Artificial Intelligence II

IMDB Sentiment Analysis - Recurrent Neural Networks

Georgios - Alexandros Vasilakopoulos
A.M. 1115202000018

Data Preprocessing

- A custom dataset class is defined, in which all of the necessary preprocessing is performed.
- Similarly to the previous two assignments, all alphabetic characters are converted to lower case, all non alphabetic characters are removed and the english stopwords are also removed.
- I decided to use the GloVe embeddings through the torchtext library, in order to resolve some compatibility issues with the models.
- For each processed review, each word of the review is converted into the corresponding index of that word within the GloVe vocabulary. When a batch of data[1] is obtained, it is padded so that all of the reviews within the batch have the same size. Padding over the entire dataset would be an unnecessary waste of memory and would certainly consume the entire available memory provided by Colab.
- Within an embedding layer of the RNN, each batch of padded reviews is converted into a tensor of size `(batch_size,max_words, input_size)`, where `max_words` is the length of the padded reviews and `input_size` is the size of the GloVe vectors (300, in our case).
- All of these adjustments were made in order for the model to be trained upon meaningful encodings of words, which have sequential meaning. It would be a waste to train the RNN upon the averaged vectors of the reviews, which don't really have sequential meaning! :)

[1] See collate_fn

Model Architecture

- The following parameters were taken into consideration in the making of the model: learning rate of Adam, cell type (LSTM/GRU), hidden size, n. of layers (aka stacked layers), dropout probability, connection skipping, use of attention.
- In order to implement connection skipping, instead of defining one LSTM/GRU cell with `num_layers = num_layers`, exactly `<num_layers>` different cells were defined and the input was passed from one layer to the next, manually.
- When connection skipping is enabled, the output of each layer is added into the input of the layer which is two steps ahead (even/odd skipping).
- Self attention was implemented manually: right after the last cell of the RNN,the dot product of the final output with the outputs of each of the previous timestamps is taken and the results are passed through softmax to estimate the prob. distribution. Then, the weighted sum of the vectors is taken the result is concatenated with the output at the final timestamp. The concatenated vector (which is twice the size than before) is then passed through the linear layer.

Hyperparameter Tuning through Optuna

- Two different studies were conducted: one without using attention and one using attention. The ranges of the parameters that were considered (in both cases) were the following:

$$\texttt{learning\_rate} \in (1e-4, 1e-2)$$
$$\texttt{cell\_type} \in \{\texttt{LSTM},\texttt{GRU}\}$$
$$\texttt{hidden\_size} \in \{16, ..., 64\}$$
$$\texttt{num\_layers} \in \{1, ..., 10\}$$
$$\texttt{dropout} \in (0.01, 0.08)$$
$$\texttt{skip\_connections} \in \{\texttt{True},\texttt{False}\}$$

- The result of the first study (without attention) produced an `f1_score` of 0.876 in the test set, through the following set of hyperparameters:

```
learning_rate = 0.00054
   cell_type = 'GRU'
   hidden_size = 35
    num_layers = 7
   dropout = 0.01589
skip_connections = True
```

- The result of the second study (using attention) produced an `f1_score` of 0.884 in the test set, with the following hyperparameters:

```
learning_rate = 0.00086
   cell_type = 'LSTM'
   hidden_size = 57
    num_layers = 3
   dropout = 0.02052
skip_connections = False
```

- The result of the second study is the one that is used in the final evaluation
- The learning curves of both of the produced models indicate that the models do not overfit after a few epochs
- The form of the ROC curves also implies a good performance, while the area under curve corresponds to a high percentage of correct predictions
- The final model takes about 5 minutes to train, but you can load the pretrained model, included in the zip file.

Comparison with previous models

- The final RNN with attention (f1 score: 0.884) outperforms the simple Feed-Forward Neural Network that was produced in the 2nd assignment (f1 score: 0.83). This is expected, as the RNN utilizes the sequential nature of the input.

- Nevertheless, the RNN was unable to outperform the simple Logistic Regression model of the 1st assignment (f1 score 0.895). Maybe the assumption that the tfidf vectors are linearly seperable is more accurate than the assumptions made by this complicated RNN, after all.
- Perhaps if more tests were made through optuna, there might have been a model which exceeds the 0.9 threshold by a little...