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

The problem we have to resonate in this case is exactly the same as in the 1st task, but this time we will use a different approach. In this paper we will build a neural network which will take as input our tweets in the form of vectors and classify them into positive, neural and negative. This time however the vectorization will be done in a different way. We will use the word2vec technique which we will present below. We will train our neural network, optimize it to optimize its hyperparameters and validate it on other data to evaluate it.

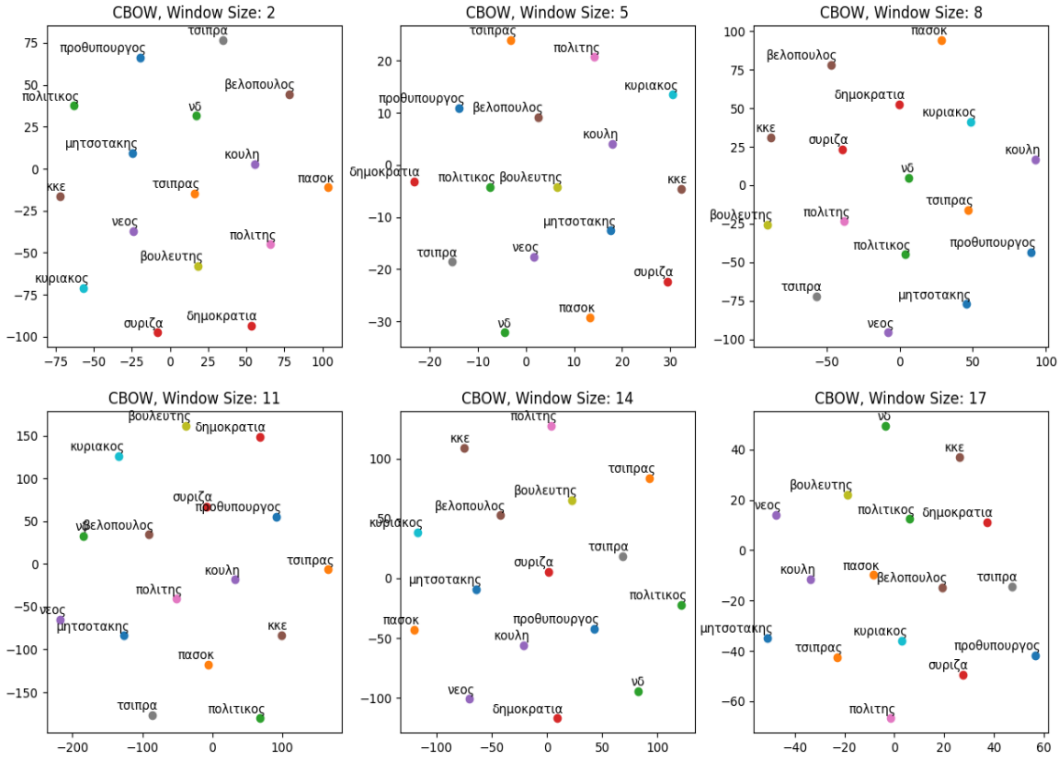
2. Data processing and analysis

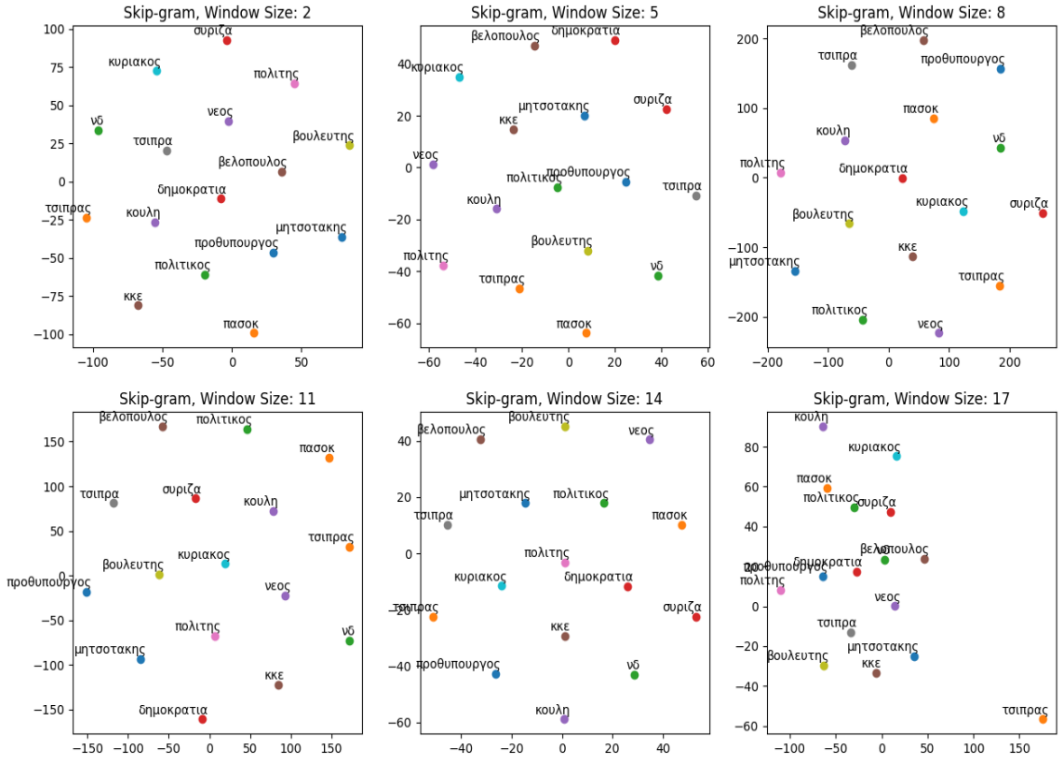
2.1. Pre-processing

For data preprocessing, we use exactly the same techniques as in the first task. That is, we made uppercase letters lowercase, removed accents, symbols and words starting with @. Then we did lematization of words and removed some key stopwords.

2.2. Analysis

We use different values for window size and word2vec algorithmn and we can show the results below





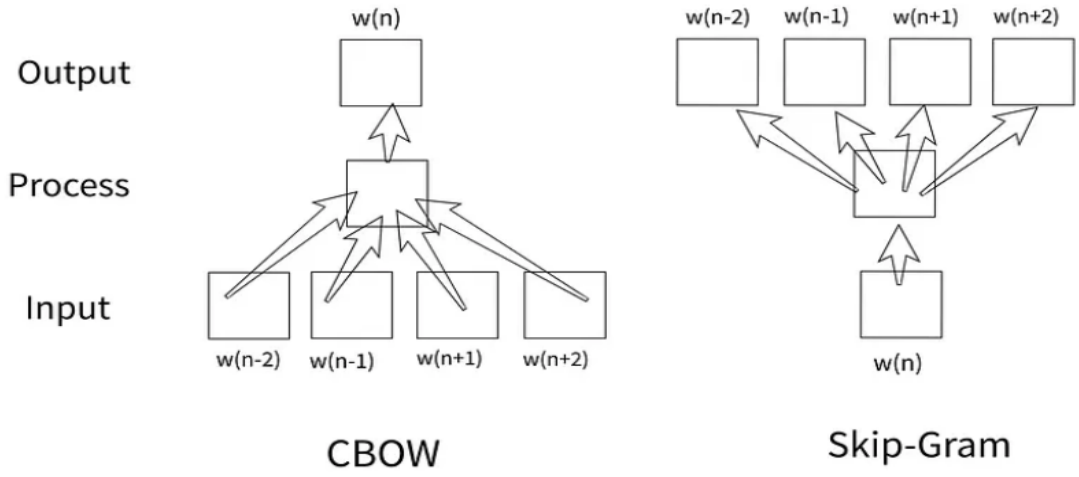
`At the end we choose CBOW algorithm with window size = 8`

2.3. Data partitioning for train, test and validation

`I choose the default sets for train valid and test`

2.4. Vectorization

The technique we used for vectorization in this paper is different from that of the first one. We used the word2vec technique which has nothing to do with the frequency of words as in the 1st paper but with the associations of words within the text. By giving each word a vector that reflects it based on its meaning within the text and finding the average of these vectors in each dimension, we get a final vector that reflects the vector of each text. The word2vec technique for each word finds its contex with a given window size, and words with similar contexts get nearby vectors and are considered similar. The algorithms used internally by word2vec are 2, CBOW and skip-gram, both of them use a neural network. The difference between them is that cbow takes as input the contex of the word and looks for the word that matches the contex, while skip-gram takes words and finds the probability of other words being close to that word.



