Deep Learning for NLP

Konstantinos Malonas
sdi: 7115112200020

Fall Semester 2023

University of Athens Department of Informatics and Telecommunications

Artificial Intelligence II (M138, M226, M262, M325)

Abstract

For this project we will build a sentiment classifier that classifies tweets written in Greek language about the Greek elections. The classifier will classify the tweets as NEGATIVE, POSITIVE and NEUTRAL classes. First, we will start by exploring my data, which includes plotting barplots, word clouds, etc., to recognize potential patterns, understand the predominant sentiment classifications for tweets associated with each party, and so on. Then we will continue with experiments to find the best parameters for our two transformers bert-base-greek-uncased-v1 and DistilGREEK-BERT and also we will plot the f1-recall-precision barplots from the training of our model in order to depict its performance.

Data processing and analysis

Pre-processing

First we start with the pre-processing of our dataset. Initially we check if there are any null values in any column with the **info** method. We observe that there are no null values.

```
RangeIndex: 36630 entries, 0 to 36629
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   New_ID     36630 non-null  int64
 1   Text       36630 non-null  object
 2   Sentiment  36630 non-null  object
 3   Party      36630 non-null  object
dtypes: int64(1), object(3)
memory usage: 1.1+ MB
```

Turn categorical values to numerical with LabelEncoder

It is a best practice to turn the categorical values of our classes from our dataset to numerical. To achieve that, we use the **LabelEncoder** object. We turn our classes from NEGATIVE to 0, from NEUTRAL to 1, and lastly from POSITIVE to 2.

```
df_train_set['Sentiment'] = le.fit_transform(df_train_set['Sentiment'])
df_valid_set['Sentiment'] = le.fit_transform(df_valid_set['Sentiment'])
```

The preprocess_tweet function

In the data preprocessing stage, the **preprocess_tweet** function plays a crucial role in standardizing and cleaning the tweet data. This function performs several key operations to ensure that the text data is in a suitable format for analysis and modeling.

Firstly, we convert all text to lowercase, which helps in maintaining consistency as text data often contains a mix of uppercase and lowercase letters, and in most contexts, these variations are not meaningful. By standardizing the case, we reduce the complexity of the text and avoid treating the same words in different cases as different tokens.

Next, we remove mentions, which are words starting with the '@' character. Mentions in tweets usually refer to usernames and do not contribute meaningful information for our analysis. Similarly, URLs are removed since they often act as noise in the text analysis, being unique and not contributing to the overall understanding of the tweet's sentiment.

We also remove all non-Greek characters, focusing our analysis strictly on Greek text. This step is crucial as it eliminates irrelevant characters or symbols that might be present in the tweets but do not contribute to their semantic meaning.

Lastly, we remove Greek stop words using the **stopwords_el_2.json** file. Stopwords are common words that appear frequently in the text but do not carry significant meaning and are often filtered out before processing text data. Removing these words helps in reducing the dimensionality of the data and focuses the analysis on the words that carry more meaning and sentiment.

Below is the code of preprocess_tweet function:

```
with open('/kaggle/input/stopwords/stopwords_el_2.json', 'r',
encoding='utf-8') as file:
    greek_stopwords = json.load(file)

def preprocess_tweet(tweet):
    tweet = tweet.lower().replace('_', ' ')
    tweet = re.sub(r'@\w+', '', tweet)
    tweet = re.sub(r'http\S+', '', tweet)
    tweet =
re.sub(r'[^αβγδεζηθικλμνξοπρστυφχψωςάέίόώύήΑΒΓΔΕΖΗΘΙΚΛΜΝΞΟΠΡΣΤΥΦΧΨΩ0
-9\s]', '', tweet)

    tweet_words = tweet.split()
    cleaned_words = [word for word in tweet_words if word not in
greek_stopwords]
    tweet = ' '.join(cleaned_words)
    tweet = tweet.strip()

    return tweet
```

Below is the code from lemmatize_tokenize_text function that we used for tokenization and lemmatization:

```
nlp =
spacy.load('/kaggle/input/el-core-news-lg-4/el_core_news_lg_3/el_core_ne
ws_lg-3.7.0')

def lemmatize_tokenize_text(text):
    doc = nlp(text)
    return ' '.join([token.lemma_ for token in doc])


df_train_set['Text'] =
df_train_set['Text'].apply(lemmatize_tokenize_text)
df_test_set['Text'] = df_test_set['Text'].apply(lemmatize_tokenize_text)
df_valid_set['Text'] =
df_valid_set['Text'].apply(lemmatize_tokenize_text)
```

