University of Athens
Department of Informatics and Telecommunications

Deep Learning for NLP

Student name: *Konstantinos Fragkos*
sdi: sdi2000207

Course: *Artificial Intelligence II (M138, M226, M262, M325)*
Semester: *Fall Semester 2023*

Contents

1. Abstract

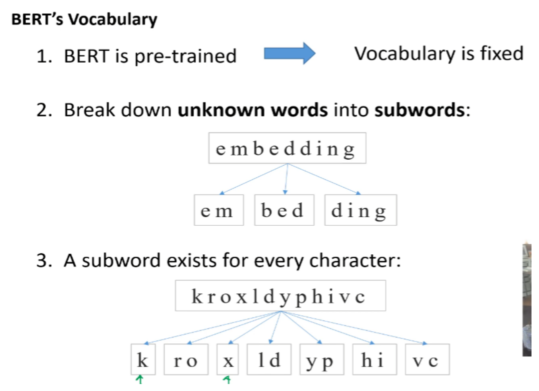
In this paper we will use the pretrained Greek model BERT (Bidirectional Encoder Representations from Transformers), more specifically GreekBERT and DistilGREEK-BERT to solve the sentimental classification problem for Greek elections. Given some tweets, we will classify them into 3 different sentiments, Positive, Neutral and Negative, using the above models. These models are already pretrained on general data, so we will fine-tune them to adapt our models to our own requirement and our own data of our problem. DistilGREEK-BERT since by maintaining only 60% of the size of GREEK-BERT, DistilGREEK-BERT managed to be 60% faster, while having a comparable performance to the one of the bigger model. Overall, it retains 96% - 97% of GREEK-BERT performance. The general philosophy and difference of these models is that they now take into account the whole contexts of the word using the logic of transformers that have embedded attension layers which are much more advanced methods than the ones we used in previous works.

2. Data processing and analysis

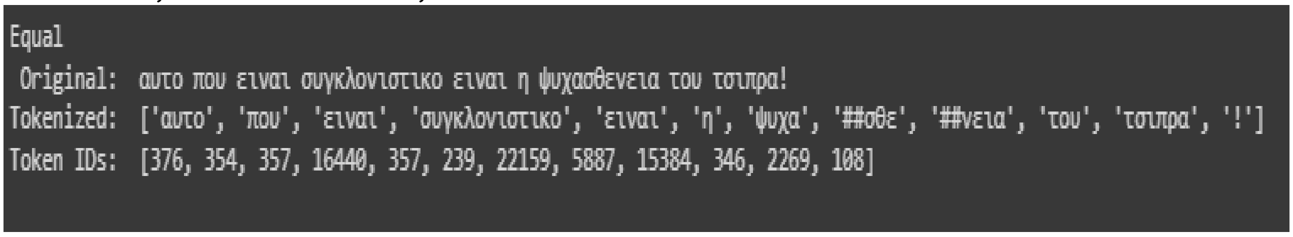
2.1. Pre-processing

The data preprocessing in this paper changes a bit compared to the previous ones. We are interested in converting uppercase to lowercase and dropping the accents as the vocabulary contains uncapitalized words and accents. We are not interested in lemmatization and steaming of words this time, as tokenization is done in a different way which we will discuss below.

2.2. Analysis



As we see, bert tokenizer, breaks the sentences like this:



```
Equal
 Original:  αυτο που ειναι συγκλονιστικο ειναι η ψυχασθενεια του τσιπρα!
Tokenized:  ['αυτο', 'που', 'ειναι', 'συγκλονιστικο', 'ειναι', 'η', 'ψυχα', '##σθε', '##νεια', 'του', 'τσιπρα', '!']
Token IDs:  [376, 354, 357, 16440, 357, 239, 22159, 5887, 15384, 346, 2269, 108]
```

Tokenized, is the tokkens of sequence. Token Id, is the index of word in Greek bert vocabulary

As I check in code, the 2 different vectorizers for 2 different Bert models, gives us the same results, for this reason I keep one of these 2

2.3. Data partitioning for train, test and validation

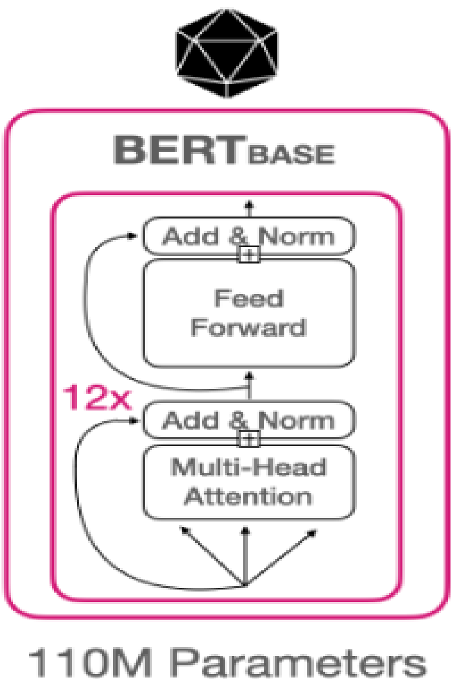
I use train set for training and validate set for validating as given

2.4. Vectorization

As you can see, the reason we did the preprocessing in this way is that if the model doesn't have a particular token in its vocabulary, it breaks it into sub tokens like the photo, until those tokens are in its vocabulary. The model is pre-trained, so the embendigs are ready, however, thanks to the attension mechanism it uses internally embendings changes according to simmilarity of input.

