UNIVERSITAT POLITÈCNICA DE CATALUNYA

CIÈNCIA I ENGINYERIA DE DADES

REPORT - ASSIGNMENT 3

PRESENTED BY

Bruno Pérez
Martí Farré

SUBJECT

PROCESSAT DEL LLENGUATGE ORAL I ESCRIT

May 2, 2023

1 Introduction

In this task, we have implemented several modifications to improve the given character-based RNN Baseline in order to create a new notebook with improvements and optimizations. A comparative study of the proposed models and tasks is accomplished related to the topics studied in class.

The report includes the modified code with our comments, results, tables and conclusions. We have compared the results of different models that we have analysed.

2 Comparative analysis

Comparative analysis of the studied models with respect to the LSTM baseline, taking into account loss, accuracy, training time, number of parameters, and hyperparameters differences with at 4 different kinds of modifications to the baseline model.

The results we get from the baseline model are the following. Training it takes 1659s, and we achieve a training accuracy of 98.729%. It is already a great accuracy, but we will try to improve it. After training, we test it and we predict the language of 3 sentences. The results we get are in the next table.

Id	Language	Id	Language
0	mwl	5	mon
1	nld	6	glk
2	ava	7	lez
3	tcy	8	bul
4	bjn	9	nan

2.1 Hyperparameter optimization

We have implemented several hyperparameter optimization by incrementing embedding size, RNN hidden size, and batch size. Have we also compared different optimizers. The best experiment result has been the experiment that modifies the baseline model by incrementing the hidden size to 512 and including 2 layers instead of only 1. We have read from studies that is one of the optimal hyperparameter configurations taking into account the task we are doing. Increasing the hidden size have some benefits to out model such as :

1. Increased model capacity
2. Improved Model Accuracy
3. Reduced Overfitting

The final model takes 5261s to train and it achieves an accuracy of 99.889%. Testing the model, where we predict the language a sentence is written in, leads to the following predictions:

Id	Language	Id	Language
0	mwl	5	mon
1	nld	6	snd
2	ava	7	lez
3	tcy	8	bul
4	bjn	9	nan

2.2 Combination of outputs

In this modification we have added to pooling layers at the end of our model, one using mean-pool and another using max-pool. Using these modifications, we have created two models we want to test. In the first one we sum the outputs of both layers, while on the second we concatenate the output of both layers.

The first model takes 1664s to train the final model, and we get a training accuracy of 99.6%. We test our model where we want to predict the language a sentence is written in and we get the following results:

Id	Language	Id	Language
0	mwl	5	mon
1	nld	6	glk
2	ava	7	lez
3	tcy	8	bul
4	bjn	9	nan

The second model, on the other hand, uses the concatenation of outputs instead of adding the results. The final model takes 1664s to train, just a few seconds more than the other model. A training accuracy of 99.34% is achieved, a little less than the other model. Overall, this differences are not relevant, however the first model performs slightly better. We have also tested our model and the language it predicts for the 10 sentences are the following:

Id	Language	Id	Language
0	mwl	5	mon
1	nld	6	glk
2	ava	7	lez
3	tcy	8	bul
4	bjn	9	nan

As it can be expected, both models predict the exact same languages for the 10 sentences. As we have mentioned, the performance of both models is very similar.

2.3 Droput

We have add a dropout layer after the pooling layer. We have tried three different dropout factors which are 0.1, 0.15 and 0.2. The best result has been obtained with the 0.1 value.

Training the final model takes 1448s, the fastest so far. However, this comes with a little drop in training accuracy, achieving only 97.621%. Testing results are displayed in the next table, and we can see this model predicts a different language for sentence 3 than the previous models.

Id	Language	Id	Language
0	mwl	5	mon
1	nld	6	glk
2	ava	7	lez
3	kok	8	bul
4	bjn	9	nan

2.4 Different RNNs and number of layers

We have made comparative analysis of the performance with other RNNs or number of layers.

We have changed the LSTM baseline model to the GRU which is another type of Recurrent Neural Network (RNN) which in some cases has advantages over Long Short-Term Memory (LSTM). GRU uses less memory and is faster than LSTM. When using datasets with longer sequences LSTM is more accurate usually. In our specific case the accuracy is better, specially in the training while in the validation remains the same. The real difference between these two models is the speed that GRU gives us which takes 2739 seconds to complete the training against LSTM which takes 3159 seconds, without improving the accuracy. So, GRU is 15.33% faster.

Id	Language	Id	Language
0	mwl	5	mon
1	nld	6	snd
2	ava	7	lez
3	tcy	8	bul
4	bjn	9	nan

3 Modified Code

In the dropout section we have added definition of the dropout in the ”init” section with the specified dropout factor.

```
self.dropout = torch.nn.Dropout(0.1)
```

And in the line just after the embedding from the output in the forward section we have added :

```
encoded = self.dropout(encoded)
```

In the hyperparameter optimization and also in number of layer optimization, we have just changed the numbers given in the baseline code and we have executed to compare the results.

The modifications we have made in the combination of outputs are adding the following lines (we have to modify the bidirectional parameter):

```
max_pool = padded.max(dim=0)[0]
mean_pool = padded.mean(dim=0)
# B x H

# concatenation
output = torch.cat([max_pool, mean_pool], dim=1)
# addition
# output = max_pool + mean_pool
```

In order to execute a different RNN we have just changed the default LSTM RNN for the GRU RNN in the CharRNNClassifier function which made the self.model being different defined both with torch.nn function extracted from the mentioned library.

4 References

https://discuss.pytorch.org/t/global-average-pooling-in-pytorch/6721/3
https://discuss.pytorch.org/t/lstm-hidden-size-num-layers-setting-question/120429/2