UNIVERSITY OF ATHENS
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

Deep Learning for NLP

Student name: *Christina Mitsiou*
sdi: 2000129

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

Contents

1. Abstract

In this assignment, the goal is to train a sentiment classification model on a dataset of tweets. Each tweet is labeled as either positive or negative, and the task is to predict this sentiment accurately using machine learning techniques.

The process began with exploratory data analysis to better understand the dataset, followed by text preprocessing. I then used TF-IDF to vectorize the cleaned text and trained a logistic regression model for classification.

Finally, I ran several experiments to fine-tune the preprocessing steps and evaluate how they affect the model's performance, using metrics like accuracy, precision, recall, and F1-score.

2. Data processing and analysis

2.1. Pre-processing

I designed a flexible preprocessing function with the option to include each cleaning step. This allowed me to experiment with different combinations and observe which ones improved model performance the most.

The steps I implemented are the following:

- Making the text lowercase
- Expanding contractions (ex., "can't" to "can not")
- Converting emojis to their textual meaning(ex. the crying face emoji becomes crying_face)
- Removing HTML entities like `&`
- Removing mentions and URLs
- Removing hashtags but keeping the word (e.g., #happy becomes "happy")
- Removing all punctuation
- Lemmatizing the words

Each step was tested and evaluated separately to see how much it contributed to model performance. This fine-tuning helped clean the data without stripping it of potentially useful sentiment words.

2.2. Analysis

In this section, we perform exploratory data analysis (EDA) on the raw tweets before any cleaning to understand the structure and content of the dataset. The goal of this analysis is to get insight on the data which we will later use to handle preprocessing and help improve the model's performance.

2.2.1. Sentiment Distribution. We first examine the distribution of sentiment labels in the training dataset. Figure 1 shows that the dataset is very balanced, with nearly equal numbers of positive and negative tweets.

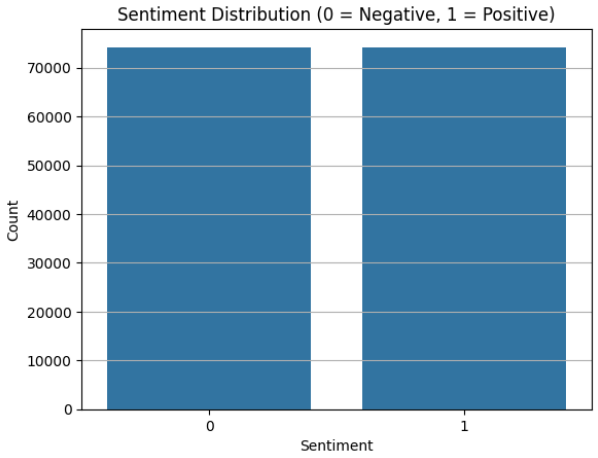

Figure 1: Sentiment Distribution (0 = Negative, 1 = Positive)

2.2.2. Most Common Words. To get an initial idea of the usage of words in sentiments, we analyzed the most frequent words in both positive and negative tweets. As shown in Figure 2, these are often punctuation, pronouns, and short common words, which confirms the need for data cleaning.

