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

In this problem we tackle the classical text classification problem for election related tweets, and try to classify them into Negative, Neurtral and Positive. In this paper we will use a different kind of neural network the bidirectional stacked RNNs with LSTM or GRU cells. The main difference with simple neural networks, is that they take into account previous data, i.e. they maintain memory states. The difference between LSTM and GRU cells is the architecture of these cells, and in this paper we will discuss their differences. We use the same techniques (word2vec) to vectorize the words.

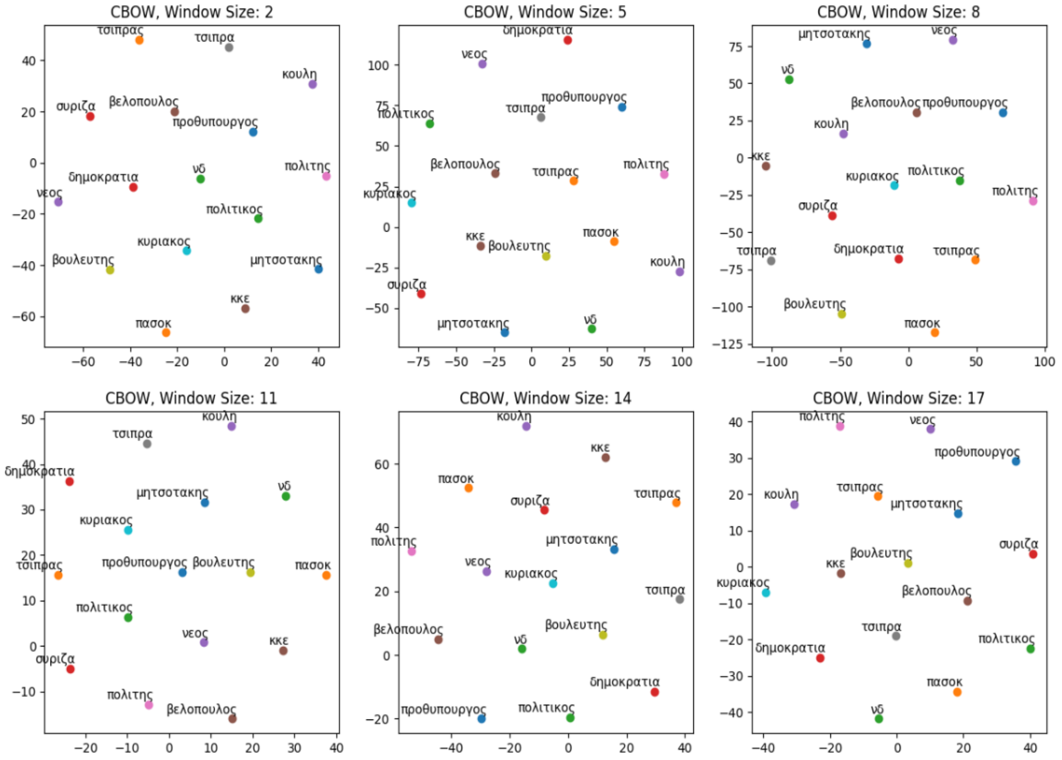
2. Data processing and analysis

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


CBOW, Window Size: 2

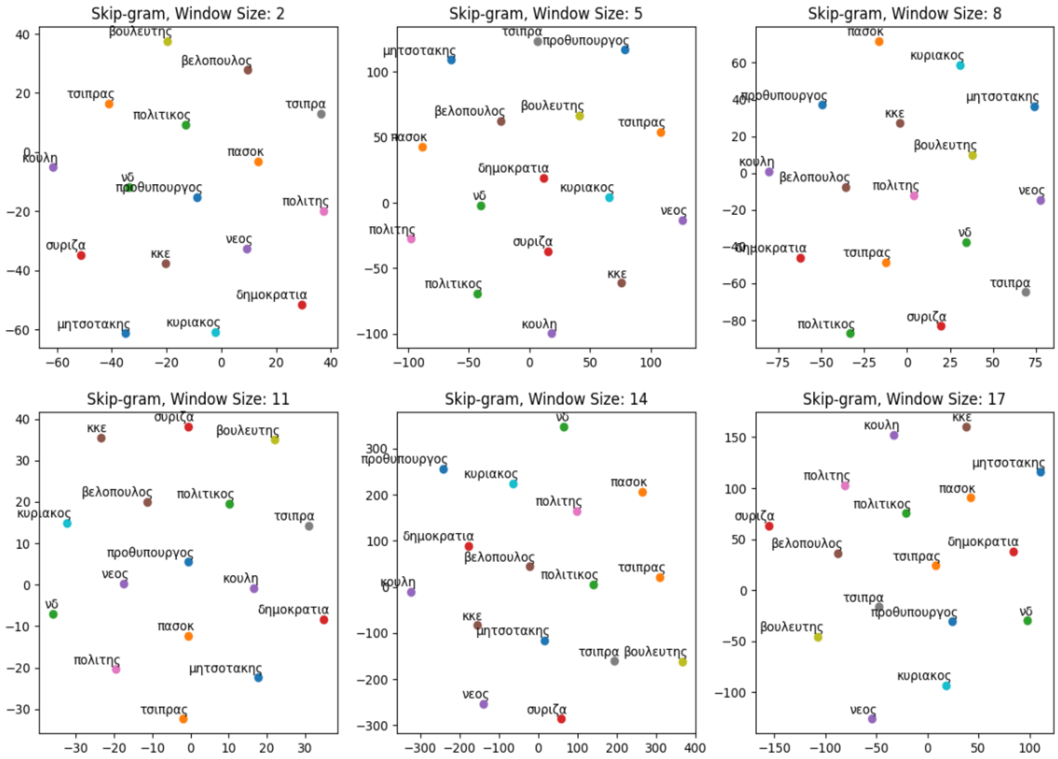
CBOW, Window Size: 5

CBOW, Window Size: 8

CBOW, Window Size: 11

CBOW, Window Size: 14

CBOW, Window Size: 17



2.3. Data partitioning for train, test and validation

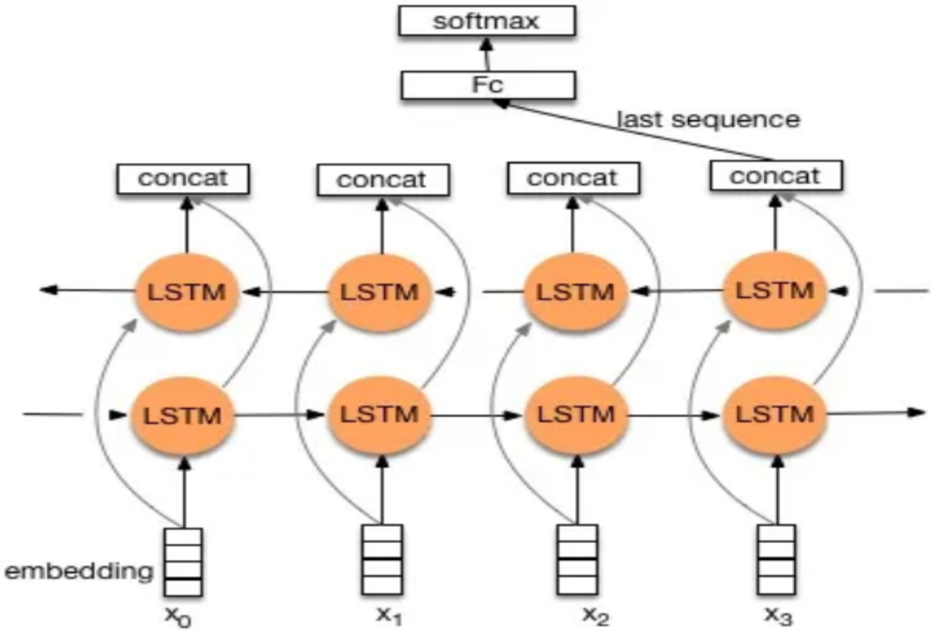
I choose the default sets for train valid and test

