UNIVERSITY OF ATHENS
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

Deep Learning for NLP

Student name: *<Konstantinos Fragos>*
sdi: <sdi2000207>

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

Contents

1. Abstract

The data of the task are tweets with id, text, party and sentiment(positive, neutral, negative). We want to train a model that takes as input a tweet with id, text, party, and predicts the sentiment of this tweet. We train our model with the training set given to us, and check for correctness with the valid set also given to us. We use the text segment to train the model as it is the one that gives us the good information. We process the text with various techniques that we will see below making our model faster and more efficient, we do vectorization, and we use logistic regression as the classifier of the problem. Then we use our validation set to make our predictions and draw our conclusions.

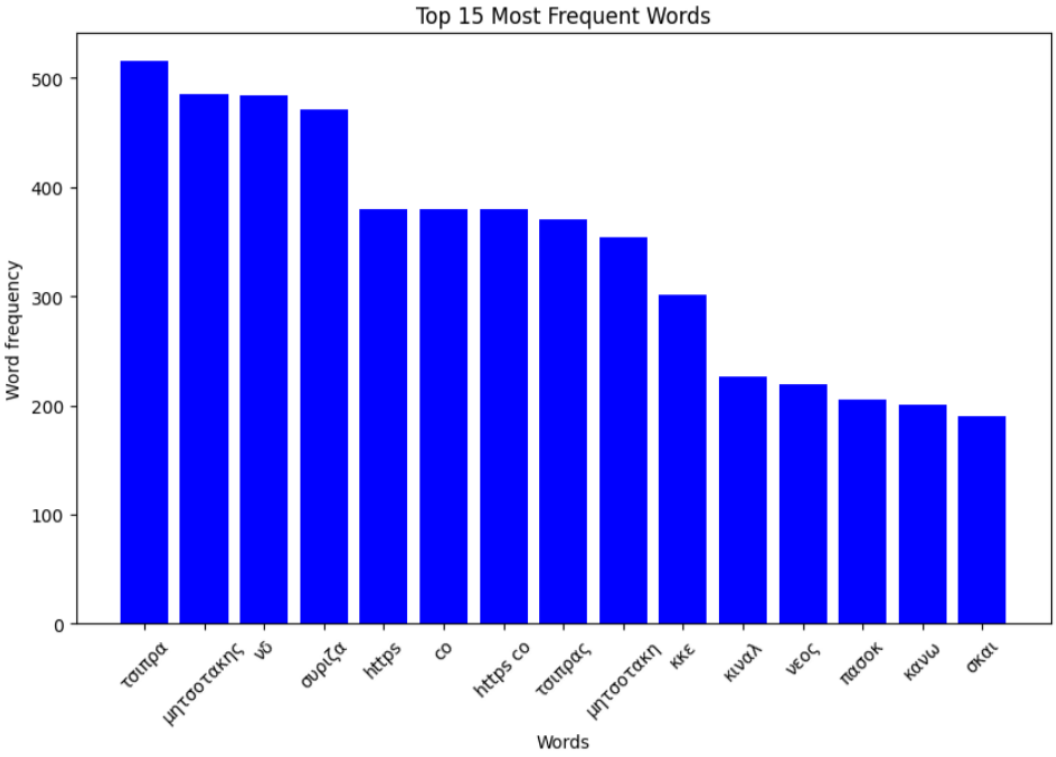
2. Data processing and analysis

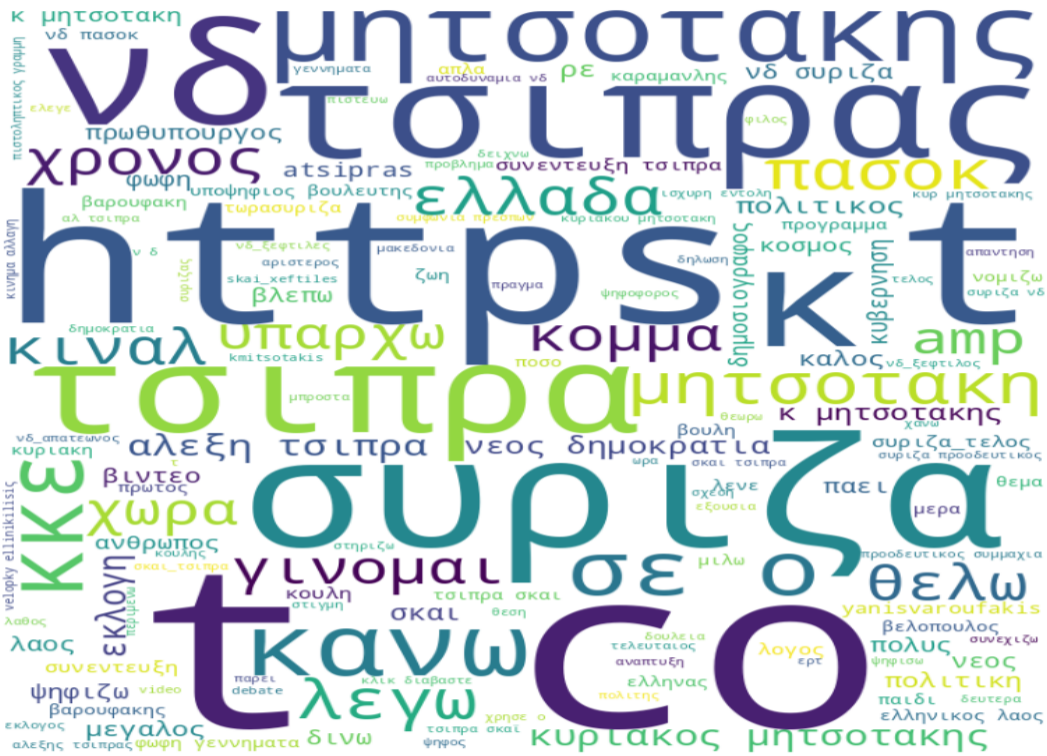
2.1. Pre-processing

The first technique I used for data preproccesing was to remove the stresses from the words and change the capitalization to lowercase. Then removing the hyphens from the words, and also removing words that started with the @ symbol. For each word also, I did lemmatization to get the root of the word in order to group the words together to make the vectorization more efficient. Finally, the removal of every word that contained a numeric digit and every word that was in a list of very common words that did not determine the sentiment of the tweet, the so-called stopwords.

2.2. Analysis

`Word frequency graph and word cloud`





2.3. Data partitioning for train, test and validation

I choose the default sets for train valid and test

2.4. Vectorization

