

CSCI-5408

**DATA MANAGEMENT,
WAREHOUSING, & ANALYTICS**

ASSIGNMENT – 2

Banner ID: B00977669

GitLab Assignment Link:

https://git.cs.dal.ca/saji/csci5408_w24_b00977669_christin_saji

Problem 1A: Reuter News Data Reading & Transformation and storing in **MongoDB**.

Flowchart:

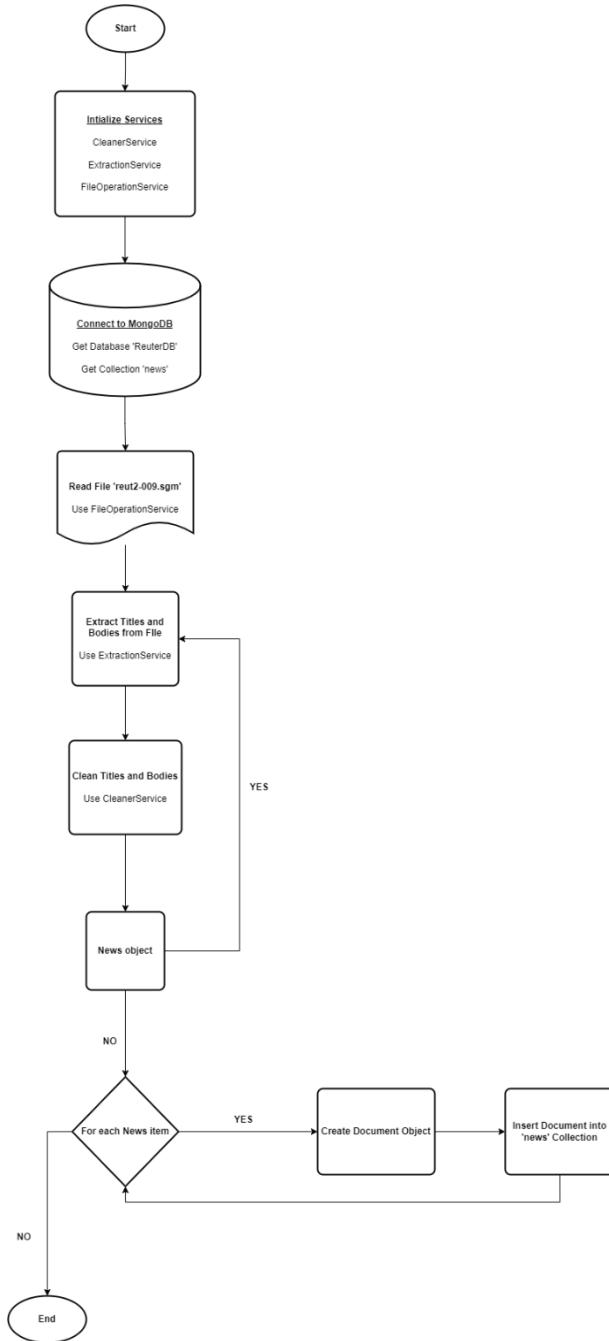


Figure 1 Flowchart of Reuters Data extracting, cleaning, and transformation.

Algorithm:

Step 1: Initialize ‘CleanerService’ for cleaning text data, ‘ExtractionService’ for use during data extraction, and ‘FileOperationService’ for handling file operations.

Step 2: Establish connection to MongoDB [1]. Access the ‘ReuterDb’ database within MongoDB and retrieve the ‘news’ collection from the database to store the news items.

Step 3: Read the data file using ‘FileOperationService’ to read “reut2-009.sgm” file’s content into memory.

Step 4: Extract, clean news items and create a ‘News’ object using ‘ExtractionService’, and ‘CleanerService’. Loop over till the end of the file.

Step 5: For each ‘News’ object, iterate over the list and create ‘Document’ objects for MongoDB with the title and body data.

Step 6: Insert ‘Documents’ object into the ‘news’ collection in the MongoDB database.

Functional Testing:

Created a ‘ReuterDb’ database and ‘news’ collection within the MongoDB. Currently, it has no documents inside the database.

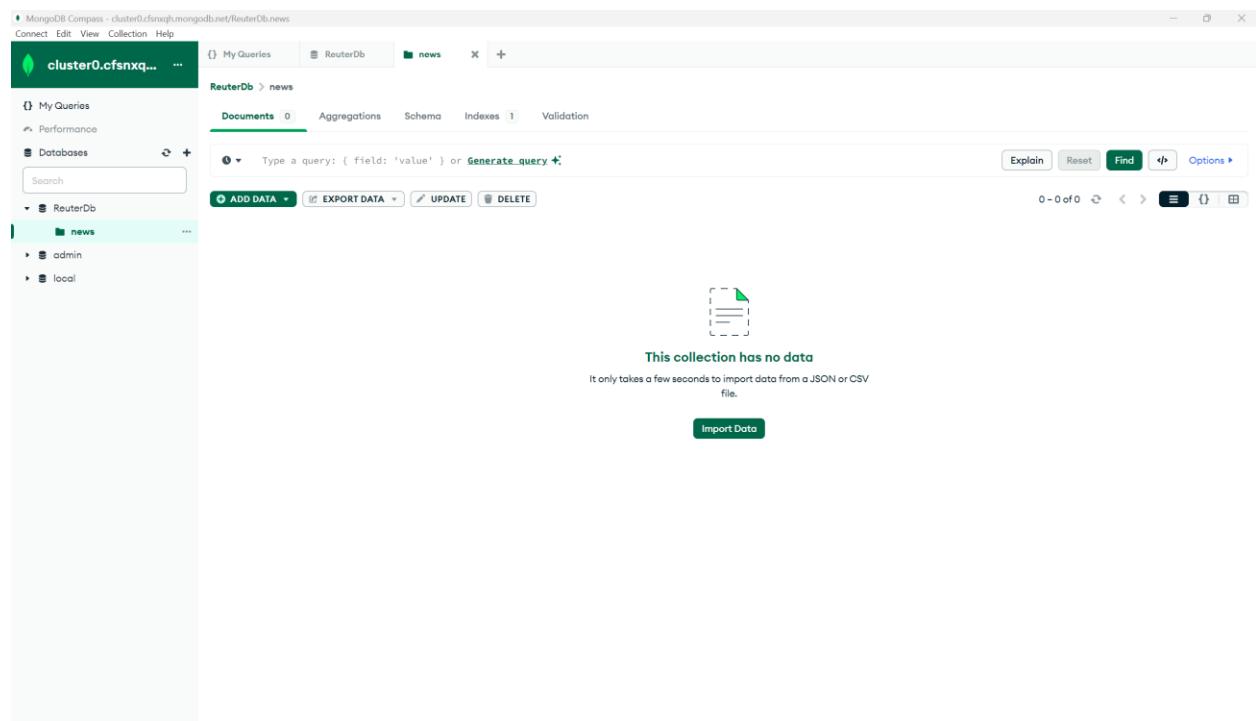


Figure 2 ReuterDb database on MongoDB

Extracted the raw data from the file to a ‘newsList’ variable.

The screenshot shows the IntelliJ IDEA interface with the code editor open to the `FileOperationService.java` file. The code is as follows:

```

15    /**
16     * Processes the given file path to extract News object
17     * @param filePath path to news data file
18     * @return list of News object
19     */
20
21    public List<News> processFile(String filePath) {
22        filePath: "reut2-009.sgm"
23        List<News> newsList = new ArrayList<>();
24        size = 0
25        StringBuilder contentBuilder = new StringBuilder();
26        contentBuilder: "<!DOCTYPE lewis SYSTEM \"lewis.dtd\"><REUTERS TOPICS=\"NO\" LEWISPLIT=\"TRAIN\""
27
28        try (BufferedReader br = new BufferedReader(new FileReader(filePath))) {
29            br: BufferedReader@2069
30            String line: Line
31            while ((line = br.readLine()) != null) {
32                br: BufferedReader@2069
33                contentBuilder.append(line).append("\n");
34            }
35
36            newsList = extractionService.parse(contentBuilder.toString());
37            contentBuilder: "<!DOCTYPE lewis SYSTEM \"lewis.dtd\"><REUTERS TOPICS=\"NO\" LEWISPLIT=\"TRAIN\""
38        } catch (Exception e) {
39            e.printStackTrace();
40        }
41
42        return newsList;
43    }

```

The code is annotated with comments explaining its purpose: it reads a file named `reut2-009.sgm`, initializes a `StringBuilder` for the document, and then reads each line of the file into the builder. Finally, it calls the `parse` method of the `extractionService` to convert the accumulated content into a list of `News` objects.

Figure 3 Extraction of raw data into variable

Title tag raw data extracted before the ‘CleanerService’ which currently has HTML entities and special characters.

The screenshot shows the IntelliJ IDEA interface with the code editor open to the `ExtractionService.java` file. The code is as follows:

```

16    /**
17     * @param cleanerService CleanerService instance
18     */
19
20    Usage
21    public ExtractionService(CleanerService cleanerService) {
22        this.cleanerService = cleanerService;
23    }
24
25    /**
26     * Parses the text and extracts news information into a list of News object
27     * @param filterData text containing multiple news entries
28     * @return list of news objects
29     */
30
31    Usage
32    public List<News> parse(String filterData) {
33        filterData: "<!DOCTYPE lewis SYSTEM \"lewis.dtd\"><REUTERS TOPICS=\"NO\" LEWISPLIT=\"TRAIN\" CGISPLIT=\"TRAIN\""
34        List<News> newsList = new ArrayList<>();
35        size = 0
36        Pattern pattern = Pattern.compile("(?s)<REUTERS(.+?)</REUTERS>|<REUTERS(.+?)</REUTERS>");
37        Matcher matcher = pattern.matcher(filterData);
38        filterData: "<!DOCTYPE lewis SYSTEM \"lewis.dtd\"><REUTERS TOPICS=\"NO\" LEWISPLIT=\"TRAIN\" CGISPLIT=\"TRAIN\""
39
40        while (matcher.find()) {
41            String reuterContent = matcher.group(1);
42            reuterContent: "TOPICS=\"NO\" LEWISPLIT=\"TRAIN\" CGISPLIT=\"TRAINING-SET\" OLDID=14522 NEWID=9001"
43            String title = extractBetweenTags(reuterContent, "TAG \"TITLE\"");
44            title: "ADVANCED MAGNETICS & ADMG IN AGREEMENT"
45            String body = extractBetweenTags(reuterContent, "TAG \"BODY\"");
46            reuterContent: "TOPICS=\"NO\" LEWISPLIT=\"TRAIN\" CGISPLIT=\"TRAINING-SET\" OLDID=14522 NEWID=9001"
47            title: cleanerService.cleanText(title);
48            title: "ADVANCED MAGNETICS & ADMG IN AGREEMENT";
49            cleanerService: CleanerService@2094
50            body = cleanerService.cleanText(body);
51
52            newsList.add(new News(title, body));
53        }
54
55        return newsList;
56    }

```

The code is annotated with comments explaining its purpose: it uses regular expressions to find `<REUTERS>` blocks in the input `filterData`. For each block, it extracts the `title` and `body` (using `cleanText` from the `CleanerService`) and adds them to a list of `News` objects.

