ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

TEAM DELTA PROJECT

# "Sentiment Analysis on Social Media Data"

## MACHINE LEARNING & CONTENT ANALYTICS

BUSINESS ANALYTICS
Master of Science

29TH OF SEPTEMBER 2024

**GEORGIOS MARKAKIS – STUDENT NUMBER: P2822319**
**VASILIKI VAMVAKA – STUDENT NUMBER: P2822304**
**CHRISTOS NINOS – STUDENT NUMBER: P2822321**
**EMMANOUIL MOSCHOGIANNIS – STUDENT NUMBER: P2822320**

# Table Of Contents

# 1 Introduction

## 1.1 Project Description

Social media has become a transformative force in how individuals, organizations, and even governments communicate and influence public opinion. Platforms like Facebook, Instagram, and Twitter(X) have become the epicenters of real-time discussions, where trends are born, products are praised or criticized, and global news spreads instantly. Sentiment analysis, the process of identifying and classifying opinions expressed in a piece of text, is particularly important in understanding these vast conversations.

In this project, we chose to start with Twitter due to its role as a global hub for major announcements, trending topics, and end-to-end feedback loops between organizations and their audience. Twitter's unique structure—limited character length, frequent updates, and broad user base—makes it an ideal platform for analyzing short, impactful messages. Whether it's product reviews, corporate announcements, political debates, or customer service interactions, Twitter provides immediate insights into public sentiment that can help guide decision-making.

The project aims to build a sentiment analysis system to classify tweets as positive, negative, or neutral using Natural Language Processing (NLP) techniques and deep learning models. By focusing on Twitter initially, we can capture trends and feedback in real time, but future iterations of the project will expand to include analysis of other major social media platforms. This multi-platform approach will offer a more comprehensive understanding of public sentiment across different user demographics and networks.

## 1.2 Business Questions Addressed

In today's world, where social media conversations can make or break a brand or political figure, being able to monitor and understand public sentiment has never been more important. This project tackles three key questions about how businesses, politicians, and marketers can better understand and react to public opinion, especially on platforms like Twitter.

How can businesses keep track of public feedback without being overwhelmed?
For big corporations, handling millions of tweets every day is no easy task. It's impossible for a single team to manually sift through all the comments, reviews, and mentions. By using sentiment analysis, businesses can quickly gauge whether people are reacting positively or negatively to their products, campaigns, or announcements. This tool can be a social monitoring system that helps large companies stay on top of public opinion without needing massive manpower, allowing them to address concerns or capitalize on positive reactions more efficiently.

How can politicians understand the real-time impact of their statements?
In politics, timing and perception are everything. When politicians make public statements, they need to know how those messages land with their audience—whether they're gaining support or facing backlash. Sentiment analysis can offer instant feedback, letting them see if their statements are resonating

positively or if they need to rethink their approach. This helps politicians tweak their strategies and messaging to connect better with voters and the public at large, ensuring they stay aligned with public sentiment.

How can marketers predict what will work based on consumer behavior? Every marketer dream of knowing exactly what will hit the mark with their audience. Sentiment analysis provides insights into which campaigns, products, or messages generate positive buzz and which ones fall flat. By tracking consumer reactions over time, marketers can better understand what resonates and model future campaigns to make sure they're hitting the right notes. It's about learning from what works and adjusting for even better results moving forward.

## 1.3 Objectives and Goals

The main objective of this project is to develop an accurate sentiment analysis system capable of processing large volumes of Twitter data, categorizing tweets as positive, negative, or neutral. By utilizing Natural Language Processing (NLP) techniques and deep learning models, specifically Long Short-Term Memory (LSTM) networks, we aim to create a system that can reliably assess public sentiment from social media conversations.

To achieve the goal of a robust sentiment analysis system, several key objectives must be met. These objectives include:

1) **Developing a robust data processing pipeline:**
   This involves extensive data cleaning, including tokenization, stopword removal and normalization to prepare raw tweet data for sentiment analysis. This is crucial to ensuring that the data fed into the model is of high quality, which directly impacts the model's performance.
2) **Implementing an LSTM-based model for sentiment classification:**
   The core of this project is to design and train an LSTM model capable of understanding the sequential nature of tweets. The model will be fine-tuned to detect and classify sentiment, utilizing the preprocessed data to identify patterns in short-form text.
3) **Batch-based sentiment analysis for large-scale data**:
   The project focuses on processing tweets in batches to deliver meaningful insights efficiently. This approach ensures the system can scale to handle large datasets while maintaining the accuracy of its results.

### 1.4 Tools and Technologies

To implement this sentiment analysis system, we leveraged a variety of tools and technologies across different stages of the project. From data collection to model training and visualization, these tools helped streamline the workflow and ensure accuracy throughout the process.

1) **Python**
   Python was our language of choice because it's powerful, flexible, and offers a wide array of libraries tailored for data science and machine learning. Its rich ecosystem, particularly in NLP, made it ideal for this project.

2) **Pandas and NumPy**
   We used Pandas to clean, filter, and organize the tweet data, ensuring it was well-prepared for analysis. NumPy helped handle more complex mathematical operations, especially when working with large datasets and matrix computations.

3) **NLTK and Langid**
   For text processing, NLTK was used to handle tasks like tokenization, stopword removal, and stemming, helping prepare the tweets for analysis. Langid was utilized to filter out non-English tweets, ensuring the data was consistent and focused on relevant content.

4) **TensorFlow and Keras**
   TensorFlow and its user-friendly API, Keras, made it easy to build and train our LSTM (Long Short-Term Memory) network, which is essential for understanding the sequential nature of short texts like tweets. TensorFlow's flexibility and Keras' simplicity allowed for quick experimentation and model optimization.

5) **Matplotlib**
   Matplotlib was used to create visualizations throughout the project. These visualizations helped us analyze the distribution of sentiments and track the model's performance, playing a key role in understanding the data and communicating results effectively.

## 2 Data

### 2.1 Data Collection and Dataset Overview

The dataset utilised in this study was obtained from Kaggle (available link here), was created specifically for sentiment analysis, with the goal of classifying tweets as positive, negative, or neutral. The data was collected automatically from Twitter, where sentiment labels were assigned based on the meaning of the tweets. The dataset has two independent sets: one containing 27,481 items designated as the training set, and one with 4,815 entries allocated as the test set.

