Univ. of Athens, Dept of Informatics & Telecoms
YS 19: Deep Learning for NLP
Fall 2022 - Homework 4

Kyriakopoulos Dimitrios - sdi1900093

Notes to the graders: I provide a fully documented ipynb notebook with a link to Google Colaboratory. In order to run the model on the test dataset

1. Go to the "Load the dataset" section (Third code cell of the Notebook).
2. Change `DATA_PATH` value to the path, where the file with the data for training is located.
3. Change `TESTING_DATA_PATH` value to the path, where the file with the data for testing is located.

1 Introduction

In this project, I perform sentiment analysis on IMDb movies' reviews. I classify the reviews into two classes (Positive and Negative) by fine-tuning the pretrained BERT-base model experimenting with several parameters.

2 Data Split - Bert Tokenizer

I split the dataset into train (80%) and validation (20%) sets, and wrap them around torch.utils.data.DataLoader objects. With its intuitive syntax, DataLoader provides an iterable over the given dataset. What is more, Bert requires to add [CLS] at the beginning of each sentence (ID 101) and [SEP] at the end of each sentence (ID 102), make sentences of the same length and create an attention mask. For this reason I used_encode plus method with bert-base-uncased, which returns an object with the following fields:

- **input_ids:** list of token IDs.
- **token_type_ids:** list of token type IDs.
- **attention_mask:** list of 0/1 indicating which tokens should be considered by the model (return_attention_mask = True). Sentences that exceed max_length are truncated, while shorter sentences are populated with

[PAD] tokens (id: 0) until they reach the desired length. An example of the result of this tokenizer is shown below in figure 1.

Tokens	Token IDs	Attention Mask
[CLS]	101	1
thought	2245	1
quiet	4251	1
good	2204	1
movie	3185	1
fun	4569	1
watch	3422	1
liked	4669	1
best	2190	1
outta	24955	1
##kes	9681	1
end	2203	1
movie	3185	1
great	2307	1
[SEP]	102	1
[PAD]	0	0

Figure 1: Encoded Text.

3 Experimenting

I experimented with the batch size, the learing rate and the number of epochs. Above are the values I experimented with for each variable:

- **batch size:** 16, 32
- **learning rate:** 5e-5, 3e-5, 2e-5
- **number of epochs:** 2, 3, 4

as recommended in https://arxiv.org/pdf/1810.04805.pdf and by the authors. In order to find the optimal hyperparameters I did grid search between all possible combinations maximizing the accuracy of the model on the validation dataset. The optimal values, I found with this method are:

- **batch size:** 32
- **learning rate:** 5e-05
- **number of epochs:** 2

The following range of possible values was found to work well across all tasks. What is more, the bert model was shown to achieve state of the art results for a few epochs for such tasks.

4 Results

As metrics for the performance of the model I use precision, recall and F-measure whose values are shown below (Figure 2). I also plot ROC curve (Figure 3) which

	precision	recall	f1-score	support
0	0.89	0.85	0.87	4550
1	0.85	0.89	0.87	4452
accuracy			0.87	9002
macro avg	0.87	0.87	0.87	9002
weighted avg	0.87	0.87	0.87	9002

Figure 2: Scores.

is a graph showing the performance of a classification model at all classification thresholds, which shows that the model is quite efficient to distinguish between positive class and negative class.

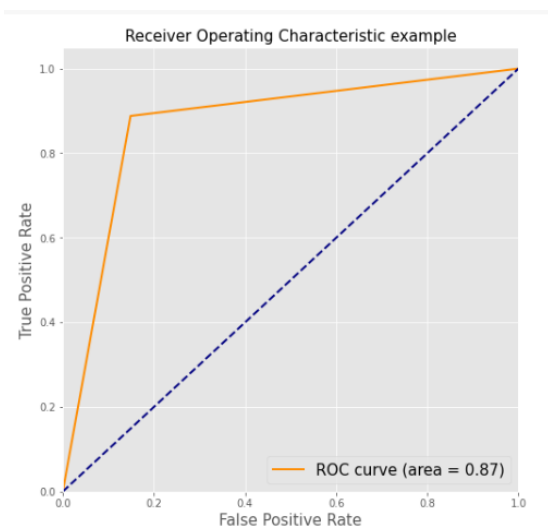

Figure 3: ROC curve.