UNIVERSITY OF ATHENS
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

Deep Learning for NLP

Student name: *Lampropoulos Konstantinos*
sdi: sdi1800092

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

Contents

1. Abstract

In this task,we were asked to tune 2 pretrained Neural Networks (GreekBert and Distil GREEK-BERT).Both are based on the Bert model which was developed by Google and is a multi-layer bidirectional transformer encoder .To tacke this task,we will first pre-process out dataset,by removing stopwords,removing whitespaces etc.Then we will pad all the other texts to the length of that one and we will tokenize our text adding the special tokens [CLS] and [SEP].Finally we will tune our models to return the best results and prevent overfitting.

2. Data processing and analysis

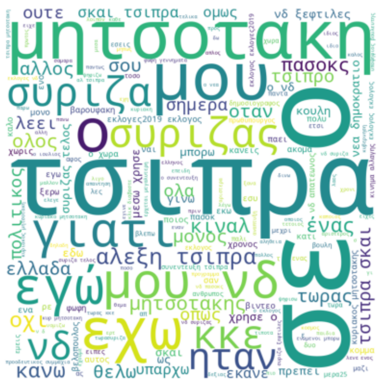
2.1. Pre-processing

In this section we will cover the pre-processing techniques involved.To begin with,we drop the unnecessary columns(NewID , Party) since we will not be using them in the classification process.After that, we convert all the text column in lowercase letters,remove the accents ,as well as, the stopwords and replace them with a single whitespace character.Following that,we replace multiple whitespace characters with a single white space,remove any links and non-greek words as well as retweets (and replace them with single whitespace characters).Finally,we lemmatize the text column to replace words with their dictionary values.The purpose of this pre-processing action is to breakdown the text and remove any unnecessary words and also provide the classifier with the dictionary value of the words so they are more easily recognized.

2.2. Analysis

2.2.1. Word Cloud.

In a word cloud,the most common words have greater size compared to the rest.Below I showcase an image with the results of the wordcloud implementation:



As we can see ,words like: Μητσοτάκη,Τσίπρα,νδ,συριζα, are really common in the database.We also have some words like: ο,σε,εχω ,which shouldn't be kept after removing stopwords.These should be removed for better results,but were not in this project.

2.2.2. Token Frequency.

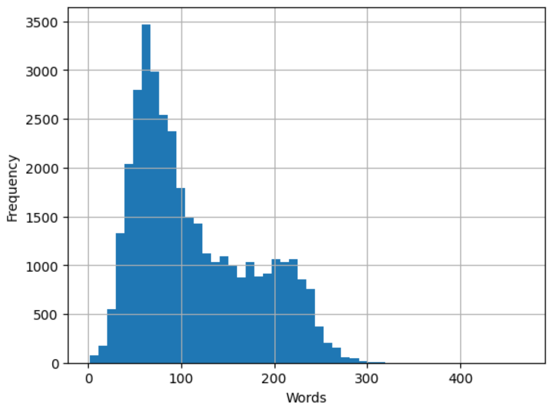
Token frequency is used to find the most common words in the dataframe.In this dataframe our 10 most common words are:

word	count
ο	18186
τσιπρα	9896
μου	8186
νδ	8168
μητσοτακη	6880
σε	4528
εκλογος	4298
εχω	4205
συριζα	3949
συριζας	3630

We can see that some stopwords appear as most common words.These should be removed manually as they may impact the training of the model,but were not for this project.As we can see some of the most common words are τσιπρα,νδ,μητσοτακη which means that most of the common words are correctly identified and will likely help us better identify the sentiment

2.2.3. Text Length Distribution.

Text length distribution is a method which finds the most common text lengths in the dataframe.



As we can see,the most common text lengths are between 60-90 words,which is the size of a common text or tweet.

2.3. Data partitioning for train, test and validation

The dataset remained the same as the one given in the assigment.

