UNIVERSITY OF ATHENS
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

Deep Learning for NLP

Student name: *Aikaterini-Methodia Zacharioudaki*

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

Contents

1. Abstract

The task of this assignment is to develop a sentiment classifier for a tweet dataset. The methods we will use are the TF-IDF Vectorizer for feature extraction and the Logistic Regression Classifier for classification.

2. Data processing and analysis

2.1. Pre-processing

We follow the steps below for pre-processing:

- Demojize the tweets and replace emoticons with their descriptive words
- Make lowercase the entire text
- Remove tags, links, HTML tags and emails
- Replace timestamps with the word time
- Remove multiple consecutive letters, leave only two
- Replace some common misspelt words with the correct spelling
- Replace common slang with the correct word
- Remove all non-word and non-whitespace characters
- Replace all excess whitespace characters with a single space
- Lemmatize all words

We also tested the following steps, but they lowered the accuracy, so they were not implemented in the final model:

- Remove stopwords
- Remove contractions
- Remove numbers

2.2. Analysis

In the countplots in Figure 1, we see that the labels are equally distributed. That means the dataset is balanced and the model will not be biased to one label.

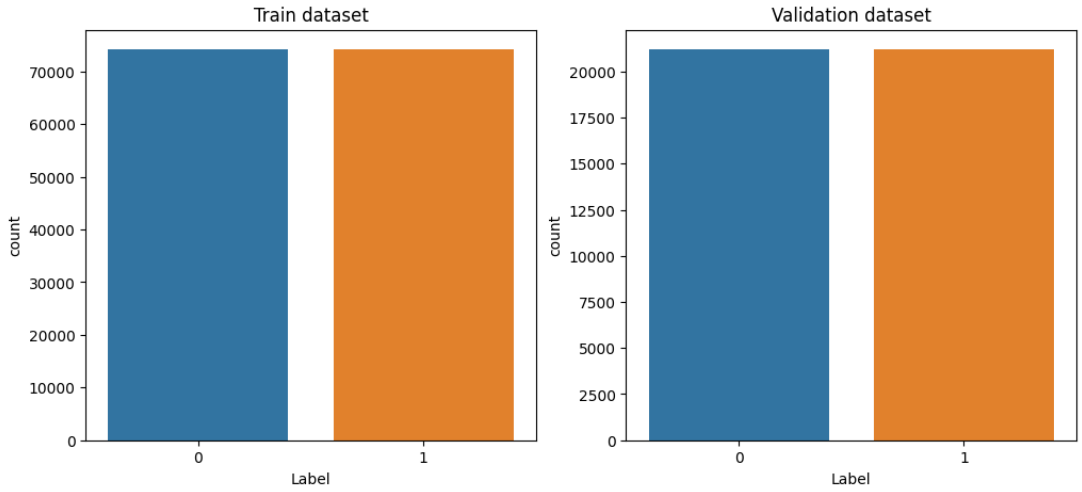

Figure 1: Label countplot

Before Pre-processing :

- Train Dataset
  The unique words in this set are 234492, with an average word length of 9.03 characters. The longest word, in true tweet fashion, is 'CheeseRavioli+Zitty+Dirk +PaperAirplanes+SportsCoats+Sex+PushPlayGroupieWannabes+GirlShoes +MyspaceWhores+MakingFunOfBands+Music=GOODTIMES!' and it is 137 characters long.

- Validation Dataset
  The unique words in this set are 90519, with an average word length of 8.36 characters. The longest word is '?? ??????????????????????????????..' and it is 84 characters long.

- Test Dataset
  The unique words in this set are 53562, with an average word length of 8.07 characters. The longest word is 'bbbbbbbbooooooooooooooooooooooooooooor rrrrrrrrrrrrrrrrrrrrrrrrrreeeeeeeeeeeeeeeeddddddddddddddddddddddd' and it is 111 characters long.