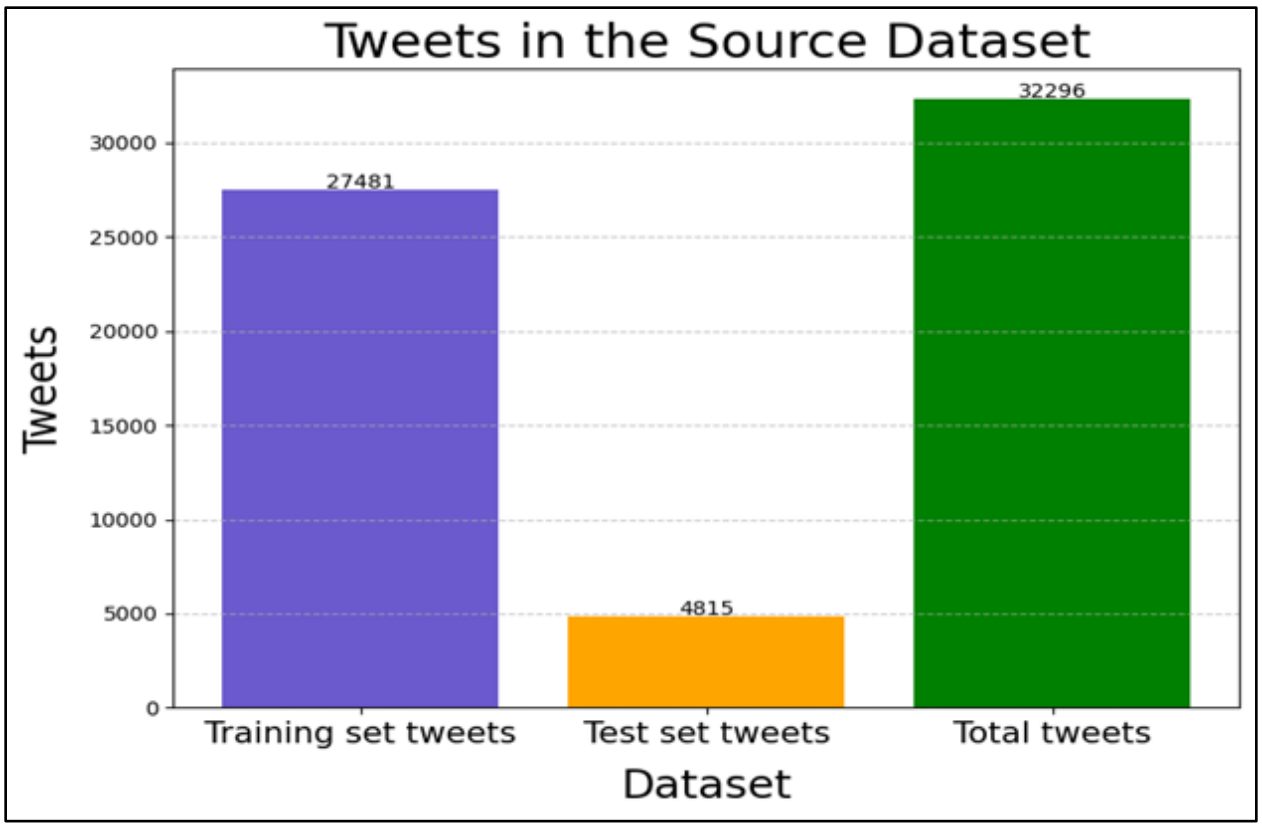


*Figure 1*

The data provides a range of information regarding each entry, including the time of the tweet, the user's age, country, etc.
For our research, we will utilise the following columns:

1) **text:** The content of the tweet, which serves as the primary input for the sentiment analysis model
2) **Sentiment:** The target label categorises each tweet as positive, negative, or neutral
3) **selected_text (in the training dataset only):** Highlights the most sentiment-relevant portion of the tweet, identifying the specific part of the tweet that reflects the assigned sentiment

At this point, we will continue our analysis by creating some visualizations with the aim of getting an initial view regarding our data.

The dataset analysis shows a minor class imbalance, with neutral tweets appearing more often than positive or negative ones in both the training and test sets, as illustrated in Figure 2 below. Although this imbalance was not severe, it was considered during model evaluation to prevent skewing predictions. Despite this, the proportional distribution of tweets across sentiments is consistent, which helps mitigate bias during model training.

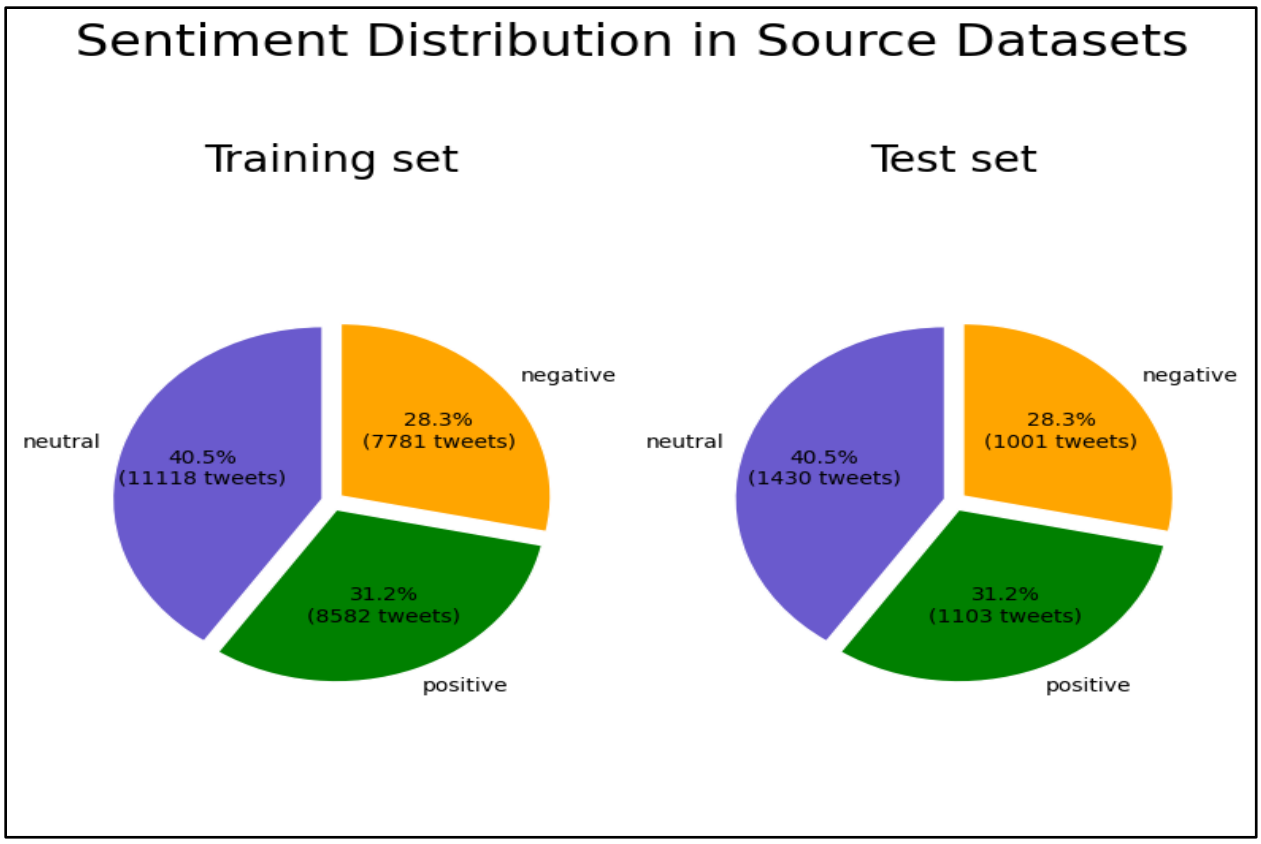


*Figure 2*

From the provided histogram, we can observe the distribution of word count per tweet in the training and test sets. The tweet length varies significantly, with most tweets falling between 5 and 15 words, which is the range where the model focuses for optimal learning. This variability is typical in social media posts due to the nature of concise communication.

To handle this variability, padding is applied during the preprocessing stage to standardize the input size for the model. Padding ensures that each input to the model has the same length, which is crucial for algorithms like LSTMs (Long Short-Term Memory networks), which require consistent input shapes. Tweets shorter than the required length are padded with zeros, and longer tweets may be truncated to fit the standard length. This uniform input helps the model process the data more effectively, ensuring that differences in tweet length do not impact the model's performance.