Figure 4 Title tag raw data before CleanerService

Title tag raw data after the 'CleanerService' by removing the HTML entities and special characters.

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project View:** Shows the file structure under the package `org.example.ReutReader`. The `ExtractionService.java` file is currently selected.
- Code Editor:** Displays the `ExtractionService.java` code. A search result for `cleanerService` is highlighted in blue, with multiple occurrences marked by red circles. The code includes annotations like `@param` and `@usage`.
- Search Results:** A floating search results window is open, listing 35 matches for `cleanerService` across various Java files, including `CleanerService.java`, `FileOperationService.java`, and `News.java`.
- Toolbars and Status Bar:** Standard IntelliJ toolbars and a status bar at the bottom showing "35.1 CRLF UTF-8 4 spaces".

Figure 5 Title tag raw data after CleanerService

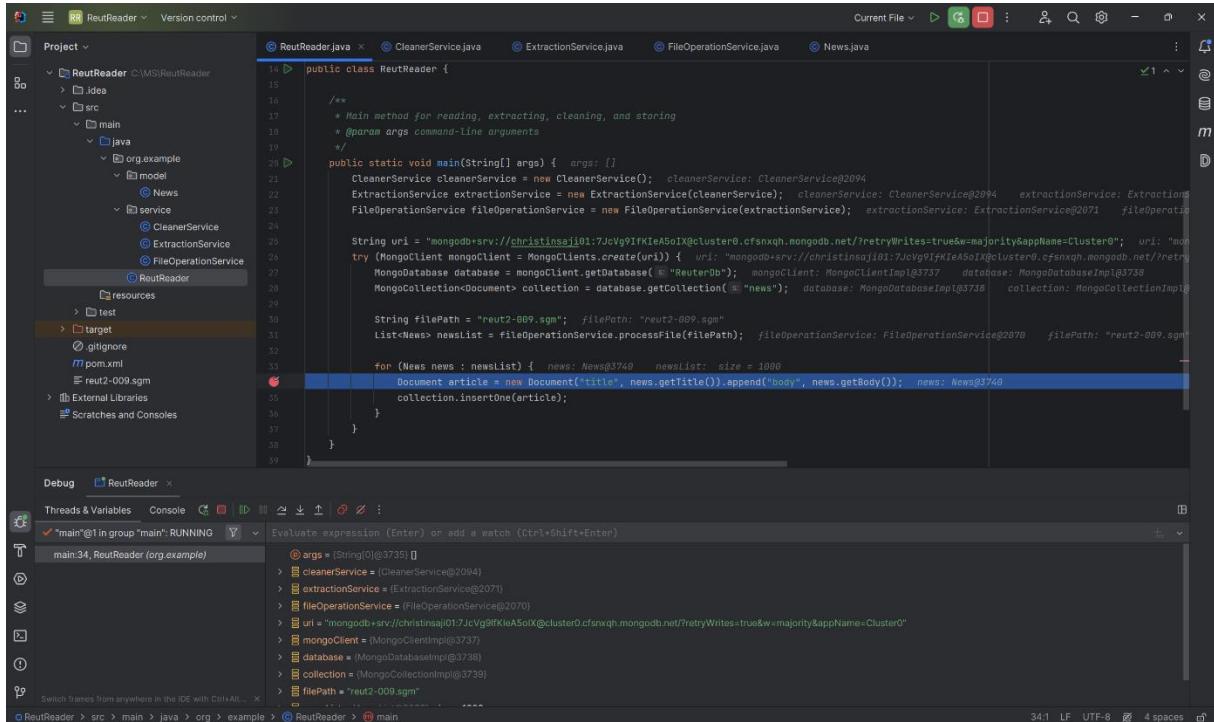
Similarly, cleaned body raw data by ‘CleanerService’.

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project View:** Shows the `ReutReader` project structure under `C:\MSI\ReutReader`. The `src/main/java/org.example.model.News` file is currently selected.
- Code Editor:** Displays the `ExtractionService.java` file. The cursor is at line 42, which contains the code `return newsList;`. A red vertical bar highlights the entire line 42, and a red horizontal bar highlights the word `newsList` in the line above.
- Toolbars and Status Bar:** The top bar shows tabs for `ReutReader`, `Version control`, and `Current File`. The status bar at the bottom right shows `57:1 CRLF UTF-8 4 spaces`.
- Bottom Navigation:** Shows the path: `ReutReader > src > main > java > org > example > service > ExtractionService > parse`.

Figure 6 Body tag raw data after CleanerService

After the extraction and cleaning process there are total of 1000 ‘newsList’. Each news is then added to the MongoDB database.



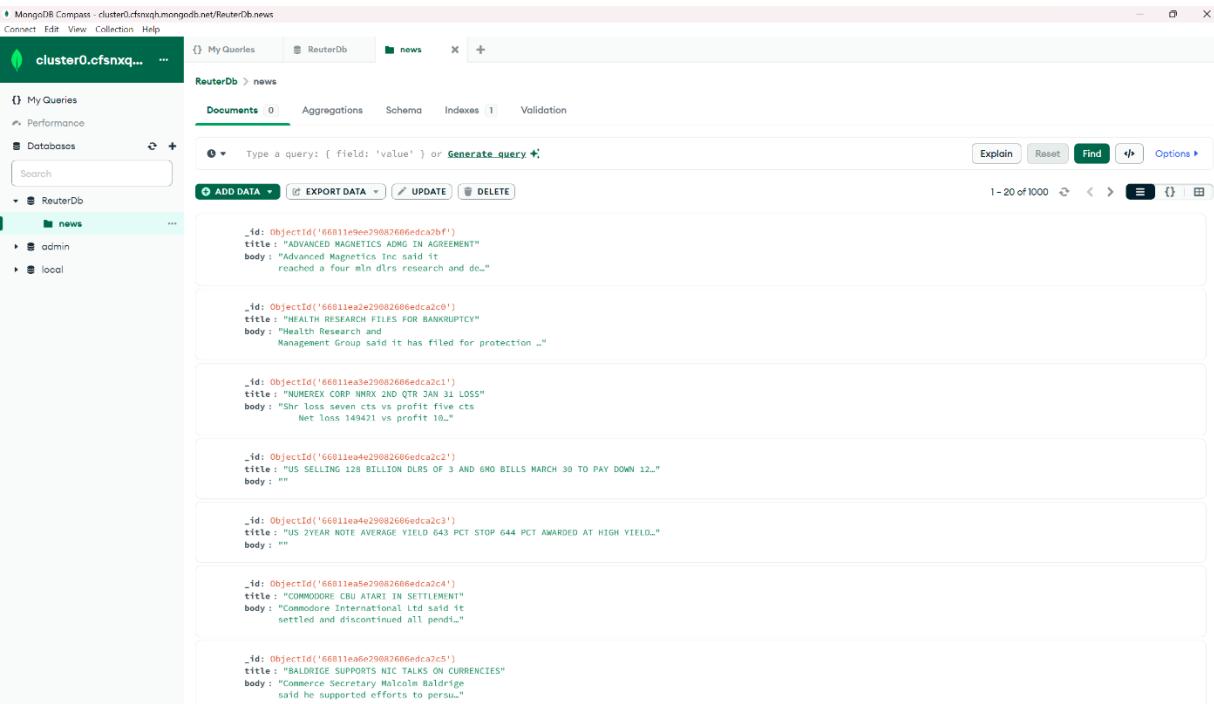
```

14  public class ReutReader {
15
16     /**
17      * Main method for reading, extracting, cleaning, and storing
18      * @param args command-line arguments
19     */
20    public static void main(String[] args) { args: []
21        CleanerService cleanerService = new CleanerService(); cleanerService: CleanerService@2094
22        ExtractionService extractionService = new ExtractionService(cleanerService); cleanerService: CleanerService@2094 extractionService: Extractions
23        FileOperationService fileOperationService = new FileOperationService(extractionService); extractionService: ExtractionService@2071 fileOperati
24
25        String url = "mongodb+srv://christina@i01:7JcVg9fKieA5oIX@cluster0.cfsnxqh.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0"; url: "mongodb+sr
26        try {
27            MongoClient mongoClient = MongoClients.create(url);
28            mongoClient = MongoClients.create(url);
29            MongoDatabase database = mongoClient.getDatabase("ReuterDb");
30            mongoClient = MongoClients.create(url);
31            database = MongoDatabaseImpl@3738;
32            collection = database.getCollection("news");
33            mongoClient = MongoClients.create(url);
34            database = MongoDatabaseImpl@3738;
35            collection = database.getCollection("news");
36            mongoClient = MongoClients.create(url);
37            database = MongoDatabaseImpl@3738;
38            collection = database.getCollection("news");
39        } catch (MongoException e) {
40            e.printStackTrace();
41        }
42
43        String filePath = "reut2-009.sgm";
44        List<News> newsList = fileOperationService.readFile(filePath);
45        fileOperationService: FileOperationService@2070 filePath: "reut2-009.sgm"
46
47        for (News news : newsList) {
48            news: News@3740
49            Document article = new Document("title", news.getTitle()).append("body", news.getBody());
50            news: News@3740
51            collection.insertOne(article);
52        }
53    }
54 }

```

Figure 7 newsList containing 1000 documents

MongoDB containing all the 1000 documents.