To visualize the data I use the Tf-Idf vectorizer which uses the following formulas to get a number for each word in each tweet. First it calculates for each word the frequency of occurrence of the word in the tweet to the total words in the tweet and we obviously get a number <=1, this number is the TF of the word, then we calculate the IDF which is the number of tweets containing that word to the total tweets in our trainig set. Finally we calculate the IDF where it is the inverse of the DF logarithm. [1]

$$TF = \frac{\text{Frequency of word in a document}}{\text{Total number of words in document}}$$

$$DF = \frac{\text{Documents containing word W}}{\text{Total number of documents}}$$

$$IDF = \log\left(\frac{\text{Total number of documents}}{\text{Documents containing word W}}\right)$$

$$TF - IDF = TF * IDF$$

3. Algorithms and Experiments

3.1. Experiments

Trial	Negative	Neutral	Positive	Score
1	0.39	0.39	0.40	0.39
2	0.39	0.40	0.40	0.40
3	0.40	0.39	0.40	0.40
4	0.44	0.36	0.40	0.40
5	0.44	0.35	0.41	0.41

Table 1: Trials

3.1.1. Table of trials.

All trials using TF-idf vectorization
Trial 1 without data preprocessing
Trial 2 without data preprocessing, combining text with party
Trial 3 with data preprocessing, combining text with party
Trial 4 I change C hyperparameter to 0.1
Trial 5 I change the n gram to (1,2)

3.2. Hyper-parameter tuning

I make changes to logistic regression max iteration, and a good value is 1.000. For C, the best value is 0,1 after testing diferent values.

```
Best: 0.389267 using {'C': 0.1, 'penalty': 'l2', 'solver': 'saga'}
0.389211 (0.003485) with: {'C': 0.1, 'penalty': 'l2', 'solver': 'newton-cg'}
0.387004 (0.002793) with: {'C': 0.1, 'penalty': 'l2', 'solver': 'liblinear'}
0.389206 (0.003639) with: {'C': 0.1, 'penalty': 'l2', 'solver': 'lbfgs'}
0.389211 (0.003485) with: {'C': 0.1, 'penalty': 'l2', 'solver': 'sag'}
0.389267 (0.003580) with: {'C': 0.1, 'penalty': 'l2', 'solver': 'saga'}
```

