UNIVERSITY OF ATHENS
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

Deep Learning for NLP

Student name: *Pavlos Ntais*
Classifier: BERT

Course: *Artificial Intelligence II*
Semester: *Fall Semester 2023*

Contents

1. Abstract

The task given is to develop a sentiment classifier using Bert & DistilBert for a twitter dataset about the Greek general elections. The classifier deals with 3 classes: POSITIVE, NEUTRAL and NEGATIVE. I plan to analyze the data, clean it by pre-processing and finally start the training process.

2. Data processing and analysis

2.1. Pre-processing

The pre-processing proved to be a rather difficult task because of the vastness of the Greek language. After experimenting I came up with the following methods:

- Remove hashtags
- Remove mentions
- Remove URLS
- Remove excess tabs and white spaces
- Remove numbers and special characters
- Remove stop words
- Remove accents
- Basic emoji replacement
- Insert the party the tweet is being referred to at the beginning of the text
- Lemmatization using the spaCy library

2.2. Analysis

Upon visualizing the dataset, it became apparent that the data had a substantial degree of noise. For example spelling and grammar errors, mentions etc. Also, it naturally contained a lot of stop words, hash tags and emojies which also add noise. So, in order to process the data and remove the noise I removed stop words, hash tags and emojies and used lemmatization to group together inflected or variant forms of a word.

