University of Athens
Department of Informatics and Telecommunications

Deep Learning for NLP

Student name: *Antigoni Stefanou*
sdi: 2000281

Course: *Artificial Intelligence II (M138, M226, M262, M325)*
Semester: *Spring Semester 2025*

Contents

1. Abstract

In this assignment, the task is to categorize tweets, depending on whether the sentiment behind them is positive or negative. For this purpose, methods such as TF-IDF vectorization and logistic regression will be used.

2. Data processing and analysis

2.1. Analysis

The first step is understanding the data. The dataframe is already split into Training, Validation and Testing datasets, each consisting of an ID and a Text column, that contains the body of the tweet. The Training and Validation datasets also have a Label column, containing the value 0, if the tweet's sentiment is negative, or 1 if it's positive. Below is a histogram of the labels' distribution.

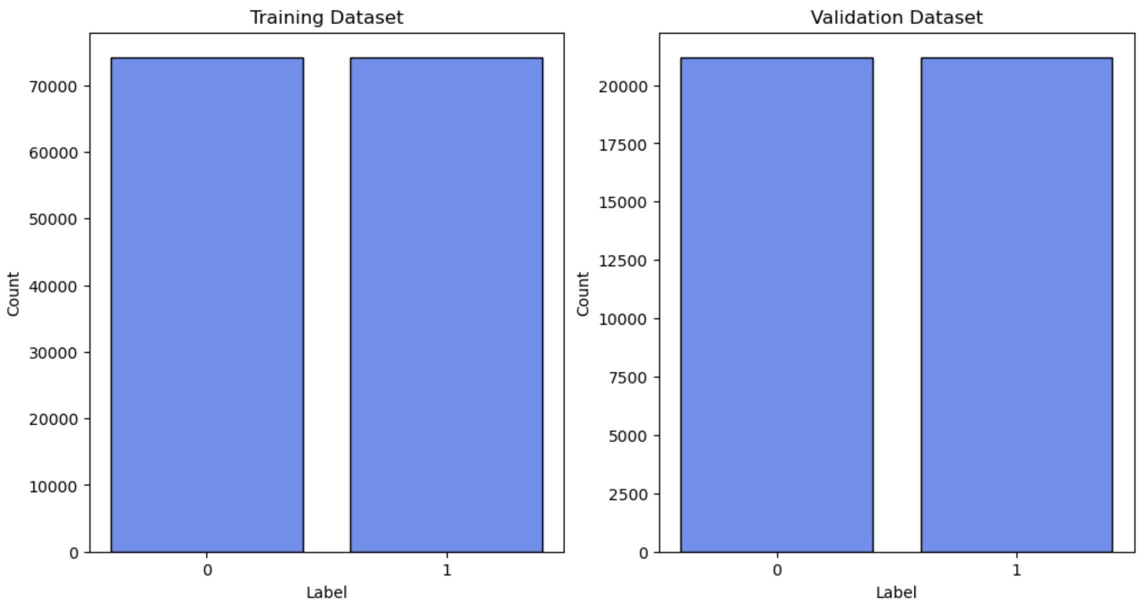

Figure 1: Label histograms

It is apparent that the data are balanced. This will be beneficial in training the model, as it will not be skewed towards one class.

Part of understanding the data is studying the content of the tweets on which the model will be trained. The following are some observations on this content.

