UNIVERSITY OF ATHENS
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

Deep Learning for NLP

Student name: ***Thodoris Minadis***
sdi: ***sdi1900113***

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

Contents

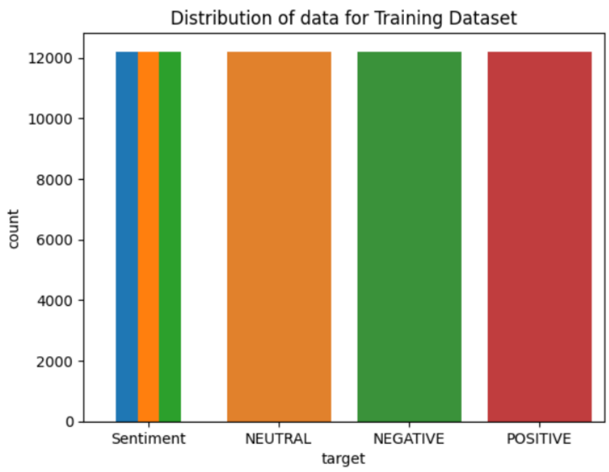
1. Abstract

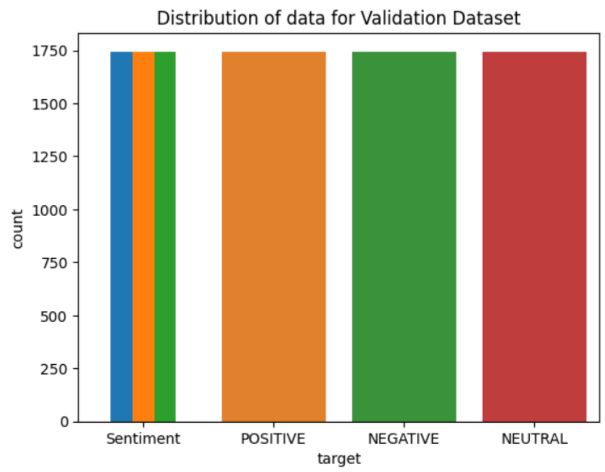
Our exercise is about sentiment analysis on tweeter comments. This is implemented using `bidirectional stacked RNNs with LSTM/GRU cells`. There are three possible sentiments:

- POSITIVE
- NEGATIVE
- NEUTRAL

We have to process some data and create a program that make a prediction about the sentiment of a tweeter text. The prediction must be as better as it cans.

2. Data processing and analysis





2.1. Pre-processing

When reading the CSV file and storing it in the DataFrame (`df`), we start the process of cleaning tweets. This cleaning includes:

- Cleaning all the '#' characters and mentions like `@minaidis` (`cleaning_txt`)
- Cleaning punctuation marks (`cleaning_punctuations`)
- Cleaning repeating characters that are often useless (`cleaning_repeating_char`)
- Cleaning URLs (`cleaning_URLs`)
- Cleaning all numbers from texts (`cleaning_numbers`)
- Cleaning all single characters (`cleaning_single_letter_words`)

After this cleaning I continued with:

- Text tokenization (`word_tokenize`)
- Cleaning stopwords (`using nltk stopwords`)
- Text lemmatization (`WordNetLemmatizer from nltk`)

2.2. Analysis

In general, I think the biggest "cleaning" is the stopwords that are many. After these, punctuation words disappearing also is a good way to improve the program. They were also some URLs , not only with the classic https://example.com form. Another useless words are the single letter words that created after the cleaning, the numbers (on 1-9 form or one-nine for example). In the end, something that improved my code was the separation of words that was stacked because a dot was missing.

2.3. Data partitioning for train, test and validation

The partitioning is already implemented.

2.4. Vectorization

Vectorization in my code is done using `TfidfVectorizer` from `sklearn.feature_extraction.text` . I use it for the training text part (`X_train`)

3. Algorithms and Experiments

3.1. Experiments

I started creating new train and valid datasets,with only the important columns that are the text,ids and the sentiment. In text I started cleaning but I realised that in cleaning I had to be careful because cleaning must become with the right order.

3.1.1. Table of trials

```
Training F1-Score: 0.3668
Training Classification Report:
              precision    recall  f1-score   support

           0       0.37      0.50      0.43     12177
           1       0.37      0.28      0.32     12162
           2       0.38      0.33      0.35     12168

    accuracy                           0.37     36507
   macro avg       0.37      0.37      0.37     36507
weighted avg       0.37      0.37      0.37     36507

Training Confusion Matrix:
[[6149 2870 3158]
 [5313 3422 3427]
 [5192 2945 4031]]
Validation F1-Score: 0.3283
Validation Classification Report:
              precision    recall  f1-score   support

           0       0.40      0.41      0.41      1731
           1       0.47      0.06      0.10      1728
           2       0.37      0.68      0.48      1733

    accuracy                           0.38      5192
   macro avg       0.41      0.38      0.33      5192
weighted avg       0.41      0.38      0.33      5192

Validation Confusion Matrix:
[[ 711   53  967]
 [ 560   96 1072]
 [ 493   55 1185]]
```

.

3.2. Hyper-parameter tuning

On general, i used Optuna to find the best hyper-parameters about my RNN function(`hidden_dim`,`layers`,`dropout`) .