If we print the values of the **Text** column of the train,test,validation sets we will notice that the words are lemmatized.

```
0    απολυμανση κοριοι απεντομωση κοριος απολυμανσε...
1    έξι νέος επιστολή μακεδονία καίνε νδ μητσοτάκη...
2     ισχυρός κκε δύναμη λαός βουλή καθημερινός αγώνας
3            μνημονιακότατο μερα25 εκλογες 2019 8 κκε
4                  συγκλονιστικός ψυχασθένεια τσίπρας
Name: Text, dtype: object

0    κυριάκος μητσοτάκης ξέρω μουσείο βεργίνας μέσω...
1    συνέντευξη υποψήφιος βουλευτής νέος δημοκρατία...
2    εκλογή μαθητής φοιτητής ψηφίζω ίδιος τρόπος αγ...
3    γεννηματά κιναλ γίνομαι δεκανίκι κανενός ενδια...
4    κυριακός εκλογή οκτώβρης 1993 ξημερώματα δευτέ...
Name: Text, dtype: object

0    θελεις μιλησεις βοσκοτοπια αιγιαλος παραγραφή ...
1    τσίπρας ζητήζω αντιπολίτευση συμμετέχω διαδικα...
2    σωστος ελληνας δημοκρατης ελληνας εξωτερικου ε...
3    βλέπεις ενδιαφέρω μητσοτακηδας γιατί πήγε κότε...
4    συνέντευξη μητσοτάκης αίρεση 13ος σύνταξη αύξη...
Name: Text, dtype: object
```

The unique_words_num function

After we pre-processed our text we printed the unique words of each set with the use of **unique_words_num** function

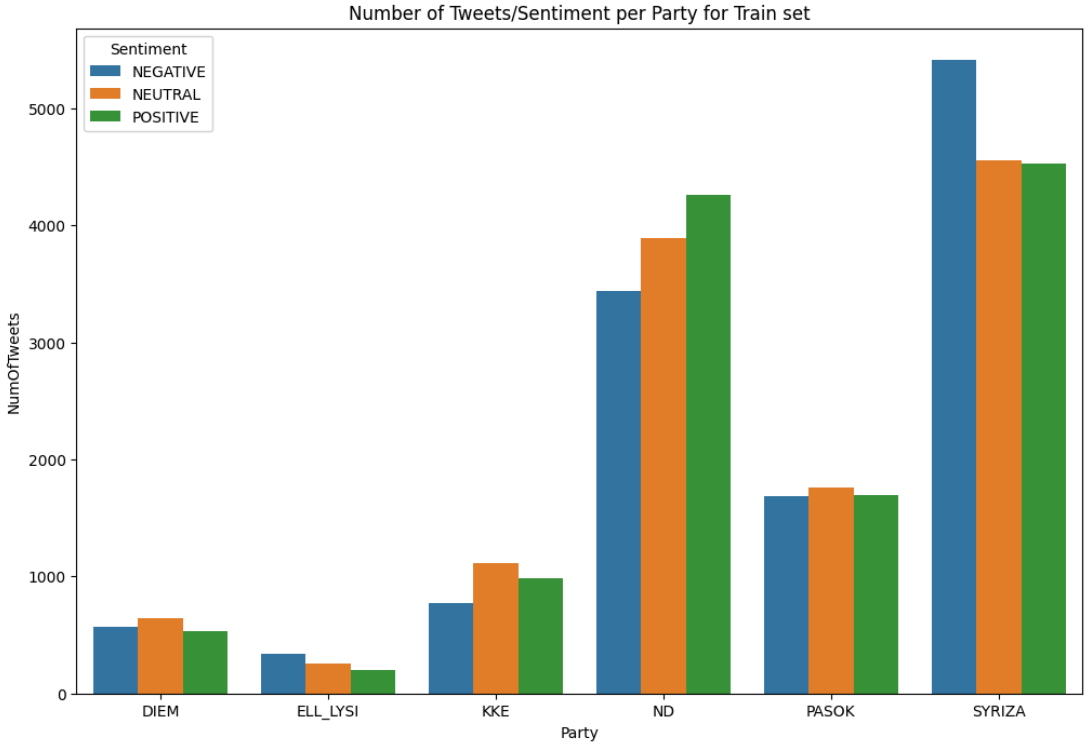
```
def unique_words_num(tweets):
    # Function that counts the number of the unique words from the Text
column of each dataframe
    words = set()
    for tweet in tweets:
        words.update(tweet.split())
    return len(words)

Result:
Num of unique words in df_train_set: 58389
Num of unique words in df_test_set: 26263
Num of unique words in df_valid_set: 16639
```