In this specific project, padding was essential to account for tweets of varying lengths and ensure that the LSTM model could focus on the content rather than being affected by the tweet's length differences.

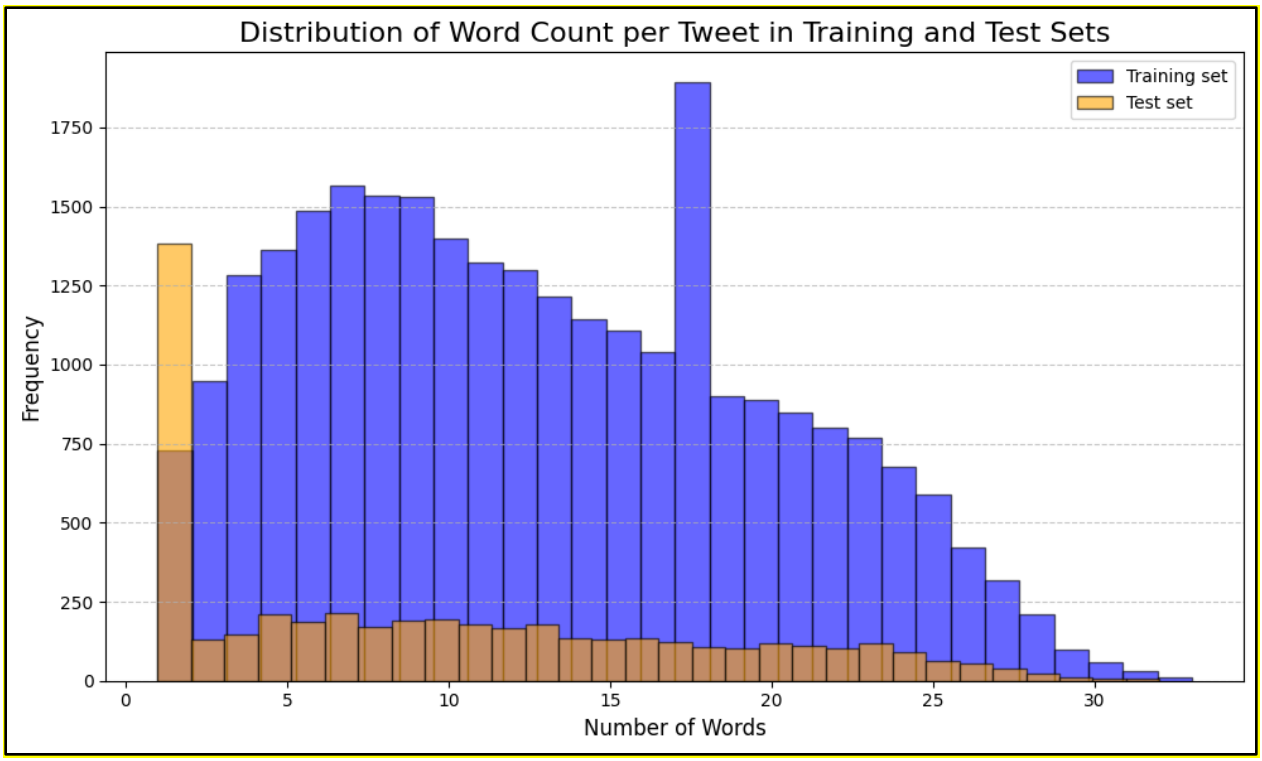


*Figure 3*

## 2.2 Data Processing

Preprocessing is essential for ensuring that raw textual data is clean and prepared for analysis; all actions must be applied to both the training and test sets to achieve optimal and correct results.

### 2.2.1 Handling Missing Values

First, we address the null values by identifying entries with at least one main column that is null. As shown in Figure 4, the missing values are consistent across all columns, indicating that the entire entry is null. Therefore, we can safely remove these entries without affecting our data integrity. In the training set, there was only one null entry, while in the test set, we found approximately 1,200 entries with null values.

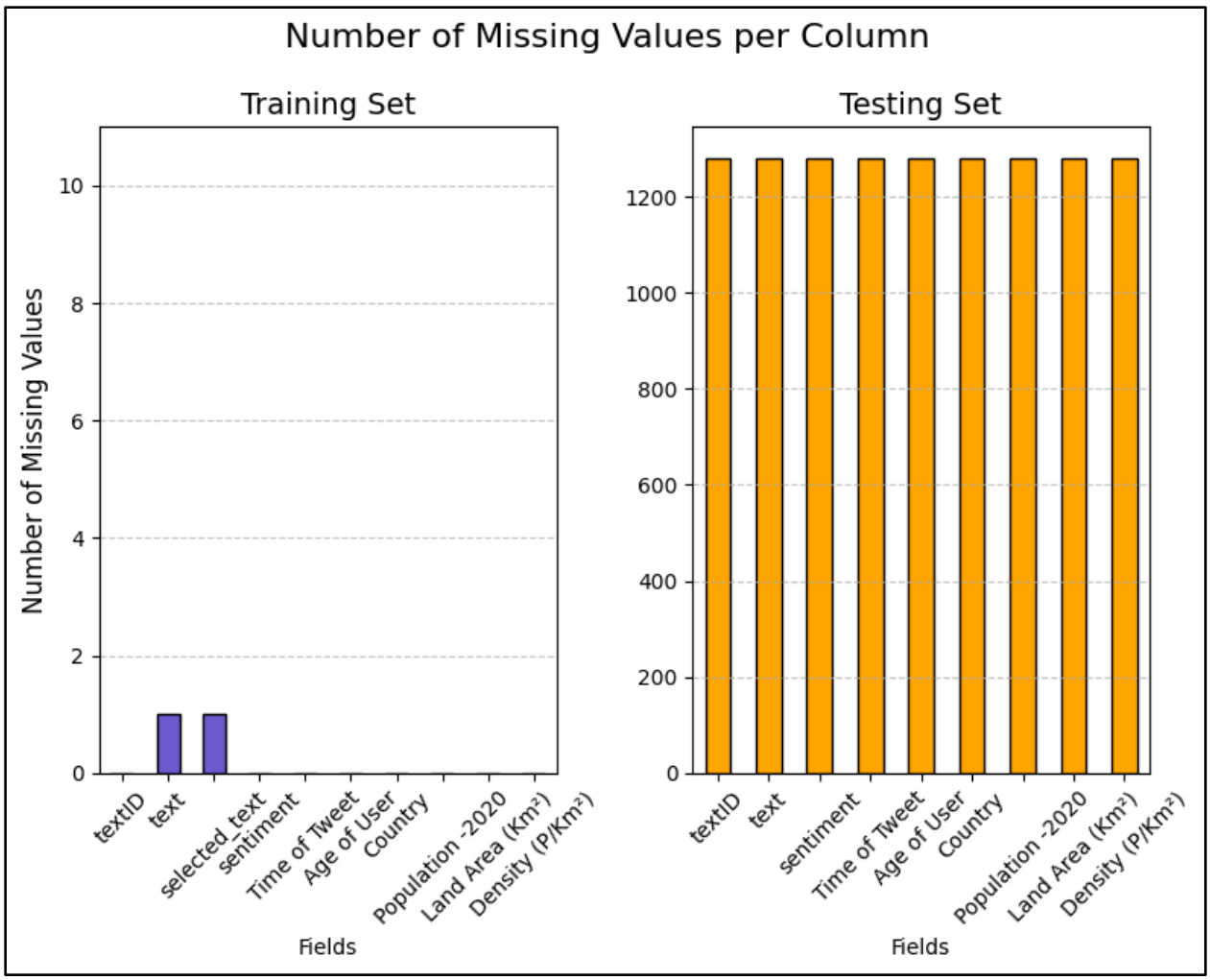


*Figure 4*

### 2.2.2 Language Filtration

We continue cleaning by deleting non-English tweets, which were recognized and filtered out using the "langid" library. This guaranteed that the dataset stayed consistent by focusing solely on English content, simplifying the model's training and limiting the possibility of noise from foreign language text. The were found 1,161 entries in the train set and 159 in the test set and these entries were removed.