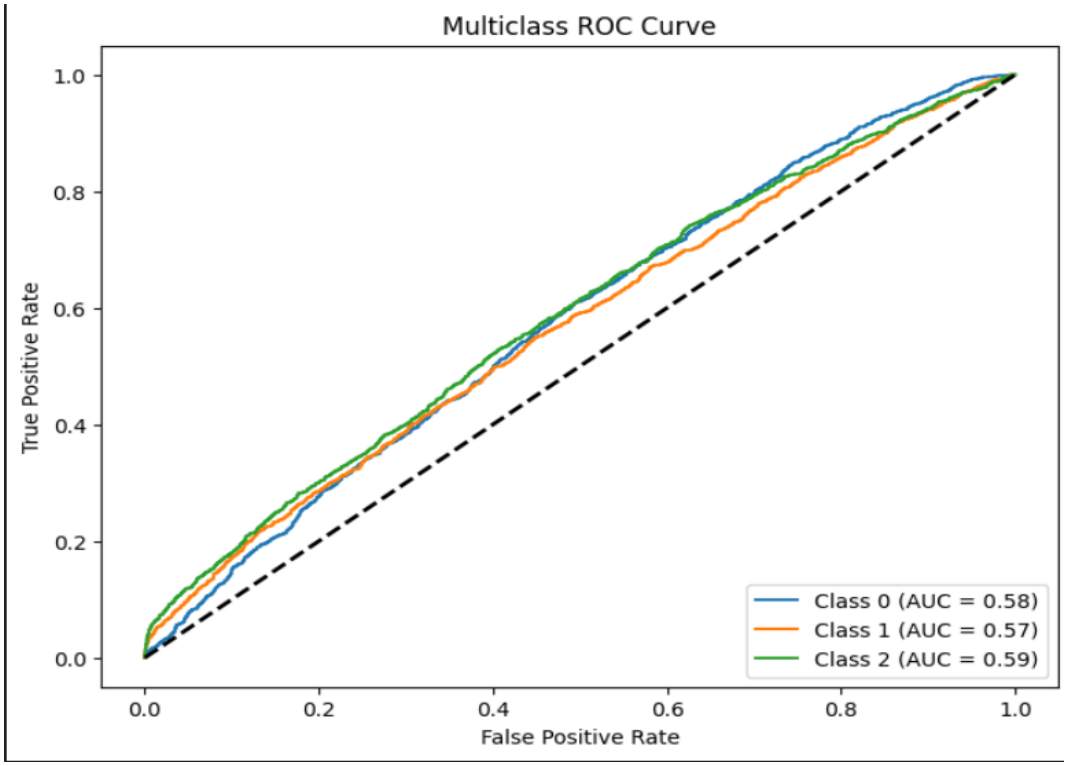
For different gradient descent solvers, I didn't find big differences to score

3.3. Optimization techniques

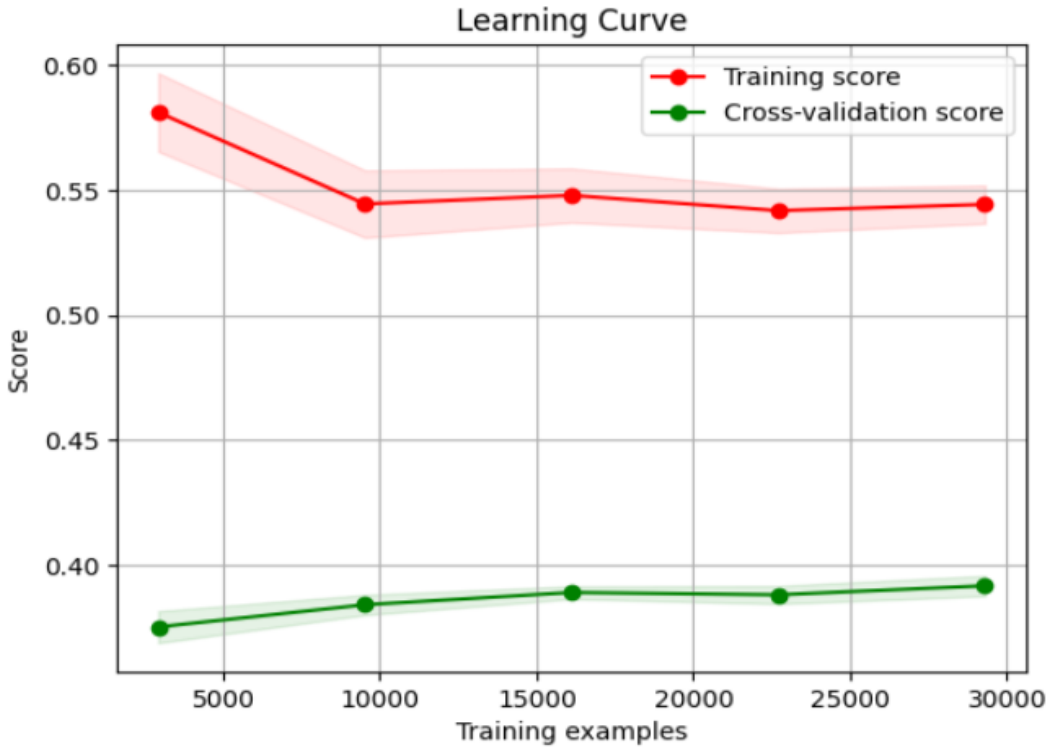
I use GridSearchCV optimization techinque, who it choose the best hyperparameters from a list of hyperparameters, by running the logistic regration code and finding the best f1 score. I change ngram to Tf-Idf to (1,2) and I get better results.

