**Reuters Sentiment Analysis**

**Submitted By –**

**Himanshi Verma**

**Algorithm –**

Below listed are the steps that I have executed in my program in order to achieve the results -

* Connect to MongoDB: Establish a connection to the MongoDB database.
* Fetch News Titles: Retrieve news titles from the MongoDB collection.
* Create Analyzer: Initialize a SentimentAnalyzer object with paths to files containing positive and negative words.
* Analyze Sentiments:
  + For each news title fetched from MongoDB:
    - Create a bag of words for the title.
    - Calculate the sentiment score based on the bag of words.
    - Find matches of positive and negative words in the title.
  + Store the sentiment scores, matches, and polarities in a SentimentAnalysisResult object.
* Save Results to CSV:
  + Use a CSVWriter object to write the sentiment analysis results to a CSV file.
  + Write the headers (e.g., "News#, Title, Match, Score, Polarity") to the CSV file.
  + For each news title, write the corresponding sentiment analysis data (e.g., news number, title, matches, score, polarity) to the CSV file.
* Output: Display a message confirming that the results have been saved in the CSV file.

**Code Explanation –**

Code execution begins from the file Problem3.java, main method -

**A screenshot of a computer

Description automatically generated**

Fig.1: Execution starts from main() method

After executing the main file the logs for each step can be seen on the console . The results will be written in a csv file named news\_sentiment\_analysis.csv. Currently I have kept the location for csv file inside the project only.

A screenshot of a computer

Description automatically generated

Fig. 2: Successful execution of program

A screenshot of a computer

Description automatically generated

Fig. 3: news\_sentiment\_analysis.csv generated

A screenshot of a computer

Description automatically generated

Fig. 4: Sentiment analysis csv

**Tests -**

I have written three test cases to verify positive, neutral and negative polarity. These testcases are present in Problem3Test.java which can be run to verify test cases.

After the execution of test cases the results can be seen on the console if the test has passed or failed.

A screenshot of a computer program

Description automatically generated

Fig. 5: Test case execution output console

**My code adheres to SOLID principle –**

**Single Responsibility Principle (SRP):** Each class, such as SentimentAnalyzer, SentimentAnalysisResult, and PositiveNegativeWordLists, has a clear and single responsibility. For instance, SentimentAnalyzer analyzes sentiments, SentimentAnalysisResult holds analysis results, and PositiveNegativeWordLists manages positive and negative word lists.

**Open/Closed Principle (OCP):** The code is designed so that you can add new features or extend existing ones without changing the existing code. For example, you can extend the functionality of the SentimentAnalyzer class by adding new methods or subclasses without modifying its original code.

**Liskov Substitution Principle (LSP):** The code allows you to use different implementations of a class interchangeably. For instance, SentimentAnalyzer accepts any implementation of PositiveNegativeWordLists, enabling you to substitute different implementations without affecting the client code.

**Interface Segregation Principle (ISP):** Each interface, such as PositiveNegativeWordLists, is small and focused on specific tasks. This allows classes to implement only the methods they need, avoiding unnecessary dependencies. For example, if a class only needs to check whether a word is positive or negative, it can implement the isPositive and isNegative methods without implementing other methods.

**Dependency Inversion Principle (DIP):** The code relies on abstractions rather than concrete implementations. For example, SentimentAnalyzer depends on the PositiveNegativeWordLists interface rather than specific implementations, making it easier to switch implementations or inject dependencies. This helps decouple classes and reduces dependencies on concrete implementations.

**Consideration –**

Before code execution it is required to add the VM Options **-Djdk.tls.client.protocols=TLSv1.2**  
This is to make mongo compatible with java 11  
(Note - the issue doesn't exist in version > jdk 13.0.5)

**References -**

1. Github Gist. Positive words[Online]. Available: <https://gist.github.com/mkulakowski2/4289437>
2. Github Gist. Negative words[Online]. Available: [gist.githubusercontent.com/mkulakowski2/4289441/raw/dad8b64b307cd6df8068a379079becbb3f91101a/negative-words.txt](https://gist.githubusercontent.com/mkulakowski2/4289441/raw/dad8b64b307cd6df8068a379079becbb3f91101a/negative-words.txt)