As a result, after cleaning, our sets contain 26,319 entries for the train set and 3,534 entries for the test set (see Figure 5). Moreover, Figure 6 depicts the sentimental distribution in both cleaned sets.

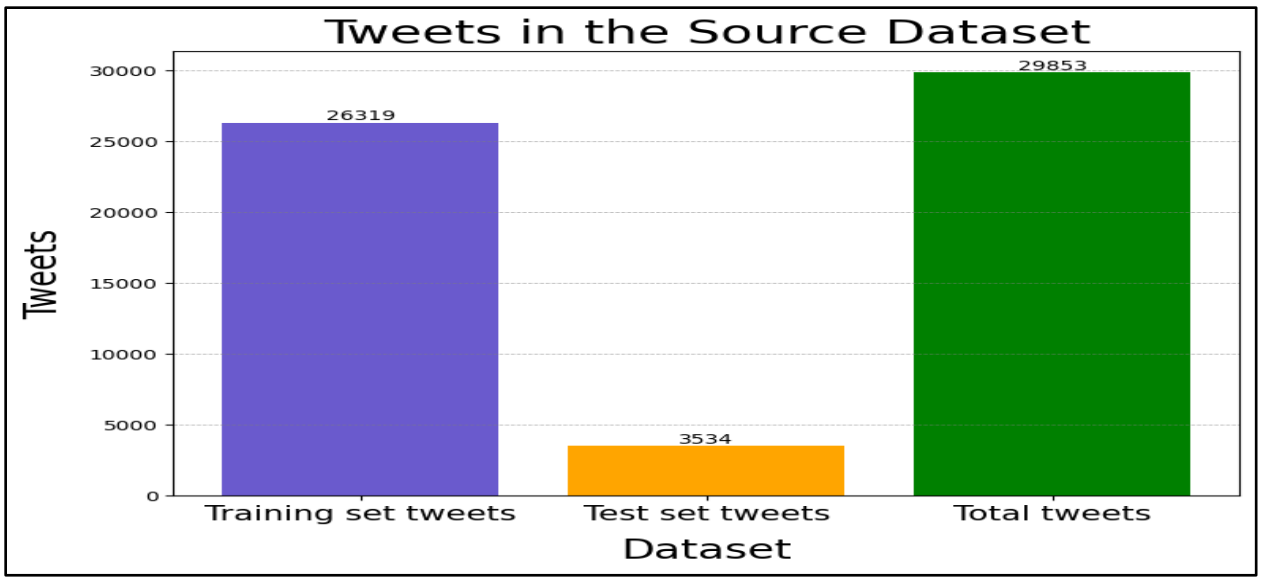


*Figure 5*

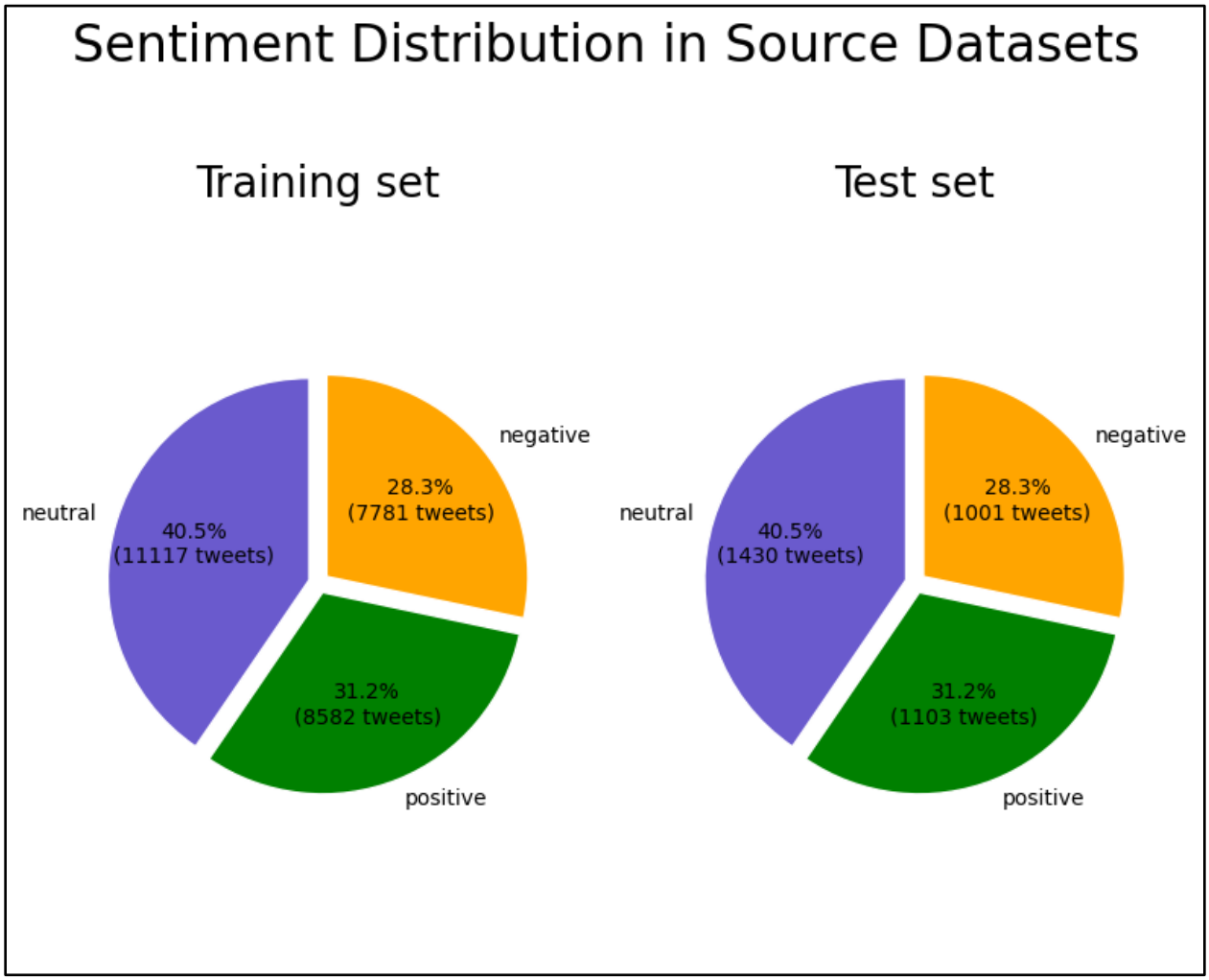


*Figure 6*

## 2.3 Normalization / Indexing

After cleaning the data, we need to normalize it to achieve a standardized format across the entire system. Data normalization is a crucial process designed to reduce errors and inconsistencies, which can significantly impact the efficiency and accuracy of data systems.