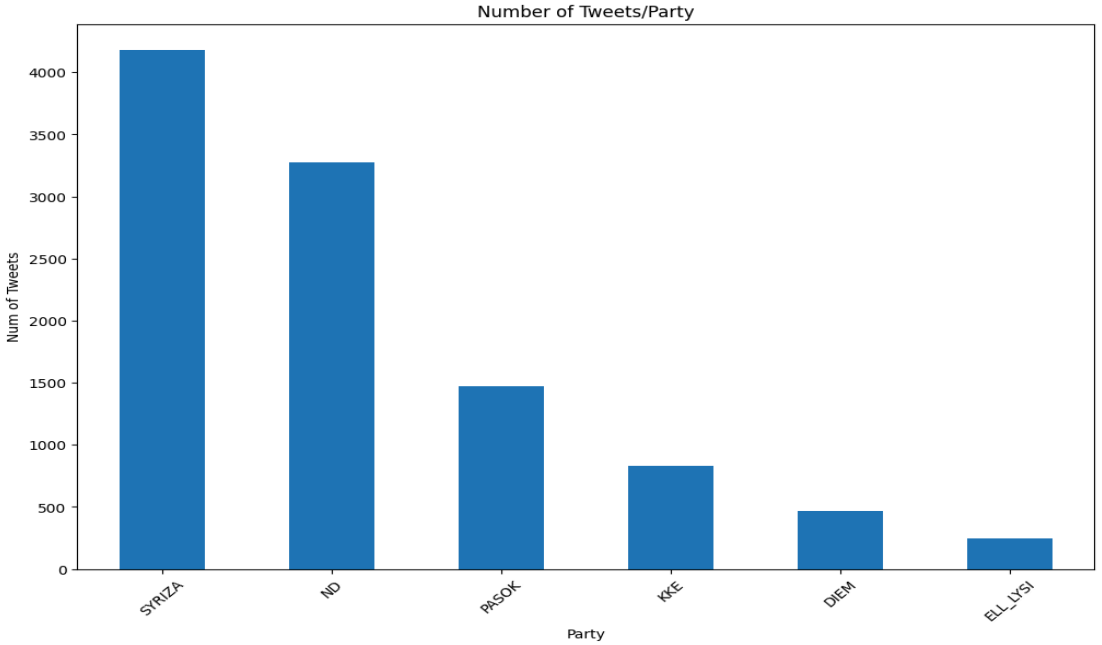
Analysis

For each set (train, validation and test) we plotted some barplots to visualize the number of tweets about each party and the emotion of these tweets. Also we plot the total number of tweets for each party. Also after lemmatization and the general preprocessing of the tweets we plotted the corresponding word cloud for each set.