The screenshot shows the MongoDB Compass interface connected to a cluster. The left sidebar lists databases: 'cluster0.cfsnxqh' (selected), 'admin', and 'local'. The main area shows the 'ReuterDb' database with the 'news' collection. The 'Documents' tab is selected, displaying 1000 documents. Each document is represented by a card with fields: '_id', 'title', and 'body'. The first few documents are:

- Document 1:** _id: ObjectId('66811ea3e290082606edca2bf'), title: "ADVANCED MAGNETICS ADMG IN AGREEMENT", body: "Advanced Magnetics Inc said it reached a four mln dlr research and de..."
- Document 2:** _id: ObjectId('66811ea3e290082606edca2cf'), title: "HEALTH RESEARCH FILES FOR BANKRUPTCY", body: "Health Research and Management Group said it has filed for protection..."
- Document 3:** _id: ObjectId('66811ea3e290082606edca2c1'), title: "NUMEREX CORP INXK 2ND QTR JAN 31 LOSS", body: "Shr loss seven cts vs profit five cts Net loss 140421 vs profit 10..."
- Document 4:** _id: ObjectId('66811ea3e290082606edca2c3'), title: "US SELLING 12B BILLION DLRS OF 3 AND 6MO BILLS MARCH 30 TO PAY DOWN 12...", body: ""
- Document 5:** _id: ObjectId('66811ea3e290082606edca2c3'), title: "US 2YR NOTE AVERAGE YIELD 643 PCT STOP 644 PCT AWARDED AT HIGH YIELD...", body: ""
- Document 6:** _id: ObjectId('66811ea3e290082606edca2c4'), title: "COMMODORE CRU ATARI IN SETTLEMENT", body: "Commodore International Ltd said it settled and discontinued all pending..."
- Document 7:** _id: ObjectId('66811ea3e290082606edca2cf'), title: "BALDRIGE SUPPORTS NIC TALKS ON CURRENCIES", body: "Commerce Secretary Malcolm Baldrige said he supported efforts to persua..."

Figure 8 MongoDB database containing the processed data

Problem 1B: Reuter News Data Processing using Spark.

Explanation:

First, I prepared the program for deployment by packaging it into a JAR file named `SparkFreqCount-1.0-SNAPSHOT.jar`. This package contains all the necessary code and dependencies to run text analysis and count highest and lowest frequencies.

Next, I navigated to the Google Cloud Platform (GCP) and initiated a new project named "assignment-2." Within this project, I chose to use Dataproc, a service that lets you run Apache Spark projects easily on Google Cloud.

Creating a Dataproc cluster was my next task. This is essentially setting up a group of virtual machines on Google Cloud that work together to process data. I opted for a single node cluster, which means all tasks would be handled by a single virtual machine. This was sufficient for needs and helps to keep costs down.

For the virtual machine, I selected a configuration with 2 CPUs and 8 GB of memory, giving enough power to process our dataset. I also set the disk size to 250 GB to ensure there was plenty of space for data and any temporary files created during process.

After setting up the cluster, I accessed the virtual machine through an SSH browser window. This is like remote-controlling the virtual machine, allowing me to run commands directly on it.

With access to the machine, I uploaded three crucial files: the `SparkFreqCount` JAR package, our dataset file `reut2-009.sgm`, and a list of stop words in `stop-words.txt`. The stop words file is used to filter out common but unimportant words from our analysis.

Finally, I ran the `SparkFreqCount` program by issuing a command through the SSH window. This command instructed Dataproc to use Spark to process our dataset, clean it from any unnecessary content, exclude common stop words, and count how frequently each significant word appeared and list the 5 highest and lowest frequencies.

Algorithm:

Step 1: Initialize the ‘SparkSession’ object with the application name “SparkFreqCount” [2].

Step 2: Check for input arguments for file paths. If not provided, use default file paths for ‘dataFilePath’ and ‘stopWordsFilePath’.

Step 3: Use ‘StopWordsService’ to load stop words from the provided file path. Read all stop words from “stop-words.txt” into a ‘Set<String>’ converting each word to lowercase.

Step 4: Read the raw data text file from “reut02-009.sgm” into a ‘Dataset<Row>’.

Step 5: Clean the raw data by mapping each row to remove HTML tags, HTML entities, non-alphabetic characters, and convert text to lowercase.

Step 6: Split the cleaned data into individual words and flatten the result into a ‘Dataset<String>’ of words.

Step 7: Filter out stop words and words that do not meet length requirements.

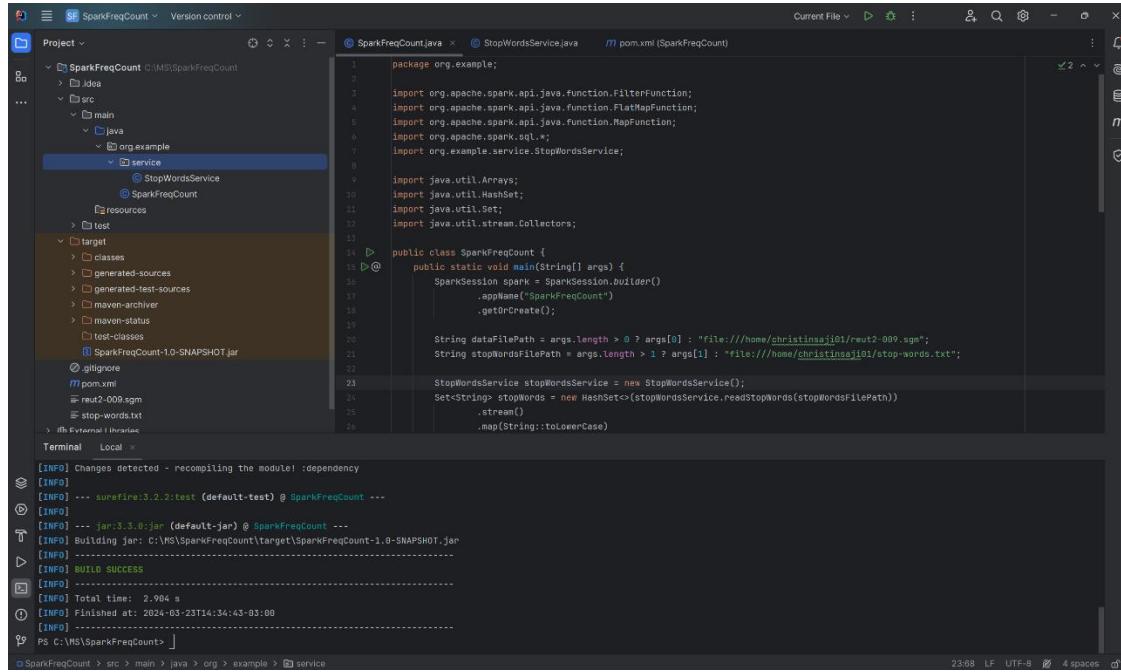
Step 8: Group the filtered words by their value and count the occurrence of each word.

Step 9: Sort the word count data in descending order to find the highest frequency words and display the top five words with the highest frequencies. Similarly, for the lowest frequencies by ordering it into ascending.

Step 10: Terminate the Spark session.

Functional Testing:

After writing the Spark program I build the jar file named SparkFreqCount-1.0-SNAPSHOT.jar



The screenshot shows an IDE interface with a project structure on the left and code editor on the right. The code editor displays Java code for a Spark application named SparkFreqCount. The terminal below shows the build process and success message.

```
package org.example;

import org.apache.spark.api.java.function.FilterFunction;
import org.apache.spark.api.java.function.FlatMapFunction;
import org.apache.spark.api.java.function.MapFunction;
import org.apache.spark.sql.*;
import org.example.service.StopWordsService;

import java.util.Arrays;
import java.util.HashSet;
import java.util.Set;
import java.util.stream.Collectors;

public class SparkFreqCount {
    public static void main(String[] args) {
        SparkSession spark = SparkSession.builder()
            .appName("SparkFreqCount")
            .getOrCreate();

        String dataFilePath = args.length > 0 ? args[0] : "file:///home/christinasj101/reut2-009.agm";
        String stopWordsFilePath = args.length > 1 ? args[1] : "file:///home/christinasj101/stop-words.txt";

        StopWordsService stopWordsService = new StopWordsService();
        Set<String> stopWords = new HashSet<>(stopWordsService.readStopWords(stopWordsFilePath))
            .stream()
            .map(String::toLowerCase);
    }
}
```

```
[INFO] Changes detected - recompiling the module! :dependency
[INFO]
[INFO] --- surefire:3.2.2:test (default-test) @ SparkFreqCount ---
[INFO] ...
[INFO] --- jar:3.3.0:jar (default-jar) @ SparkFreqCount ---
[INFO] Building jar: C:\MS\SparkFreqCount\target\SparkFreqCount-1.0-SNAPSHOT.jar
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 2.904 s
[INFO] Finished at: 2024-03-23T14:34:43+03:00
[INFO] -----
PS C:\MS\SparkFreqCount>
```

Figure 9 Successful build of SparkFreqCount jar file

In the GCP cloud, I created a project named assignment-2 then selected Dataproc.

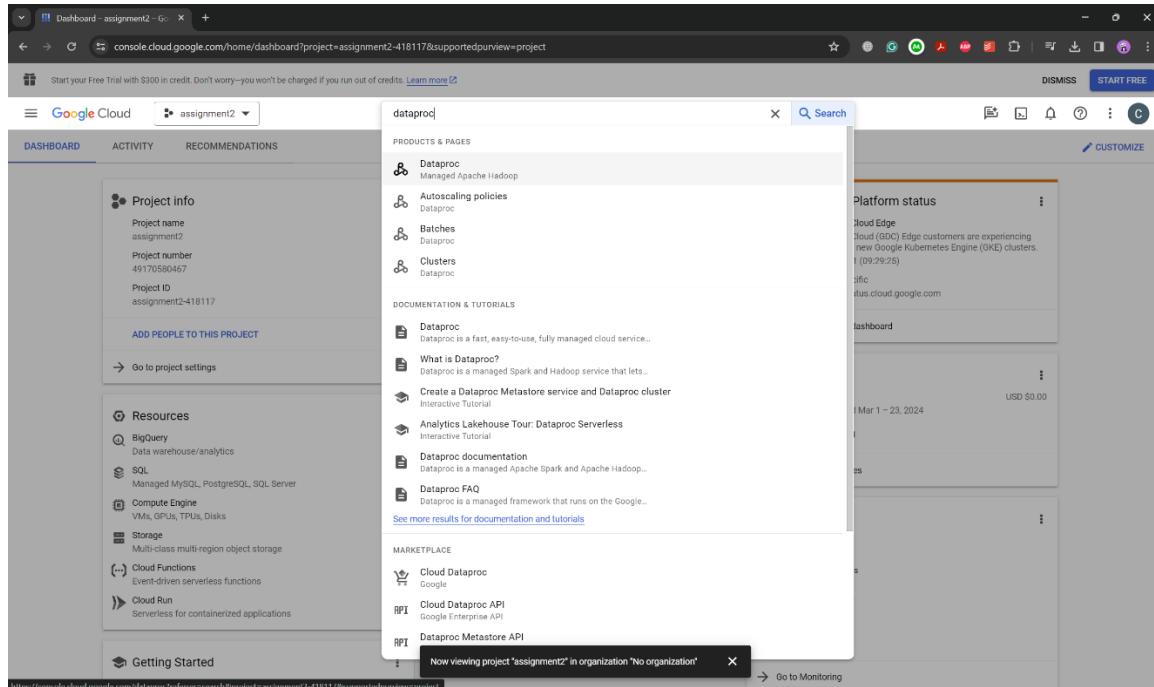


Figure 10 Under assignment-2 selected Dataproc

Created a Cloud Dataproc Cluster.

