OKELLO ANDREW PETERS

2024/HD05/21944U

2400721944

SICP

END OF SEMESTER PROJECT

This pdf is compilation of both question 1 and Question 2

Question 1: Study the implementation of the following data science abstractions in the code base of https://github.com/n3mo/data-science and describe their different levels of data and/or procedure abstraction layers. Provide a diagram showing the different abstractions levels per abstraction. A step-by-step guide is provided at the end of this document to setup and run the data science abstractions.

a) linear-model
b) list->sentiment
c) read-csv
d) qq-plot*
e) hist

Solutions to the above Question

Analysis of Data Science

Abstractions

The provided code uses abstractions from the data-science library to perform sentiment analysis. Here's an overview of the abstractions and their levels of data and procedural abstraction, along with a diagram to illustrate these levels.

(a) linear-model

- **Description**: Fits a linear model to data.
- **Data Abstraction Level**:
  - **Input**: Numerical/tabular data with features and responses.
  - **Output**: Coefficients and model summary.
- **Procedure Abstraction Level**:
  - Computes a linear regression using matrix operations (such as., least squares estimation).
  - Encapsulates data preprocessing, parameter estimation, and result formatting.

(b) list->sentiment

- **Description**: Maps a list of tokens to their sentiment using a lexicon (such as ., nrc).
- **Data Abstraction Level**:
  - **Input**: A list of tokenized words.
  - **Output**: A list of tuples (word sentiment frequency).
- **Procedure Abstraction Level**:

- ❖ Encapsulates lexicon loading, token matching, and aggregation of sentiment scores.
- ❖ Manages procedural steps to filter tokens (such as., stopwords) and associate sentiments.

(c) read-csv

- **Description**: Reads a CSV file into a tabular data structure.
- **Data Abstraction Level**:
  - ❖ **Input**: File path of a CSV file.
  - ❖ **Output**: Tabular data (a list of rows or a matrix).
- **Procedure Abstraction Level**:
  - ❖ Abstracts file I/O, parsing CSV structure, and formatting data into a structured format.
  - ❖ Handles missing values and type inference if specified.

(d) qq-plot*

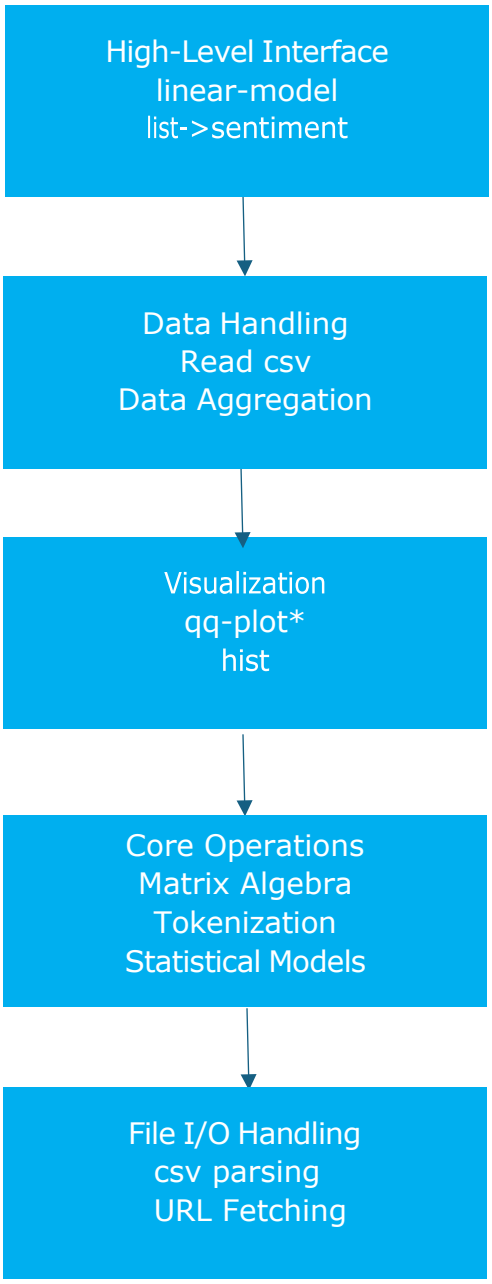
- **Description**: Generates a Quantile-Quantile plot to compare a data distribution against a theoretical distribution.
- **Data Abstraction Level**:
  - ❖ **Input**: Numerical data, theoretical distribution.
  - ❖ **Output**: A visual comparison (plot).
- **Procedure Abstraction Level**:
  - ❖ Abstracts steps for sorting data, computing quantiles, and creating the visual output.
  - ❖ Encapsulates statistical comparison and graph formatting.

