UNIVERSITY OF ATHENS
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

Deep Learning for NLP

Student name: *Dimitrios Rammos*
sdi: sdi1900161

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

Contents

1. Abstract

The purpose of our work was the creation of an algorithm that can read a text from twitter and analyze its emotional basis. More specifically, I read two csv files(train and valid), I process this files in 'Text' column and aftrer I call the Logistic Regression Algorithm to create the model. Also, I do pre-process in test file.

2. Data processing and analysis

2.1. Pre-processing

Let's describe the preprocessing. First aff all I have seperated my code in cells. I do pre processing for 3 csv file(train, valid and test).

I read the file and I change the 'Text' column for dataset removing hashtags, at(@) etc.

1. First I delete the hashtags with the word. For example, if my string is "Hello #november" the result will be "Hello", I remove "#november".

2. After, I remove all urls from my strings texts in dataset, for example the "Hello https://www.youtube.com" will beacame "Hello".

3. Subsequently, I remove all numbers as words. For example, the "Hello 123 Jim" will become "Hello Jim".

4. I remove all stopwords, greek stopwords.

5. I remove the accents from the words.

6. Because some word may have been not accent from the beginning, I create a new table with not accents and I remove again the stopwords from dataset.

7. I remove the @words(at-words). For example the string "Hello @jim hello" will beacame "Hello hello".

8. I remove all punctuatuions from dataset.

9. In the end I do tokenizer for my strings. For example, the "hello jim goodmorning" will became ['hello', 'jim', 'became'].

2.2. Analysis

I read the files .csv from input and I create the dataset for train, valid and test.
I do the pre-processing for evrey dataset and I save it. In my code from kaggle, I have functions from Spacy library. I try do pre-process with spacy functions but I noticed that the preprocessing was slowly. In nootebook you can see this functions but I not used.

At first I did lemmatization but I saw that my score instead of increasing towards the positive was worse and slower so I deleted it.

2.3. Data partitioning for train, test and validation

The files I used were from the work compention Project - 1.

2.4. Vectorization

For my project, I used logistic regression on train, validation and test datasets. The text data was converted into numerical vectors using techniques like TF-IDF. These approaches helped the logistic regression model understand the text's meaning through numerical features, facilitating accurate predictions or classifications.

3. Algorithms and Experiments

3.1. Experiments

The score was very close to 0.386. When for example I removed single letters like "s" or anything else the score decreased instead of improving. That's why I haven't implemented any functionality in my pre-process yet.

At first I tried everything with the spacy library as it seemed easier to delete words or symbols but I noticed that it took too long for each process and so I changed the method and tried other ways to do the process.

Trial				Score

Table 1: Trials

3.1.1. Table of trials.

3.2. Hyper-parameter tuning

I tried the hyperparameter tuning via optuna. I basically ran the logistic regression 50 times and it took about 6-8 minutes to run. The result was that the score increased from 0.386 to 0.398 depending on the parameters tol and c.(Figure 1)

```
Best params: {'tol': 0.0009646442741544528, 'C': 0.010001837828563419}
Vali acc: 0.3958333333333333
Vali Class rep:
              precision    recall  f1-score   support

           0       0.39      0.44      0.41      1744
           1       0.39      0.41      0.40      1744
           2       0.42      0.35      0.38      1744

    accuracy                           0.40      5232
   macro avg       0.40      0.40      0.40      5232
weighted avg       0.40      0.40      0.40      5232
```

Figure 1: Hyper-parameter tuning

3.3. Optimization techniques

I try change the preprocess for a better results. For example, lemmatization, emojils, english stopwords, not accents stopwords.

3.4. Evaluation

The dataset did not provide the best result for the diagrams, however I tried to do my best to have a good f1-score and good diagrams which I will present below.