3. Model

3.1. Architecture



Bert's architecture is essentially the encoding member of a transformer. It takes as input sequences, and these are passed through 12 contiguous layers, where each layer is composed of a multihead attension mechanism, and also an fc layer. As output, out of the many tokens that each sentence puts out, we are interested in the first one, which gathers the overall information of the sentence, and based on this by passing it through a linear layer (this is done inside the BertForSequenceClassification function not implemented by us) we get after a softmax the sentiment predicted by the model.[1][2]

3.2. Fine tuning

Our model is pretrained on a more general set of words. This alone is not enough for us to make predictions on this problem, as we need to train our model a bit more on the philosophy of our problem. For this we do fine-tuning which is essentially like training.

4. Algorithms and Experiments

4.1. Experiments

Below I note some important results, relating to the experiments carried out in both models

model	epochs	Lr	batch size	eps	F1-score
Simple	1	$8 \cdot 10^{-5}$	32	$4 \cdot 10^{-8}$	0.37
Simple	3	$4 \cdot 10^{-5}$	16	$4 \cdot 10^{-8}$	0.42
Simple	2	$1 \cdot 10^{-4}$	16	$9 \cdot 10^{-8}$	0.35
Distil	2	$1 \cdot 10^{-4}$	32	$3 \cdot 10^{-8}$	0.35
Distil	2	$7 \cdot 10^{-5}$	32	$5 \cdot 10^{-8}$	0.4
Distil	1	$1 \cdot 10^{-4}$	32	$7 \cdot 10^{-8}$	0.33

Table 1: Trials

4.1.1. Table of trials.

4.2. Hyper-parameter tuning

Epochs : we notice that you need much less than other tasks, we experimented with numbers 1,2,3,4 which after 2 becomes overfiting quickly, so a number from 1 or 2 is good.

Lr: we experimented over a range of values of 10^{-4} to 10^{-6}, we notice that it plays a crucial role in learning the model and a number close to $4 * 10^{-5}$ is good

eps : it is the factor that reduces the learning rate, and as it is small it does not play a decisive role, but it certainly improves our model

batch size : We notice that both 16 and 32 gave us quite satisfactory results

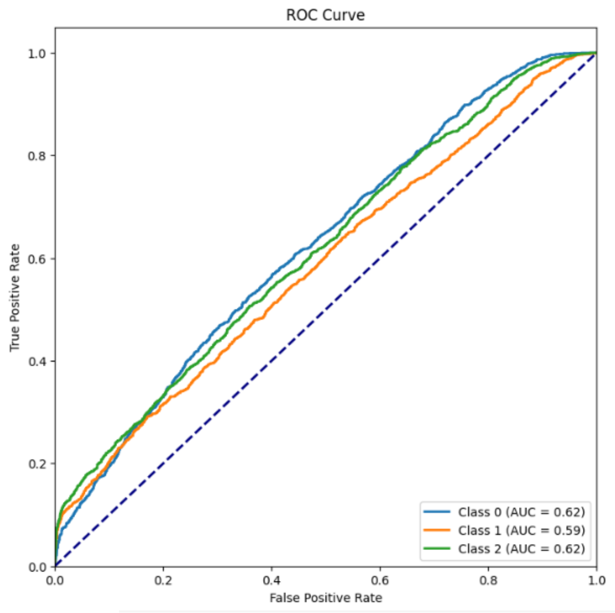
4.3. Optimization techniques

In this paper as in previous ones, I used optuna over a reasonable range of hyperparameter values to maximize the f1 score.

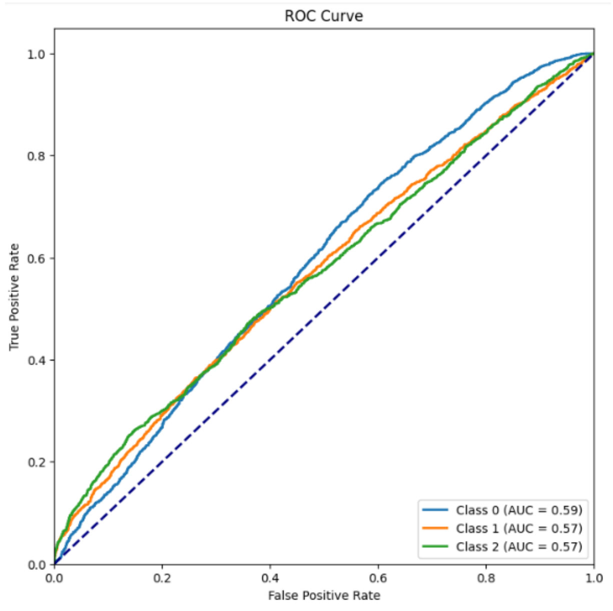
4.4. Evaluation

To evaluate my results I used the following curves. The Roc curve to find the relationship between true positive and false negative, the Learning curve to avoid overfitting and the comfusion matrix which gives me a bigger picture of the predictions of the results