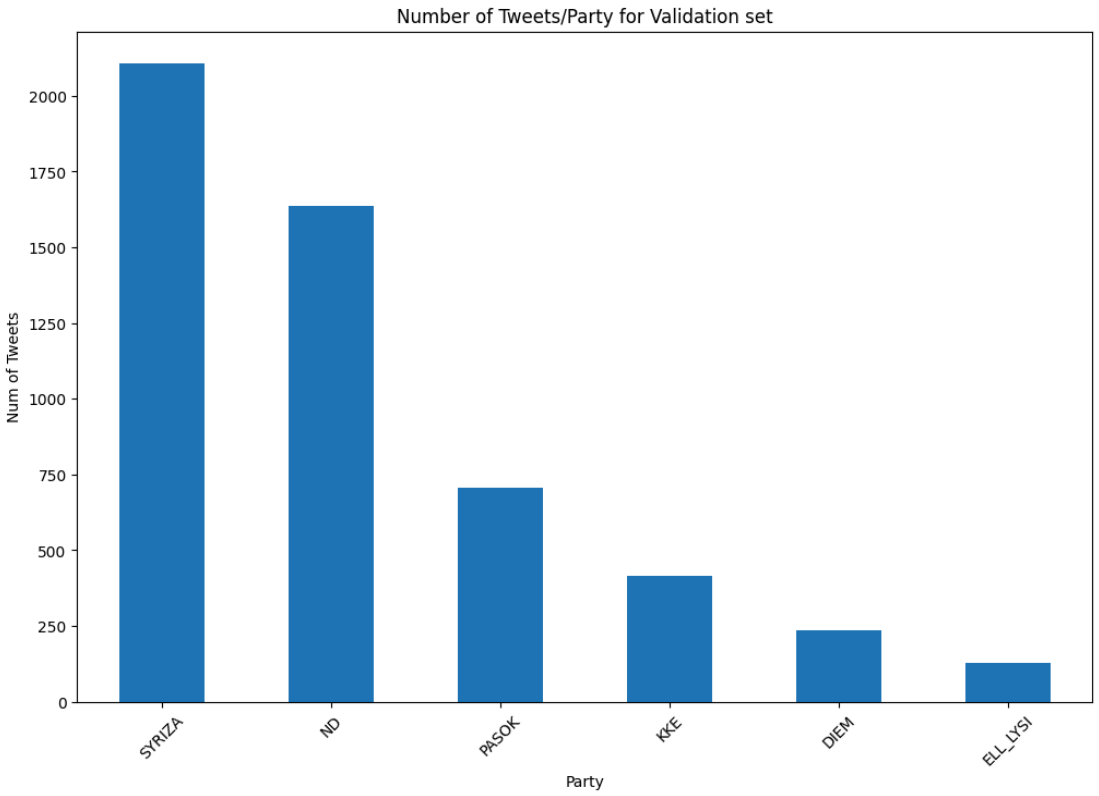
Number of tweets and sentiment for each party



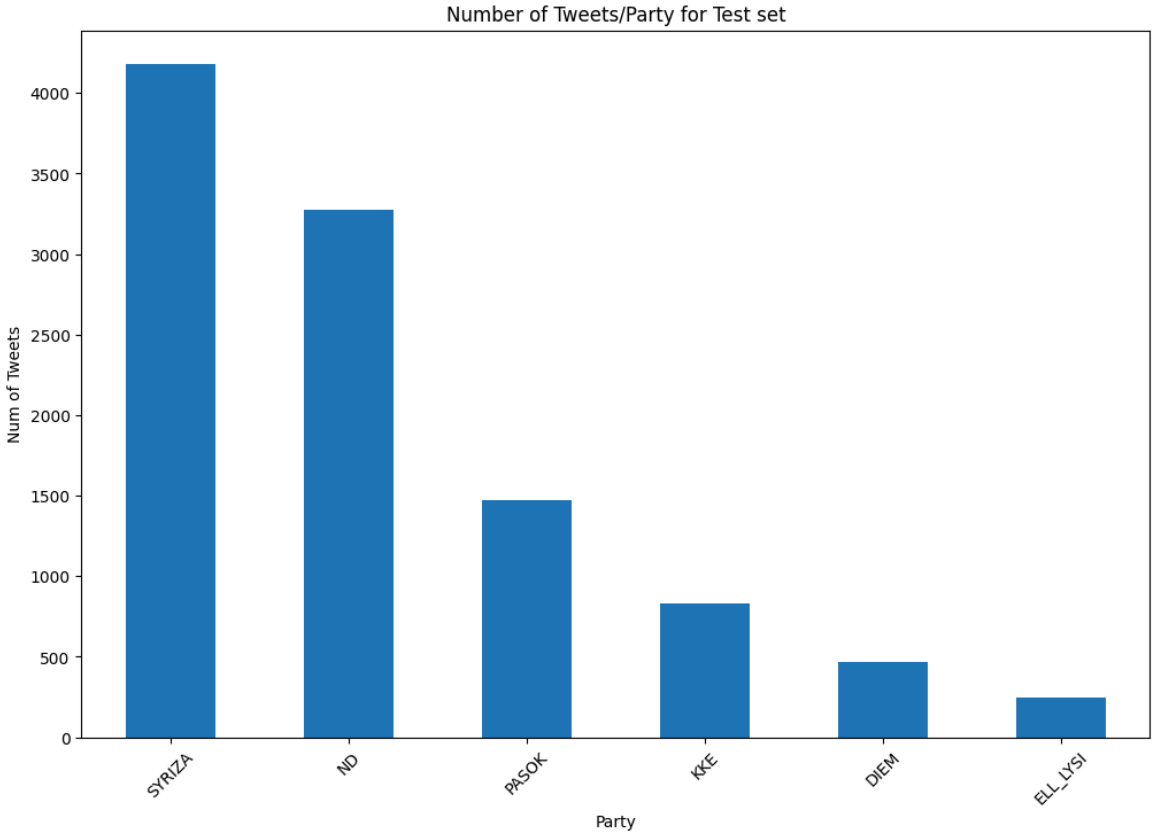
Number of tweets for each party for the train set