[1] <Explain the technique used for vectorization>

3. Algorithms and Experiments

3.1. Experiments

At first, I use brute force method (data without preprocessing) to train and validate my model and I get Trial 1
Using preprocessing data, Trial 2
Using aproximatly optimization to vectorization, Trial 3
Upgrade epoch to 1000 and batch size to 128 Using optuna optimizer and find the optim hyperparameters

Trial	Negative	Neutral	Positive	Score
1	0.4593	0.1632	0.1356	0.2527
2	0.0866	0.2620	0.4608	0.2698
3	0.2798	0.3581	0.4008	0.3462
4	0.4165	0.3664	0.3926	0.3918
5	0.4393	0.3312	0.4164	0.4015

Table 1: Trials

3.1.1. Table of trials.

3.2. Hyper-parameter tuning

Our neural network has several hyperparameters that we have parameterized with optimization techniques and found the appropriate ones. First, the layers number on which after experiments we concluded that 3 is a good value. The size of the nodes of each layer is also a parameterizable size which after many experiments using the optuna framework which will be described below we came up with some specific values which you can see below in the picture with the optimal hyperparameters. Similarly with the batch sizes for train set

and validation set. Finally, we also found the best initial value for learning rate and also for dropout which we use after the activation function for which in this work we chose ReLu.

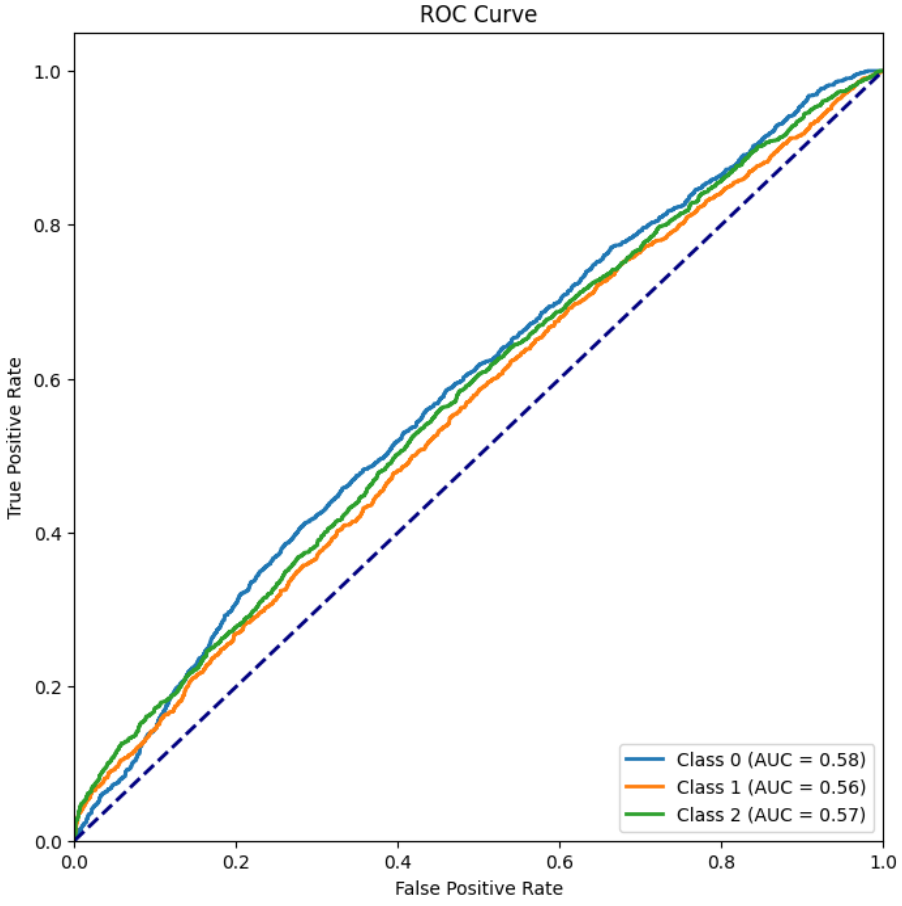
3.3. Optimization techniques

To optimize the model, I used the optuna framework which allowed me to train the model several times with different combinations of hyperparameters each time, so that based on which ones gave me the best f1 score, I could select those hyperparameters as well.