We begin by creating the cleanUP() function. Using regular expressions, we remove html tags, emoticons, symbols and pictographs, transportation and map symbols, and flags. Next, we convert all the text in the text column to lowercase. This step ensures consistency in text representation by eliminating case sensitivity, allowing words like "Happy" and "happy" to be treated as the same. We also proceed to remove digits and any single or two-letter words.

Next, we proceed with the removal of the stopwords. Stopwords, such as “the,” “is,” and “and,” were removed using NLTK’s stopword list to reduce noise in the dataset. Stopword removal improves the model’s efficiency and reduces the dimensionality of the dataset, making it easier for machine learning algorithms to process.

Following data normalization, we proceed with indexing.

Indexing improves models’ performance by keeping only the essential to our analysis words of each tweet. During indexing, a new index column was produced.

For example:

| Original Tweet | Final Outcome post indexing |
|---|---|
| its at 3 am, im very tired but i can`t sleep | tired sleep try |

### 2.4 Tokenization

The purpose of the tokenization process is to convert words into numerical tokens.
For example:

| Word | Token |
|---|---|
| day | 1 |
| good | 2 |
| get | 3 |
| like | 4 |

In our case, the tokenizer will take into account only the top 5.000 most frequent words in the training data.
The number of unique words in all the tweets (vocabulary size) is equal to 23.450.

## 3 Methodology/Model

### 3.1 Conceptual Approach

After completing the essential tasks of text processing and feature extraction, the next crucial step is to develop a sentiment analysis model. We will be using Recurrent Neural Networks (RNNs), a type of artificial neural network that is well-suited for processing sequential data. Unlike feedforward neural networks, which process information in a single pass, RNNs analyze data over multiple time steps, making them ideal for text modelling and sequence analysis. Specifically, we will implement the Long Short-Term Memory (LSTM) architecture, a variant of RNNs.

Let's explore how an RNN works. By feeding the output from the previous step as input to the current step, RNNs capture dependencies between inputs. This contrasts with traditional neural networks, where inputs and outputs are independent of one another. The most important feature of an RNN is its *hidden state*, which retains some information from previous steps in the sequence. This makes RNNs well-suited for tasks like time series classification, where the order of time points is crucial. RNNs leverage feedback loops to account for dependencies across time.

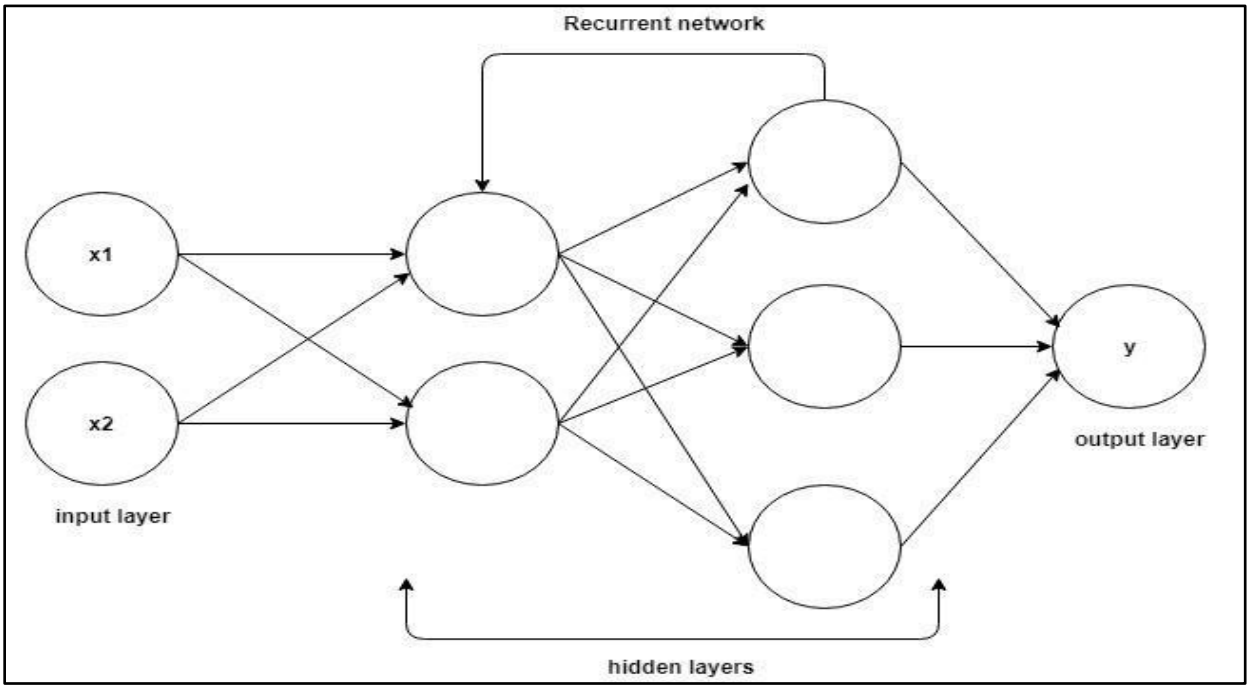


*Figure 7*

Figure 7 provides a graphical representation of a recurrent neural network.

The LSTM network, a more advanced form of RNN, developed to better model sequential data and address the limitations of standard RNNs, particularly for long-term dependencies. In their research cite, the authors tackle the issue of long-term dependencies. Traditional RNNs struggle with such cases, as they may not accurately predict the current state if the relevant past information is not recent.

LSTMs resolve this by using "cells" in their hidden layers, each containing three types of gates: the input gate, output gate, and forget gate. These gates regulate the flow of information, allowing the network to decide which information is important to retain or discard to make accurate predictions.

In a neural network, nodes are interconnected through weighted links, where each weight represents the significance of a given input in determining the desired output. To compute the output, the input value is multiplied by the weight associated with each link, and then the results from all nodes in that layer are summed. This sum is passed through a non-linear function called the activation function. Activation functions improve the model's ability to learn complex patterns by introducing non-linearity, enabling the network to capture non-linear decision boundaries.

There are various types of activation functions available. In our model, we will use the following functions:

- **The Tanh function:** outputs values ranging from -1 to +1, allowing it to handle negative values more effectively than the sigmoid functio*n.*
- **ReLU (Rectified Linear Unit):** This function sets negative input values to zero, while returning the input itself for positive values.
- **Softmax function:** generates a probability distribution where all probabilities sum to one, and each element represents the likelihood that the input belongs to a particular class.