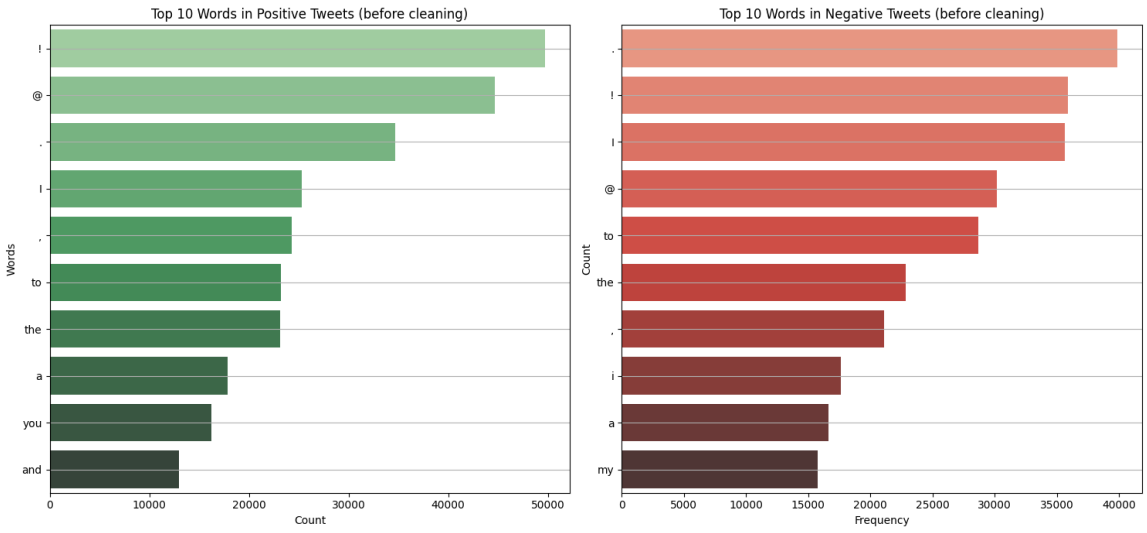

Figure 2: Top 10 Words in Positive vs. Negative Tweets (Before Cleaning)

2.2.3. Tweet Length Distribution. The histogram in Figure 3 shows that most tweets are under 150 characters, with a strong peak around 50 or so. This makes sense as tweets are usually short.

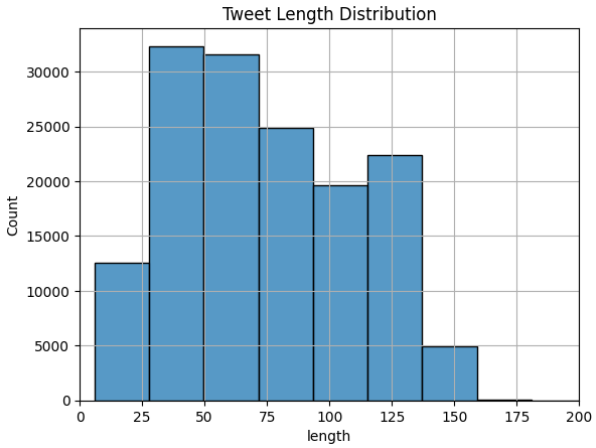

Figure 3: Tweet Length Distribution

We also examined how tweet lengths differ between sentiments. Figure 4 shows that positive tweets tend to generally be even in length with negative ones, but negative tweets show more variety in length.