2.4. Vectorization

The technique for vectorization in this paper is quite close to the previous paper but changes a bit compared to the previous paper. In the previous work, we created 2-dimensional tensors of type (size × tweet_embedding), where each tweet_embedding was based on the average of the embeddings of each word. In this phase, we will create 3-dimensional tensors of type (size × max_sentence_length × tweet_embedding), i.e., we will go one step ahead of the previous work and not find the average of each sentence by the words but stick to 3 dimensions by padding with arrays of zeros in the tweets up to the largest tweet so that the 2nd dimension reflects the time step needed by RNN to work efficiently.
[3]

2.5. Model

Our model is a bidirectional rnn with LSTM or GRU cells. It is a many to one model as it takes as input a sequence of words (sentence) and the result we get is a sentiment. I use pack padding to skip the meaning of zero padded vectors[1][2]



2.6. Experiments

cell	lr	epochs	layers	hidden	skip-dropout	attenionHeads	gc	f1score
LSTM	0.0012	13	2	30	False	False	False	0.4
LSTM	0.00045	10	2	30	0.4	2	True	0.38
GRU	0.0177	12	2	2	False	1	True	0.17
LSTM	0.0276	14	1	2	False	1	False	0.3
GRU	0.0395	17	2	30	0.0	False	True	0.33
LSTM	0.0106	30	3	False	0.4	2	False	0.26
GRU	0.000156	12	3	64	0.4	2	True	0.39
LSTM	0.0304	18	1	30	False	False	False	0.36

2.6.1. Table of trials.

2.7. Hyper-parameter tuning

- cell, we choose the cell architecture between LSTM and GRU where we don't see big differences between the options
- learning rate, it plays a key role in the way the model learns and a value less than 0.001 is needed to keep the model from escaping
- epochs, a low value close to 10 as the model is observed to learn quickly to avoid overfitting
- num of stacked layers, after experimentation, a value close to 2 is ideal, with a higher value I notice that the model converges to only one value
- hidden-size, a value close to 30 is quite good because it drops the dimension from 100 to 30 and then drops smoothly to 3
- skip connections and dropout, I didn't notice that using it there was any improvement in the model
- attenion mechanism, similarly as the skip connection didn't improve my model
- gradient clipping, similarly I didn't notice that it helped much

