Abbottabad University of Science & Technology

SOFTWARE REQUIREMENTS SPECIFICATION

(SRS DOCUMENT)

For

< Social Media Data Analysis >

Submitted To	**Sir Jamal Abdul Ahad**
Submitted By	**Ayesha Jadoon**
Roll no	**14638**
Subject	**Data Structure and Algorithm**
Date	**26/12/2024**

Table of Contents

1. Introduction

1.1 Purpose

This Software Requirements Specification (SRS) document outlines the requirements for the **Social Media Data Analysis** project. The system processes and analyzes **social media datasets** (e.g., **Sentiment140** and others) to classify the sentiment of each entry as **Positive**, **Negative**, or **Neutral**. The project applies **Data Structures** like **hashmaps** for word frequency analysis, **sorting algorithms** for sentiment sorting, and **machine learning algorithms** like **Logistic Regression** for sentiment classification. The system is designed to work with multiple datasets, providing flexibility to adapt to different types of social media data.

1.2 Project Scope

1.2.1 Scope Definition

The **Social Media Data Analysis** system will:

- Preprocess tweets or posts from various datasets, including Sentiment140 and others.
- Classify the sentiment of each entry as Positive, Negative, or Neutral using TextBlob and Logistic Regression.
- Perform hashing to calculate word frequencies and visualize the most frequent terms.
- Apply sorting algorithms to sort entries based on sentiment values.
- Generate visualizations such as sentiment distribution charts, word clouds, and top 10 frequent words.
- Allow flexibility for users to input different social media datasets for analysis.

1.2.2 Core Features

- **Text Preprocessing**: Clean and preprocess any dataset by removing unwanted characters, stopwords, and applying stemming.
- **Sentiment Classification**: Classifying sentiment using TextBlob and training a Logistic Regression model for sentiment prediction.

- **Word Frequency Analysis**: Calculate word frequencies using hashmaps (`defaultdict`).
- **Sorting by Sentiment**: Sorting entries based on sentiment scores using efficient sorting algorithms.
- **Visualization**: Generating sentiment distribution, word frequency, and word cloud visualizations.
- **Dataset Flexibility**: The system is capable of processing various datasets with different column names for sentiment and text data.

1.2.3 Subsequent Enhancements

- Real-time sentiment analysis via **Twitter API**.
- Deployment of a web interface for interactive analysis and visualization.
- Expansion to handle multi-language datasets.

1.3 References

- **Dataset**: Sentiment140 Dataset from Kaggle (available at: Kaggle - Sentiment140))
- **Libraries**: `pandas`, `sklearn`, `TextBlob`, `matplotlib`, `seaborn`, `wordcloud`, `nltk`

2. Overall Description

2.1 Product Perspective

This system is a standalone application that processes, analyzes, and visualizes **social media datasets**, including **Sentiment140** and others. It is capable of classifying sentiment, performing word frequency analysis using **hashmaps**, sorting data based on sentiment, and generating various visual insights. The system is designed to be adaptable for different datasets, making it versatile in handling various social media data types.

2.2 User Classes and Characteristics

- **Data Scientists/Researchers**: Interested in sentiment analysis for academic or research purposes, using multiple datasets.

- **Marketers/Brand Managers**: Interested in understanding public sentiment and feedback trends across various platforms.
- **Casual Users**: Users who want to explore sentiment analysis of social media data, adaptable to various datasets.

2.3 Operating Environment

The software will run on any platform supporting Python 3.x with the necessary libraries:

- OS: Windows, macOS, or Linux
- Python 3.x
- Required Libraries: `pandas`, `sklearn`, `matplotlib`, `seaborn`, `TextBlob`, `wordcloud`, `nltk`

2.4 Design and Implementation Constraints

- The system should be capable of processing datasets of up to **1.6 million rows** efficiently.
- Data will be processed via a **command-line interface (CLI)**.
- The system is designed to accept any dataset in **CSV format**.