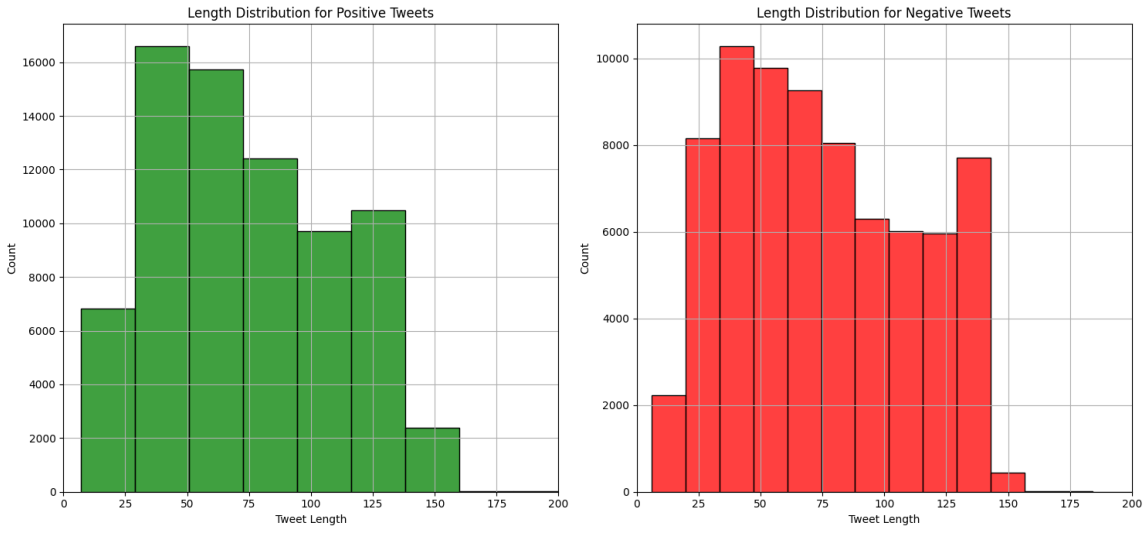

Figure 4: Tweet Length Distribution by Sentiment

2.2.4. Word Clouds. To get better insight in the most commonly used words, I generated word clouds across different categories. Figure 5 shows the most common words across all tweets.

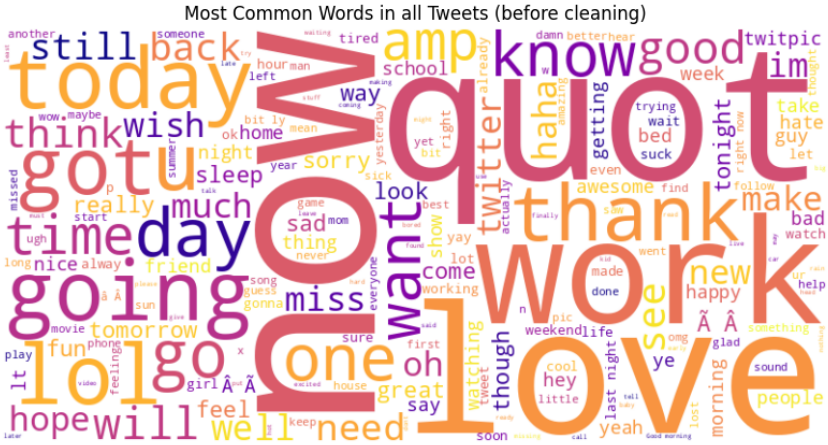

Figure 5: Word Cloud for All Tweets (Before Cleaning)

I also created sentiment-specific word clouds, shown in Figure 6. Positive tweets include words like “love”, “thank”, “good” and “great”, while negative tweets contain words like “work”, “sorry”, “miss” and “sad”.

Figure 6: Word Cloud for Positive vs. Negative Tweets (Before Cleaning)

2.2.5. Hashtag Analysis. I first generated a word cloud of all hashtags across all tweets (Figure 7), then 2 separate one based on sentiment (Figure 8).

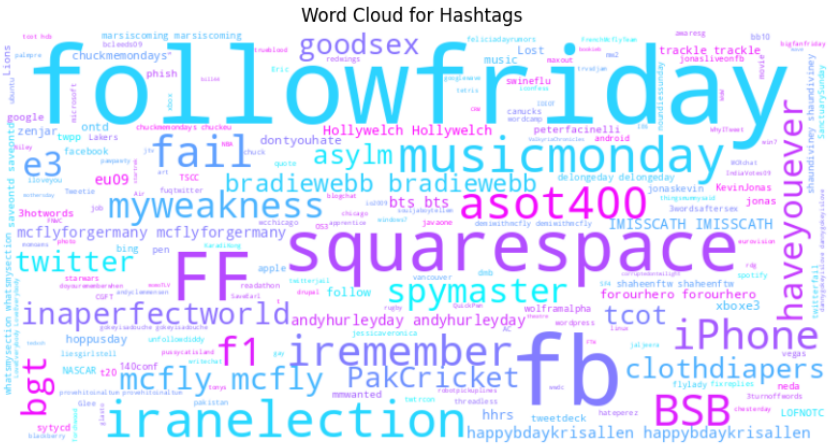

Figure 7: Word Cloud for Hashtags

We observe that the hashtags used for positive tweets tend to be words like "followfirday" and "musicmonday" while negative tweets hashtags words like "fail" and "iranelection". We can also note that the use of hashtags is more prominent in the negative tweets

Figure 8: Hashtag Word Cloud by Sentiment

2.3. Vectorization

To transform the cleaned tweets into a numerical format suitable for machine learning, I used the **TF-IDF (Term Frequency-Inverse Document Frequency)** vectorizer.