— Training Dataset —

The average number of words in a sentence is: 13.29

The number of unique words is: 148388

— Validation Dataset —

The average number of words in a sentence is: 13.31

The number of unique words is: 42396

— Testing Dataset —

The average number of words in a sentence is: 13.36

The number of unique words is: 21199

A sentence is, on average, only 13 words long, which is expected given Twitter's 280-character limit. Upon printing a sample of the training dataset, it is apparent that this limit encourages users to use more abbreviations. Furthermore, the non-serious nature of Twitter explains the wealth of slang terms, typos, and general noise.

```
@whoisralphie dude  I'm so bummed ur leaving!

oh my god, a severed foot was foun in a wheely bin in cobham!!! where they found is literally minutes from my
house! feel sick now!

I end up &quot;dog dialing&quot; sumtimes. What's dog dialing, u ask?My dogs will walk across my phone &amp;
 end up calling someone. aka &quot;dog dialing&quot;!

@_rachelx meeeee toooooo!

I was hoping I could stay home and work today, but looks like I have to make another trip into town

says plurk karma finally reached the 50s. still no heartsy smileys. boo.  http://plurk.com/p/z2x3y

Good to hear it @Arth This is a bit more, but a la four tet  Do you know Free Rotation festival?
 Am thinking ... Ã¢â¢Â« http://blip.fm/~7hcvo

@davorg in that case im gonna start tweeting about nymphomanic pub owners who like to cook, well worth a
shot, eh

@belunyc its alright love, how are you?

@brightondoll haha that has to be the best analogy ever. mogwai to gremlin. love it. i love gizmo and the
gremlins movies
```

Figure 2: Text sample from the training dataset

These issues will be fixed in pre-processing.

2.2. Pre-processing

One of the most important tasks in this assignment is the pre-processing, the goal of which is to remove the noise often found in tweets, in order to improve the model's performance. The following methods were used to clean the datasets.

1. Replace all emojis with their CLDR short name, aka a word or small phrase that describes them. This is done with the help of the demojize function found in the emoji module.

2. Convert all letters to lowercase. In this way, there will be no distinction between words such as "Hello" and "hello".

3. Remove all HTML character entities. These entities start with an ampersand and end with a semicolon (ex. <), so they can be found using RegEx.

4. Find all links starting with "http(s)://" or "www." and either remove them completely or replace them with the word "link".

5. Find all emails and either remove them completely or replace them with the word "email". This is also important for privacy reasons.

6. Find all Twitter handles and remove them completely or replace them with the word "name". This is also important for privacy reasons.

7. Remove all punctuation marks and special characters.

8. Remove all digits.
9. Remove non-ASCII characters.
10. Search for all instances where a letter appears more than twice in a row and replace it with just two characters. This way, words such as "good" and "goooooood" will be recognized as the same word.
11. Remove all stop words with the help of the NLTK library. This practice is actually shown to decrease a model's accuracy when dealing with Twitter datasets. [3] However, it was implemented on the basis of trial and error experimentation.
12. Lemmatize the text using NLTK's WordNetLemmatizer
13. Use NLTK's SnowballStemmer to apply stemming to the text.

While all of these methods are implemented, not all of them are used in the final model, as some have an adverse effect in its accuracy.

Below are the word clouds generated from the Training dataset, before and the after the pre-processing.

Figure 3: Before pre-processing

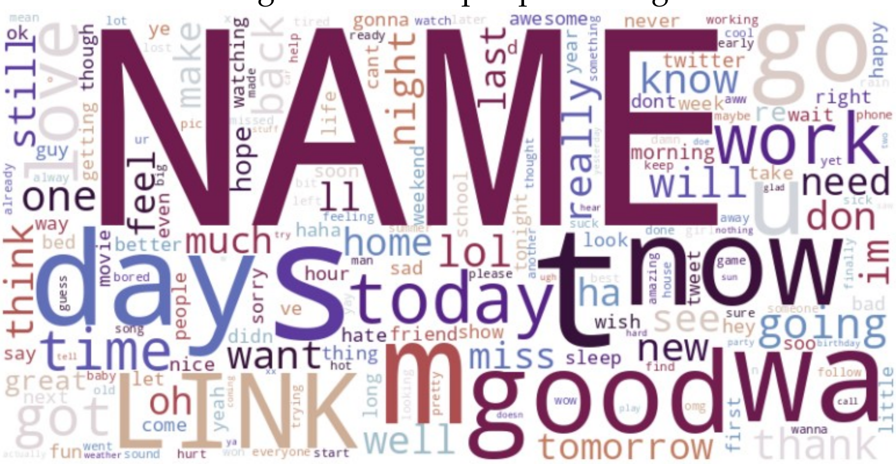

Figure 4: After pre-processing

3. Algorithms and Experiments

3.1. Algorithms

The model's training is a two-step process. The first step is feature extraction, which will be accomplished with the TF-IDF method. SK-learn's TfidfVectorizer will be used to convert the data into a matrix of TF-IDF features. The next step is classifying the data. Since this is a binary classification problem, a logistic regression classifier will be used; specifically SK-learn's LogisticRegression classifier.