- In the complete dataset, meaning the 3 datasets together, there are 306856 unique words.

Due to the nature of Twitter as social media and its character limit, we see that many users will join words together without whitespaces and forego proper spelling/grammar. This could intervene with our model's training and accuracy, so we implement steps in the pre-processing to clean the data.

After Pre-processing :

- Train Dataset
  The unique words in this set are 71224, with an average word length of 6.63 characters. The longest word is 'youthinkimkiddingimfuckingseriousiam havingmeatloafwithdrawalsandamthisclosetorobbingameatloafrestaurant' and it is 103 characters long.
- Validation Dataset
  The unique words in this set are 33253, with an average word length of 6.42 characters. The longest word is 'whyareallthegoodmedscontrolledsubstances' and it is 40 characters long.
- Test Dataset
  The unique words in this set are 21858, with an average word length of 6.28 characters. The longest word is 'unfollowanyonewhoisfollowingoprahcampaign2009' and it is 45 characters long.
- In the complete dataset, meaning the 3 datasets together, there are 88998 unique words.

We see that the unique word count dropped to a third of the original after the pre-processing.

Below are the wordclouds of the 3 datasets. Figure 2 is the wordcloud of the train datset before pre-processing, while Figures 3, 4 and 5 are the wordclouds of the train, validation and test dataset, respectively, after pre-processing.

