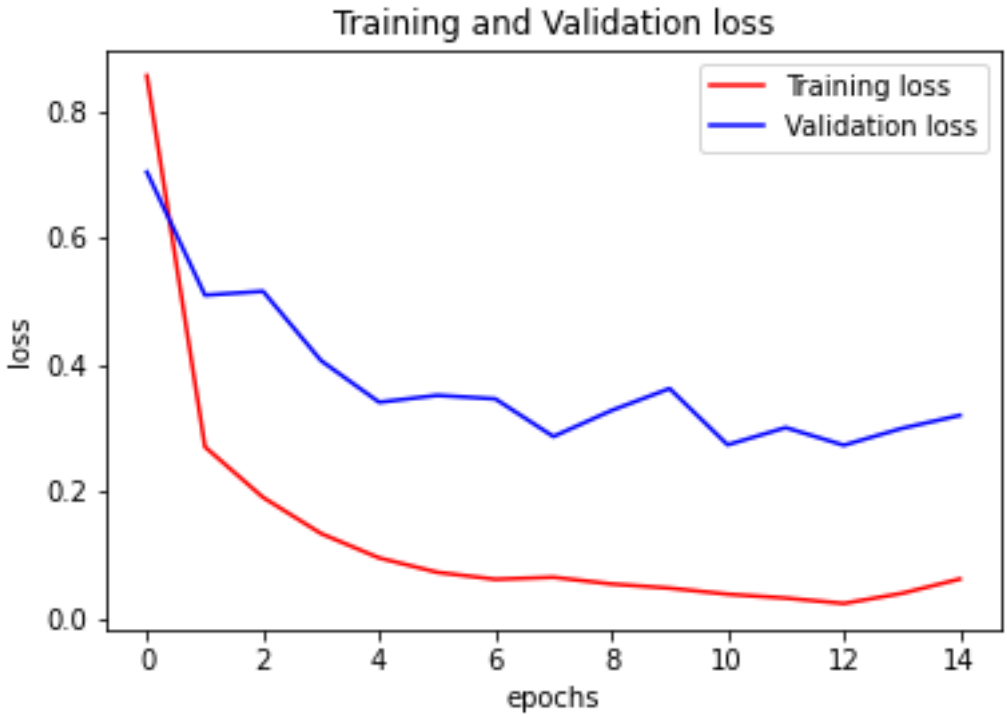
We tested multiple models during the hackathon. We found out that the best optimizer for fine-tuning is the Adam optimizer with a learning rate of 1e-4. Even though the learning rate is pretty small, we also used the torch.nn.utils.clip_grad_norm_ function to clip the gradients so that we don't suffer of explosive gradients. The main tests we did were about the number of epochs trained and the number of frozen layers. The results obtained are shown in the following table.