Number of tweets for each party from the validation set



Number of tweets for each party from the test set

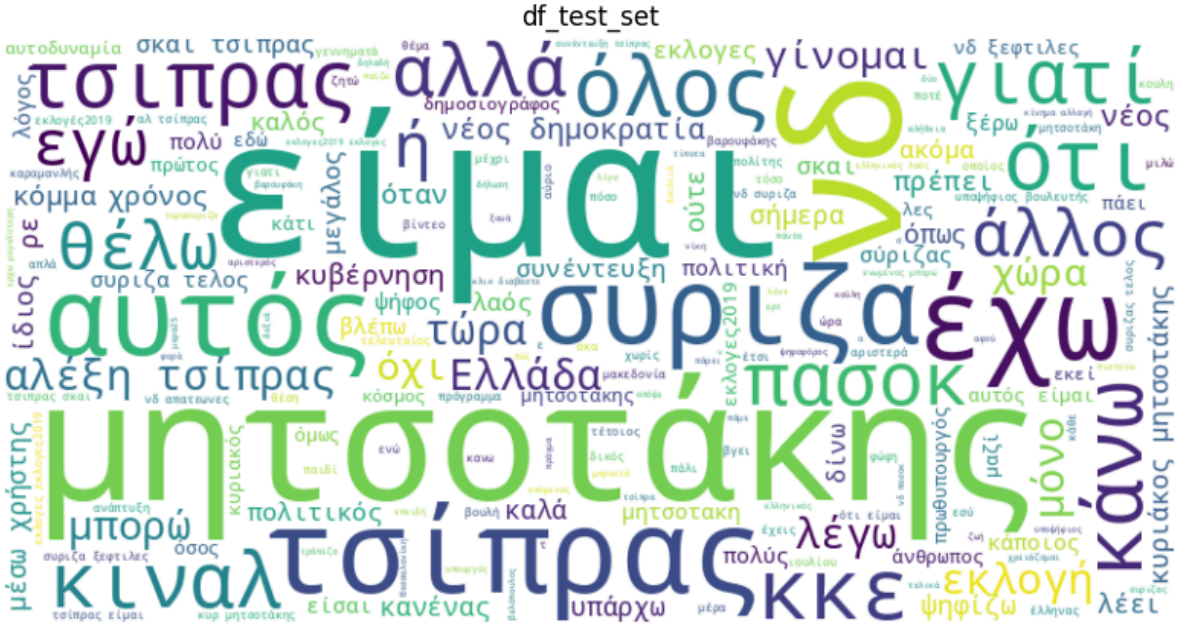


Word cloud for train set

In our analysis, we have utilized word clouds as a visual tool to represent the data from our train, test, and validation sets. Word clouds, or tag clouds, are graphical representations where the frequency of each word in the text data is depicted with various font sizes. The more frequently a word appears in the dataset, the larger and bolder it appears in the word cloud. This visualization technique is particularly useful for quickly identifying key themes and terms that are most prominent in large volumes of text.