TF-IDF uses weights to showcase how important a word is in a tweet relative to the rest of the text. The words that are common across many tweets (like "the" or "to") will get lower weights, while the more unique words or words that carry more sentimental value will get higher weights.

Before vectorization, all tweets were cleaned and stored in a new column called `cleaned_text`. After that I used **TfidfVectorizer** to fit on the training data and transform all datasets (train, validation, and test) into matrices. Said matrices were then used as input for training and evaluation.

3. Algorithms and Experiments

3.1. Experiments

To understand the impact of different cleaning steps on the models performance, we tested each preprocessing step individually. This allowed us to isolate and measure the contribution of each step in the success of the model. We used a Logistic Regression model trained on TF-IDF vectorized text and evaluated on a validation set with accuracy, precision, recall, and F1-score.

The first set of experiments was simpler and included configurations where only one preprocessing step was applied at a time. Interestingly, some simple steps like removing punctuation or mentions/URLs resulted in higher scores than applying all steps together. For example, "Only Removing Punctuation" achieved an F1-score of 0.7889, outperforming the "All Steps" configuration, which only reached 0.7638.

After that I tested combinations of configurations based on top-performing and worst-performing steps. I created experiments like "Top 3 Only", "Noise Reduction Only", "All Except Stopwords", etc. The best configuration turned out to be "Top 3 Only" and "Only Removing Punctuation", both achieving F1-scores close to 0.789.

Experiment	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Only Removing Punctuation	0.7870	0.7820	0.7959	0.7889
Only Removing Mentions and URLs	0.7868	0.7810	0.7971	0.7889
Only Expanding Contractions	0.7863	0.7811	0.7956	0.7883
Only Removing Hashtags	0.7862	0.7806	0.7963	0.7883
Top 3 Only	0.7862	0.7806	0.7963	0.7883
Noise-Reduction Only	0.7856	0.7804	0.7949	0.7876
Only Removing HTML Entities	0.7855	0.7798	0.7957	0.7877
Only Removing Stopwords	0.7849	0.7794	0.7948	0.7870
Only Lemmatizing	0.7723	0.7653	0.7865	0.7757
All Except Lemmatization	0.7837	0.7788	0.7926	0.7856
All Steps	0.7601	0.7522	0.7759	0.7638

Table 1: Summary of selected cleaning configurations and their performance.

3.1.1. Table of Trials. Each experiment showed us something about which cleaning steps are helpful and which may actually hurt performance. In particular, stopword removal and lemmatization did not contribute as much as expected, especially when used with other steps.

3.2. Evaluation

To evaluate the performance of the sentiment classifier, I used four metrics: **accuracy**, **precision**, **recall**, and **F1-score**.

- **Accuracy** measures the proportion of total correct predictions over all predictions.
- **Precision** evaluates how many of the predicted positives are actually positive.
- **Recall** checks how many of the actual positives were correctly identified.
- **F1-score** is the mix of precision and recall.

After selecting the best preprocessing configuration, I plotted a **confusion matrix** (Figure 9) to visualize how well the model distinguishes between positive and negative tweets. Most predictions are in the diagonal cells, something that shows a strong performance.

