Romanian Sentence Classification

Gavrila Alexandru

1. The Task

This task was about classifying Romanian sentences into 10 classes based on style and topic. Given the nature of the task NLP was used.

2. Dataset

The dataset consists of 2,500 labeled Romanian phrases used for the training set, 574 for the validation set and 1,600 examples for the testing set. The .txt files for the training and validation sets contain the id on the first column, followed by the label on the second column, and the corpus on the third column, while the test set only contains the id on the first column and the corpus on the second column.

3. Data processing

The data was read using Pandas Dataframe. After naming the columns “id”, “class” and “data” I formatted everything in Unicode.

For the NLP part, I used HashingVectorizer on words and N-grams. Using unigrams and bigrams proved to be the best option. The number of features for the HashingVectorizer was 2^18. I noticed an increase in performance proportional to the number of features. The limiting factor was RAM capacity. I didn’t remove the stopwords because it would affect the style of the corpus even though it would help in determining the topic. This proved to be the correct choice after testing. I reshaped the output of the vectorizer to match a time series (input for LSTM). I didn’t try other pre-processing methods as I focused mostly on the other task.

4. Model

I used a sequential model with the following layers: Input, bidirectional LSTM, BatchNormalization, Dropout, Dense with activation function tanh, BatchNormalization, Dropout, Dense (Output). I tried a combination of multiple LSTM layers but it yielded worse results. The Batch Normalization and Dropout were used to prevent overfitting. For optimizer I used Adam and for the loss functions I used Categorical Crossentropy.

5. Training

I trained the model for about 15 to 20 epoch with a batch size of 64. This proved to be a fast training model.

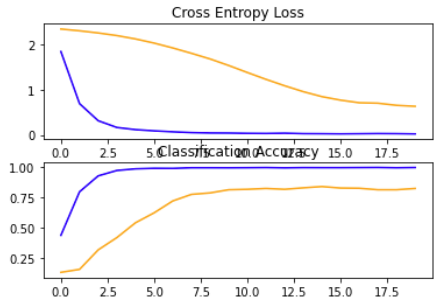
6. Hyperparameters

I used 64 cell LSTM with an input shape of (1, number of features). For the Dropout layer I used a dropout rate of 0.5 after tuning it to this value. The number of cells on the Dense layer is 64 and the activation function is “tanh” (which worked better than “ReLu”). On the last Dense layer I used 11 cells because the labels were numbers from 1 to 10. I tried using other optimizers and loss functions but they proved to be worse. I tried changing the hyperparameters for the optimizer but the default worked best.

7. Accuracy and Confusion Matrix

The validation accuracy ranged between 76% and 83% during the making of the model. The model was clearly not generalizing well based on the graph below but the confusion matrix looks good.

```
> 82.404
```




```
Confusion Matrix:
 [[50  2  0  5  0  0  1  1  1  2]
 [ 0 58  2  0  1  2  0  0  0  1]
 [ 1  4 35  0  1  0  3  1  2  0]
 [ 0  0  2 53  0  0  0  2  1  0]
 [ 0  6  1  1 23  1  1  0  0  0]
 [ 0  0  3  2  0 27  1  0  1  1]
 [ 0  1  0  0  0  0 64  2  4  2]
 [ 1  2  0  3  0  1  5 47  2  1]
 [ 0  2  1  1  0  0  2  0 76  1]
 [ 3  1  0  2  0  1  8  0  2 40]]
```

8. Conclusion

Even though this task was done in a very short time and was mostly recycled from the image classification task the results were satisfactory. Comparing the accuracy of this model to the leaderboard of the challenge I consider this to be a good approach for the task.