The word clouds generated for our datasets reveal significant insights, especially concerning the political context of the tweets. Given that our data is centered around the Greek general elections, it is not surprising to find that the names of the two dominant parties at the time, Syriza (Σύριζα) and Nea Dimokratia (Νέα Δημοκρατία), along with the names of their party leaders, are among the most prominently featured words in the tweets. This observation aligns with the expected discourse during an election period, where discussion and commentary are often focused on the major political parties and their leaders.

Word cloud for test set



Word cloud for validation set

Data partitioning for train, test and validation

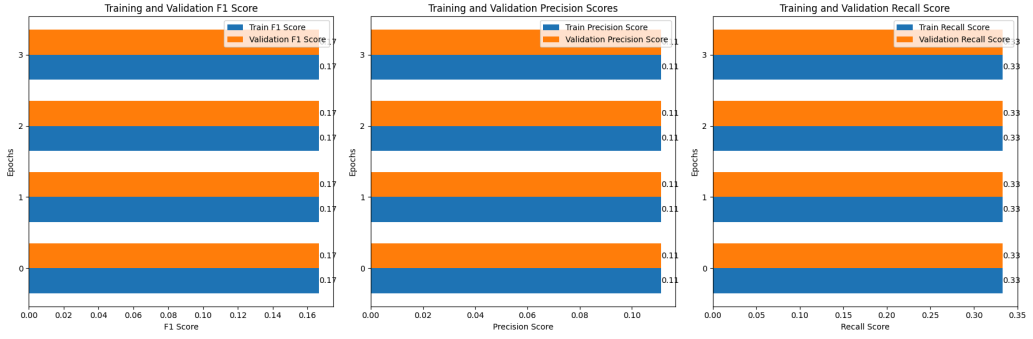
In our project, we initially worked with the given datasets as they were originally partitioned into separate training, validation, and testing sets. This conventional partitioning is a standard practice in machine learning, ensuring that models are trained, tuned, and tested on distinct data subsets.

Algorithms and Experiments

All the experiments were conducted with **V100 GPU** at the **google collab plus.** The results we acquired were disappointing to say the least. From the following graphs you can observe we got pretty much the same result from every experiment. The **train f1 score** is **0.1667**, the **validation f1 score** is **0.1667**, the training-validation **precision** is 0.1111 and the training-validation **recall** is 0.3333 at all cases.

bert-base-greek-uncased-v1

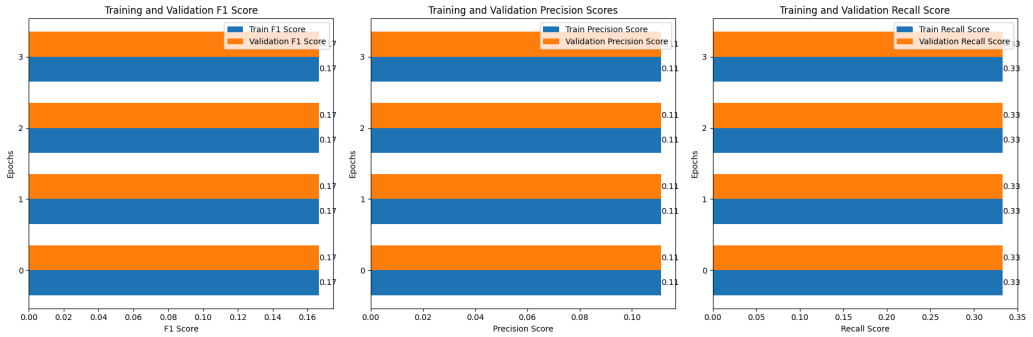
Learning rate experiments

F1 - Precision - Recall	Learning rate
	0.1
	0.01

	0.001

Mean training f1	Mean validation f1	Mean training precision	Mean validation precision	Mean training recall	Mean validation recall	Learning rate
0.1667	0.1667	0.1111	0.1111	0.3333	0.3333	0.1
0.1667	0.1667	0.1111	0.1111	0.3333	0.3333	0.01
0.1667	0.1667	0.1111	0.1111	0.3333	0.3333	0.001