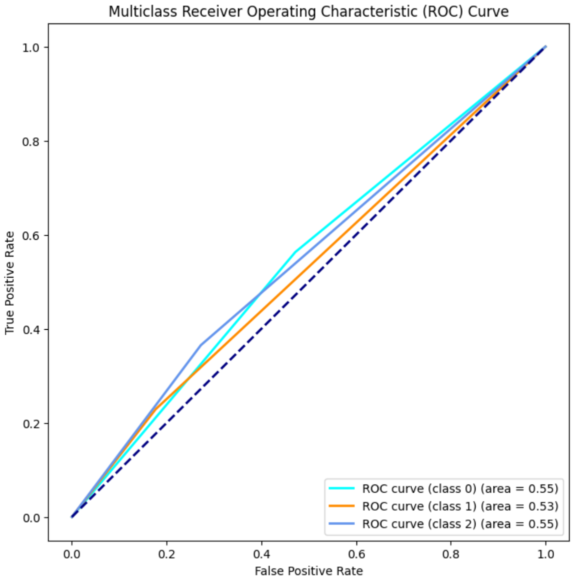
3.3. Optimization techniques

Starting from the cleaning-tokenization-lemmatization, I used spaCy but i saw that it was too slow. So the cleaning is done by using re.sub , or str.
Also, for tokenization and lemmatization i used `word_tokenize(text)` and `WordNetLemmatizer()`.

3.4. Evaluation

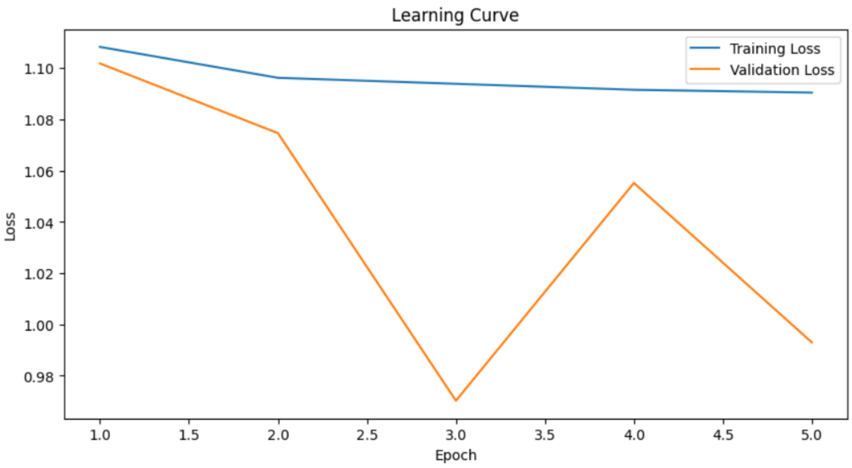
```
Epoch [1/5], Loss: 1.0759, Accuracy: 35.42%
Validation Loss: 1.0931, Validation Accuracy: 37.65%
Epoch [2/5], Loss: 1.0759, Accuracy: 36.31%
Validation Loss: 1.0889, Validation Accuracy: 37.44%
Epoch [3/5], Loss: 1.0759, Accuracy: 36.90%
Validation Loss: 1.0895, Validation Accuracy: 37.44%
Epoch [4/5], Loss: 1.0759, Accuracy: 37.52%
Validation Loss: 1.0863, Validation Accuracy: 38.54%
Epoch [5/5], Loss: 1.0759, Accuracy: 37.26%
Validation Loss: 1.0873, Validation Accuracy: 38.37%
```

3.4.1. ROC curve



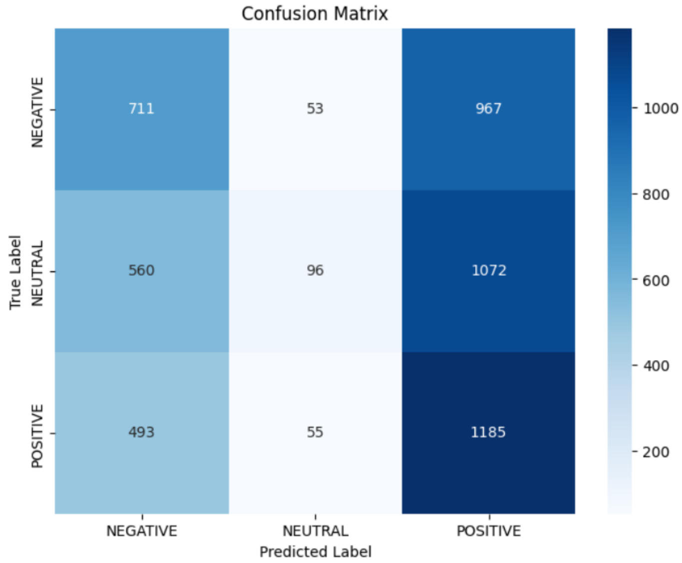
•

3.4.2. Learning Curve



•

3.4.3. Confusion matrix



.

4. Results and Overall Analysis

4.1. Results Analysis

I think it was a good try because I have a good time and an good result. Because of my work, the most difficult part is to push myself to learn and stay more hours on laptop, after 8-hours in the office on my PC. Of course, without my job , i would have more time to spend on this and I would have implemented something better, but that's life, we cannot have everything ☻.

4.2. Comparison with the first project

The first exercise was a lot easier . The only difficulty was the adjustment to kaggle, Latex again and on this type of AI.

4.3. Comparison with the second project

I think the difficulty was the same , maybe third was a little easier. Having already implemented SentimentClassifier i just made a change on this and after an edit on the training / validation / test part. The neural network was a huge part, difficult to understand.

5. Bibliography

References

References

[1] Scikit-learn.org. URL: https://scikit-learn.org

[2] Problem solver: Stackoverflow . URL: https://stackoverflow.com

[3] YouTube channel: Computer Science . URL: https://www.youtube.com/watch?v=ujId4ipkBio

[4] analyticsvidhya.com . URL: https://www.analyticsvidhya.com/blog/2021/06/twitter-sentiment-analysis-a-nlp-use-case-for-beginners/