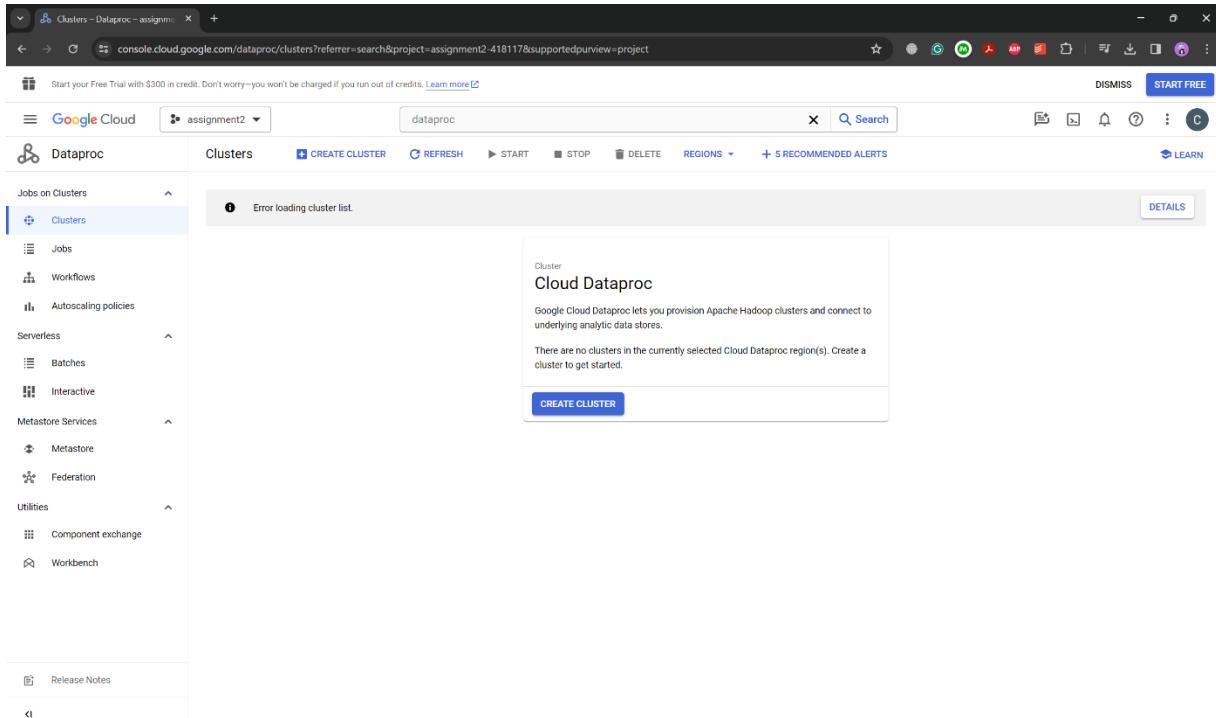


Figure 11 Created a Dataproc Cluster

Created cluster on Compute Engine.

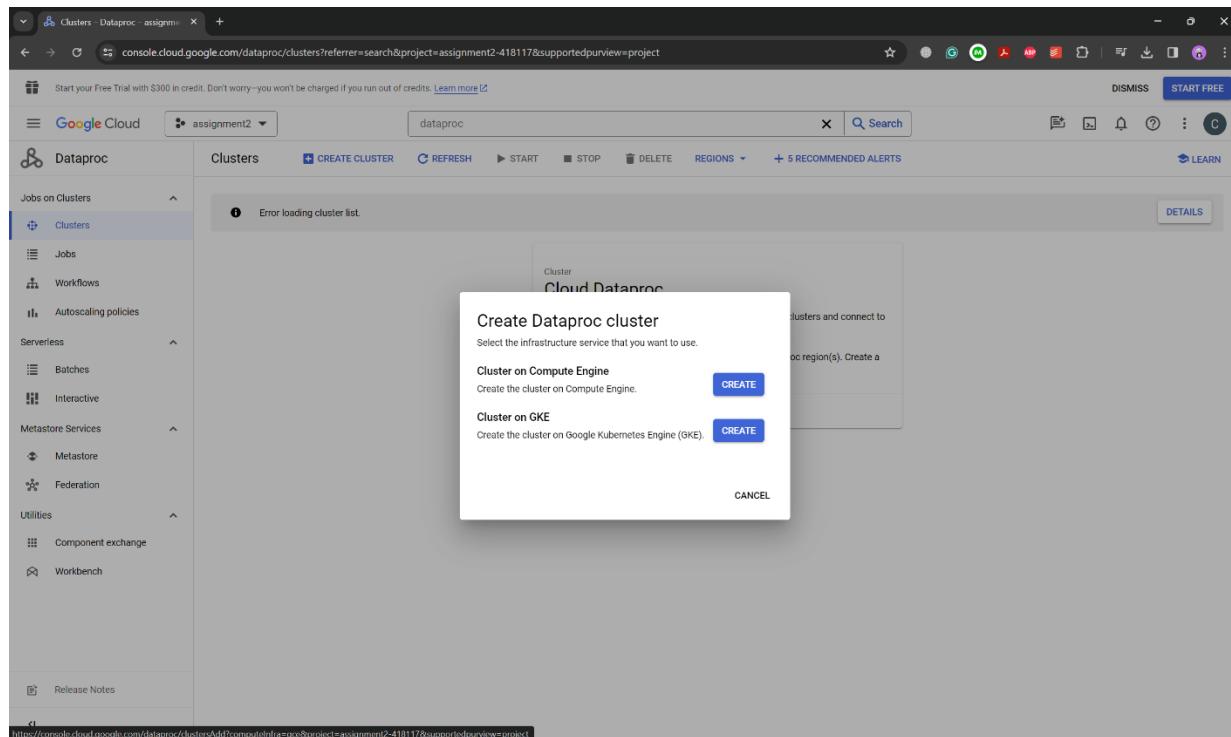


Figure 12 Created cluster on Compute Engine

Renamed the cluster name to sparkfreqcount-cluster and selected single node option for cluster type.

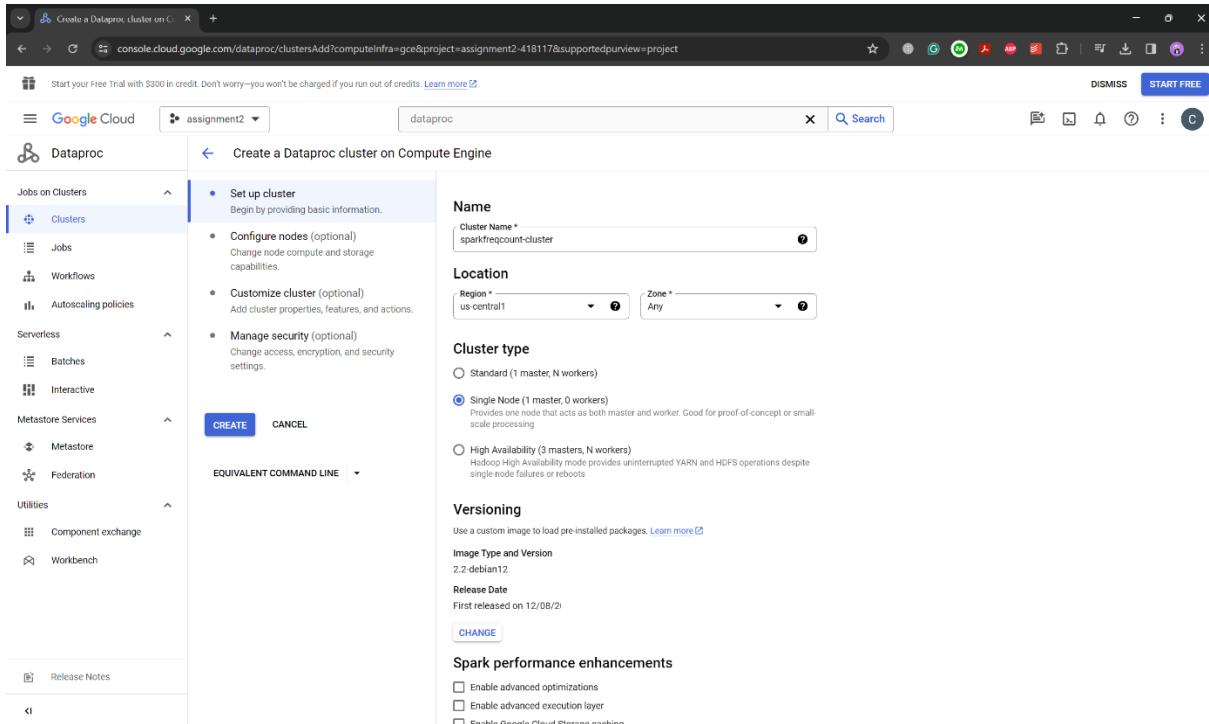


Figure 13 Settings for set up cluster

In configure nodes, I changed machine type to 2vCpu with 8 GB memory and primary disk size to 250 GB.