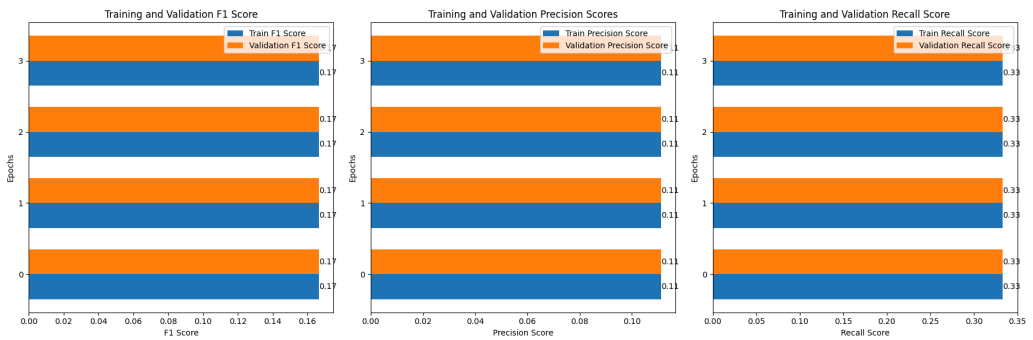
Optimizer experiments

F1 - Precision - Recall	Optimizer
	AdamW
	SGD

	Adam

Mean training f1	Mean validation f1	Mean training precision	Mean validation precision	Mean training recall	Mean validation recall	Learning rate
0.1667	0.1667	0.1111	0.1111	0.3333	0.3333	0.1
0.1667	0.1667	0.1111	0.1111	0.3333	0.3333	0.01
0.1667	0.1667	0.1111	0.1111	0.3333	0.3333	0.001

Batch size experiments

F1 - Precision - Recall	Batch size
	32
	64

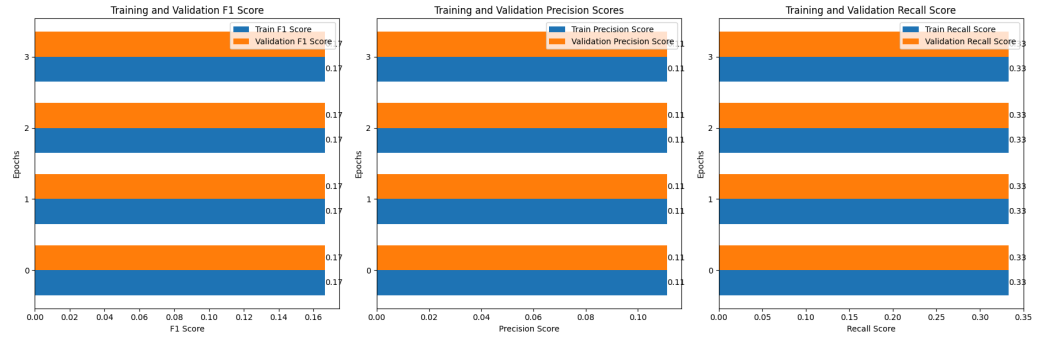
	128

Mean training f1	Mean validation f1	Mean training precision	Mean validation precision	Mean training recall	Mean validation recall	Batch size
0.1667	0.1667	0.1111	0.1111	0.3333	0.3333	32
0.1667	0.1667	0.1111	0.1111	0.3333	0.3333	64
0.1667	0.1667	0.1111	0.1111	0.3333	0.3333	128