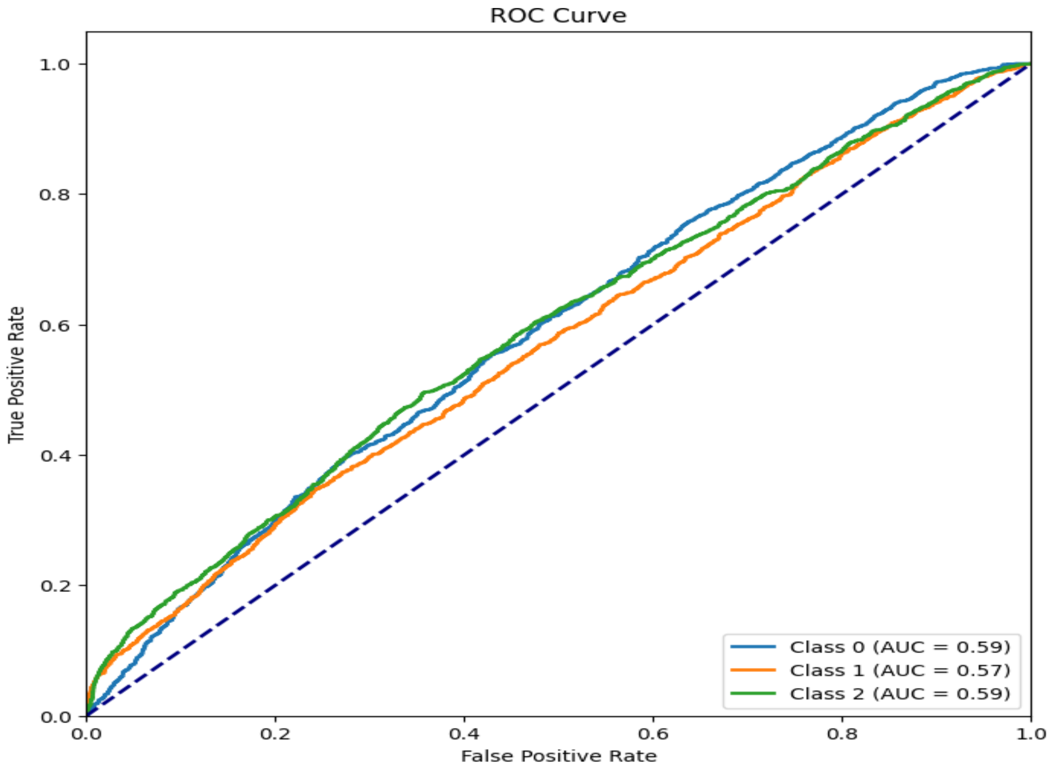
2.8. Optimization techniques

For optimization, I used optuna framework as I used in the previous assignments but with differnet hyparameters this time. I make a run of 100 trials to find the best results

2.9. Evaluation

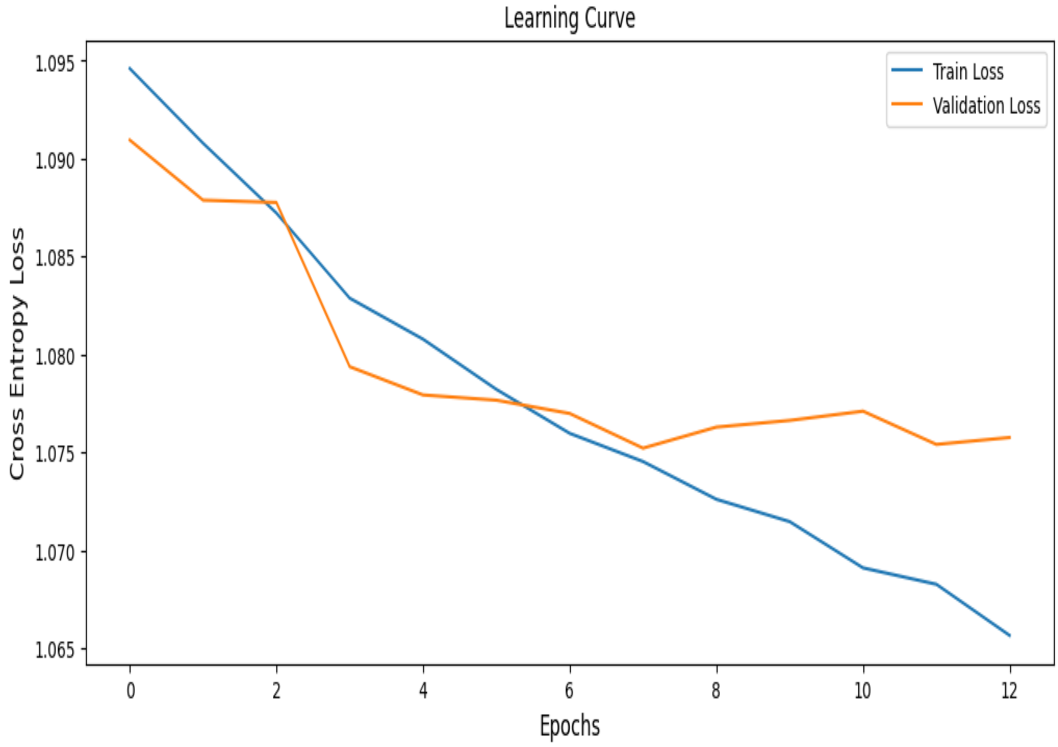
I use learning curve to see how the model learning, and roc curve and matrix to evaluate the score