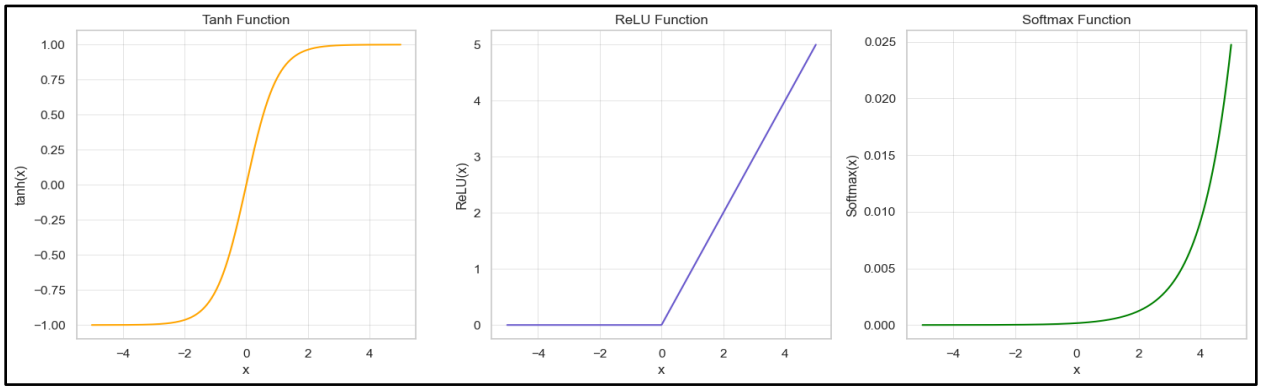


*Figure 8*

Finally, in our model we used as loss function the "Categorical crossentropy loss function".

$$L = \sum_{i=1}^{N} \sum_{j=1}^{c} y \; log\left(\hat{y}_{ij}\right)$$

Where:

- N is the number of samples (tweets in our case).
- C is the number of classes (3 sentiment categories: positive, negative, neutral).
- $y_{ij}$ is a binary indicator (0 or 1) if the class label j is the correct classification for sample i.
- $\hat{y}_{ij}$ is the predicted probability that sample i belongs to class j.
- $log\left(\hat{y}_{ij}\right)$ is the natural logarithm of the predicted probability for numerical stability and precision.

Generally, a loss function is a mathematical function used in machine learning to measure the difference between the predicted values produced by a model and the actual values (the true labels). It quantifies how well or poorly the model is performing during training. The main purpose of the loss function is to guide the optimization process.

## 3.2 LSTM Implementation and Model Architecture

The implementation of the neural network was done using a high-level neural network API known as Keras. This module allowed us to construct our neural network layer by layer and specify the activation functions, loss functions, dropouts, etc.

In the table below is presented the summary of the neural network:

| Layer (type) | Purpose | Number of Neurons | Activation Function |
|---|---|---|---|
| Embedding Layer | Converts the input words into dense vectors of a fixed size (in our case 50) | 50 | |
| LSTM Layer | The LSTM layer captures sequential dependencies from the input data | 64 | Tanh function as activation function & sigmoid as recurrent activation function |
| GlobalMaxPooling1D | It downsamples the data from the LSTM to a fixed-size representation, no matter the sequence length | | |
| First Dense Layer | Applies transformations to the input from the LSTM and pooling layers | 128 | ReLU (Rectified Linear Unit) |
| Dropout Layer | This layer randomly drops 50% of the connections during training to prevent overfitting | | |
| Second Dense Layer | Further refines the learned features from the first Dense layer | 32 | ReLU (Rectified Linear Unit) |
| Dropout Layer | This layer randomly drops 20% of the connections during training to prevent overfitting | | |
| Output Layer | This layer produces the final output of the model | 3 | Softmax |

- **Hidden Layers:** 2 dense layers
- **Batch size:** We set it to 32
- **Number of epochs:** 5, so the model will go through the entire training dataset 5 times during training
- **Loss function:** categorical crossentropy, since the problem in hand is a multi-class classification problem
- **Regularization:** L2 Regularization was applied to both dense layers
- **Optimizer:** Adam (Adaptive Moment Estimation), which is a first-order gradient-based optimization algorithm
- **Learning rate:** We set the learning rate to 0.01 as this is a standard value which is optimal in most cases.

## 3.3 Evaluation Protocol

### 3.3.1 Precision

Precision is the proportion of true positive predictions out of all the positive predictions made by the model.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives \& False Positives}}$$

### 3.3.2 Recall

Recall is the proportion of true positive predictions out of all the actual positives in the dataset.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives \& False Negatives}}$$

### 3.3.3 F1-Score

F1-score is the harmonic mean of precision and recall, providing a balanced measure that accounts for both false positives and false negatives.

$$\text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision \& Recall}}$$

# 4 Model Performance

## 4.1 Enhancing Performance (hyperparameter tuning)

To achieve the best performance, we utilize a process called hyperparameter tuning. This process involves selecting the optimal values for a machine learning model's hyperparameters settings that control how the model learns. Hyperparameters are distinct from model parameters, which are the internal weights and biases learned from the data during training. Below are the different hyperparameter options that we used in our model:

- **Epochs:** [5, 10] [1]– the number of times the model will process the entire training dataset
- **LSTM Neurons:** [32, 64, 128] – the number of neurons in the LSTM layer
- **Layers:** [1, 2, 3] – the number of layers in the neural network
- **Neurons per Layer:** [[32], [128, 32], [128, 64, 32]] – the number of neurons in each layer, matching the number of layers
- **Dropout Rate:** [0.2] – the rate at which neurons are randomly dropped to prevent overfitting
- **Learning Rate:** [0.001] – controls how fast the model updates its internal weights during training
- **Embedding Output Dimensions:** [50] – the size of the vector representing each word in the embedding layer
- **Batch Size:** [32] – the number of training samples processed before updating the model's internal weights

Given these hyperparameter options, a total of 54 different model configurations are tested.
After evaluating each model's accuracy, the one with the highest accuracy is selected as the best model, which is then described in detail in section 3.2.

## 4.2 Performance Metrics for Each Sentiment Class

| | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| **Class 0 (negative)** | 0.78 | 0.64 | 0.71 | 955 |
| **Class 1 (neutral)** | 0.67 | 0.74 | 0.71 | 1,371 |
| **Class 2 (positive)** | 0.76 | 0.79 | 0.77 | 1,049 |
| | | | | |
| **Accuracy** | | | 0.73 | 3,375 |
| **Macro avg** | 0.74 | 0.72 | 0.73 | 3,375 |
| **Weighted avg** | 0.73 | 0.73 | 0.73 | 3,375 |

[1] [x,y] refers to a set of elements not a range

The LSTM model's results reveal that it performed well overall, with an accuracy of 73% across all three classes. The precision, recall, and F1-scores for each class show a fairly balanced performance.

- Class 0 - negative sentiment: has a precision of 0.78 but a significantly lower recall of 0.64, indicating that the model is competent at predicting negative sentiment but misses some real instances.
- Class 1 (neutral sentiment) : has a recall of 0.74, suggesting strong sensitivity, but a slightly lower accuracy (0.67), implying that this class produces more false positives.
- Class 2 (positive sentiment); has the best performance, with both high precision (0.76) and recall (0.79), showing that the model correctly detects this class.

## 4.3 Learning Curve Analysis

As mentioned earlier, we use accuracy to track how well the model is learning and generalizing throughout the training process. These metrics are depicted in Figure 9, where the training accuracy (blue line) shows consistent improvement, rising from 50% to about 85%, indicating that the model is effectively learning patterns from the training data. On the other hand, the validation accuracy (orange line), which reflects the model's performance on unseen data, initially improves but levels off at around 70% after several epochs, even showing a slight decline. This gap between training and validation accuracy suggests that the model may begin overfitting after the second epoch.