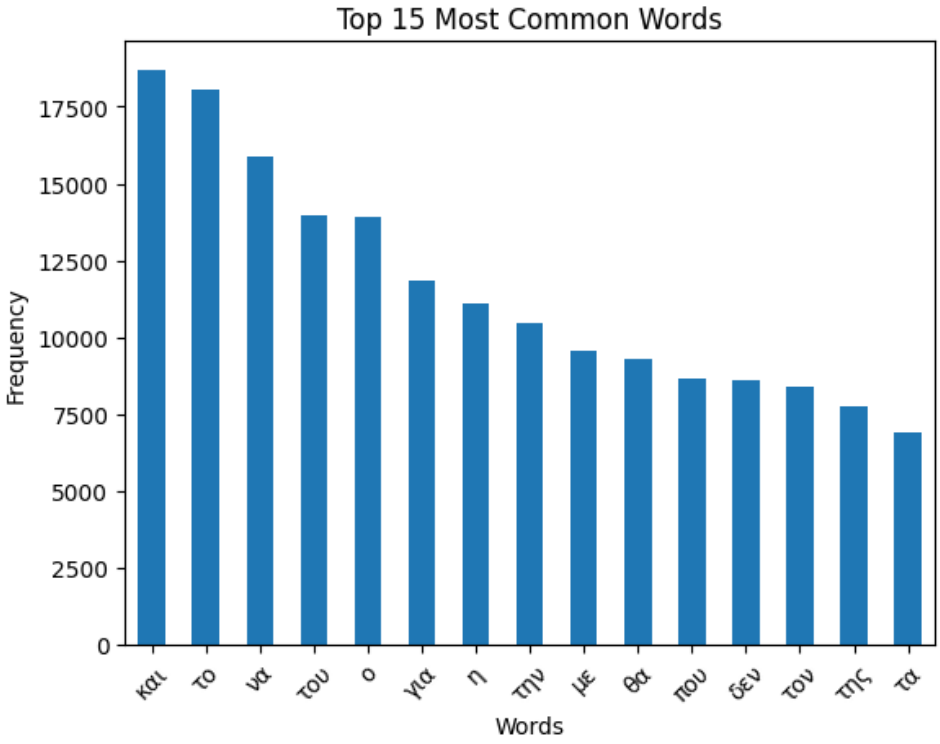

Figure 1: Before pre-processing

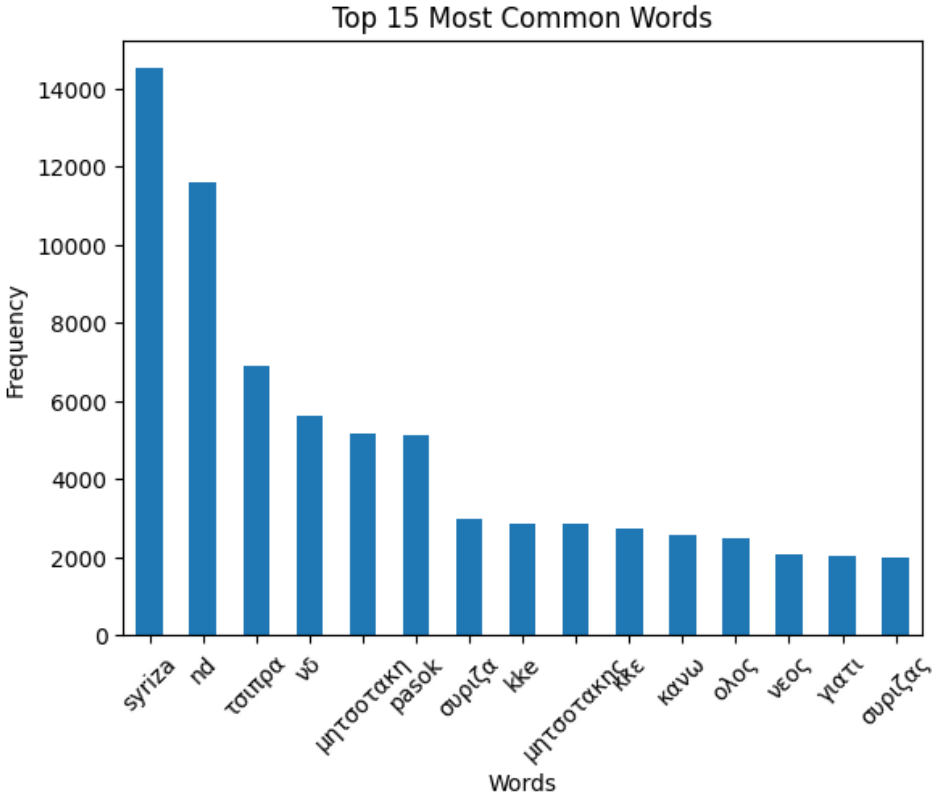

Figure 2: After pre-processing

2.3. Data partitioning for train, test and validation

For the training and testing I used the given datasets because I evaluated that their respective size is sufficient, so no partitioning occurred.

2.4. Vectorization

We start by tokenizing each sentence, which involves breaking down the sentence into individual words or subwords, and then converting these tokens into their corresponding IDs using the provided tokenizer. Special tokens, [CLS] and [SEP], are added at the beginning and end of each sentence, following the standard practice for BERT models. The [CLS] token is used as an aggregate representation for classification tasks, while the [SEP] token helps the model distinguish between sentences.

The sequences of token IDs are then adjusted to a specified, maximum, length. If a sentence has fewer tokens than the maximum length, the sequence is padded with zeros at the end. If a sentence has more tokens, it is truncated. Corresponding attention masks are created, with 1s for actual tokens and 0s for padding. These sequences of token IDs and attention masks are then converted into PyTorch tensors. These are the essential preprocessing steps required to prepare our text data for the BERT model, converting the raw sentences into the suitable numerical format.

3. Algorithms and Experiments

3.1. Experiments

1. I initially experimented with **just** using the bert models. I will not be providing the results because I quickly realized the model was far too unstable and a lot of overfitting occurred. To stabilize the model and to reduce overfitting I added a dropout layer, creating a model that first passes through a bert layer, then a dropout layer and finally an output layer. This addition seems to yield much better results compared to just using the model on its own.
2. The optmizer used is AdamW.
3. The models were trained on 3 epochs, primarily because of the long training time needed and because with bert, not many epochs are needed.

Trial	Precision	Recall	F1 score
Bert + default hyperparameters	0.4377136139478	0.4149464831804	0.4025035414255
DistilBert + default hyperparameters	0.4177344325920	0.4032874617737	0.3826948095225
Optuna 1 - Bert	0.4329110491993	0.4162844036697	0.3911958464625
Optuna 1 - DistilBert	0.4268671327125	0.3985091743119	0.3710174906567
Optuna 2 - Bert	0.4315181719858	0.4271788990825	0.4220312415361
Optuna 2 - DistilBert	0.4406315096540	0.4113149847094	0.3844559618843

Table 1: Trials

3.1.1. Table of trials. Overall, as expected from the paper found here, the Bert model outperforms the DistilBert model by about 3%. The upside though is the fairly noticeable faster training time.

3.2. Hyper-parameter tuning & Optimization techniques

For optimization I used the Optuna framework, an open-source tool designed for automating hyperparameter optimization. I primarily fine-tune my model running it, ultimately enhancing its performance. More specifically, ran optuna 2 times for both Bert & DistilBert (for about 50 iterations each), each time reducing the search range in order to find the hyperparameters that give the best results.
In the optimization function we tune:

1. **The dropout rate**
2. **The learning rate**

3.3. Evaluation

I am evaluating the predictions using 3 curves:

- **ROC curve**
  The ROC curve is a graphical representation of a binary classification model's ability to distinguish between positive and negative classes.
- **Loss learning**
  Loss is the difference between the predicted values of a model and the actual ground truth labels, guiding the optimization process to minimize this discrepancy and improve the model's performance.
- **Learning rate**
  A learning curve visualizes how a model's performance improves with more training data using the fscore. Fscore is a performance metric that combines precision and recall into a single value.

