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 **2**

2 Data processing and analysis **2**
2.1 Pre-processing . . . 2
2.2 Analysis . . . 3
2.3 Analysis . . . 3
2.4 Data partitioning for train, test and validation . . . 3
2.5 Vectorization . . . 3

3 Algorithms and Experiments **3**
3.1 Experiments . . . 3
3.1.1 Table of trials . . . 4
3.2 Hyper-parameter tuning . . . 4
3.3 Optimization techniques . . . 4
3.4 Evaluation . . . 4
3.4.1 ROC curve . . . 4
3.4.2 Learning Curve . . . 4
3.4.3 Confusion matrix . . . 4

4 Results and Overall Analysis **5**
4.1 Results Analysis . . . 5
4.1.1 Best trial . . . 5
4.2 Comparison with the first project . . . 6
4.3 Comparison with the second project . . . 6

5 Comments **6**

6 Bibliography **6**

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 Optuna take from it the best params and after I call the RNN model with best params. I training my RNN model. Also, I do pre-process in test file and I create submission 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. 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.

2.4. Data partitioning for train, test and validation

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

2.5. Vectorization

For my project, I used FFNN on train, validation and test datasets. I create a new model for it and I use FFNN model algorithm.

3. Algorithms and Experiments

3.1. Experiments

(THIS IS TEST-STEPS FROM PREVIOUS EXERCISE)

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

You can see the code for hyper-parameter tuning in my notebook. The result is: Best params: hiddenSize: 101, numLayers: 2, dropout: 0.16703239798054653

In Solution I use this Parameters from optuna nTrials = 50

3.3. Optimization techniques

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

I check with different best params from optuna, for example i try with nTrials=100, 50, 20, 5 and 10. But I decide use the best params from nTrials=50.