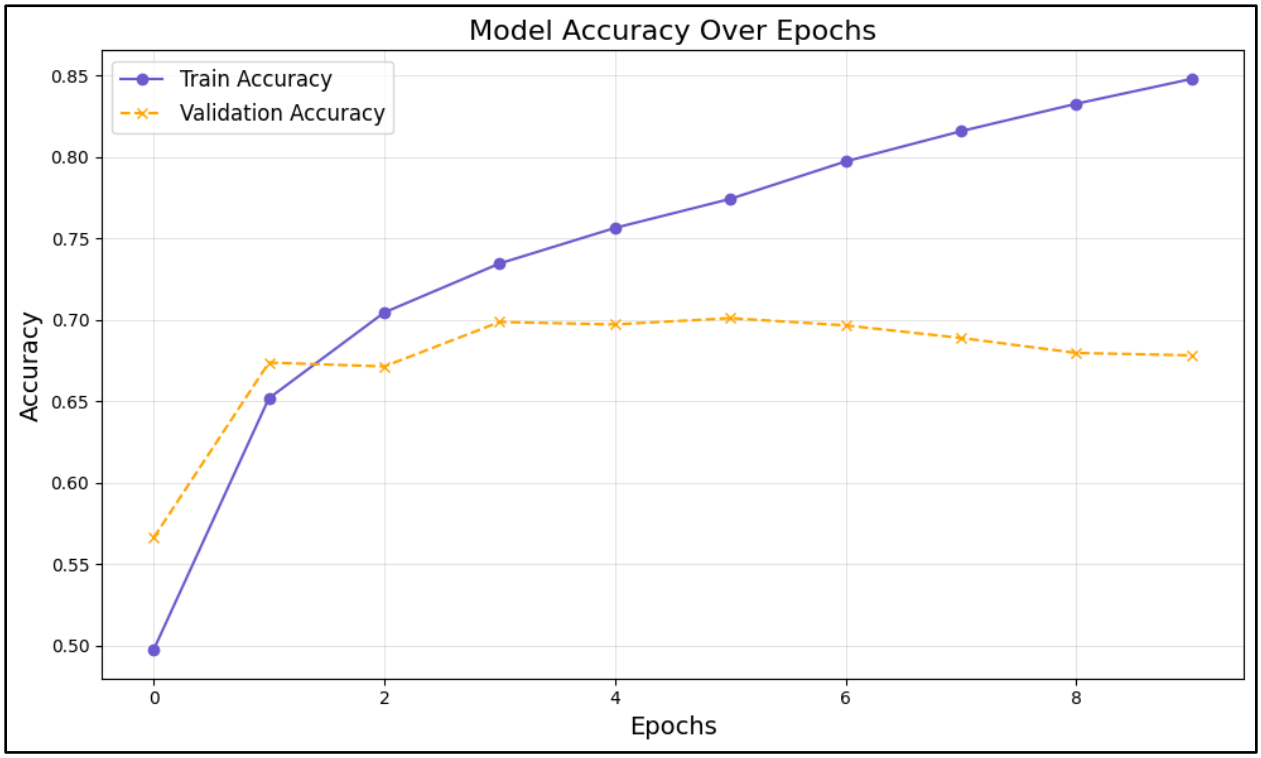


*Figure 9*

Next Figure 10 shows the training loss and validation loss over a series of epochs. The training loss (blue line) decreases consistently, indicating that the model is becoming better at minimizing errors in the training data. In contrast, the validation loss (orange line) initially decreases but then starts to increase after a few epochs, indicating that the model's performance on unseen data is deteriorating over time.

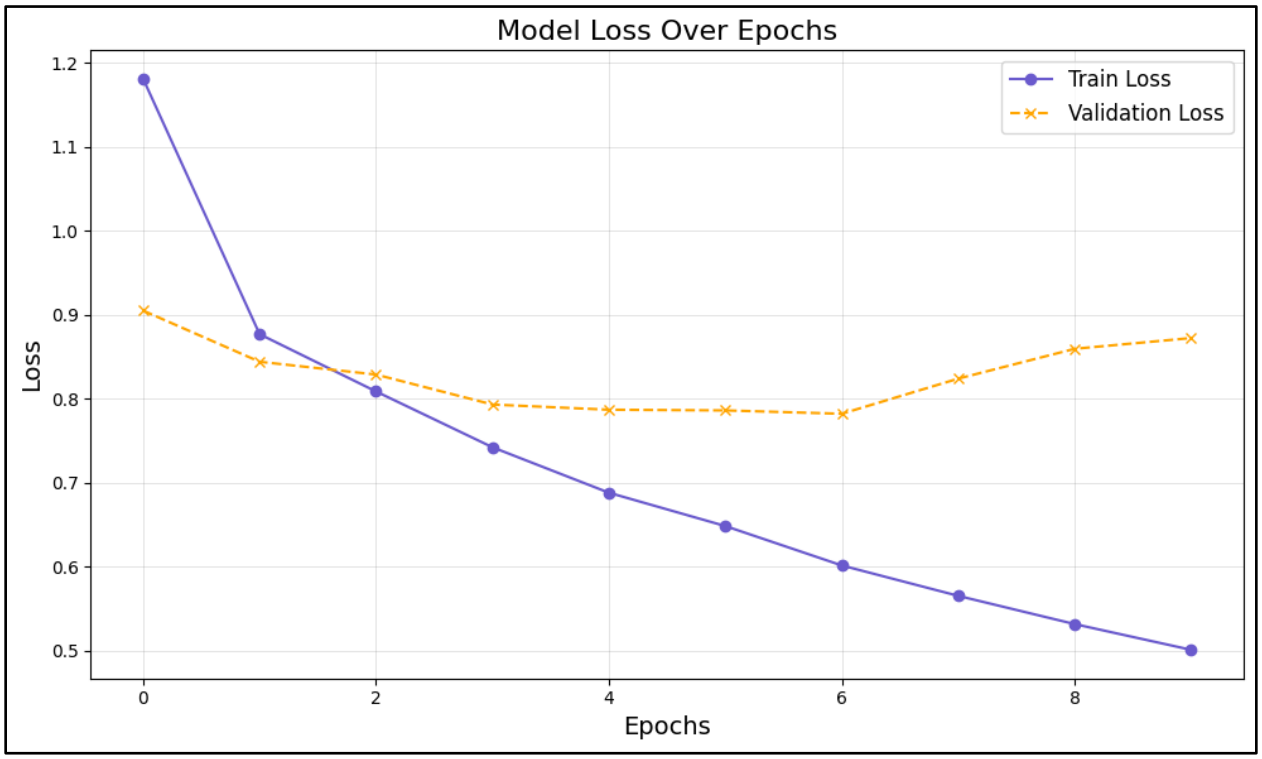


*Figure 10*

# 5 Discussion and Conclusion

## 5.1 Insights and Observations

The LSTM model demonstrated a clear capability for capturing sentiment in tweets, particularly for positive and negative sentiments. However, distinguishing neutral sentiment remained a challenge, partly due to the complexity of identifying neutrality in brief, informal texts.

## 5.2 Challenges and Obstacles Encountered

Our main problem was that our model was overfitting, since it performed better with the training data in comparison to the evaluation data.