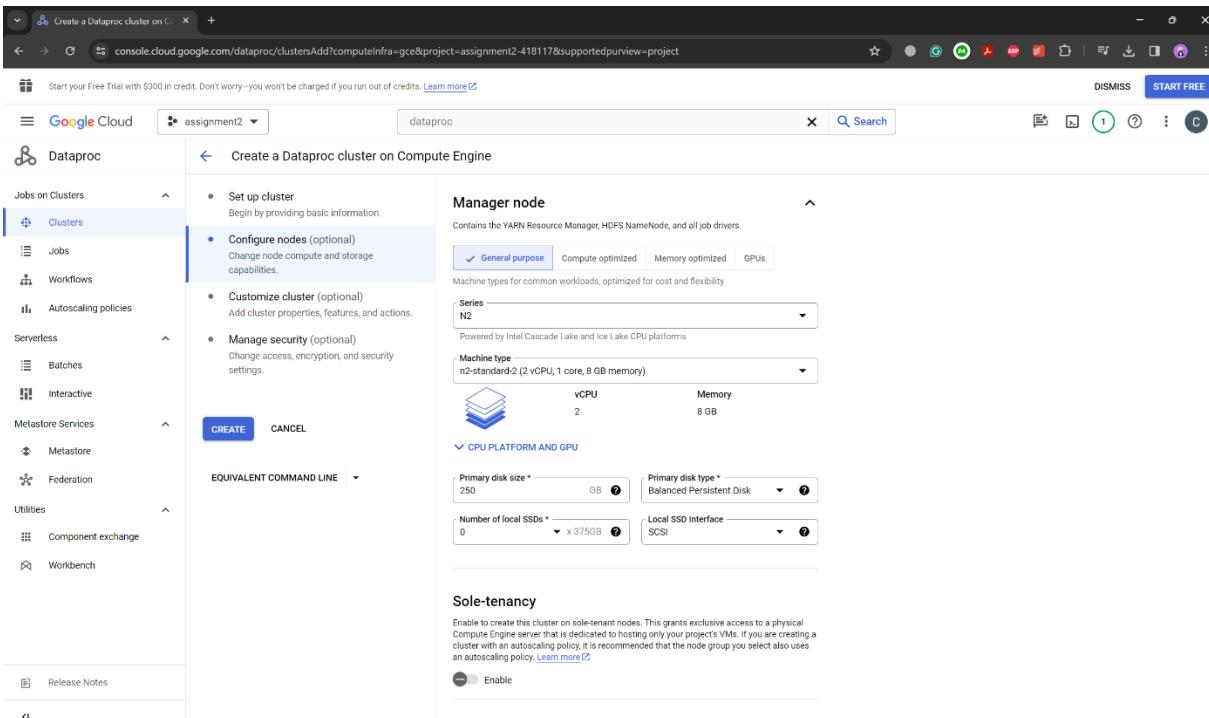


Figure 14 Settings for configure nodes

Selected the sparkfreqcount-cluster.

The screenshot shows the Google Cloud Dataproc Clusters menu. On the left, there's a sidebar with categories like Clusters, Jobs, Workflows, Serverless, Metastore Services, Utilities, and Workbench. The 'Clusters' section is selected. In the main pane, there's a table titled 'Clusters' with columns: Name, Status, Region, Zone, Total worker nodes, Flexible VMs?, Scheduled deletion, and Cloud Storage stage. One row is selected, showing 'sparkfreqcount-cluster' as Running in us-central1. A message at the bottom right says 'Please select at least one resource.'

Figure 15 Clusters menu

Then, selected the VM instance of the sparkfreqcount cluster.

The screenshot shows the Google Cloud Dataproc Cluster details page for 'sparkfreqcount-cluster'. The left sidebar is identical to Figure 15. The main pane has tabs for Cluster details, Submit Job, Refresh, Start, Stop, Delete, and View Logs. Under 'Cluster details', there's a note about service account permissions. Below that, it shows the cluster's name, UUID, type (Dataproc Cluster), and status (Running). There are tabs for Monitoring, Jobs, VM Instances, Configuration, and Web Interfaces. The 'VM Instances' tab is selected, showing a table with columns: Name, Role, and Machine type. One row is selected, showing 'sparkfreqcount-cluster m' as Master with 'n2 standard-2' machine type. There's also a 'Filter instances' dropdown.

Figure 16 VM instances for sparkfreqcount-cluster

Opened SSH browser window for the sparkfreqcount-cluster-m.

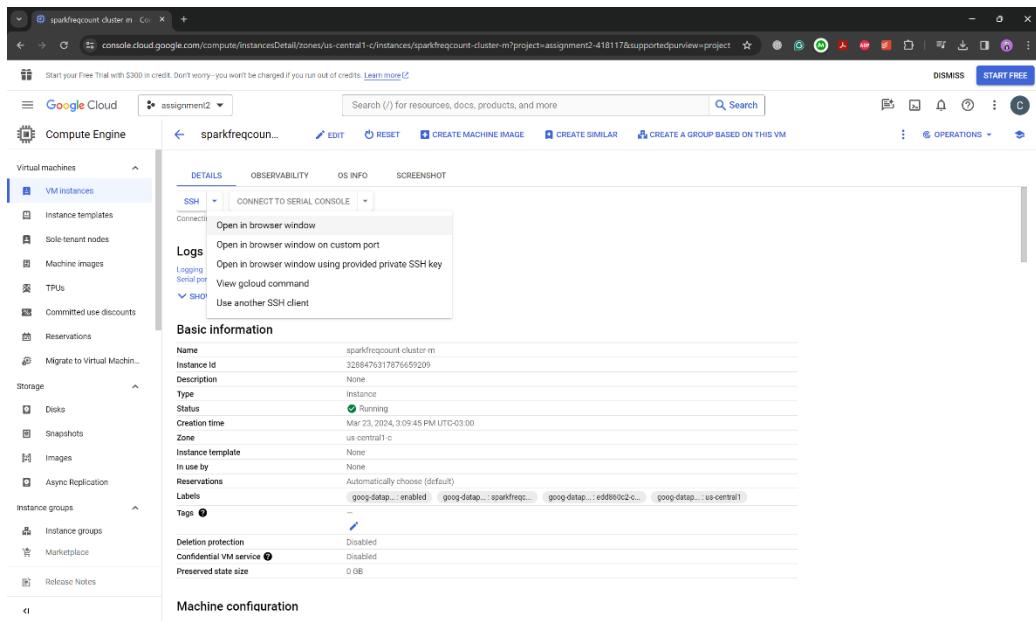


Figure 17 SSH in browser window

I uploaded SparkFreqCount-1.0-SNAPSHOT.jar, reut2-009.sgm, and stop-words.txt.

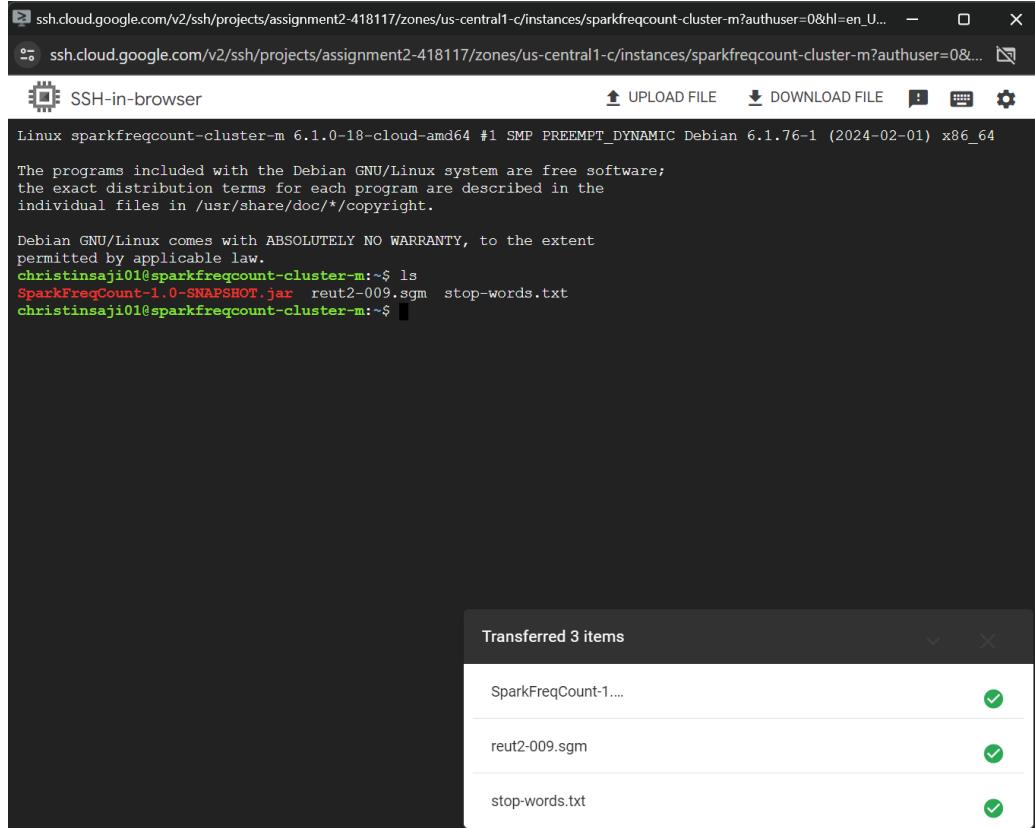


Figure 18 Uploaded files in the SSH cloud

Executed the SparkFreqCount program using the command “spark-submit” –class org.example.SparkFreqCount-1.0-SNAPSHOT.jar file:///home/christinsaji01/reut2-009.sgm stop-words.txt

```

christinsaji01@sparkfreqcount-cluster-m:~$ ls
SparkFreqCount-1.0-SNAPSHOT.jar  reut2-009.sgm  stop-words.txt
christinsaji01@sparkfreqcount-cluster-m:~$ spark-submit --class org.example.SparkFreqCount SparkFreqCount-1.0-SNAPSHOT.jar file:///home/christinsaji01/reut2-009.sgm stop-words.txt
24/03/23 18:32:21 INFO SparkEnv: Registering MapOutputTracker
24/03/23 18:32:21 INFO SparkEnv: Registering BlockManagerMaster
24/03/23 18:32:21 INFO SparkEnv: Registering JobHistoryHeartbeat
24/03/23 18:32:21 INFO SparkEnv: Registering OutputCommitCoordinator
24/03/23 18:32:22 INFO DefaultToHARMFailoverProxyProvider: Connecting to ResourceManager at sparkfreqcount-cluster-m.us-central1-c.c.assignment2-418117.internal./10.128.0.2:8032
24/03/23 18:32:22 INFO ARSProxy: Connecting to Application History server at sparkfreqcount-cluster-m.us-central1-c.c.assignment2-418117.internal./10.128.0.2:10200
24/03/23 18:32:24 INFO Configuration: resource-types.xml not found
24/03/23 18:32:24 INFO ResourceUtils: Unable to find 'resource-types.xml'.
24/03/23 18:32:25 INFO YarnClientImpl: Submitted application application_1711217486103_0006
24/03/23 18:32:26 INFO DefaultToHARMFailoverProxyProvider: Connecting to ResourceManager at sparkfreqcount-cluster-m.us-central1-c.c.assignment2-418117.internal./10.128.0.2:8030
24/03/23 18:32:27 INFO MetricsConfig: Loaded properties from hadoop-metrics2.properties
24/03/23 18:32:27 INFO MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
24/03/23 18:32:27 INFO MetricsSystemImpl: google-hadoop-file-system metrics system started
24/03/23 18:32:29 INFO GoogleCloudStorageImpl: Ignoring exception of type GoogleJsonResponseException; verified object already exists with desired state.
24/03/23 18:32:31 INFO GoogleHadoopOutputStream: hflush(): No-op due to rate limit (RateLimiter[stableRate=0.2gops]): readers will *not* yet see flushed data for gs://dataproc-temp-us-central1-49170580467-vvtzpq6e/eddb60c2-c1d9-429d-b329-dd87dfa4f02/spark-job-history/application_1711217486103_0006.in-progress [CONTEXT rateLimit_period="1 MINUTES"]
Highest frequency words:
+-----+-----+
| value | count |
+-----+-----+
| said | 2730 |
| mln | 1284 |
| march | 1051 |
| dirs | 1013 |
| mar | 1003 |
+-----+-----+
only showing top 5 rows

Lowest frequency words:
+-----+-----+
| value | count |
+-----+-----+
| tripolis | 1 |
| omisions | 1 |
| landover | 1 |
| bestdefended | 1 |
| disrupting | 1 |
+-----+-----+
only showing top 5 rows

christinsaji01@sparkfreqcount-cluster-m:~$ 

```

Figure 19 Output of the SparkFreqCount program

Problem 2: Building Neo4J Graph Database for relation visualization.