3.4. Evaluation

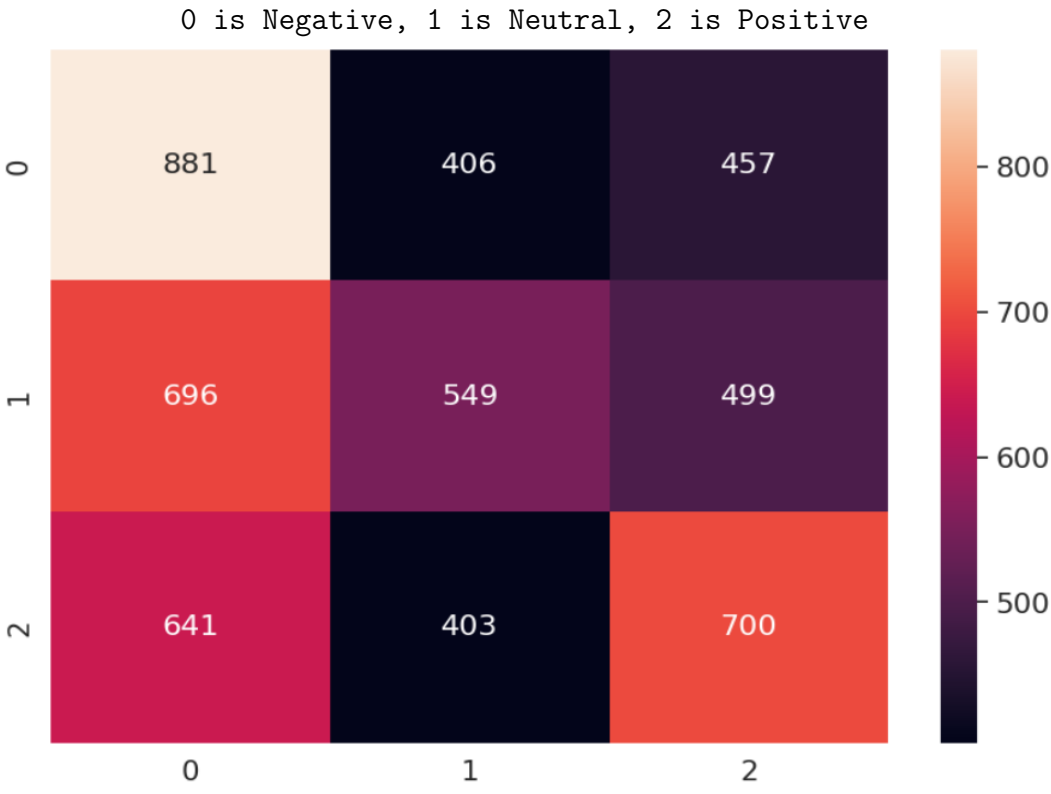
3.4.1. ROC curve.



3.4.2. Learning Curve.



3.4.3. Confusion matrix.



4. Results and Overall Analysis

4.1. Results Analysis

4.1.1. Best trial.

`My best trial find f1 score 41. The best trial had preprocessed data, combining with party column, using 1000 max iteration in logistic regration, and normalize c to 0.1 and n-gram of Tf-Idf to (1,2)`

4.2. Comparison with the first project

<Use only for projects 2,3,4>
<Comment the results. Why the results are better/worse/the same?>

4.3. Comparison with the second project

<Use only for projects 3,4>
<Comment the results. Why the results are better/worse/the same?>

4.4. Comparison with the third project

<Use only for project 4>
<Comment the results. Why the results are better/worse/the same?>

5. Bibliography

References

[1] *Vectorization techniques*.

[1] <Tf-Idf Vectorizer>