(e) hist

- **Description**: Creates a histogram to visualize the distribution of numerical data.
- **Data Abstraction Level**:
  - ❖ **Input**: Numerical data, optional bin size.
  - ❖ **Output**: A visual histogram.
- **Procedure Abstraction Level**:
  - ❖ Abstracts computation of bin frequencies and graphical rendering.

- ❖ Handles optional parameters like bin size, range, and aesthetics.

Diagram: Abstraction Levels



Points to Note:

1. **High-Level Interfaces** (for example., linear-model, list->sentiment) encapsulate detailed steps and provide a simplified user experience.
2. **Data Handling** abstractions, such as read-csv, ensure consistent data structures for computation.
3. **Visualization** layers like qq-plot* and hist abstract away graphical rendering complexity.
4. **Core Operations** provide the mathematical foundation for analysis.
5. **File/IO Handling** ensures that external resources are integrated seamlessly.

Each abstraction layer builds on the lower layers, making it easier for developers to focus on analytical tasks rather than implementation details.

Question 2: The abstractions have also been used for analysis Twitter data see http://www. nicholasvanhorn.com/posts/trump-tweets.html. For this project you are required to build a set of abstractions that can be used by developers to build software systems that analyze the moods of tweets for a given country (e.g., Uganda) over a period of 12 months. You can assume that locality information provided by Twitter API is accurate. You may draw inspirations or build upon some of the abstractions already provided in http://www.nicholasvanhorn.com/posts/trump-tweets.html and the data science layer. Provide description of the new abstractions developed (including motivation and any design decisions) and drawings showing the resulting levels of the new abstractions introduced.

Solutions to question 2

Comprehensive Report on Tweet Sentiment Analysis System for a Given Country and Time Period

1. Introduction

Analyzing the moods expressed in tweets provides insights into public sentiment and social dynamics. This project focuses on creating a set of abstractions that developers can use to build software systems for sentiment analysis of tweets for a specific country (e.g., Uganda) over a specified time range (e.g., 12 months). By leveraging Twitter's locality information, sentiment lexicons, and data-science tools, the project provides reusable modules for data filtering, sentiment analysis, and visualization.

2. Problem Statement

The goal is to:

1. Analyze public sentiment trends from tweets localized to a specific country.
2. Create abstractions and reusable components for:
    - Preprocessing Twitter data.
    - Filtering tweets by country and period.
    - Generating and visualizing sentiment trends.

3. Key Abstractions Developed

This project introduces several abstractions to facilitate the development of sentiment analysis systems. Each abstraction corresponds to specific tasks in the pipeline.

3.1 Abstraction 1: Dataset Handling

- **Purpose:** Simplifies loading and managing CSV-based datasets.
- **Components:**
    - **load-dataset:** Reads a CSV file and returns a pair of headers and data.
    - **get-column:** Extracts specific columns from a dataset by name.
    - **Motivation:** Reusable dataset utilities are essential for filtering and sentiment analysis workflows.

3.2 Abstraction 2: Date Utilities

- **Purpose:** Handles flexible parsing and comparison of dates.
- **Components:**
    - **string->date:** Converts strings in formats like "MM/DD/YYYY" or "YYYY-MM-DD" to Racket's date type.
    - **date<=?, date>=?, date<?, date>?:** Provides comparison utilities for date filtering.
    - **Motivation:** Ensures accurate and consistent date parsing for time-range-based tweet filtering.