Figure 2: Word cloud of the Train dataset before Pre-processing

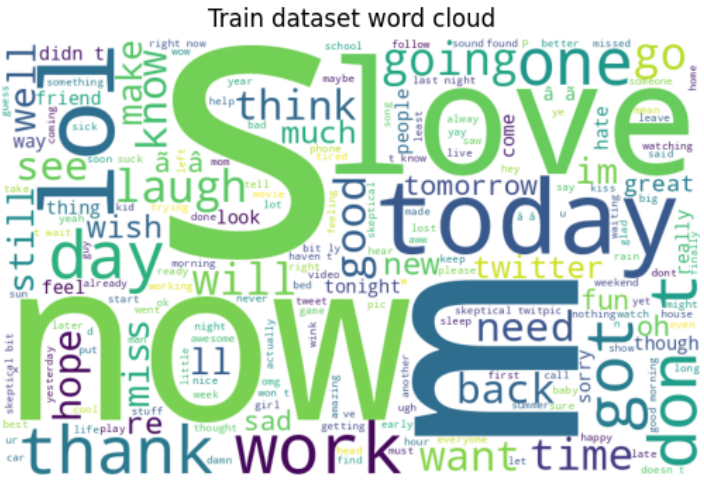

Figure 3: Word cloud of the Train dataset after Pre-processing

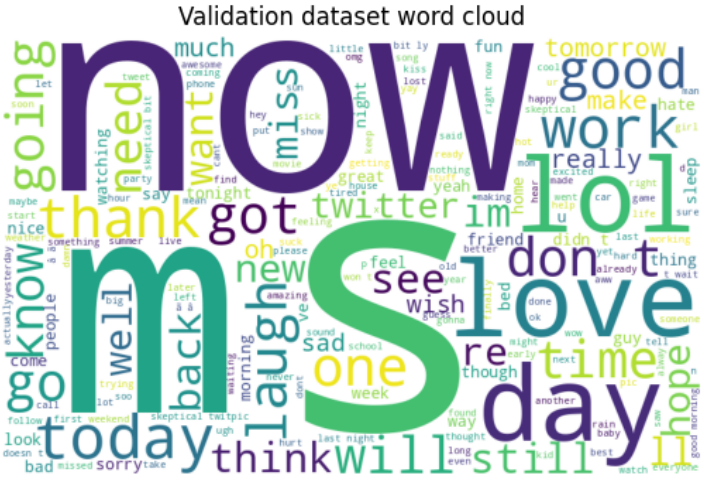

Figure 4: Word cloud of the Validation dataset after Pre-processing

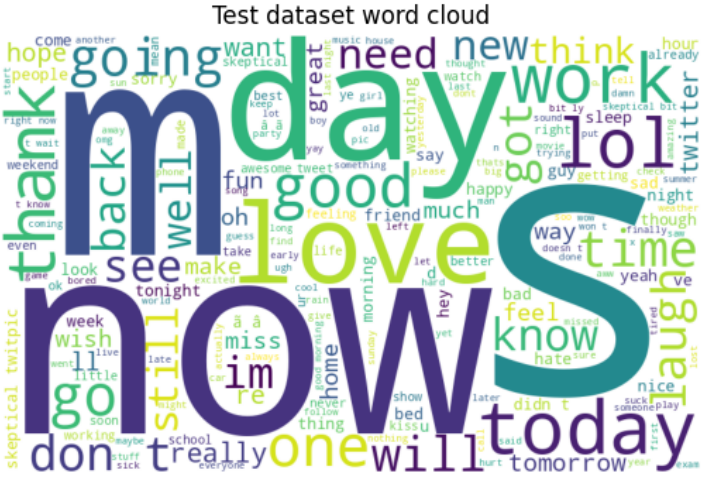

Figure 5: Word cloud of the Test dataset after Pre-processing

3. Algorithms and Experiments

3.1. Baseline Model

Before the pre-processing step, we train a simple model, using the default parameter of the vectorizer and the classifier. We achieve an accuracy of 0.7909, which will be our base.
Below is the learning curve of the model.

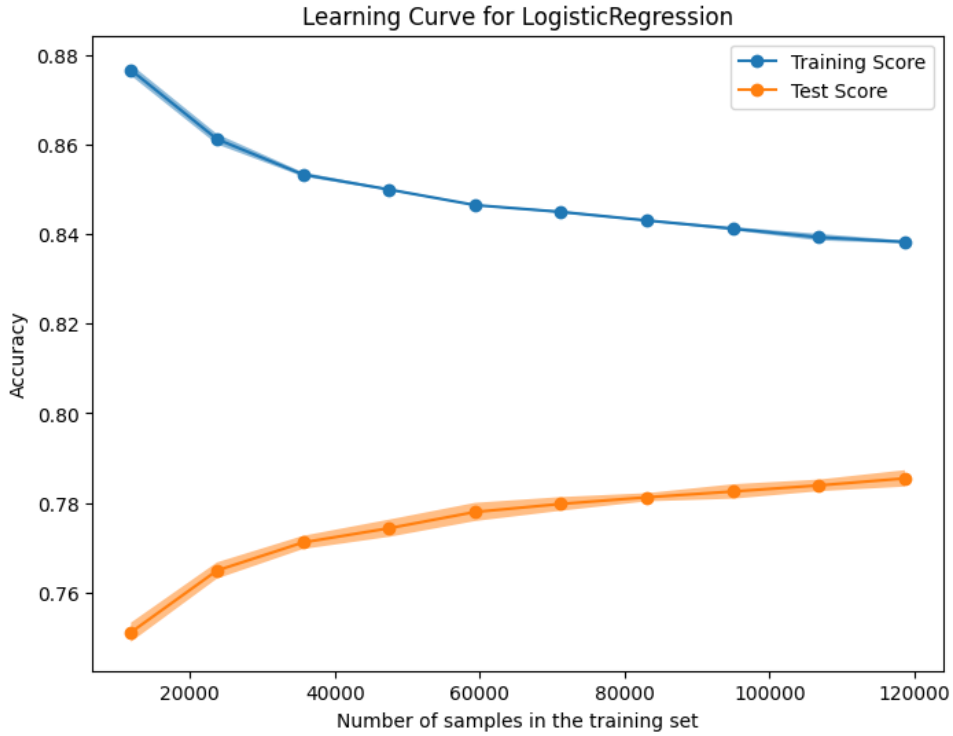

Figure 6: Learning Curve for Baseline model

3.2. Experiments

After finding the baseline and applying the pre-processing steps as mentioned in Section 2.1, we need to tune the model's parameters to improve the accuracy. This was achieved by tuning the vectorizer manually and the classifier with the help of GridSearch.