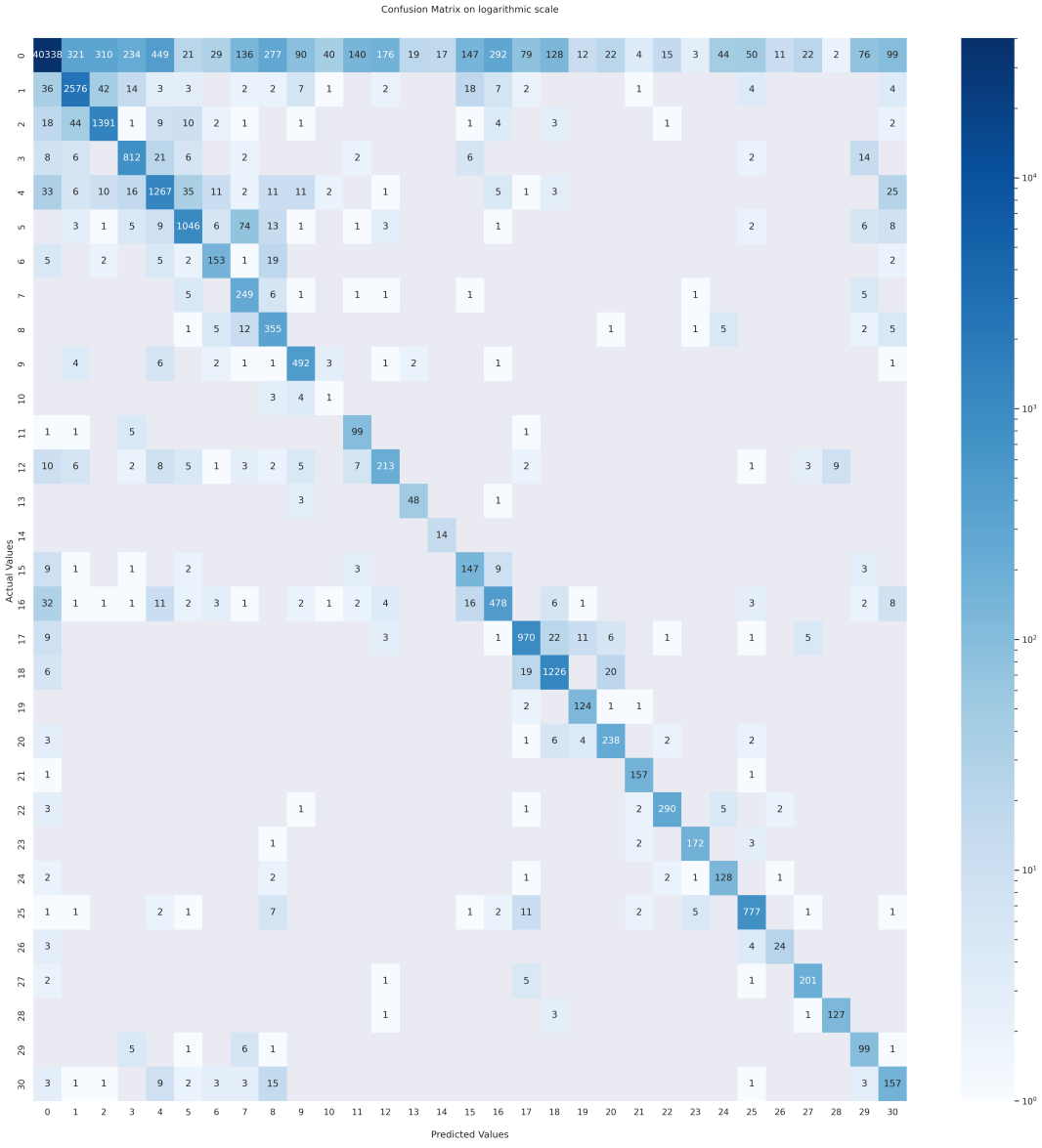
The contest required us to use the F1 strict score which was obtained from the evaluator that they provided. [1]

	Epcohs	Frozen layers	Accuracy	Loss	F1-score
Base	10	8	91.0%	0.346	72.1
Big-Base	10	6	91.2%	0.362	73.1
Base-15	15	8	92.1%	0.324	78.6
Big-Base-15	15	6	93.1%	0.303	**80.4**

Table 1: Results of the models.


Training and Validation accuracy
accuracy
epochs
Training accuracy
Validation accuracy
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0
2
4
6
8
10
12
14


Training and Validation loss
loss
epochs
Training loss
Validation loss
0.8
0.6
0.4
0.2
0.0
0
2
4
6
8
10
12
14



Conclusion

The final model that obtained the score of 80.4 obtained us the second spot for the chosen task. We managed to train 15 epochs in 12 minutes on a cloud A100, which was meeting the requirement of maximum 20 minutes of training.

Bibliography

[1] Constantin Gabriel-Adrian and Sociu Daniel. *The model that we submitted for the LiRo Hackathon*. URL: https://github.com/gabrielconstantin02/nlp-hackathon-2022/blob/main/upload/AYAYA/task1/model.py (visited on 01/15/2023).

[2] LiRo Hackathon. *NER Task*. URL: https://sites.google.com/view/liro-hackathon/challenge-1-liro-tasks?authuser=0 (visited on 01/15/2023).

[3] Huggingface. *BERT model*. URL: https://huggingface.co/docs/transformers/model%5C_doc/bert (visited on 01/15/2023).

[4] Dumitrescu Stefan. *Romanian BERT*. URL: https://huggingface.co/dumitrescustefan/bert-base-romanian-cased-v1 (visited on 01/15/2023).