DistilGREEK-BERT

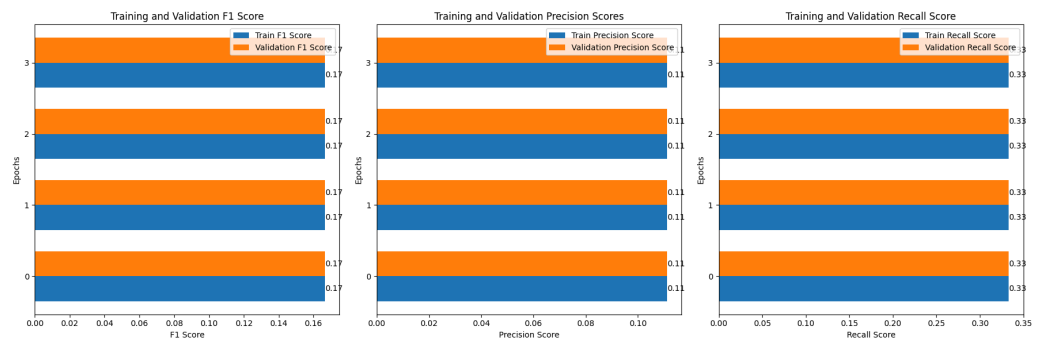
Learning rate experiments

F1 - Precision - Recall	Learning rate
	0.1

	0.01
	0.001

Mean training f1	Mean validation f1	Mean training precision	Mean validation precision	Mean training recall	Mean validation recall	Learning rate
0.1667	0.1667	0.1111	0.1111	0.3333	0.3333	0.1
0.1667	0.1667	0.1111	0.1111	0.3333	0.3333	0.01
0.1667	0.1667	0.1111	0.1111	0.3333	0.3333	0.001

Optimizer experiments

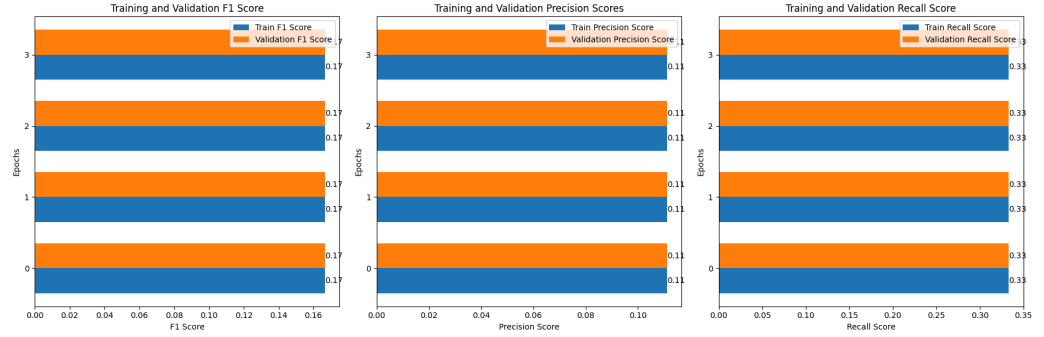
F1 - Precision - Recall	Optimizer
	AdamW

	SGD
	Adam

Mean training f1	Mean validation f1	Mean training precision	Mean validation precision	Mean training recall	Mean validation recall	Learning rate
0.1667	0.1667	0.1111	0.1111	0.3333	0.3333	0.1
0.1667	0.1667	0.1111	0.1111	0.3333	0.3333	0.01
0.1667	0.1667	0.1111	0.1111	0.3333	0.3333	0.001

Batch size experiments

F1 - Precision - Recall	Batch size
	32

	64
	128
	256

Mean training f1	Mean validation f1	Mean training precision	Mean validation precision	Mean training recall	Mean validation recall	Batch size
0.1667	0.1667	0.1111	0.1111	0.3333	0.3333	32
0.1667	0.1667	0.1111	0.1111	0.3333	0.3333	64
0.1667	0.1667	0.1111	0.1111	0.3333	0.3333	128
0.1667	0.1667	0.1111	0.1111	0.3333	0.3333	256

Hyper-parameter tuning

For the hyper-parameter tuning we conducted experiments with the learning rate, the optimizers and the batch size. In all our experiments we got almost identical results.

Optimization

For the optimization we used the AdamW optimizer. It was evident for every experiment that our model is not performing well.

Evaluation

For evaluation of every hyperparameter and for every experiment we were plotting the learning curves that was showing the f1, recall and precision training and validation scores. We focused mostly on increasing the f1 score since we are building a classifier.

f1: F1 Score is a measure that combines recall and precision. As we have seen there is a trade-off between precision and recall, F1 can therefore be used to measure how effectively our models make that trade-off. One important feature of the F1 score is that the result is zero if any of the components (precision or recall) fall to zero. Thereby it penalizes extreme negative values of either component.

Precision: Precision is a metric that gives you the proportion of true positives to the amount of total positives that the model predicts. It answers the question “Out of all the positive predictions we made, how many were true?”

Recall: Recall focuses on how good the model is at finding all the positives. Recall is also called true positive rate and answers the question “Out of all the data points that should be predicted as true, how many did we correctly predict as true?”

Results and overall analysis

Best trial

All the trials scored low and identical results

Comparison with the previous projects

In comparison with the previous projects all our models scored a lot lower.