2.4. Tokenization

Bert accepts as an input a sentence of Ids,where each id corresponds to a word.So we need to map each word to a unique ID and convert the sentences to a list of Ids,instead of a list of words.To do that,we first need to make sure all sentences are of the same length.We start by firstly finding the maximum length of a sentence across all 3 datasets(training,validation,testing).We then pad all sentences to that length.Finally we convert out text to a list of indices,add the special tokens needed and create attention masks,which help our model identify which of the indices are part of the sentence and which (most commonly 0s) are part of the padding process and can be ignored. After conducting an experiment and comparing the max length of both the sentences created by the DistilGreekBert tokenizer and the GreekBert tokenizer,I came to the conclusion that the maximum length of a sentence was the same for both cases,and thus it was called only using the sentences created by the GreekBert tokenizer.

3. Algorithms and Experiments

3.1. Experiments

The training of both GreekBert and DistilGreek-Bert was done using the optuna framework and the hyperparameters tuned were the learning rate,the number of epochs and the batch size.The optuna framework creates a study consisting of all the trials perfomred.After creating the study it analyzes the importance of each hyperparameter as to improvement (or not) of the performance of the model.In this assigment the objective function was to be maximized,as I chose to use F1-score as my main metric (accuracy could be used as well,since the dataset is balanced both f1 and accuracy will converge to the same value) and thus it became an maximization problem.The hyperparameters specified in the optuna framework as well as their potential values are listed below (for Greek Bert):

3.1.1. Table of trials.

HyperParameter	1st Value	2nd Value	3rd Value
Batch Size	32	64	128
Epochs	2	5	
Learning rate	2e-6	2e-5	

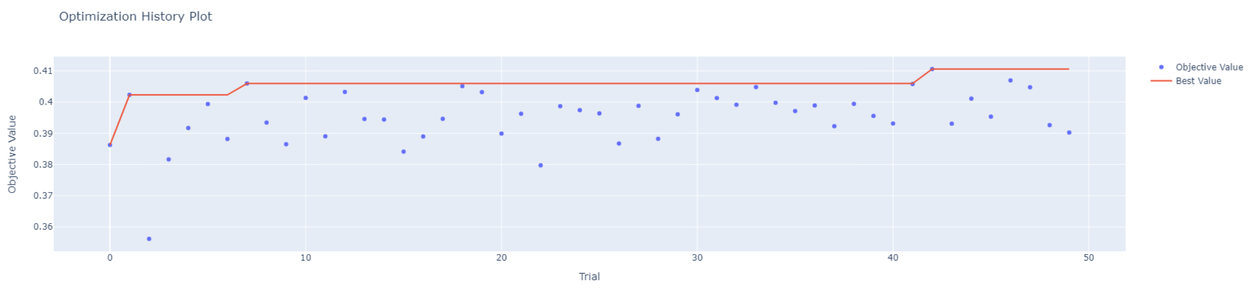
and for DistilGreek-Bert:

HyperParameter	1st Value	2nd Value	3rd Value
Batch Size	32	64	128
Epochs	2	8	
Learning rate	2e-6	2e-5	

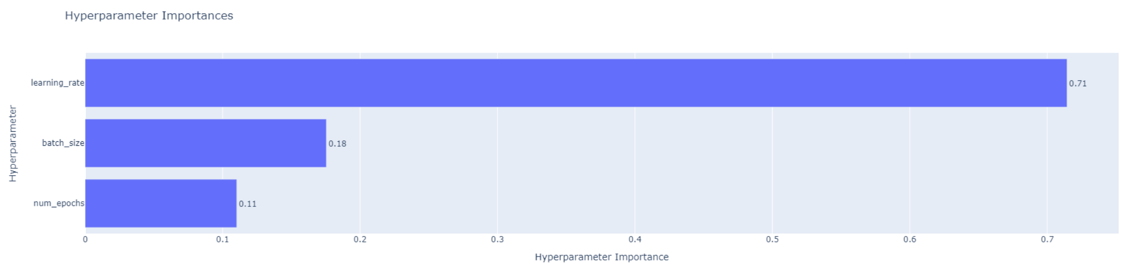
At both of those tables, both learning rate and Epochs specify a range at which optuna chooses values.For example at GreekBert the Epochs are specified as 2,5 which means that optuna in the course of the 50 trials done,will choose any integer value between 2,5 (including 2 and 5) as the number of epoch ,for that trial.