3.4. Evaluation

3.4.1. ROC curve. The ROC curve.

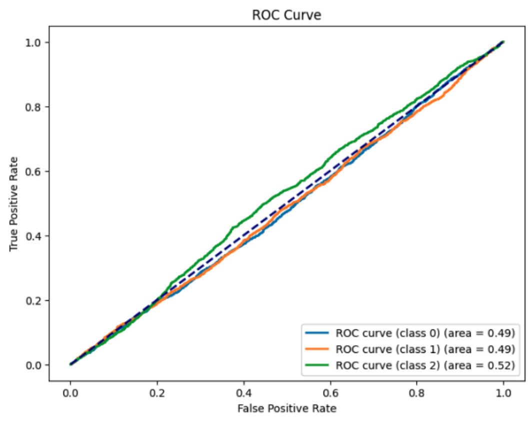


Figure 1: Roc Curve

3.4.2. Learning Curve. The learning curve.

3.4.3. Confusion matrix. The confusion matrix.

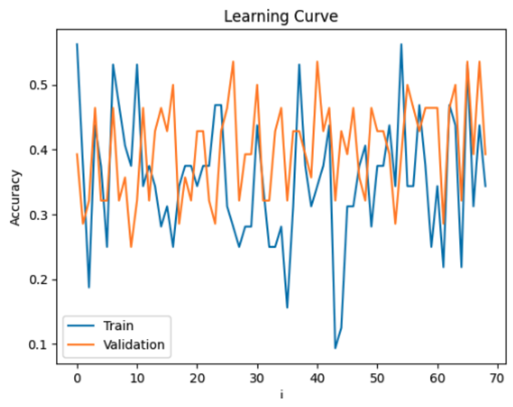


Figure 2: Learning Curve

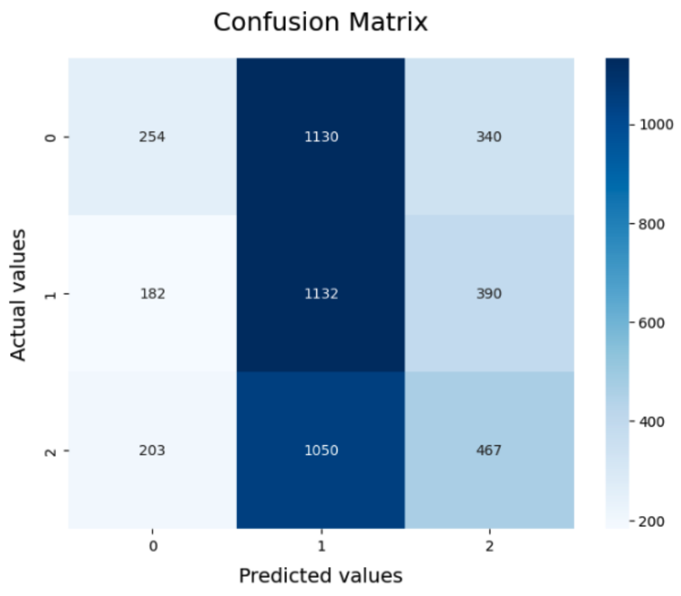


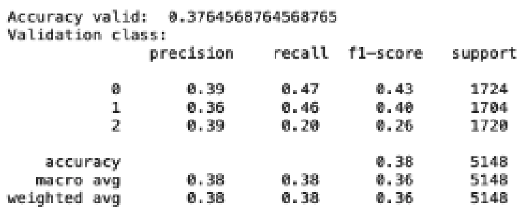
Figure 3: Confusion Matrix

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.



```
Accuracy valid:  0.3764568764568765
Validation class:
              precision    recall  f1-score   support

           0       0.39      0.47      0.43      1724
           1       0.36      0.46      0.40      1704
           2       0.39      0.20      0.26      1720

    accuracy                           0.38      5148
   macro avg       0.38      0.38      0.36      5148
weighted avg       0.38      0.38      0.36      5148
```

Figure 4: Best score

4.2. Comparison with the first project

While Logistic Regression offers simplicity and interpretability, RNNs excel in capturing temporal dependencies. Thus, for tasks requiring nonlinear data understanding or sequential pattern recognition, RNNs typically outperform Logistic Regression.

4.3. Comparison with the second project

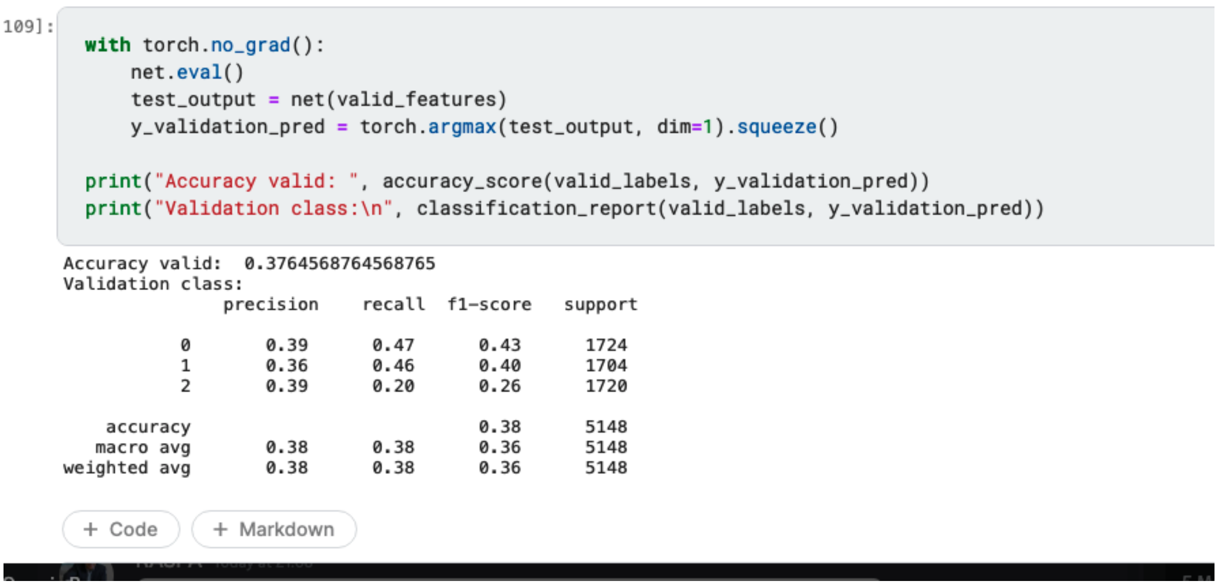
Comparing the results between FFNN and RNN, we find variations in performance based on data type. FFNNs shine in tasks with fixed-length input and intricate nonlinear patterns, while RNNs excel in sequential or time-series data analysis due to their adeptness at capturing temporal dependencies. Choosing between the two hinges on dataset characteristics and problem requirements.

5. Comments

First of all, I apologize for the order of the photos, but I couldn't arrange them in the right order.

My top f1 score was 0.376

6. Bibliography



```
109]:
with torch.no_grad():
    net.eval()
    test_output = net(valid_features)
    y_validation_pred = torch.argmax(test_output, dim=1).squeeze()

print("Accuracy valid: ", accuracy_score(valid_labels, y_validation_pred))
print("Validation class:\n", classification_report(valid_labels, y_validation_pred))
```

```
Accuracy valid:  0.3764568764568765
Validation class:
               precision    recall  f1-score   support

           0       0.39      0.47      0.43      1724
           1       0.36      0.46      0.40      1704
           2       0.39      0.20      0.26      1720

    accuracy                           0.38      5148
   macro avg       0.38      0.38      0.36      5148
weighted avg       0.38      0.38      0.36      5148
```

Figure 5: score