2.5 Assumptions and Dependencies

- The system assumes that the dataset provided contains at least a **text column** (for the content of posts/tweets) and a **sentiment column** (if available).
- The system is adaptable to handle different formats of social media data and automatically detect or allow users to specify the text and sentiment columns.

3. System Features

Feature 1: Text Preprocessing (Algorithm - String Processing)

- **Description**: Clean the text data (tweets/posts) by removing non-alphabetical characters, stop words, and applying stemming.
- **Data Structures**: **List** for storing words, **Set** for stopwords.

- **Functional Requirements**:
    - The text will be processed using regex to remove all non-alphabetical characters.
    - **Stopwords** will be removed using a **hashset** for **O(1)** lookups.
    - Words will be stemmed using **Porter Stemmer**.

Feature 2: Sentiment Classification (Algorithm - Classification)

- **Description**: Classify the sentiment of each text entry (Positive, Negative, Neutral).
- **Algorithms**: **TextBlob** for polarity analysis and **Logistic Regression** for classification.
- **Functional Requirements**:
    - Sentiment classification will be based on **TextBlob**'s polarity.
    - **Logistic Regression** will be used to train a model for sentiment classification.

Feature 3: Word Frequency Analysis (Data Structure - Hashing)

- **Description**: Calculate the frequency of words in the dataset using **hashmaps** (`defaultdict`).
- **Data Structure**: **defaultdict(int)** for storing word frequencies.
- **Functional Requirements**:
    - The frequency of each word will be counted using a **hashmap** for **O(1)** insertions.
    - The top 10 most frequent words will be identified and displayed.

Feature 4: Sorting by Sentiment (Algorithm - Sorting)

- **Description**: Sort the entries by sentiment score using an efficient sorting algorithm.
- **Algorithms**: **Merge Sort** or **Quick Sort** with custom comparators based on sentiment.
- **Functional Requirements**:
    - Sort entries based on sentiment using a **custom sorting function** for **O(n log n)** performance.

Feature 5: Model Training and Evaluation (Algorithm - Machine Learning)

- **Description**: Train a **Logistic Regression** model and evaluate its performance using metrics like **accuracy**, **classification report**, and **confusion matrix**.
- **Algorithm**: **Logistic Regression** with **TF-IDF** features.
- **Functional Requirements**:
  - The model will be trained using the **Logistic Regression** algorithm on the processed dataset.
  - Performance will be evaluated using **accuracy**, **classification report**, and **confusion matrix**.

4. Data Requirements

- **Input Data**: The system is designed to accept **any dataset in CSV format** containing at least the following:
  - `target`: Sentiment label (e.g., 0 = Negative, 2 = Neutral, 4 = Positive)
  - `id`: Unique identifier for each entry
  - `text`: Content of the tweet/post (the text to be analyzed)
- **Output Data**:
  - A CSV file with additional columns for `processed_text` and `sentiment`.
  - Graphical outputs saved as PNG files (e.g., sentiment distribution, word cloud, confusion matrix).

5. External Interface Requirements

5.1 User Interfaces

- **Input**: The user provides the file path for the social media dataset (CSV format).
- **Output**: The system will generate and save graphical outputs such as sentiment distribution, word frequency charts, confusion matrix as PNG files.

5.2 Software Interfaces

- **Libraries**: The system uses **pandas** for data manipulation, **TextBlob** for sentiment analysis, **sklearn** for model training and evaluation, **matplotlib** and **seaborn** for visualizations, and **nltk** for text preprocessing.

6. Quality Attributes

6.1 Performance

- The system must process any dataset (including large datasets with up to 1.6 million entries) efficiently, completing data processing and analysis in less than **30 minutes**.

6.2 Reliability

- The system should handle missing or invalid data gracefully, without crashing, and should be able to process large datasets without memory issues.

6.3 Usability

- The CLI should guide the user through data input, processing, sentiment analysis, and visualization, providing feedback and results clearly.

6.4 Security

- The system does not store any data permanently and ensures that input data is processed in-memory only.

6.5 Maintainability

- The system's code will be modular and well-documented, making it easy to update or extend for future features, including the ability to handle new types of social media datasets.