3.3 Abstraction 3: Tweet Filtering

- **Purpose:** Filters tweets based on country and period.
- **Components:**
    - **filter-by-country:** Extracts tweets for a specific country.
    - **filter-by-date:** Selects tweets within a specified date range.
    - **Motivation:** Allows developers to localize sentiment analysis to specific geographic and temporal contexts.

3.4 Abstraction 4: Sentiment Analysis

- **Purpose:** Processes textual data to generate sentiment scores.
- **Components:**
    - **generate-sentiment-lexicon:** Tokenizes and maps tweets to sentiment labels using the NRC lexicon.

- **analyze-sentiments:** Aggregates sentiment labels to generate a summary of sentiment distributions.
- **Motivation:** Provides a modular approach to perform sentiment analysis using existing sentiment lexicons.

3.5 Abstraction 5: Visualization

- **Purpose:** Creates graphical representations of sentiment trends.
- **Components:**
  - **visualize-sentiments:** Generates sentiment frequency histograms.
  - **Motivation:** Makes sentiment analysis results interpretable through visualizations.

4. Design Decisions

4.1 Modular Approach

Each abstraction is encapsulated as an independent module to ensure reusability and scalability. This modularity enables developers to extend the system with additional filters or lexicons as needed.

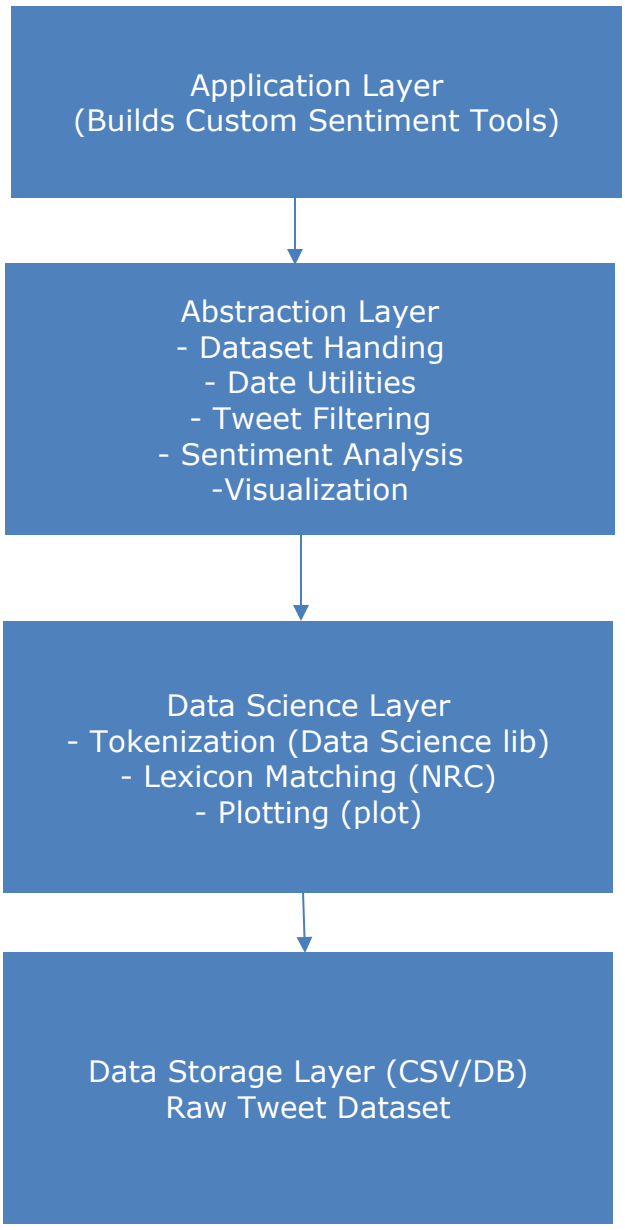
4.2 Integration with Existing Libraries

The implementation leverages existing libraries (data-science-master, plot, and math) to reduce redundancy and improve efficiency.