3.2. Experiments

To effectively tackle this problem, a baseline is needed. Therefore, the first step is running the vectorizer and the classifier without changing the default parameters, before the pre-processing step. Doing so yields an accuracy score of 0.79 and the following learning curve:

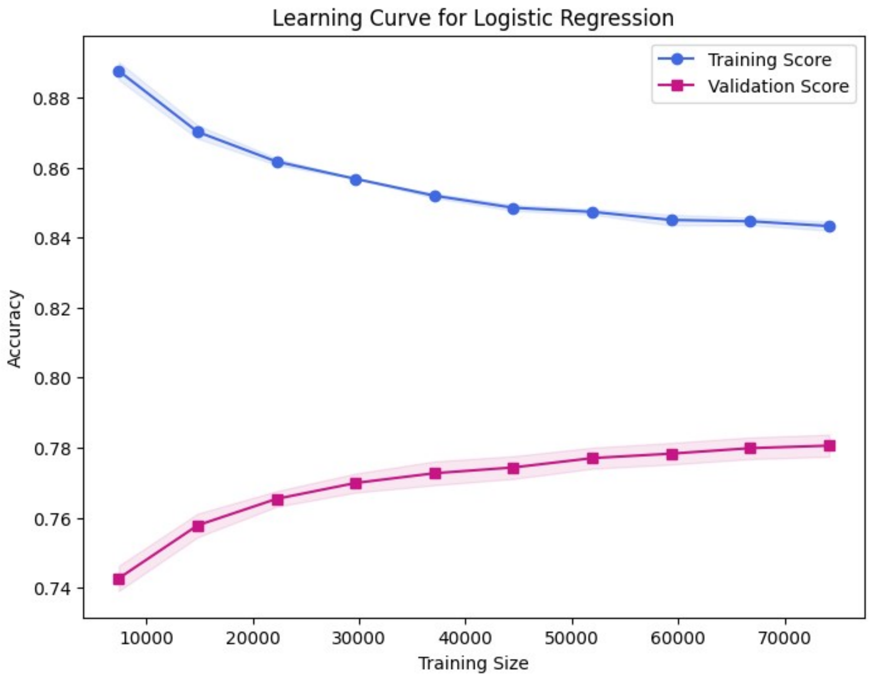

Figure 5: Base case learning curve

These scores are already quite high, meaning that the model does not randomly choose between the two labels. This is partially due to the data being balanced. However, it is also safe to assume that the default parameters are a good fit.

One way to improve the model is to remove the noise from the data with the pre-processing, in order to make the features in the TF-IDF matrix more valuable. After applying all the methods of data cleaning mentioned in section 2.2, the accuracy drastically drops to 0.76 and the confusion matrix is the following:

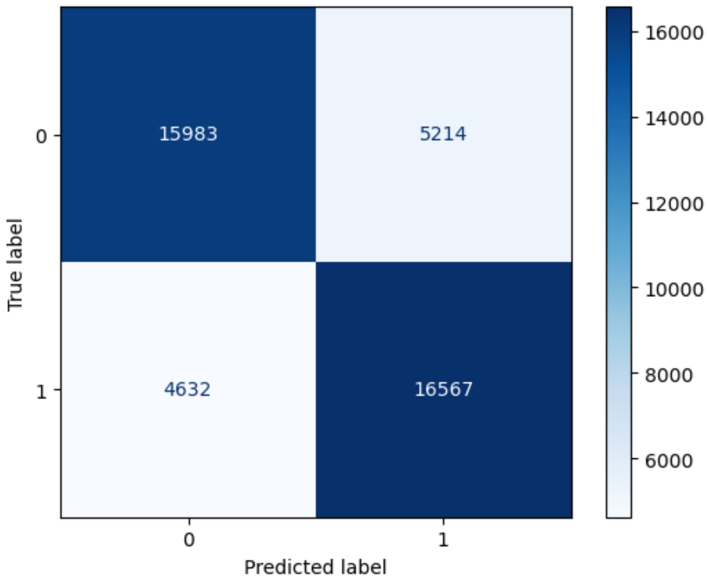

Figure 6: Confusion matrix

It is clear that not all of the methods were beneficial to the model. This is likely due to the nature of the data. For example, because each tweet only contains around 13 words (see 2.1), the model is sensitive to changes such as removing stop words, which removes a large percentage of the sentence, or stemming, which can make words appear more similar than they actually are. After experimenting with different combinations of these methods, the following seem to yield the best results:

1. Replace all emojis with their CLDR short name.
2. Convert all letters to lowercase.
3. Remove all HTML character entities.
4. Replace all links with the word "link".
5. Replace all emails with the word "email".
6. Replace all Twitter handles with the word "name".
7. Remove all punctuation marks and special characters.
8. Remove non-ASCII characters.
9. Replace all multiple instances of a character with only two characters.
10. Apply lemmatization.

