UNIVERSITY OF ATHENS
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

Deep Learning for NLP

Student name: ***Theodoros Minadis***
sdi: ***sdi1900113***

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

Contents

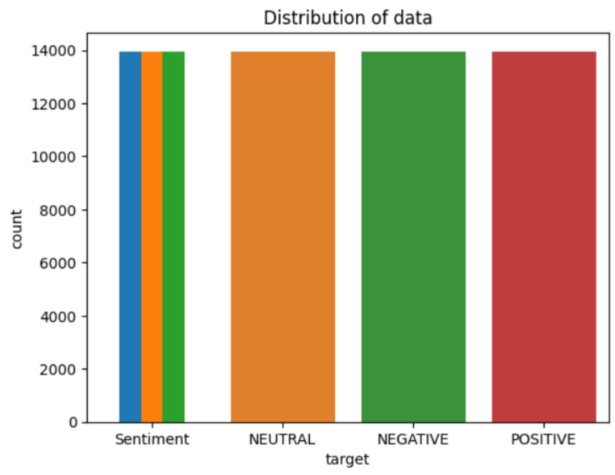
1. Abstract

Our exercise is about sentiment analysis on tweeter comments. This is implemented using `LogisticRegression` from `scikit-learn`. 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

As we saw in lesson, a good split is usually 80-20. In this exercise I also tried a 95-5 split but the CPU time was worst and the difference in the result was small, only 0.01. I found out that the best split for me was the 85-15 because of the combination of time and accuracy.

2.4. Vectorization

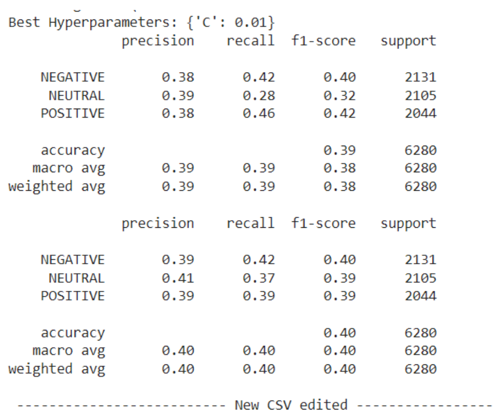
Vectorization in my code is done using `TfidfVectorizer` from `sklearn.feature_extraction.text` . I use it for the training text part (`X_train`)

3. Algorithms and Experiments

3.1. Experiments

I started creating a new CSV file from train and valid ,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. After all, the GridSearch gave me the best C for LogisticRegression and I used it.
One of my problems that I still have and as I saw, there is no any solution, is the time compilation for the learning Curve diagram. It took me a lot of time but I created the diagram and didn't run it again.

3.1.1. Table of trials



```
Best Hyperparameters: {'C': 0.01}
              precision    recall  f1-score   support

    NEGATIVE       0.38      0.42      0.40      2131
     NEUTRAL       0.39      0.28      0.32      2105
    POSITIVE       0.38      0.46      0.42      2044

    accuracy                           0.39      6280
   macro avg       0.39      0.39      0.38      6280
weighted avg       0.39      0.39      0.38      6280

              precision    recall  f1-score   support

    NEGATIVE       0.39      0.42      0.40      2131
     NEUTRAL       0.41      0.37      0.39      2105
    POSITIVE       0.39      0.39      0.39      2044

    accuracy                           0.40      6280
   macro avg       0.40      0.40      0.40      6280
weighted avg       0.40      0.40      0.40      6280

 ------------------------- New CSV edited ----------------
```

.

3.2. Hyper-parameter tuning

In this point, I used GridSearchCV that gave me that the best C and after i was running this with this

3.3. Optimization techniques

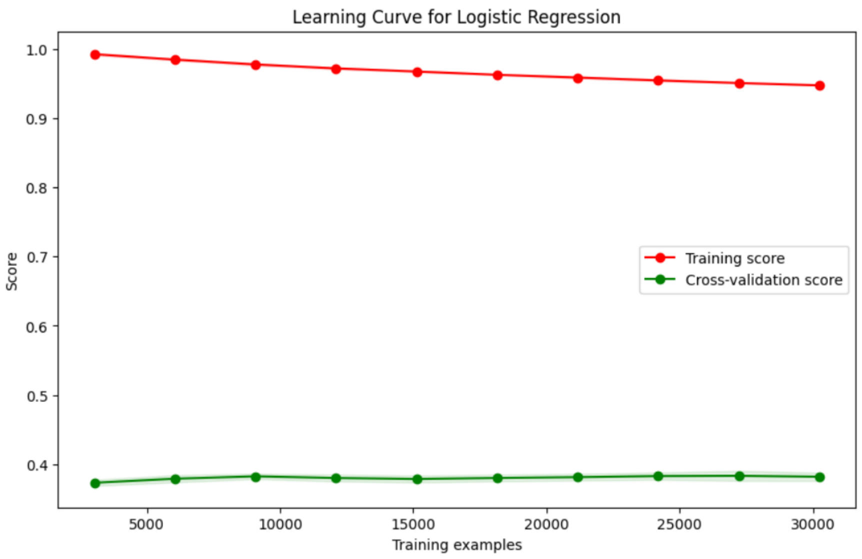
Starting from the cleaning-tokenization-lemmatization, I used spaCy but i saw that it was too slow. So the cleaning is done by using re.sub , or str.
Also, for tokenization and lemmatization i used `word_tokenize(text)` and `WordNetLemmatizer()`.
Now, for split to train and test part, vectorization i totally used nltk , like for the LogisticRegression that we used.
Now, for the diagrams , it was one way road.

3.4. Evaluation

<Provide and comment diagrams and curves>

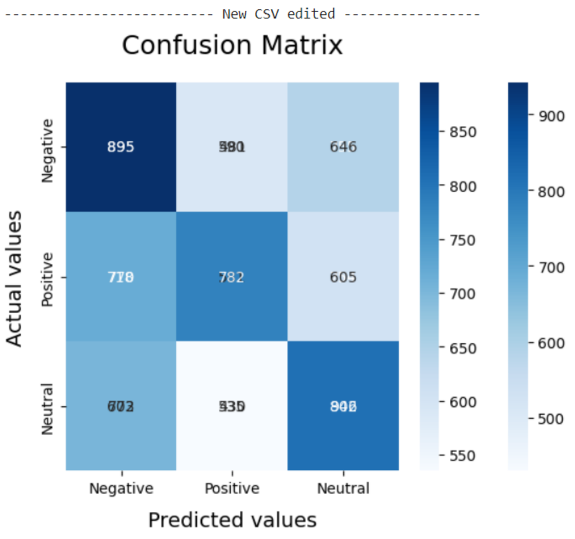
3.4.1. ROC curve.

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 ☻.

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/