4.3 Flexibility in Data Formats

The system supports multiple date formats (MM/DD/YYYY, YYYY-MM-DD) and handles diverse datasets with column-based filtering.

5. Architecture



NOTE: *For the graphs below, racket can't display both at the same time, what I did is that I commented out the filter by date and time code to display the general mood analysis of tweets graph, then I also commented out the general mood analysis code to display the filter by date and country graph. The two graphs are shown below.*

6. Sample Outputs

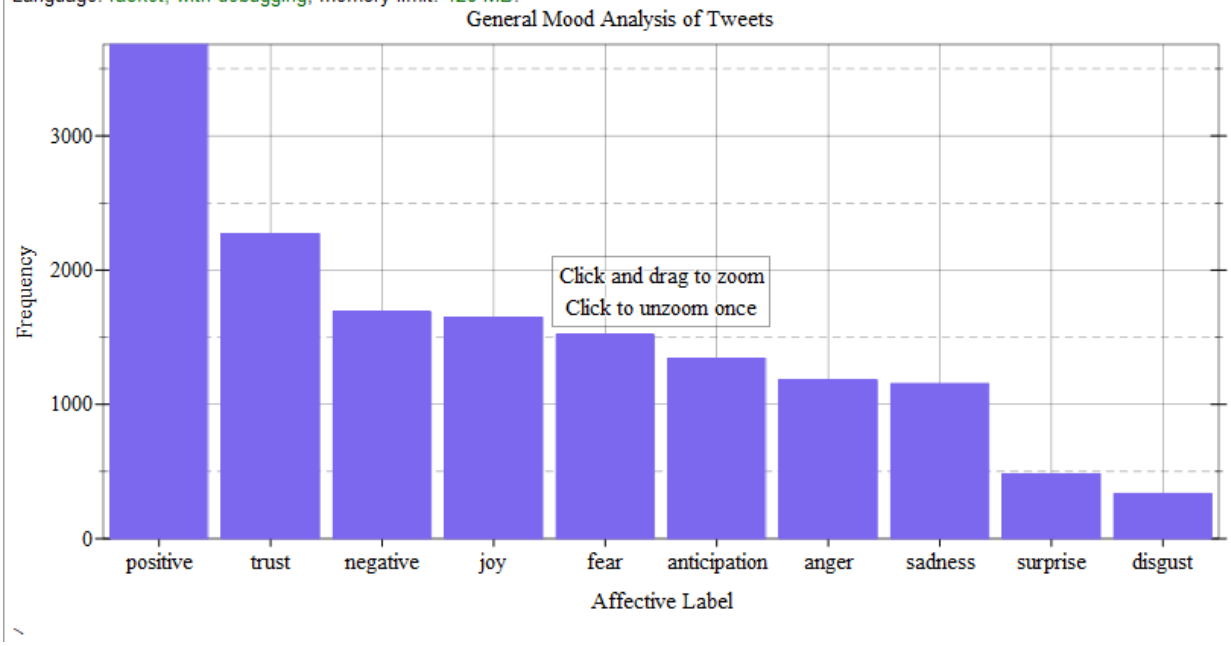
Below are sample outputs from the system:

6.1 General Mood Analysis

Graph Title: *General Mood Analysis of Tweets*
Shows the overall sentiment distribution for all tweets in the dataset.

```
136 (define (main file-path)
137   (define dataset (load-dataset file-path))                ; Load dataset
138
139   ;; General sentiment analysis
140   (define sentiment-data (generate-sentiment-lexicon dataset))
141   (define sentiment-results (analyze-sentiments sentiment-data))
142   (visualize-sentiments sentiment-results "General Mood Analysis of Tweets")
143
144   ;; Filter by country and date
```