3.2. Hyper-parameter tuning

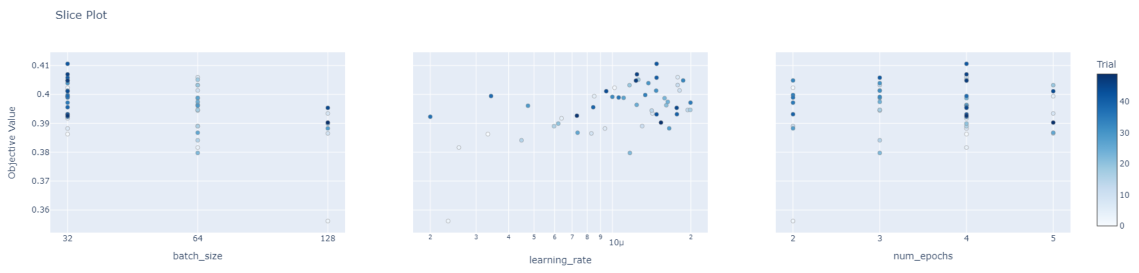
First we can check the optimization history of the GreekBert model:



It is not significant,but we can see that the hyperparameters for the model are found near the end of the trial(at trial 42). Next we can see the importance of each parameter at the performance of our model:

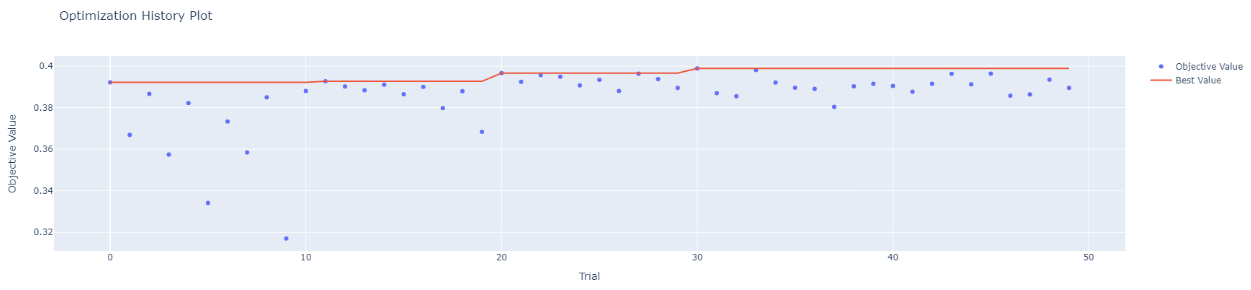


It is obvious that the learning rate is the most importance one,after that batch size and lastly the number of epochs.We can deduct from that the number of epochs don't increase signficantly the validation F1 Score of our model.It is of little surprise that among these values,the learning rate is of the greatest importance ,since it is considered one of the most important hyper parameters when training a Neural Network.Lastly we will take a peek at the slice plot:

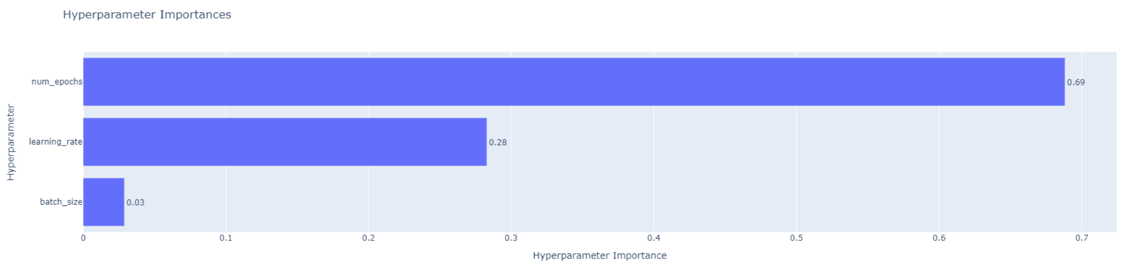


The slice plot gives us an idea as to the values of the hyperparameters and their corresponding F1-Score.Firstly we can check the batch size.As noted above the batch size and the number of epochs were not really as significant to the performance of the model as the learning rate.Nonetheless,batch size 32 was the best for our model,with the selection of batch size 64 being acceptable performance-wise. Batch size 128 didn't perform well and was not preferred by the framework.As for the learning rate,we can

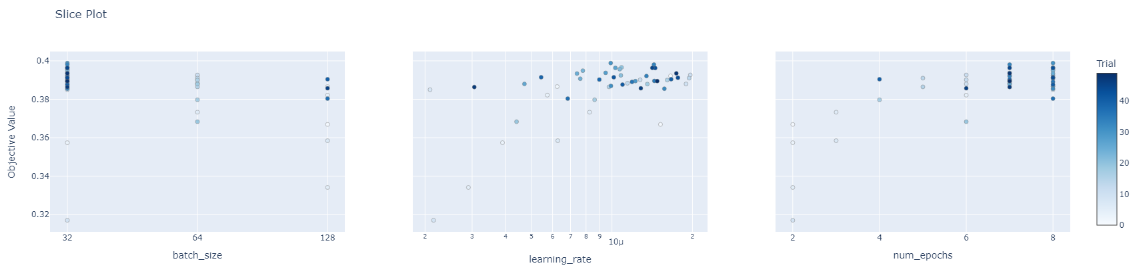
see that it was the most important parameter.We can verify that,by the fact that,even the smallest of changes in the learning rate,changed the validation f1 score of our model significantly.After following up on a paper concerning the performance of the Bert model with different hyper parameters, learning rates of 2e-5 ,5e-5 and 6e-5 were preferred,yet I decided to check all learning rates at 2e-5 and 2e-6 and the results were that values in between 10e-6 and 2e-5 were the best.Lastly the number of epochs that the paper(which will be at the bibliography at the end of the report) were [2,4],so I decided to add one more epoch and check all epochs between [2,5].Best number of epochs showed to be 4,but as we will see,the final result will be 3 to prevent overfitting.
The same course of action was taken as to the training of the DistilBert model.Firstly we have the plot optimization history:



This time around the best model was found at the middle of the training and not at the end.To be specific the best model was found at the 30th trial.Next we can check the importance of each parameter,as to the performance of the model:



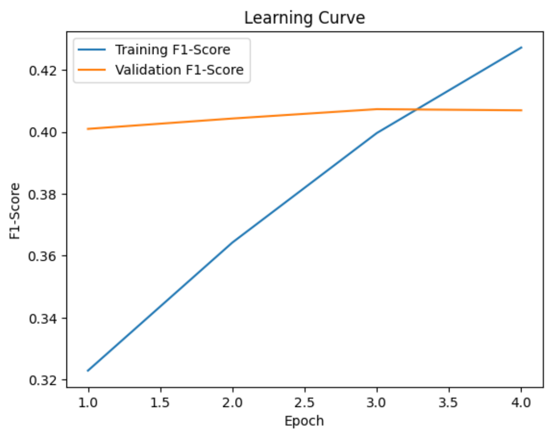
This came to me as a surprise,since having already ran and completed my experiments for the GreekBert model,I expected similar results.As we can see from the plot,the most important parameter this time around was the number of epochs the model was trained and not the learning rate.Finally we have the slice plot:

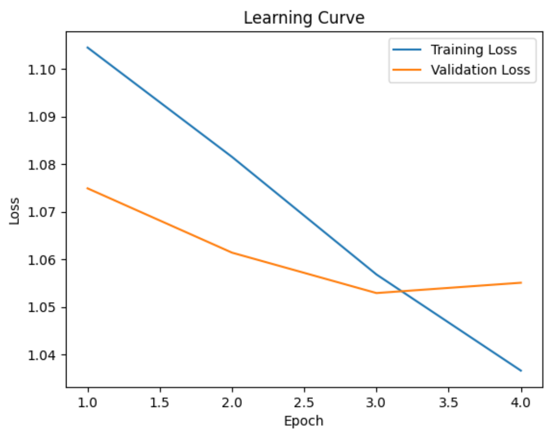


By looking at the slice plot we can see that batch size of 32 was preferred here as well.The learning rate this time was again in the previous stated ranges but alsmost all the values in that ranged returned similar results.Lastly the most important parameter for the DistilBert,the number of epochs.By reading the paper for Bert the number of epochs preferred were at most 4,we can see by looking at the results in the notebook,that (for the Bert Model) after the 3rd or 4th epoch,the model would start to overfit.Yet training times for the DistilBert were significantly lower(1 Epoch for Bert was around 2:41 minutes but for Distil 1:11) so I decided to set epochs to vary from 2 to 8.By looking at the notebook and at the slice plot,we will see that after 4 epochs the improvement is minimal,yet there is improvement.What is also interesting is that especially for the DistilBert,the Training scores were lower than the validation scores,which would hint that the validation set was easier to predict than the training set.This occured to a degree for the GreekBert too,yet after some epochs the training scores would improve ,where the validation scores would remain the same,so overfitting would occur.

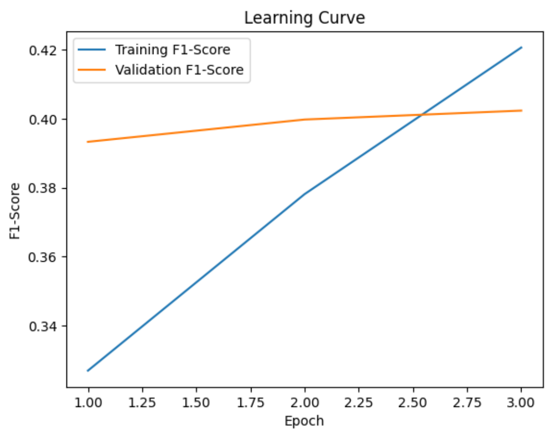
3.3. Evaluation

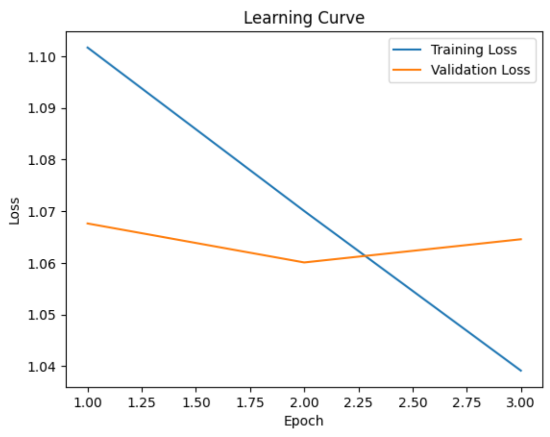
After the trials,a final trial for each model was performed using the best parameters that optuna recommended. After performing the final trial for the GreekBert these were the learning curves:





By looking at the learning curve,we can see that the validation set stops improving after the 3rd epoch,whereas the training set improves.So we can deduct that after the 3rd epoch our model overfits,so I performed one more trial with 1 less epoch:

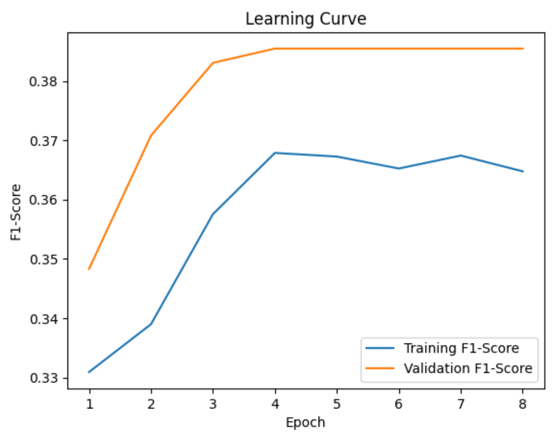


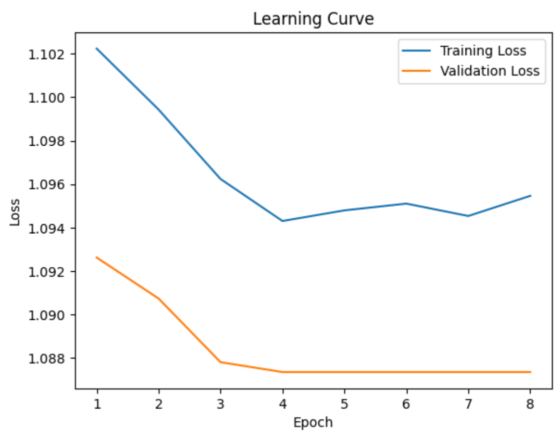


The final model still overfits slightly,but I think it is acceptable since the fact that a difference of 0.02 means that in 100 test cases the model will not predict correctly 2 tweets in the validation set(compared to the training set). For the GreekBert the final Scores were:

Metric	Score
Training Accuracy	0.43
Training F1	0.42
Validation Accuracy	0.41
Validation F1	0.40

As to the performance of the DistilBert,its performance was not as good as that of GreekBert.The learning curves show that the validation set was easier to predict compared to the training set and the difference was a greater than the one of GreekBert:





We can see clearly that the model performs better on the validation set on all epochs.This could hint an urepresentative dataset.Below is the table of the final Scores:

Metric	Score
Training Accuracy	0.37
Training F1	0.36
Validation Accuracy	0.39
Validation F1	0.39

4. Results and Overall Analysis

4.1. Results Analysis

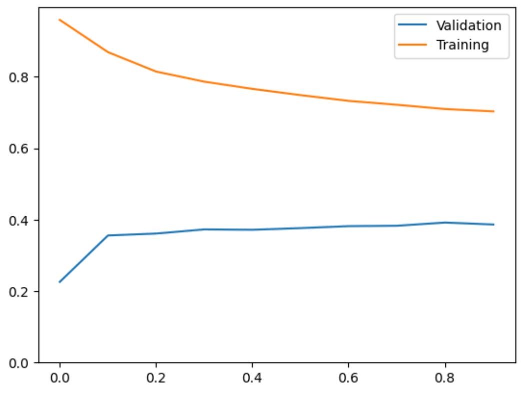
These results are expected.NLP is a difficult task to perform,as is sentiment classification of text.By taking that in to account,the fact that the classification task is in Greek and in English(for which most pre-processing libraries are mostly used for),and the fact that some of the tweets on the training set are falsely classified,we can deduct that our performance is acceptable.BERT is a state of the art Neural Network and if the Dataset was completely correct,we could maybe achieve higher scores.In this project(compared to the previous ones) both Accuracy and F1 were computed,but F1 was preferred.This of little importance,since our dataset is balanced both F1 and Accuracy will converge to the same value and both can be used as valid metrics in this classification task.The comparison below was done between the best model found which was GreekBert.