4.4.1. ROC curve.

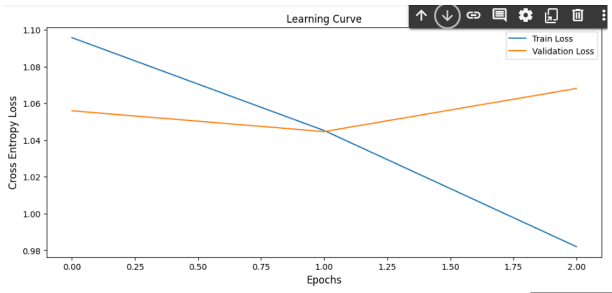


Greek bert best case

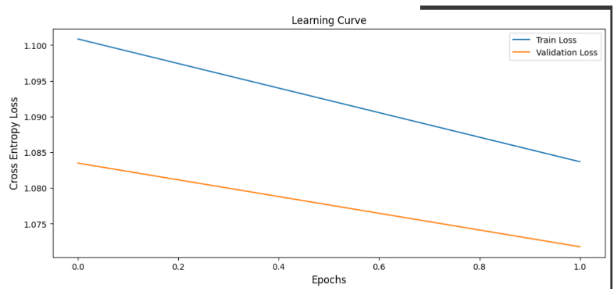


Distil Greek bert best case

4.4.2. Learning Curve.

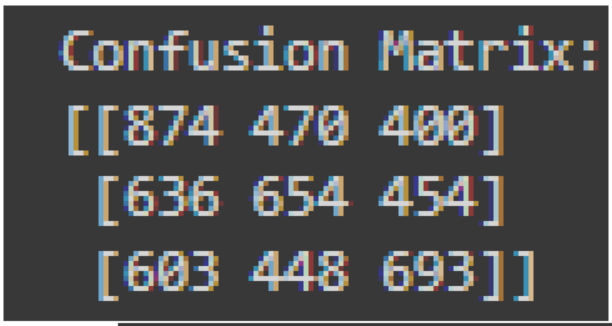


Greek bert best case



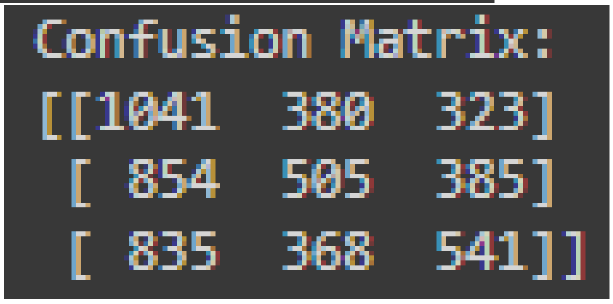
Distil Greek bert best case

4.4.3. Confusion matrix.



```
Confusion Matrix:
[[874 470 400]
 [636 654 454]
 [603 448 693]]
```

Greek bert best case



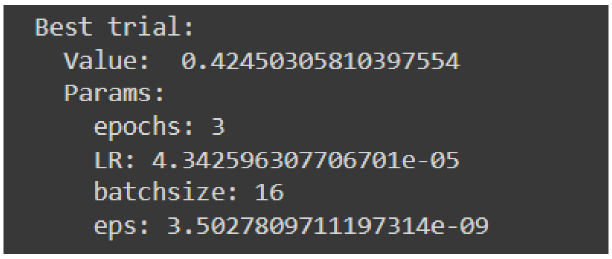
```
Confusion Matrix:
[[1041  380  323]
 [ 854  505  385]
 [ 835  368  541]]
```

Distil Greek bert best case

5. Results and Overall Analysis

5.1. Results Analysis

In conclusion, the bert model was the most effective model of the models we used in the previous papers as we got the best results we have gotten. This is obviously due to the complex methods it uses such as multiple layers and the use of attension. Definitely the point where it lags behind is the training time which is much longer compared to the other methods.



```
Best trial:
  Value:  0.42450305810397554
  Params:
    epochs: 3
    LR: 4.342596307706701e-05
    batchsize: 16
    eps: 3.5027809711197314e-09
```

5.1.1. Best trial.

5.2. Comparison with the first project

Compared to the 1st paper, I don't think there is any issue of comparison as bert's methods are much more complex and effective than a simple linear regression.

5.3. Comparison with the second project

Compared to the 2nd paper, again a neural is not as efficient a method as bert's methods.

5.4. Comparison with the third project

Compared to the 3rd paper, I would say that the rnn network is a pretty good model but still bert is more forward.

6. Bibliography

References

[1] bert.

[2] bert explain.

[1] <More about Bert>
[2] <More about Bert>