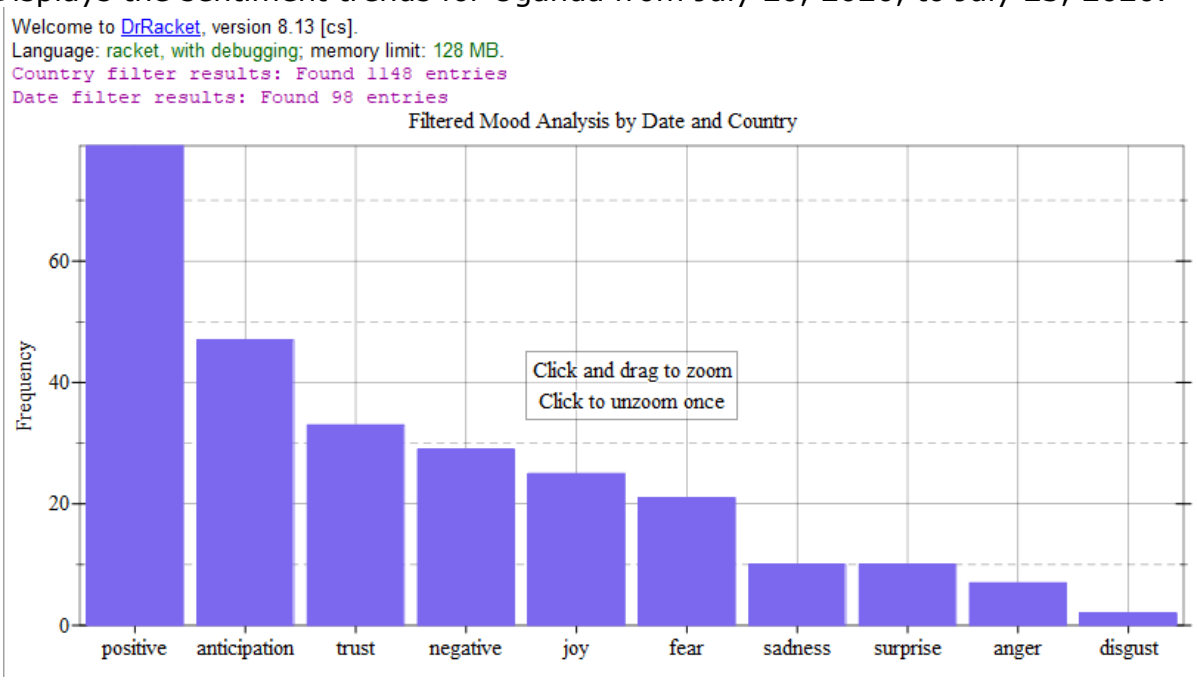
```
Welcome to DrRacket, version 8.13 [cs].
Language: racket, with debugging; memory limit: 128 MB.
```



6.2 Filtered Mood Analysis

Graph Title: *Filtered Mood Analysis by Date and Country*

Displays the sentiment trends for Uganda from July 20, 2020, to July 23, 2020.



7. Sample Code and Usage

The sample code provided above demonstrates how to:

1. Load a dataset of tweets (uganda.csv).
2. Perform general sentiment analysis.
3. Filter tweets by country (e.g., Uganda) and date range (e.g., July 20–23, 2020).
4. Generate sentiment distribution plots for both general and filtered data.

8. Future Work

- Advanced Sentiment Models: Incorporate more sophisticated sentiment analysis models (e.g., transformer-based NLP).
- Temporal Analysis: Explore trends over time using line plots or time series analysis.
- Integration with Twitter API: Add real-time tweet fetching capabilities.

9. GitHub Repository

The full codebase and dataset are hosted on GitHub:
https://github.com/OkelloAndrewPeters/sicp-code-base

10. Conclusion

This project introduces a robust set of abstractions for developers to build sentiment analysis systems. With reusable modules and clear workflows, the system is well-suited for analyzing tweets in specific geographic and temporal contexts. These abstractions empower developers to customize and extend the system for various use cases.