2.9.1. ROC curve



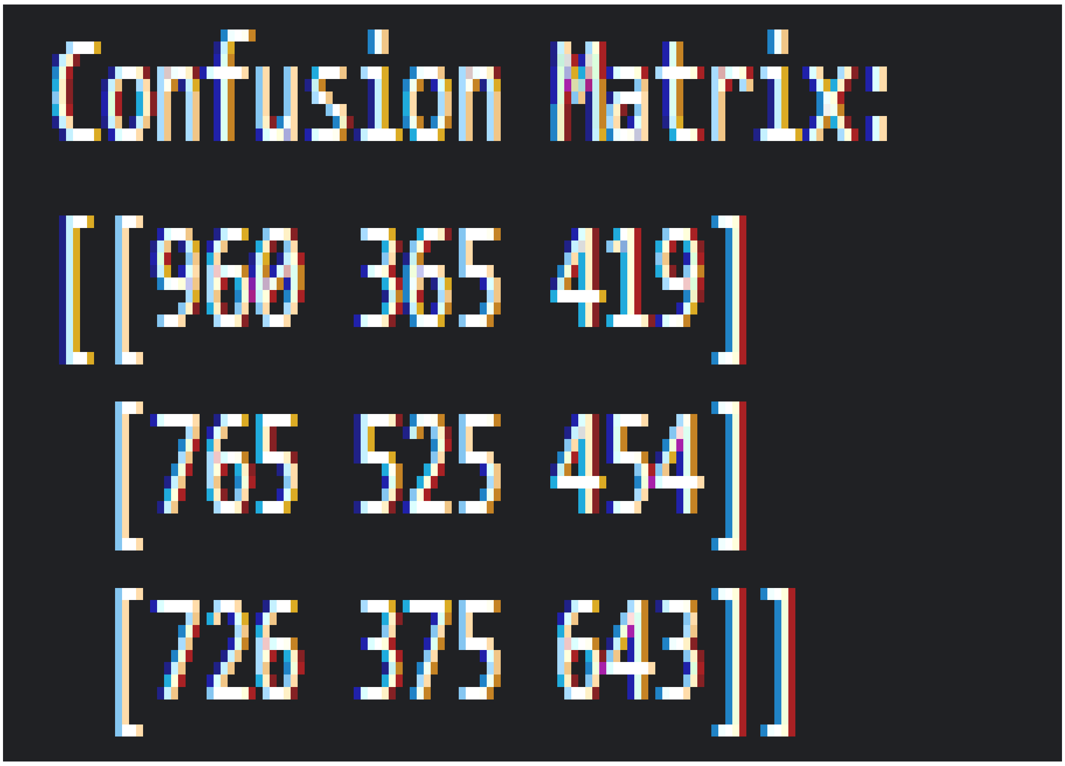
.

2.9.2. Learning Curve



.