3.4.1. ROC curve. The ROC curve.

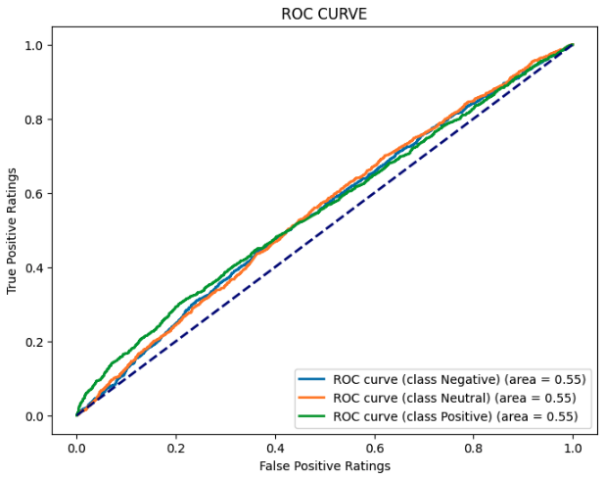


Figure 2: ROC curve

3.4.2. Learning Curve. The learning curve. I see the example from lab for project-1.

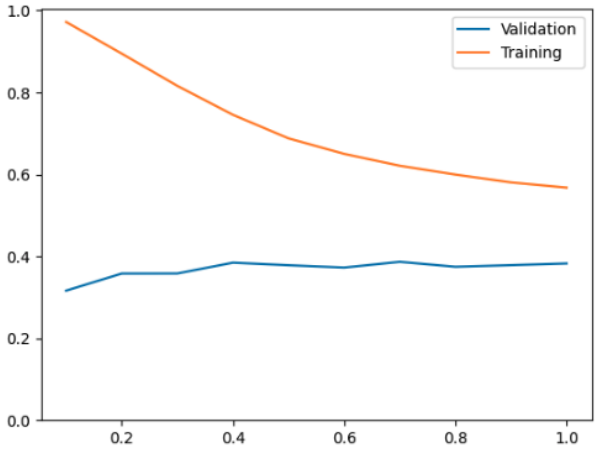


Figure 3: Learning curve

3.4.3. Confusion matrix. The confusion matrix, the middle diagonal shows the correct results.

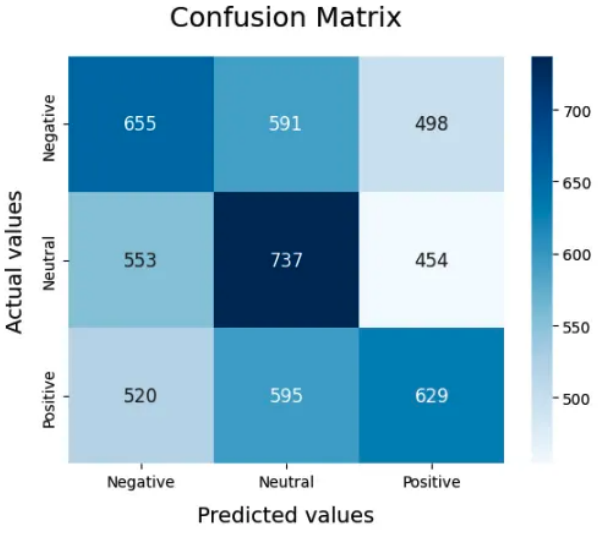


Figure 4: Confusion Matrix

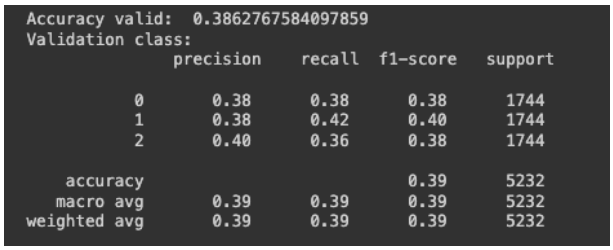
4. Results and Overall Analysis

4.1. Results Analysis

I don't think it's a good performance and I could definitely have a better performance. Maybe it's due to the pre process. I think some corrections could be made there.
However, the dataset may also create some problems and difficulties in the code. Surely other corrections could be made, such as words without meaning in English or Greek. Sticky words from syntactical errors of the users who wrote the texts and we could

separate them. Spelling mistakes that could be fixed. Make sure to do lemmatization without spoiling the score and definitely other corrections that I haven't considered yet.

4.1.1. Best trial. My best trial without hyper-parameter tuning or anything else was:



```
Accuracy valid:  0.3862767584097859
Validation class:
              precision    recall  f1-score   support

           0       0.38      0.38      0.38      1744
           1       0.38      0.42      0.40      1744
           2       0.40      0.36      0.38      1744

    accuracy                           0.39      5232
   macro avg       0.39      0.39      0.39      5232
weighted avg       0.39      0.39      0.39      5232
```

Figure 5: Best trial

I coding for the project in Google Colab and Kaggle.

5. Bibliography

References