4.1.1. Best trial. The best Trial had an F1-Score of 0.41 with parameters:

- Learning Rate 7.305090422351797e-06
- Batch Size 32
- Epochs 4

4.2. Comparison with the first project

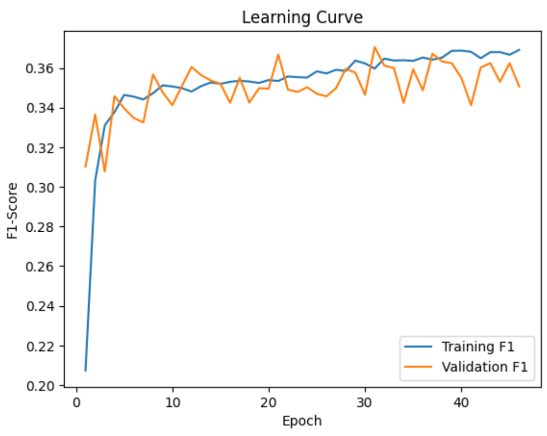
Comparing our current results with the ones of the first project were Logistic Regression was used,we can see a huge difference. To begin with the F1 score was 0.37 in comparison to 0.41 ,and we can see major signs of over-fitting by looking at the learning curve of the logistic regression classifier:



We can also see that both curves seem to converge,and with a larger dataset we could maybe converge.We can deduct that with increase in performance,we also managed to prevent this major over-fitting,even though our current model still over-fits to a degree.

4.3. Comparison with the second project

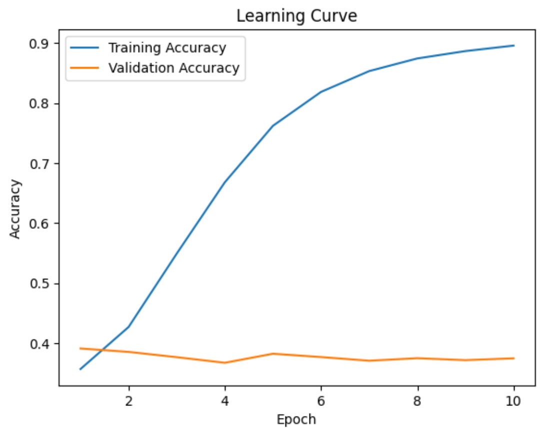
Compared to the second project,our scored increased,but both NNs prevented overfitting.Below is the learning curve when using the Feed Forward Neural Network:



The F1-Score from the training of this model was : 0.37,which is a lot lower than our current f1 score.Still both NNs could benefit from a larger dataset and no falsely classified items in the training set but in comparison to over-fitting both models performed relatively the same.Even though the learning curve of Bert shows a much larger difference between the curves of validation and training,but as we can see by the values on the y-axis of the curve the difference is the same(of around 0.02).In conclusion we managed to performed much greater than the Feed Forward Neural Network while not increasing the over-fitting already occurring.

4.4. Comparison with the third project

At the 3rd project,the metric F1 was not used and accuracy was preferred.As I stated before,both metrics are valid and can even be compared since on a balanced dataset both converge to the same value.The best run for the Recurrent Neural Network had an accuracy of : 0.396 and ours was 0.42 which is an improvement.A learning curve for the RNN (Recurrent Neural Network) is not provided since the best number of epochs was 1.Below is a learning curve which was of a trial (not the best one) where 10 epochs were used,and the other parameters were the optimal ones:



We can see that the validation and training accuracies are close and our model generalizes well,but after that it over-fits.With that being said,we can deduct that both models generalize well with slight over-fitting taking place.In conclusion,the Greek-Bert model is the best one so far,which is expected since it is a tranformer based model.That being said,it is also the most hardware demanding.The 50 trials optuna performed took about 9.6 hours running on an RTX 3080.To compare that with the other models,the RNN model took for about 200 trials around 5 minutes.

5. Bibliography

References

[1] Pre-training of Deep Bidirectional Transformers for Language Understanding

[2] BERT Fine-Tuning Tutorial with PyTorch