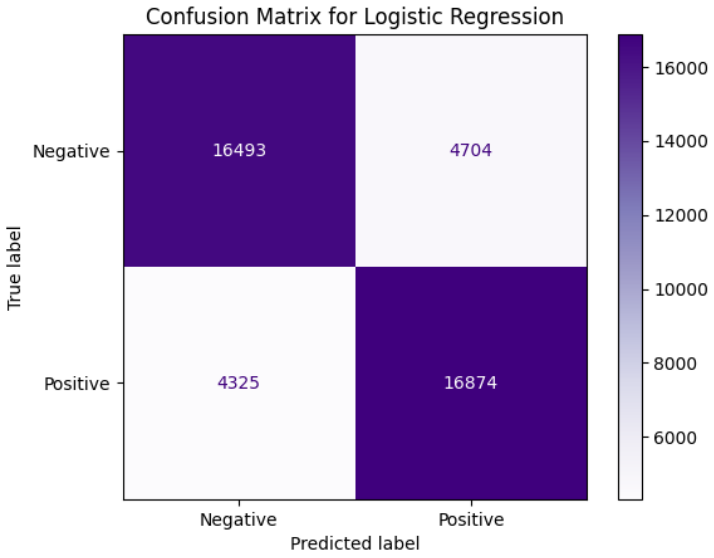

Figure 9: Confusion Matrix on Validation Set

Additionally, I plotted the **ROC curve** (Figure 10) and found the **AUC (Area Under Curve)** to evaluate the classifier across different thresholds. A high AUC value shows that the model performs well regardless of decision threshold.

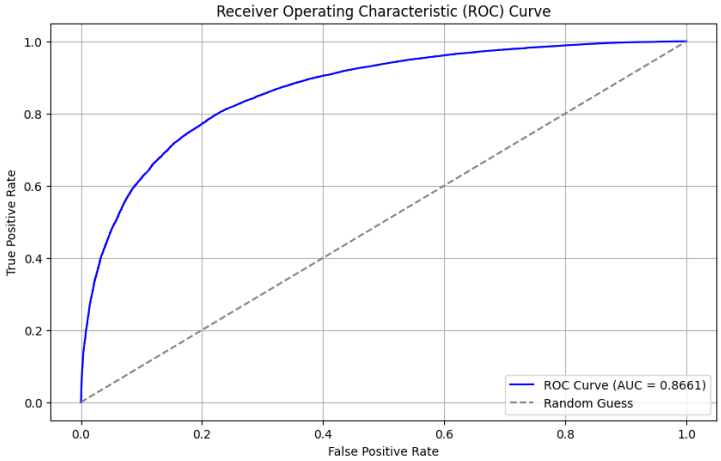

Figure 10: ROC Curve with AUC Score

In addition, I generated a learning curve to better understand how the model behaves with increasing amounts of training data. The curve shows that both training and validation scores are relatively close and stable, which is a good indication that the model does not suffer from high variance (overfitting) or high bias (underfitting). This gave me confidence that the model is generalizing well to unseen data.

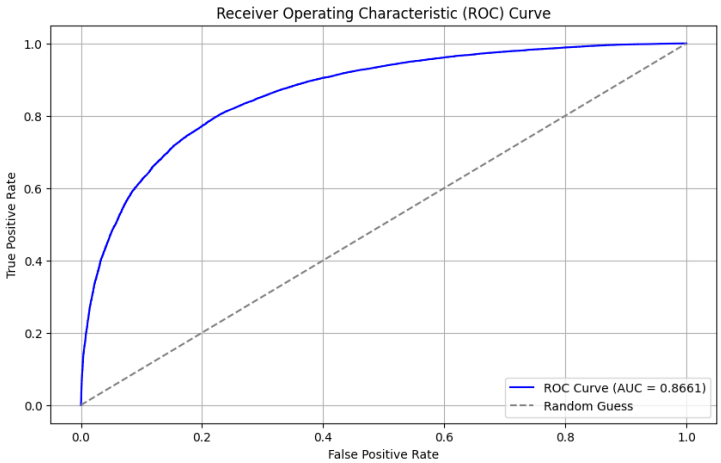

Figure 11: Learning curve

4. Results and Overall Analysis

The final model achieved strong results on the validation set, with an accuracy of approximately 78% and similarly high precision, recall, and F1-score values. These results indicate that the logistic regression model was able to effectively learn sentiment patterns from the tweet data, especially after careful preprocessing and feature extraction.

Although the performance is solid there is still room for improvement. Maybe utilizing more semantic understanding (ex. through word embeddings or deep learning models) could help the model capture the sentiments of the tweets even better (for example if emojis could be utilized more for capturing sentiments, there's a chance the model would perform a lot better).