I also provide the precision & recall scores. The most important thing I'll be taking in mind though, since neither precision nor recall seem to be our priority, is the fscore. It provides a balance between these 2 and makes the model more "generic".

3.3.1. Curves. Here are the curves of the trials showed above:
You have to zoom in order to see the results clearly.

At first, we will be testing models using the default hyperparameters.

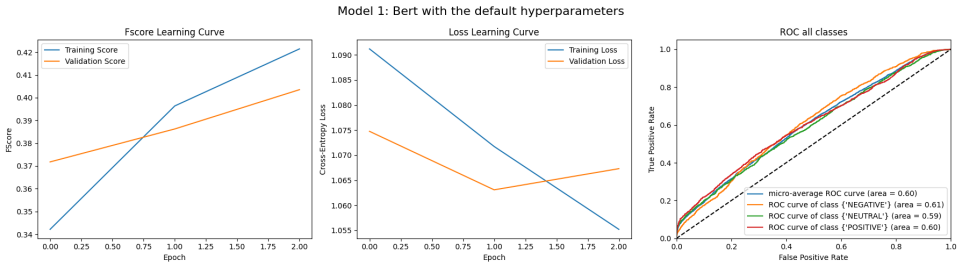

Figure 3: Bert & default hyperparameters

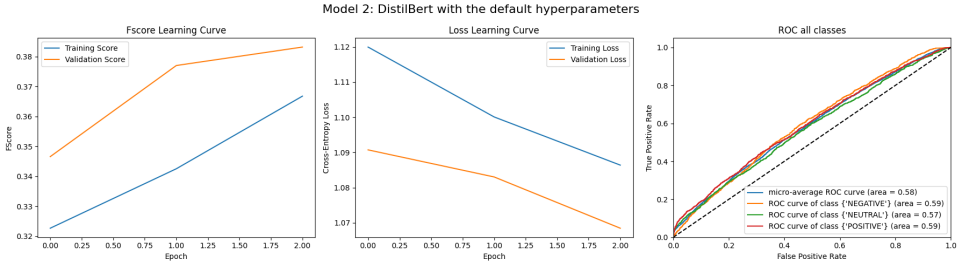

Figure 4: DistilBert & default hyperparameters

Now we will be running Optuna to hopefully optimize our model.