The accuracy is now 0.78, which is higher than before, but still lower than in the base case.

After the pre-processing is finished, the next step is configuring the model. This is achieved using grid search, and more specifically SK-learn's GridSearchCV. By setting a parameter grid, it becomes possible to check many parameter values and pinpoint those that improve the model's accuracy the most.

Regardless of whether the pre-processing methods are applied to the data or not, the final accuracy score is 0.80. Therefore, the data were cleaned, as that is the better practice.

3.2.1. Hyper-parameter tuning. The hyper-parameters will be found with the grid search, so it is important to focus on the parameters that will affect the model the most.

In terms of the TF-IDF vectorizer, those are:

- the n-gram range (ngram_range). This is the lower and upper boundary of the range of n-values. For example, (1, 2) means unigrams and bigrams.[2] The values checked here are [(1, 1), (1, 2), (2,2)].
- the maximum document frequency (max_df). This is an upper threshold. When building the vocabulary, terms that have a document frequency strictly higher than the given threshold will be ignored.[2] The values checked here are [0.7, 0.8, 0.9].
- the minimum document frequency (min_df). This is a lower threshold. When building the vocabulary, terms that have a document frequency strictly lower than the given threshold will be ignored.[2] The values checked here are [0.0, 0.00001, 0.001].

For the logistic regression classifier, the focus is on:

- the maximum number of iterations (max_iter). The values checked here are [100, 150, 200, 250].
- the inverse of regularization strength (C), which acts as a penalty for the model, when the magnitude of parameter values is increased. The values checked here are [0.5, 0.7, 1.0, 1.3, 1.5, 1.7, 2.0, 2.5].
- the solver (solver), which is the algorithm used for optimization.[1] The values checked here are ['lbfgs', 'liblinear', 'saga'].

3.2.2. Optimization techniques. The model's optimization was achieved by testing different solvers for the logistic regression classifier, specifically lbfgs, which is the default option, liblinear and saga. Liblinear handles by default binary classification problems and performs better on smaller datasets. Saga on the other hand, is faster on larger, sparse datasets. Lbfgs generally performs the best in comparison to the other methods and it saves a lot of memory.[1] This was the case for this dataset as well, given that it produced the best accuracy score.

3.3. Evaluation

The following metrics will be used to evaluate the models.

3.3.1. *Accuracy.*

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Accuracy is a metric that measures how often a model correctly predicts the outcome, in this case the tweet's sentiment. A high accuracy score means that the model can correctly categorize the data. However, it is not enough on it's own, as a very high accuracy is also a byproduct of overfitting.

3.3.2. *Learning Curve.* A learning curve is just a plot showing the progress over the experience of a specific metric related to learning during the training of a machine learning model The learning curve used is a plot showing the Training and Validation set's accuracy scores in relation to the training size. If the lines diverge, the model is overfitting.

3.3.3. *Confusion matrix.* The confusion matrix is a 2x2 table with the number of correct predictions (true positives and true negatives) and incorrect predictions (false positives and false negatives) for both classes. It is a very important metric, since it clearly presents the model's weak points.

4. Results and Overall Analysis

4.1. Results Analysis

In the end, the best model can correctly predict a tweet's sentiment with around 80% accuracy, which is a relatively high score. According to the learning curve (Figure 8), the model is not overfitted either- the training score slowly decreases while the validation score increases and the lines slowly converge. Therefore, the performance seen here could be generalized to other similar Twitter datasets as well. The model could potentially be further improved with adjustments to the pre-processing step, such as utilizing a spell checker. In addition, it could benefit from testing more variables in the grid search, which would be, however, very time consuming.