2.9.3. Confusion matrix

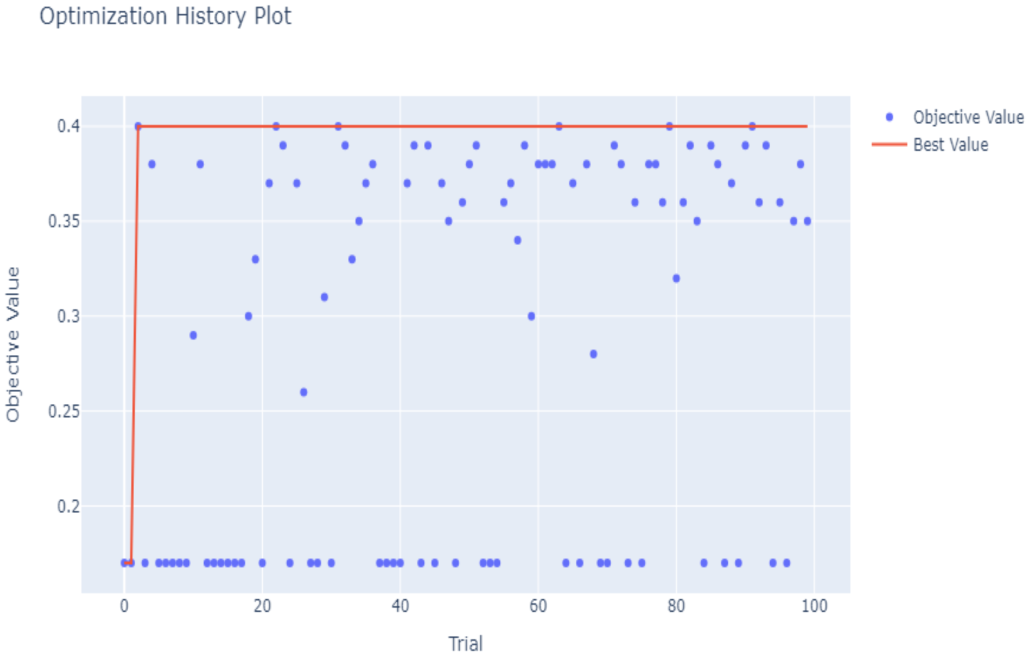


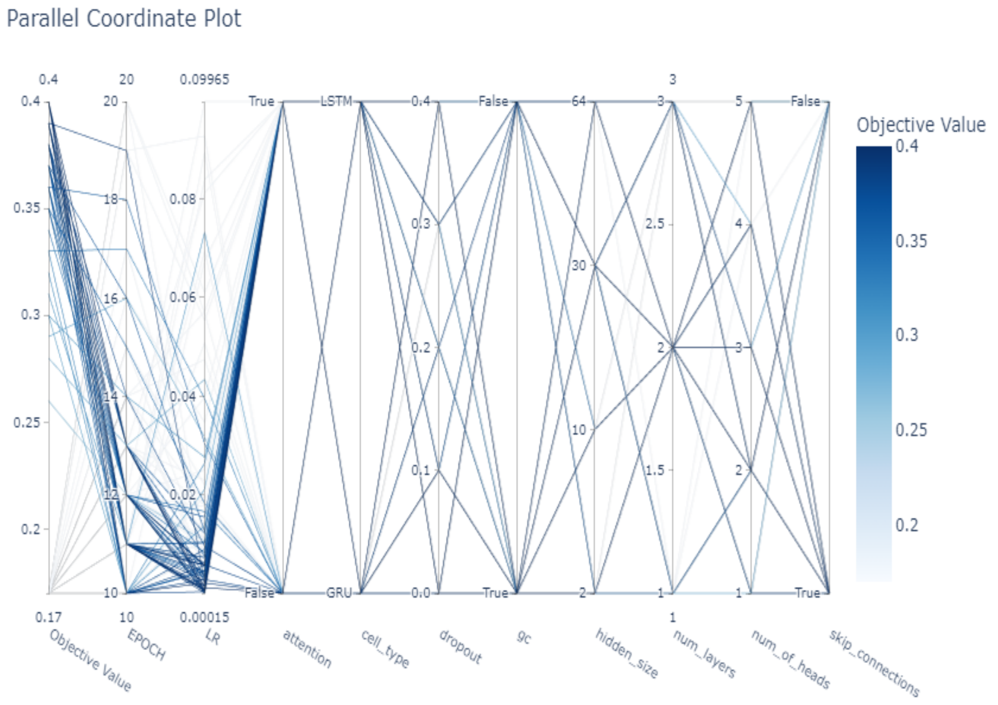
```
Confusion Matrix:
[[960 365 419]
 [765 525 454]
 [726 375 643]]
```

.

3. Results and Overall Analysis

3.1. Results Analysis





3.1.1. Best trial. F1 score 0.4

3.2. Comparison with the first project

Compared to the first one, the paper has changed quite a bit in all parts, both in the way we vectorized the data and in the way we classify it, as now instead of a simple logistic regression we use a bidirectional rnn.

3.3. Comparison with the second project

Compared to the second paper we changed the way we vectorize the data as we said above by adding another dimension, which was necessary to adapt to our canonical model (RNN), compared to the simple NN before.

3.4. Comparison with the third project

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

4. Bibliography

References

[1] Bidirectional rnn.

[2] Pack padding.

[3] Word2vec.

[3] <More about Word2vec> [2] <More about Pack Padding>
[1] <More about bidirectional rnn in pytorch>