In order to deal with the overfitting, we executed the following actions:

### 5.2.1 Action 1

Upon analyzing the distribution of the sentiment labels in the larger training set it becomes evident that the dataset is highly imbalanced, as it can be observed in the below pie (no neutral tweets)[2]:

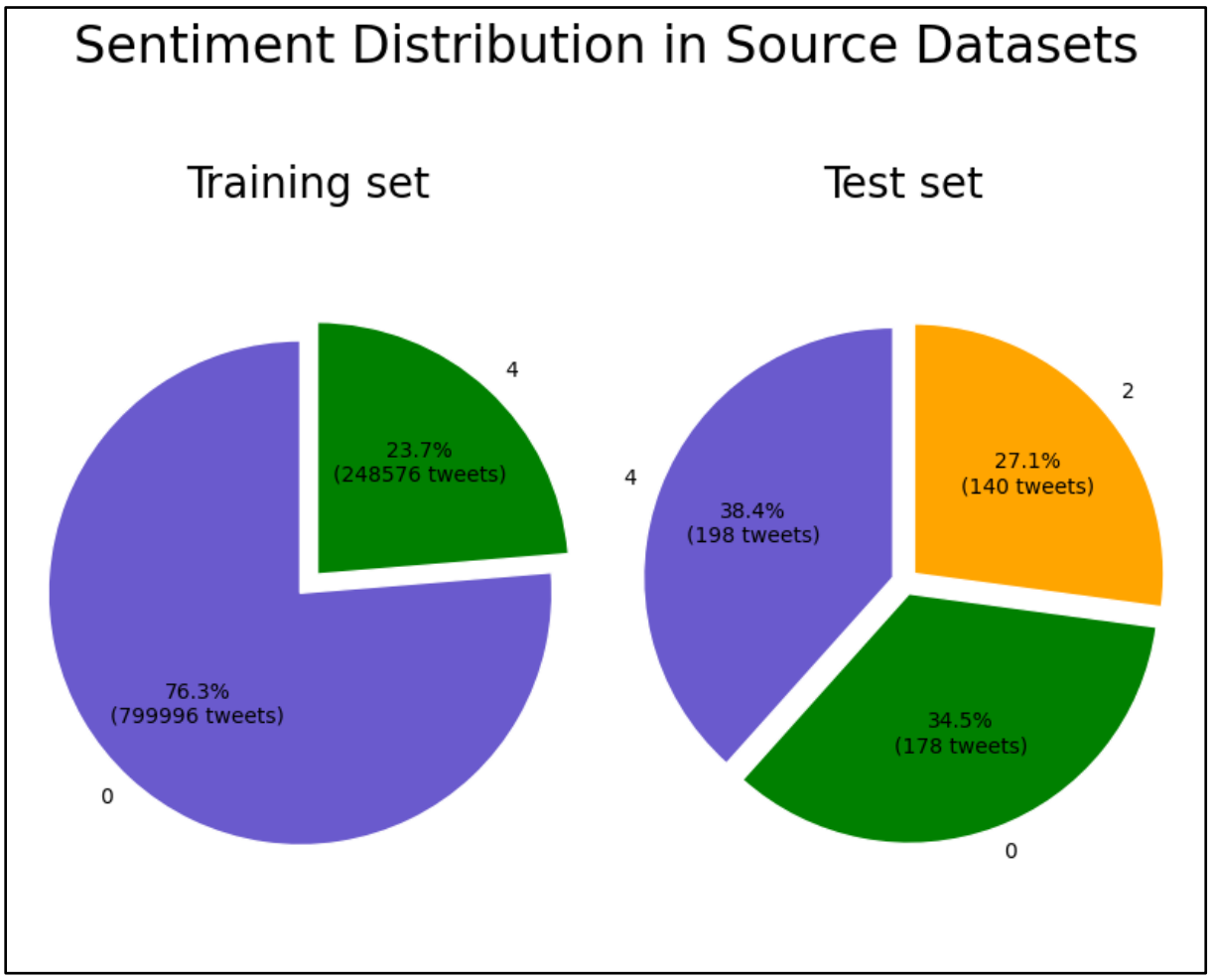


*Figure 11*

The impact of this imbalance on the model performance was severe, since our model turned out to be biased towards the majority classes.

Since we had no other way to collect more data, we decided to discard the larger dataset.

[2] 0 = Negative, 2 = Neutral & 4 = Positive

### 5.2.2 Action 2

Tried to run the model with the "selected text" instead of the "index" column, in order to check if our indexing procedure has a negative impact on overfitting. The results were worse in terms of overfitting, so we kept our indexing method.

### 5.2.3 Action 3

To improve the performance of the sentiment classification model, we performed hyperparameter tuning, experimenting with the following parameters:

- Batch Size
- Number of Neurons in LSTM and Dense Layers
- Dropout Rates
- Number of Epochs

By fine-tuning these parameters (see § 4.1 for more information) we got better results (based on our evaluation method).

## 5.3 Future Work

### 5.3.1 Current model

The macro average and weighted average F1-scores of 0.73 reflect overall balanced performance, but there is room for improvement, especially in better capturing Class 0 (negative) and Class 1 (neutral) instances.

### 5.3.2 Incorporating a transformer-based model

Incorporating a transformer-based model such as BERT could improve the model's contextual understanding of tweets.

### 5.3.3 Expansion to other platforms

Although the current focus is on Twitter, the long-term vision is to expand this sentiment analysis system to other social media platforms like Facebook and Instagram.
By expanding to other platforms, we aim to capture a richer, more diverse set of public sentiments, enabling businesses and organizations to respond more effectively across multiple channels.

# 6 Team Members and Roles

This project was conducted by four part-time MSc students in Business Analytics, each bringing their professional experience to both technical and strategic aspects:

| SN | Name | Role | Profession | Responsibilities |
|---|---|---|---|---|
| 1 | Georgios Markakis | Business Strategist, Project Manager, and Data Collection Lead | Growth & Performance Marketing at PeopleCert | Georgios coordinated the project, led data collection, and ensured the results aligned with business applications like feedback and social monitoring strategies. |
| 2 | Vasiliki Vamvaka | Technical Lead and Model Developer | Developer at Adzuna | Vasiliki led the technical implementation of the model, working closely with Manos on model development and managing NLP tasks like stopword removal and text cleaning to optimize the system. |
| 3 | Christos Ninos | Risk and Data Processing Specialist | Underwriter, Health & Personal Accident Department at Allianz European Reliance | Christos handled data processing and led the model evaluation, ensuring the model's performance using accuracy and precision metrics while minimizing bias. |
| 4 | Emmanouil (Manos) Moschogiannis | Model Developer and NLP Engineer | IT Public Tender Consultant at TCS Toolbox | Manos focused on building and training the LSTM model and handled data preprocessing tasks like tokenization and normalization, preparing the dataset for analysis. |

# 7 Timeline and Milestones

| Stages | Description | W1 | W2 | W3 | W4 | W5 | W6 | W7 | W8 |
|---|---|---|---|---|---|---|---|---|---|
| Stage 1: Data Collection and Exploration | Relevant datasets were gathered, and initial exploration was performed to assess data suitability for sentiment analysis. | | | | | | | | |
| Stage 2: Data Preprocessing | The dataset underwent cleaning, tokenization, normalization, and stopword removal to prepare it for model training. | | | | | | | | |
| Stage 3: Model Development and Training | The LSTM model was developed and trained, with several iterations run to optimize its accuracy in classifying sentiment. | | | | | | | | |
| Stage 4: Model Evaluation and Refinement | The model was evaluated using performance metrics such as accuracy, precision, and recall. Refinements were made based on the results. | | | | | | | | |
| Stage 5: Final Report and Presentation | The final report was compiled, and the results were presented. All deliverables were completed on time. | | | | | | | | |