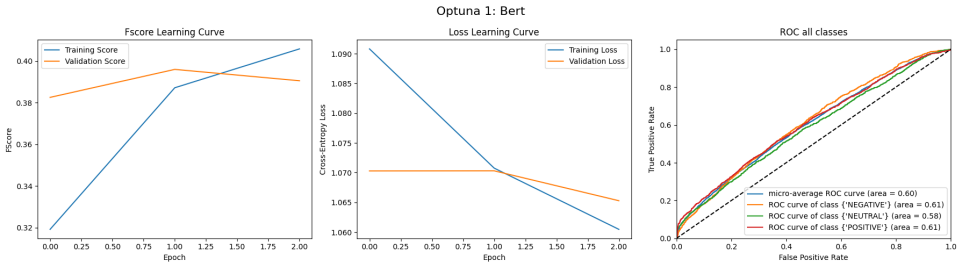

Figure 5: Optuna 1 - Bert

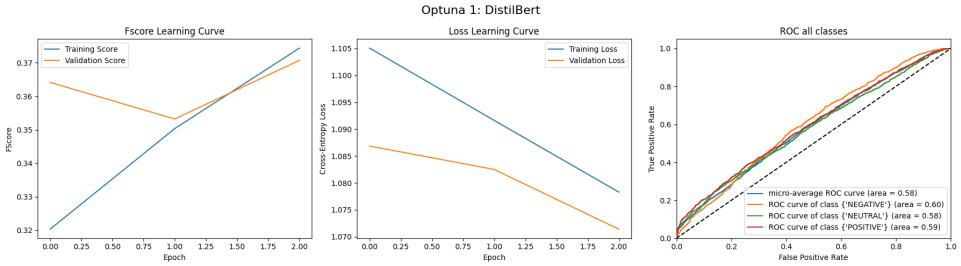

Figure 6: Optuna 1 - DistilBert

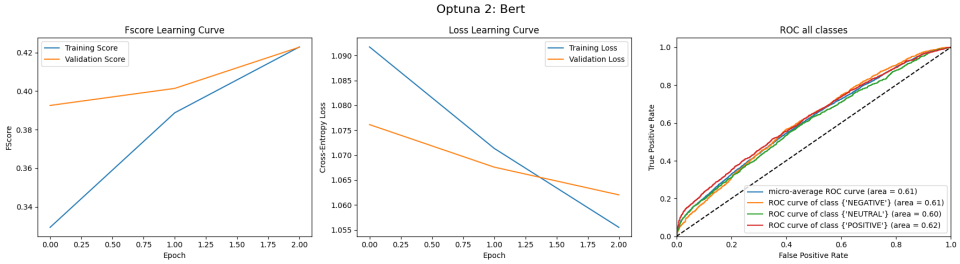

Figure 7: Optuna 2 - Bert

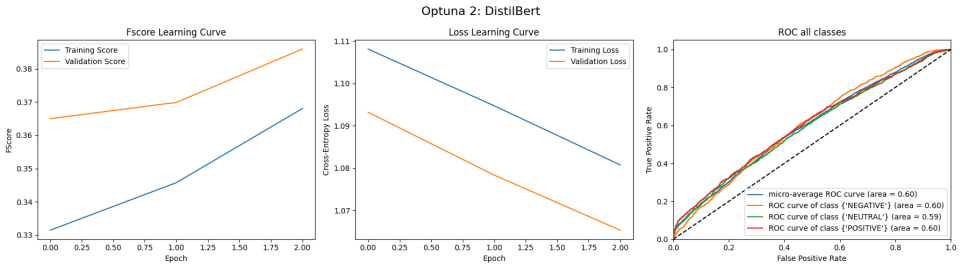

Figure 8: Optuna 2 - DistilBert

4. Results and Overall Analysis

4.1. Results Analysis

Overall, although the performance of the model fell short of my expectations, there are several factors contributing to this outcome:

- The dataset presented inherent challenges due to its Greek language content. Greek, being linguistically complex and distinct from widely-used NLP languages, posed inherent difficulties for the model. Also, the limited availability of good performing Greek NLP libraries posed a significant challenge.
- The dataset itself may have contributed to the suboptimal performance. A good number of labels were not correct thus hindering the ability of the model to learn.

4.1.1. Best trial. Ultimately my best trial came after running hypertuning optimization.

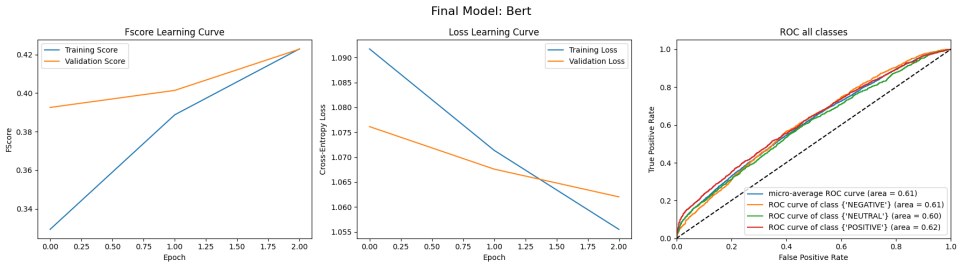

Figure 9: Final Model - Bert

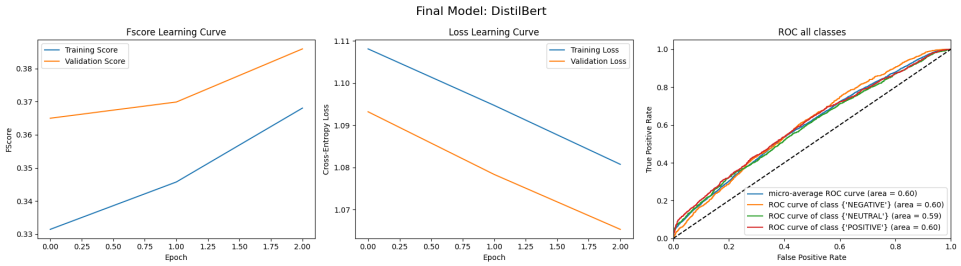

Figure 10: Final Model - DistilBert

4.2. Comparison with Softmax Regression

Overall, in comparison with the Softmax Regression, our model outperforms it, by a percentage of 2.2%.

4.3. Comparison with Feed Forward Neural Network

Overall, in comparison with the Feed Forward Neural Network, our model significantly outperforms it, by a percentage of about 3%.

4.4. Comparison with Recurrent Neural Network

Overall, in comparison with the Recurrent Neural Network, our model significantly outperforms it, by a percentage of about 2%.

5. Bibliography

References

[1] Optuna. Optuna hyperparameter tuning. https://optuna.org/#code_pytorch.

[2] Stack Overflow. Stack overflow q&a. https://stackoverflow.com/.

[3] Pytorch. Pytorch tutorial. https://pytorch.org/tutorials/.

[4] Stanford. Speech and language processing. https://web.stanford.edu/~jurafsky/slp3/.

[4] [2] [3] [1]