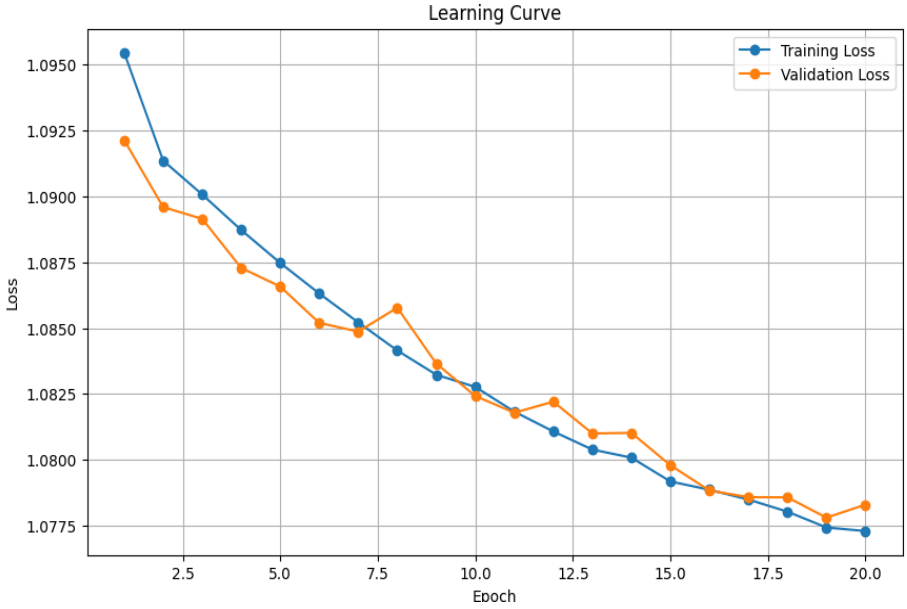
3.4. Evaluation

I evaluate my predictions using f1 score and all curves bellow

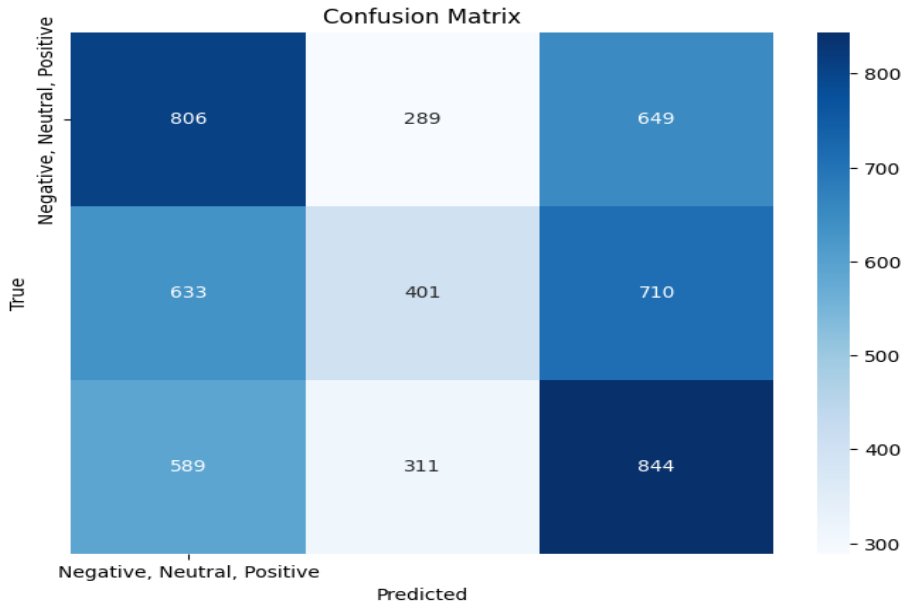
3.4.1. ROC curve.



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 40. Using data preprocesing , word2vec vectorizer, and pytorch newral network for predictions`

4.2. Comparison with the first project

<Use only for projects 2,3,4>
`My results was a litle bit worst than first project because Tf-idf vectorizer was better in this case than word2vec.`

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] Word2vec.

[1] <More about word2vec>