From the Assignment 1, final Erd I took, Student-Enrollment-Program-Course relationship.

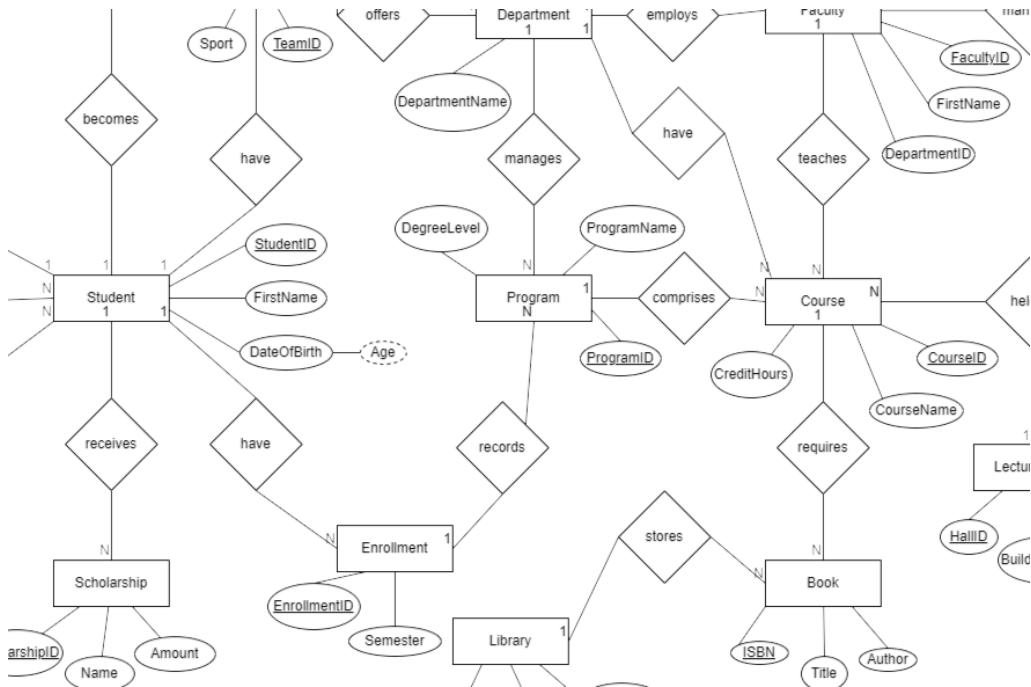


Figure 20 Part of final Erd from Assignment 1 Student-Enrollment-Program-Course relation

Firstly, I created a Assignment 2 project in the Neo4J desktop application.

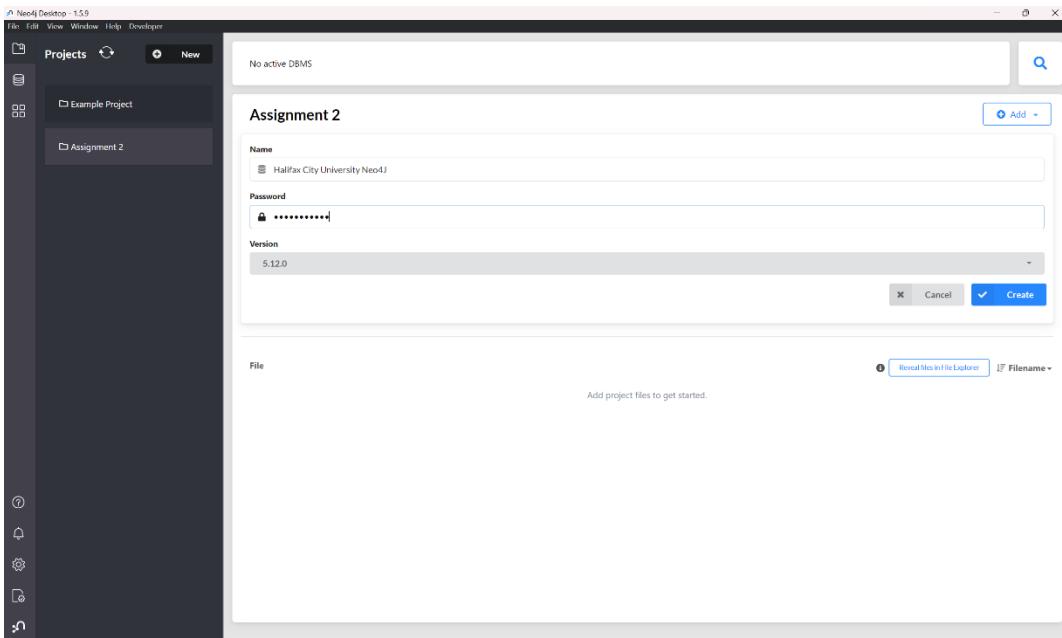


Figure 21 Assignment 2 project in Neo4J application

Inside the Assignment 2 project, I created database named Halifax City University Neo4J and activated it.

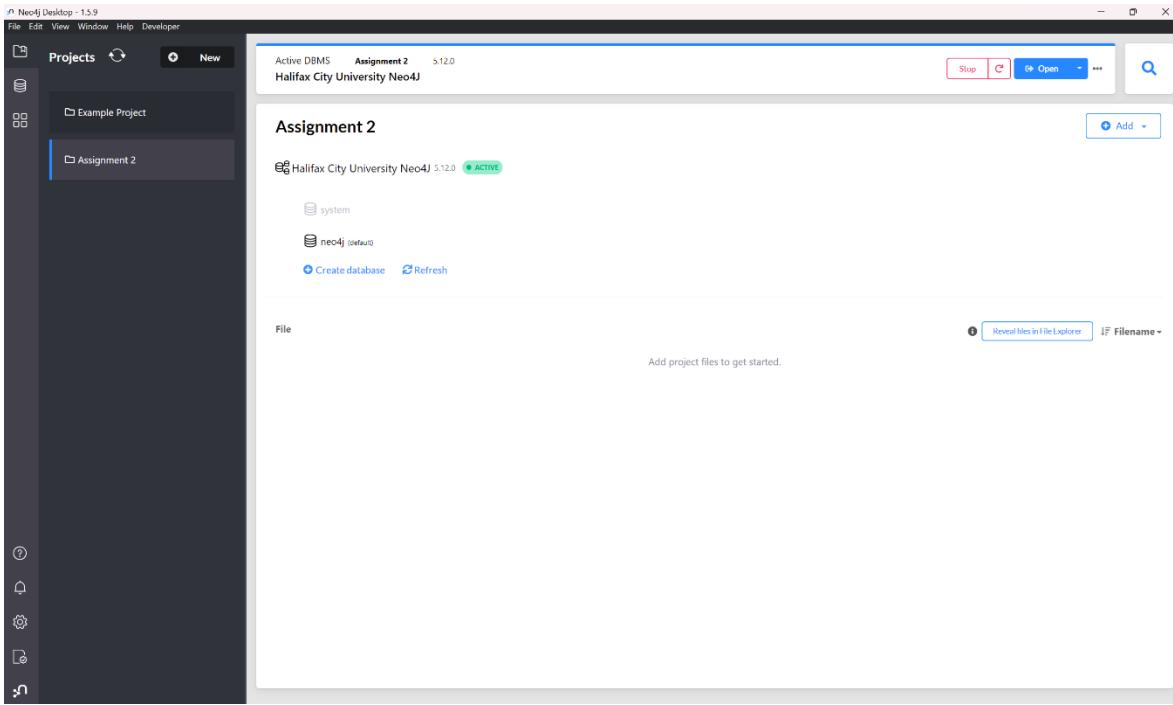


Figure 22 Halifax City University Neo4J database inside Assignment 2

Then I opened the Neo4J Browser from the apps section in the Neo4J Desktop application to run the cypher query language.

I began by establishing the primary entities as nodes within graph. Each 'Student' node captures essential attributes like 'StudentID', 'FirstName' and 'DateOfBirth'. 'Enrollment' nodes is tied to a specific semester. 'Program' nodes encapsulate distinct academic pathways such as 'MACS' and 'Data Science,' detailing the degree level. 'Course' nodes represent individual subjects, complete with unique identifiers and descriptive names.

Students are linked to their respective enrollments through `:ENROLLED` relationships, suggesting a enrollment in the institution [3]. The `:RECORDS` relationships from enrollments to programs establish which courses are undertaken during a particular enrollment period. Finally, programs are decomposed into the courses they comprise through `:COMPRIMES` relationships, allowing us to trace a student's academic trajectory down to the specific subjects studied.

Using Cypher's `CREATE` statements, I created the nodes and relationships. Each `CREATE` action sequentially built graph, starting with students to the courses they study.

```

1 CREATE (student1:Student {StudentID: 101, FirstName: 'Christin', DateOfBirth: '08-03-2001'})
2 CREATE (student2:Student {StudentID: 102, FirstName: 'Mithun', DateOfBirth: '23-07-1999'})
3
4 CREATE (enrollment1:Enrollment {EnrollmentID: 1001, Semester: 'Winter 2024'})
5 CREATE (enrollment2:Enrollment {EnrollmentID: 1002, Semester: 'Summer 2024'})
6
7 CREATE (program1:Program {ProgramID: 5001, ProgramName: 'MACS', DegreeLevel: 'Masters'})
8 CREATE (program2:Program {ProgramID: 5002, ProgramName: 'Data Science', DegreeLevel: 'Masters'})
9
10 CREATE (course1:Course {CourseID: 5408, CourseName: 'Database Management, Warehousing & Analytics'})
11 CREATE (course2:Course {CourseID: 5308, CourseName: 'Advance Topics in Software Development'})
12 CREATE (course3:Course {CourseID: 6409, CourseName: 'Process of Data Science'})
13
14 CREATE (student1)-[:ENROLLED]-(enrollment1)
15 CREATE (student2)-[:ENROLLED]-(enrollment1)
16
17 CREATE (enrollment1)-[:RECORDS]-(program1)
18 CREATE (enrollment1)-[:RECORDS]-(program2)
19 CREATE (enrollment2)-[:RECORDS]-(program1)
20 CREATE (enrollment2)-[:RECORDS]-(program2)
21
22 CREATE (program1)-[:COMPRISES]-(course1)
23 CREATE (program1)-[:COMPRISES]-(course2)
24 CREATE (program1)-[:COMPRISES]-(course3)
25 CREATE (program2)-[:COMPRISES]-(course3)

```

Figure 23 Cypher query to create node and make relationship

Finally, I used cypher query ‘MATCH (n) RETURN n’ to select and return all nodes in the graph database.