3.3. Hyper-parameter tuning and Optimization techniques

For the vectorizer, we tested the following parameters:

- **tokenizer** : We used the TweetTokenizer from the package nltk.tokenize
- **ngram_range** : We tried the following ranges: { (1,1), (1,2), (1,3), (2,2) }, and we found that the best one is (1,2)

- **max_df , min_df , max_features** : We experimented with some different values but the default ones were the best for accuracy

For the classifier, we tested the following parameters using GridSearch:

- **C** : We tried a range of floats between 1 and 3 and some integers up to 100. We found that GridSearch always returned a float between 2 and 3.
- **solver** : We tried all the available solvers from sklearn, and GridSearch usually returned one of the following 3: lbfgs, sag, and saga.

Using the range for C and the 3 solvers, we found manually that the best option was 2.3 for C and lbfgs for the solver.

Below are the modified parameters used to train the final model:

TfidfVectorizer	
tokenizer	TweetTokenizer.tokenize
ngram_range	(1,2)

LogisticRegression	
C	2.3
solver	lbfgs

3.4. Evaluation

To evaluate our model, we will use the following metrics:

- **Accuracy** : The proportion of all predictions that are correct. It is calculated as:

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

- **Confusion matrix** : In our case a 2×2 matrix that displays the number of true positives, true negatives, false positives, and false negatives. This aids in calculating the accuracy and other metrics.
- **Learning Curve** : Determines cross-validated training and test scores for different training set sizes. It shows us how good the model is at making correct predictions on the training data as it goes through the training process.
- **ROC curve** : Illustrates the performance of a classifier model at varying threshold values. The ROC curve is the plot of the true positive rate (TPR) against the false positive rate (FPR) at each threshold setting.

4. Results and Overall Analysis

4.1. Results Analysis

Using the above parameters we train the final model, which has an accuracy of 0.8085 in the validation set. This is a relatively high score, higher than the baseline, and can give meaningful insight on a tweet's sentiment. Below we have the learning curve (Figure 7) and the ROC curve (Figure 8) of the model. At first glance, one might say that our model is overfitting, because the gap between the test score and the train score curves in the learning curve plot is not as small as expected. However since we have a high accuracy in the validation set and the ROC curve covers a large area, the model is not overfitting.

We also achieve an accuracy of 0.80763 in the test set, but no other metrics can be calculated since we don't have the true labels for the test set.

4.1.1. Best trial.

Metric	Score
Accuracy	0.8085
Recall	0.8026
Precision	0.8121
F1 score	0.8074

Confusion Matrix:

		Predicted	
		0	1
True	0	17261	3936
	1	4184	17015

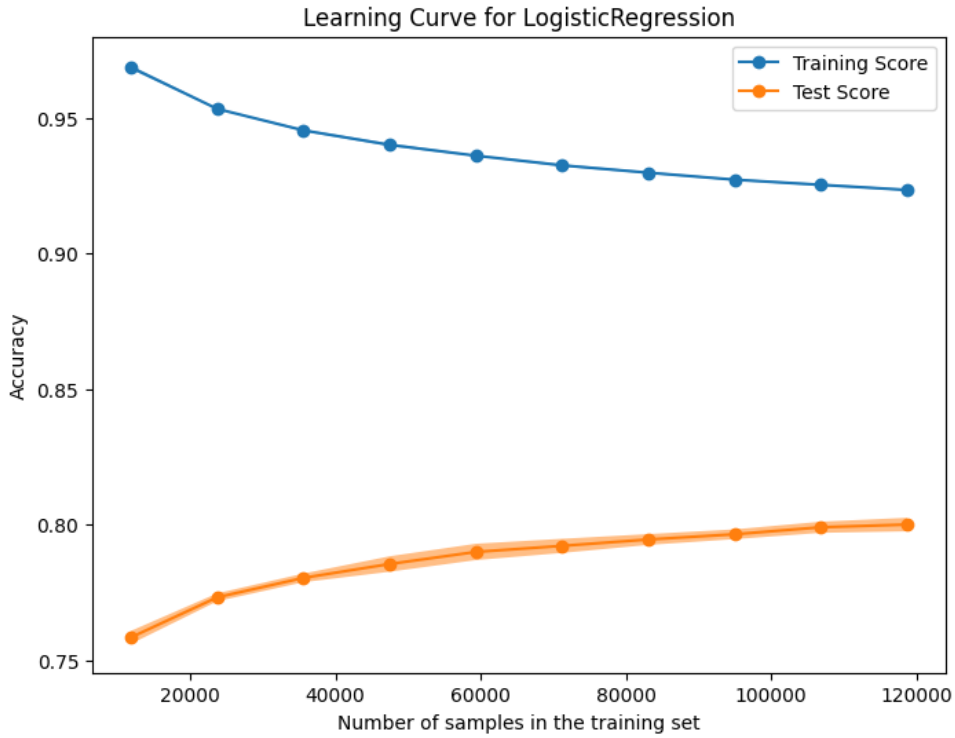

Figure 7: Learning Curve for best model

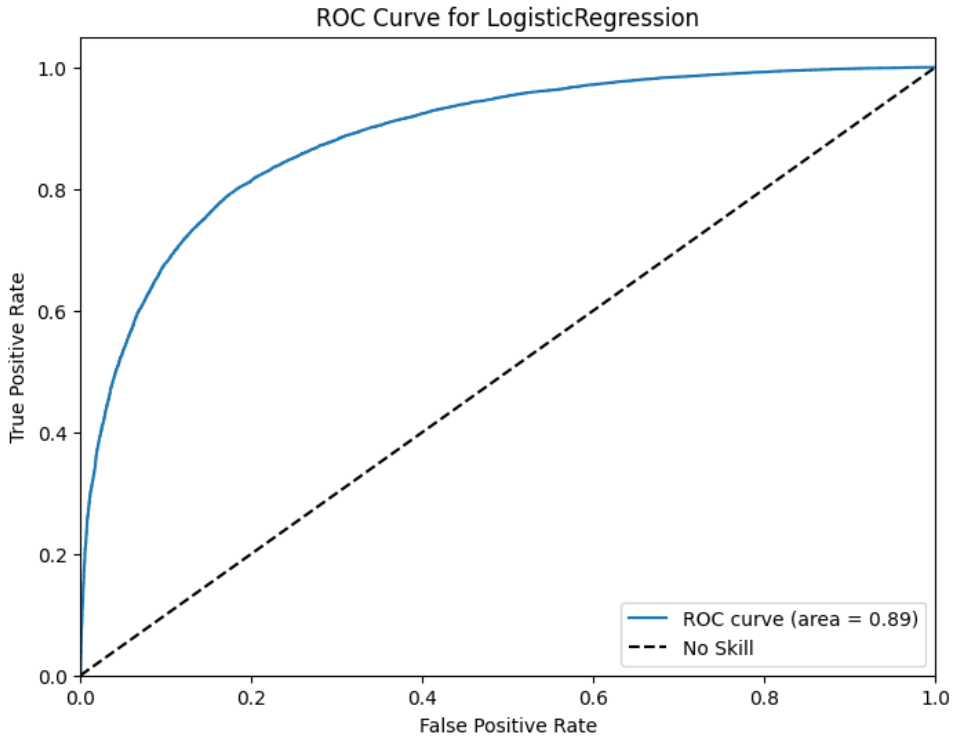

Figure 8: ROC curve for best model