4.1.1. *Best trial.* The best model was achieved with the following parameters:

TF-IDF Vectorizer	
ngram range	(1, 2)
max df	0.7
min df	0.0
Logistic Regression	
max iter	100
C	2.5
solver	lbfgs

The model achieved the following scores on the validation dataset:

Accuracy	0.8062
Precision	0.8095
Recall	0.8008
F1 Score	0.8051

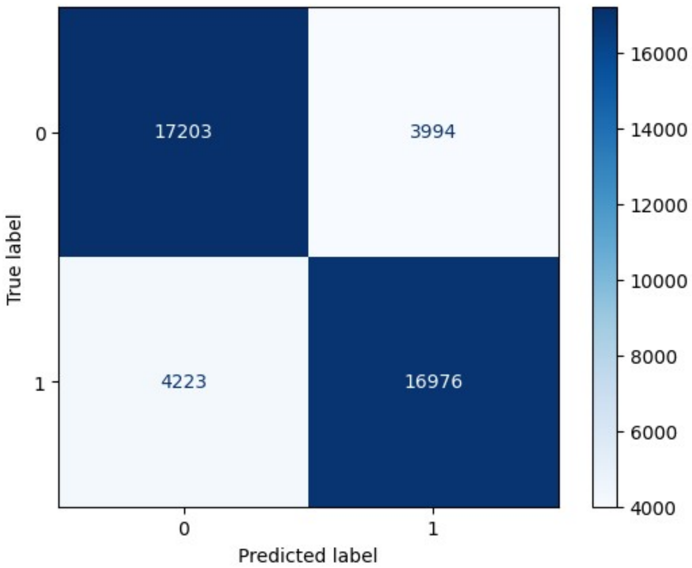

Figure 7: Final Model Confusion Matrix

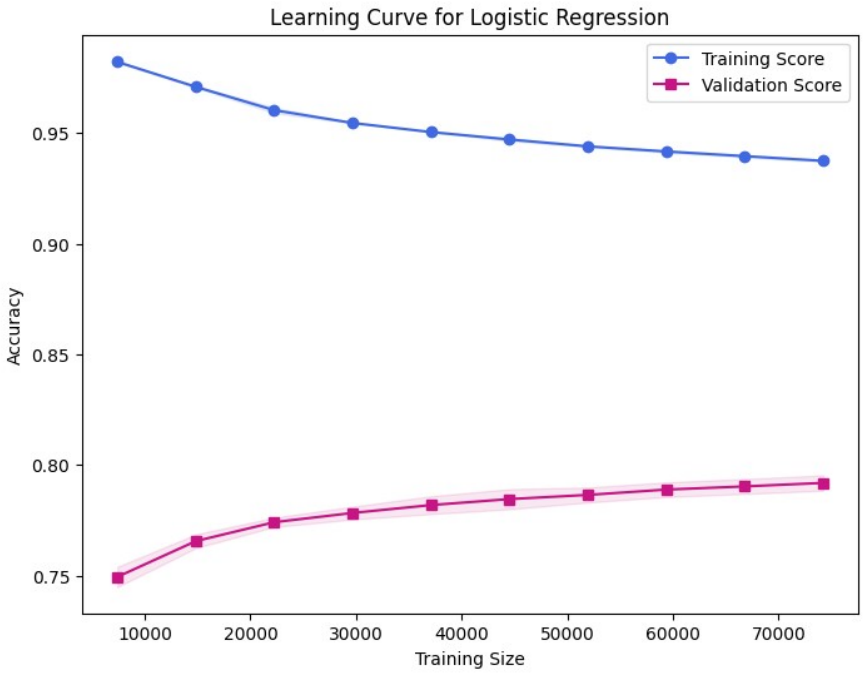

Figure 8: Final Model Learning Curve

The Test dataset's accuracy score is 0.80409.

5. Bibliography

References

[1] Scikit learn: Machine Learning in Python. Logistic Regression, 2023. Accessed: 2025-03-28.

[2] Scikit learn: Machine Learning in Python. TfidfVectorizer, 2023. Accessed: 2025-03-28.

[3] Hassan Saif, Miriam Fernandez, Yulan He, and Harith Alani. On stopwords, filtering and data sparsity for sentiment analysis of Twitter. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 810–817, Reykjavik, Iceland, May 2014. European Language Resources Association (ELRA).