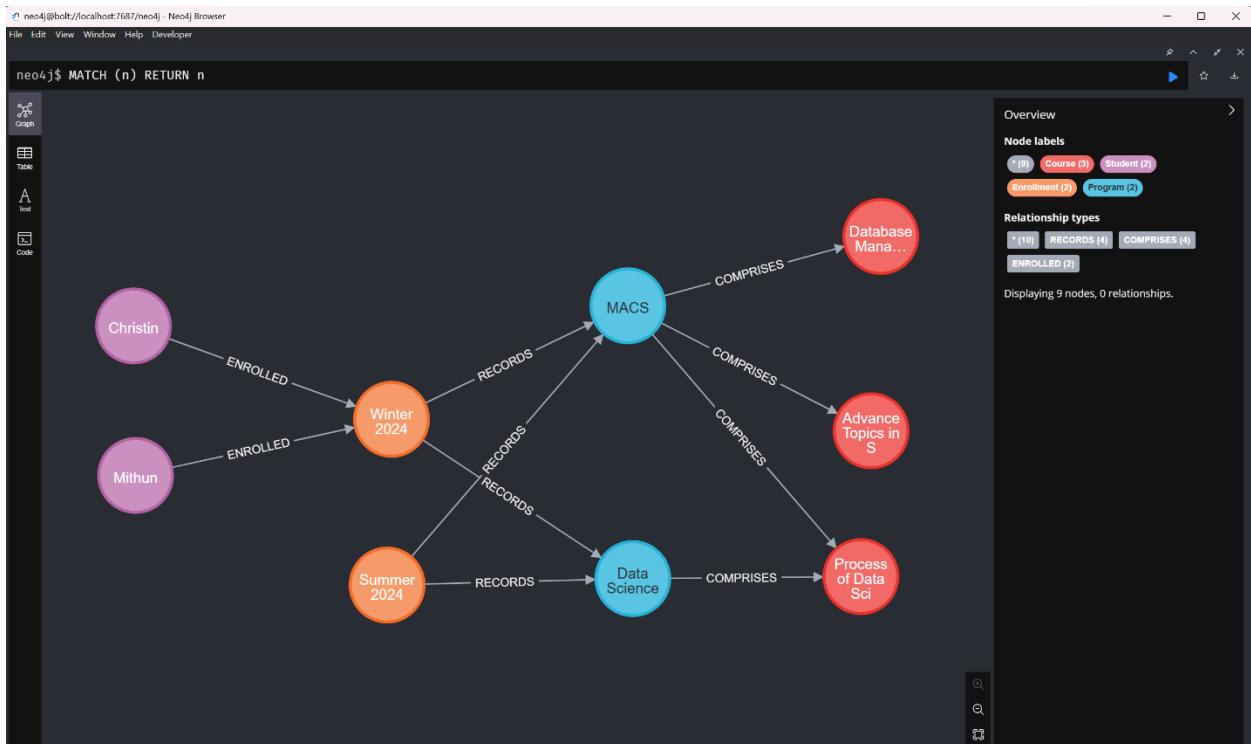


Figure 24 Displaying the nodes of the graph database

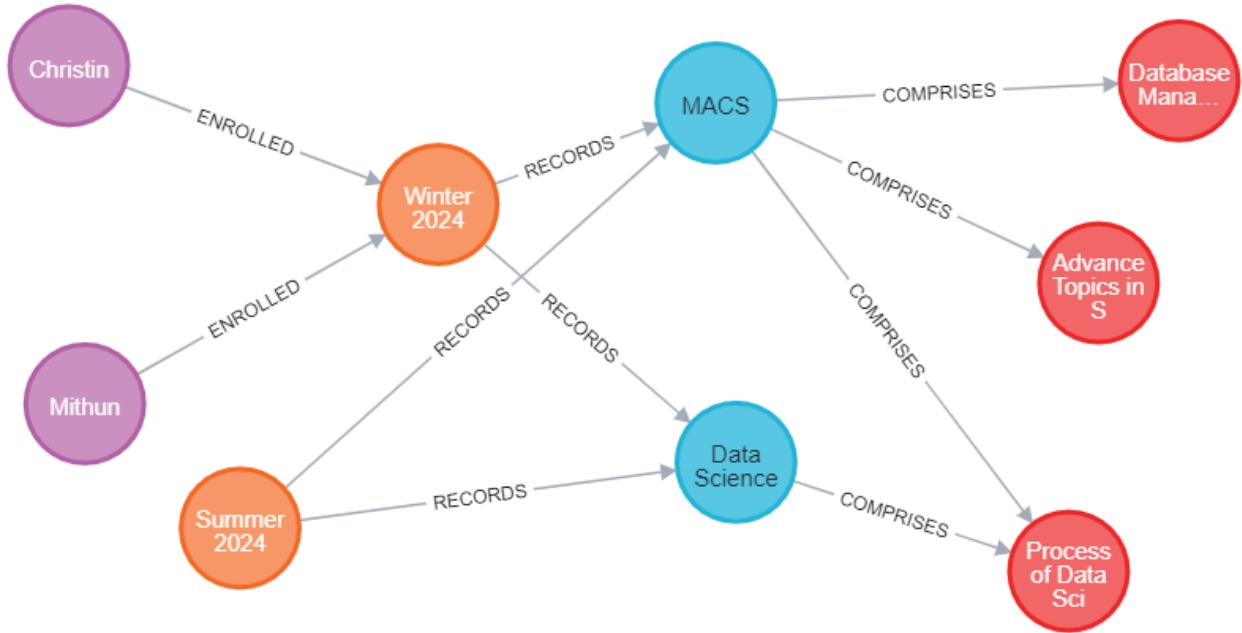


Figure 25 Final Neo4J Graph

Problem 3: Sentiment Analysis using BOW model on title of Reuters News Articles.

First, I loaded the positive-words.txt and negative-words.txt. It had 2006 positive words and 4783 negative words.

```

public static void main(String[] args) {
    ...
    String filePath = "reut2-009.sgm";
    try {
        Set<String> positiveWords = wordListService.loadWordsFromFile("positive-words.txt");
        Set<String> negativeWords = wordListService.loadWordsFromFile("negative-words.txt");
        List<String> titles = fileOperationService.extractTitlesFromFile(filePath);
        SentimentAnalysisService sentimentService = new SentimentAnalysisService(positiveWords, negativeWords);
        List<SentimentResult> sentimentResults = new ArrayList<>();
        for (int i = 0; i < titles.size(); i++) {
            SentimentResult result = sentimentService.analyzeSentiment(titles.get(i), newIndexNumber[i + 1]);
            sentimentResults.add(result);
        }
    }
}

```

Figure 26 positive-words.txt and negative-words.txt loaded on the program

Extracted the whole raw data from the ‘reut2-009.sgm’ file on the ‘contentBuilder’ variable.

```

public List<String> extractTitlesFromFile(String filePath) {
    StringBuilder contentBuilder = new StringBuilder();
    try (BufferedReader br = new BufferedReader(new FileReader(filePath))) {
        String line;
        while ((line = br.readLine()) != null) {
            contentBuilder.append(line);
        }
    }
    titles = extractionService.extractTitle(contentBuilder.toString());
    return titles;
}

```

Figure 27 contentBuilder variable having whole raw data

Then, extracted the 'TITLE' tag from inside the 'REUTES' tag, which is currently having html entities and special characters inside it.

The screenshot shows an IDE interface with the following details:

- Project:** SentimentAnalysis
- File:** ExtractionService.java
- Code Content (Excerpt):**

```
public class ExtractionService {
    private final CleanerService cleanerService; cleanerService: CleanerService@877

    public ExtractionService(CleanerService cleanerService) {
        this.cleanerService = cleanerService;
    }

    /**
     * Extracts new titles from the given Reuters content
     * @param filterData Reuters content
     * @return list of extracted titles
     */
    public List<String> extractTitles(String filterData) { filterData: <!DOCTYPE lewis SYSTEM "lewis.dtd"><!ELEMENTS TOPICS="NO LEWISPLIT="TRAIN" COISPLIT="TRAI
        List<String> titles = new ArrayList<>(); titles.size() = 0
        Pattern pattern = Pattern.compile("(?s:<REUTERS.*?>/REUTERS)", Pattern.DOTALL); pattern: "<REUTERS.*?>/REUTERS"
        Matcher matcher = pattern.matcher(filterData); filterData: "<!DOCTYPE lewis SYSTEM "lewis.dtd"><!ELEMENTS TOPICS="NO LEWISPLIT="TRAIN" COISPLIT="TRAI
        while (matcher.find()) {
            String reuterContent = matcher.group(); matcher: "java.util.regex.Matcher@974" region@0,1338903 lastmatch@<REUTERS TOPI
            String title = extractTitleBetweenTags(reuterContent); reuterContent: "TOPICS="NO LEWISPLIT="TRAIN" COISPLIT="TRAINING-SET" OLID@>"14522" NEWID@>"9001" DATE@>"24-MAR-1987 15:59:10" View
            title = cleanerService.cleanText(title); title: "ADVANCED MAGNETICS &ADMG> IN AGREEMENT" cleanerService: CleanerService@877
            if (!title.isEmpty()) {
                titles.add(title);
            }
        }
        return titles;
    }
}
```

- Debug View:** Shows a stack trace for an exception. The top frame is `main@1 in group 'main': RUNNING`.
- Threads & Variables View:** Shows variables for the current thread, including `this` pointing to `ExtractionService@878` and `filterData` containing the XML snippet.
- Console View:** Displays the output of the application's standard error stream.
- Status Bar:** Shows the file path as `src > service` and the status bar with `28:1 CRLF UTF-8 4 spaces`.

Figure 28 Title tag before the CleanerService

Cleaned out TITLE tags data using the 'CleanerService' which removes the html entities and special characters.

```
public class ExtractionService {
    private final CleanerService cleanerService; cleanerService: CleanerService@877

    Usage
    public ExtractionService(CleanerService cleanerService) {
        this.cleanerService = cleanerService;
    }

    /**
     * Extracts new titles from the given Reuters content
     * @param filterData Reuters content
     * @return list of extracted titles
     */
    Usage
    public List<String> extractTitles(String filterData) { filterData: <!DOCTYPE lewis SYSTEM "lewis.dtd"><n>REUTERS TOPICS="NO" LEWISPLIT="TRAIN" CGISPLIT="

    List<String> titles = new ArrayList<>(); titles: size = 1
    Pattern pattern = Pattern.compile(regex "<REUTERS(.*)>/<REUTERS>", Pattern.DOTALL); pattern: <REUTERS(.*)>/<REUTERS>
    Matcher matcher = pattern.matcher(filterData); filterData: <!DOCTYPE lewis SYSTEM "lewis.dtd"><n>REUTERS TOPICS="NO" LEWISPLIT="TRAIN" CGISPLIT="TRAIN

    while (matcher.find()) {
        String reuterContent = matcher.group(1); matcher: java.util.regex.Matcher@pattern=<REUTERS(.*)>/<REUTERS> region=0,1338903 lastmatch=<REUTERS TOP
        String title = extractBetweenTags(reuterContent); reuterContent: TOPICS="NO" LEWISPLIT="TRAIN" CGISPLIT="TRAINING-SET" OLID=14523 NEWID="9002
        title = cleanerService.cleanText(title); title: "HEALTH RESEARCH FILES FOR BANKRUPTCY" cleanerService: CleanerService@877

        if (!title.isEmpty()) {
            titles.add(title);
        }
    }
}
```

Figure 29 Cleaned data after the CleanerService

After extracting and cleaning the raw data there are about total 996 titles for which Sentimental analysis is to be done.

```

24
25     while (matcher.find()) {
26         String reuterContent = matcher.group(1); matcher: "java.util.regex.Matcher@pattern=<REUTERS(.*)?>/REUTERS> region=0,1338903 lastmatch="
27         String title = extractBetweenTags(reuterContent);
28         title = cleanerService.cleanText(title); cleanerService: CleanerService@877
29         if (!title.isEmpty()) {
30             titles.add(title);
31         }
32     }
33     return titles; titles: size = 996
34 }

35 /**
36  * Extract content enclosed within Title tags
37  * @param content string containing title tags with content
38  * @return content between the tags
39 */
40 Usage
41 private String extractBetweenTags(String content) {
42     Pattern tagPattern = Pattern.compile("(?s)<TITLE>.*<(.*)></TITLE>"); tagPattern.matcher(content);
43     Matcher tagMatcher = tagPattern.matcher(content);
44     if (tagMatcher.find()) {
45         return tagMatcher.group(1);
46     }
47     return "";
48 }

```

Figure 30 Number of titles in the file

Each title is converted to a List<String> carrying a bag of words for the process of sentimental analysis.

```

9
10     private final Set<String> negativeWords; negativeWords: size = 4783
11
12 Usage
13 public SentimentAnalysisService(Set<String> positiveWords, Set<String> negativeWords) {
14     this.positiveWords = positiveWords;
15     this.negativeWords = negativeWords;
16 }
17
18 /**
19  * Analyze the sentiment of a given title by counting the frequency of positive and negative words
20  * @param title title content to analyze
21  * @param newsNumber number associated with bag of words
22  * @return a SentimentResult object
23 */
24 Usage
25 @
26 public SentimentResult analyzeSentiment(String title, int newsNumber) { newsNumber: 1 title: "ADVANCED MAGNETICS ADMG IN AGREEMENT"
27     String[] words = title.split(" "); words: ["ADVANCED", "MAGNETICS", "ADMG", "IN", "AGREEMENT"] title: "ADVANCED MAGNETICS ADMG IN AGREEMENT"
28     int score = 0; score: 0
29     List<String> matches = new ArrayList<>();
30
31     for (String word : words) {
32         word = word.toLowerCase();
33         if (positiveWords.contains(word)) {
34             score++;
35             matches.add(word);
36         } else if (negativeWords.contains(word)) {
37
38     }
39 }

```

Figure 31 Bag of words of a title

Each bag of words is then compared to the positive words and negative words. If a word is matched to one of the positive words, then score is increased by 1 or decreased by 1 if it matches to a negative word. The matched words are stored in a ArrayList so that they can be displayed comma separated under one column while displaying in the csv file. If the total score is greater than 0 then the polarity is set to ‘Positive’ or if it’s less than 0 then it is set to ‘Neutral’, and finally if score is 0 then polarity is set to ‘Neutral’.

```

public SentimentResult analyzeSentiment(String title, int newsNumber) {
    String[] words = title.split("\\s+");
    int score = 0;
    List<String> matches = new ArrayList<>();
    matches.size = 0;

    for (String word : words) {
        word = word.toLowerCase();
        if (positiveWords.contains(word)) {
            score++;
            matches.add(word);
        } else if (negativeWords.contains(word)) {
            score--;
            matches.add(word);
        }
    }

    String polarity = score > 0 ? "Positive" : score < 0 ? "Negative" : "Neutral";
    polarity = Positive;
    return new SentimentResult(newsNumber, title, String.join(", ", matches), score, polarity);
}

```

Figure 32 Sentimental Analysis of a title

Then after the sentimental analysis the object values are stored in the csv file named ‘sentimental-analysis.csv’.

```

Set<String> positiveWords = wordListService.loadWordsFromFile("filePath: positive-words.txt");
positiveWords.size = 2000
Set<String> negativeWords = wordListService.loadWordsFromFile("filePath: negative-words.txt");
negativeWords.size = 4783

List<String> titles = fileOperationService.extractTitlesFromFile(filePath);
fileOperationService.FileOperationService@879 filePath: "reut2-009.sgm"
titles.size = 996

SentimentAnalysisService sentimentService = new SentimentAnalysisService(positiveWords, negativeWords);
sentimentService.SentimentAnalysisService@1033
sentimentService.sentimentService = sentimentService

List<SentimentResult> sentimentResults = new ArrayList<>();
sentimentResults.size = 996

for (int i = 0; i < titles.size(); i++) {
    SentimentResult result = sentimentService.analyzeSentiment(titles.get(i), newsNumber: i + 1);
    sentimentService.sentimentService.add(result);
}

excelService.writeResultsToCSV("filePath: sentiment-analysis.csv", sentimentResults);
excelService.ExcelOutputService@881 sentimentResults.size = 996

```

Figure 33 sentimental-analysis file creation

Finally, while printing in the csv file it is formatted as mentioned in the task.

```

public class ExcelOutputService {
    /**
     * Writes the sentiment analysis results to a CSV file
     * @param fileName name of the file
     * @param results list of sentiment analysis results
     * @throws IOException if an I/O error occurs writing to or creating the file
     */
    public void writeResultsToCSV(String fileName, List<SentimentResult> results) throws IOException {
        try (PrintWriter writer = new PrintWriter(new BufferedWriter(Paths.get(fileName)))) {
            writer.println("News#|Title|Content|Match|Score|Polarity");
            for (SentimentResult result : results) {
                String matches = String.join(delimiter, ", " + result.matches());
                writer.println(String.format("%d|%s|%s|%s,%s,%s", result.newsNumber(),
                    result.title(),
                    matches,
                    result.score(),
                    result.polarity()));
            }
        }
    }
}

```

Figure 34 Formatting of the content

Output:

News#	Title	Content	Match	Score	Polarity
1	ADVANCED MAGNETICS AGING TO AGREEMENT		advanced	1	Positive
2	HEALTH RESEARCH FILES FOR BANKRUPTCY			0	Neutral
3	3 NUMEREX CORP NMRX 2ND QTR JAN 31 LOSS		loss	-1	Negative
4	US SELLING 12B BILLION DLR'S OF 3 AND 8MO BILLS MARCH 30			0	Neutral
5	US 2YEAR NOTE AVERAGE YIELD 643 PCT STOP-644 PCT			1	Positive
6	COMMODORE CBU ATARI IN SETTLEMENT		awarded	0	Neutral
7	US 2YEAR NOTE AVERAGE YIELD 643 PCT STOP-644 PCT			0	Neutral
8	TRIMBLE TRI BEGINS EXCHANGE OFFER		supports	1	Positive
9	SOUTHEMARK 9M UNIT IN PUBLIC OFFERING OF STOCK			0	Neutral
10	EASTMAN KODAK CO TO SELL HOLDINGS IN ICN			0	Neutral
11	FEUD PERSISTS AT US HOUSE BUDGET COMMITTEE			0	Neutral
12	TREASURY BALANCES AT FED ROSE ON MARCH 23			0	Neutral
13	FARM CREDIT SYSTEM SETS NEEDING 600 MLN DLR'S AID			0	Neutral
14	UNISTAT URGES RETALIATION AGAINST JAPAN			0	Neutral
15	EXXON XON GETS 992 MLN DLR CONTRACT			0	Neutral
16	EATON ETN GETS 530 MLN DLR CONTRACT			0	Neutral
17	PAIRE AUTHORIZES TO BUY PL480 RICE USA			0	Neutral
18	MCDONNELL DOUGLAS GETS 306 MLN DLR CONTRACT			0	Neutral
19	MC DONNELL DOUGLAS ACQUIRES ASSETS OF BUSINESS AVIATION			0	Neutral
20	SWITZERLAND SWISSAIR TAKES OVER AIRLINES			0	Neutral
21	2Q DOLLAR EXPECTED TO FALL DESPITE INTERVENTION		fall	-1	Negative
22	US TO SELL 38 BILLION DLR'S IN BILLS			0	Neutral
23	INLAND STEEL IAD TO BUILD NEW PLANT IN INDIANA			0	Neutral
24	EASTMAN KODAK EKT TO SELL HOLDINGS			0	Neutral
25	GUINNESS SUES BOESEY IN FEDERAL COURT		sues	-1	Negative
26	PRODUCES SOFT DRINK SOURCES SPRL HOLDINGS			0	Neutral
27	GLENCOOP INC WITHDRAWNS PROPOSALS TO STAGGER			0	Neutral
28	ACHILLES LTD 1ST QTR FEB 28 NET			0	Neutral
29	BULL AND BEAR GROUP A&B Cuts Fund PAYOUTS			0	Neutral
30	CALTEX TO RAISE BAHRAIN OIL PRODUCT PRICES			0	Neutral
31	CHARTER CO OCHAN TO COMPLETE REORGANIZATION			0	Neutral
32	NASHUA NSH TO PURCHASE PRIVATE DISC MAKER			0	Neutral
33	ALUMINUM ALTAIR Q1 QTR 2000			0	Neutral
34	GENOCORP CY PROPOSALS DOWN FROM MEETING			0	Neutral
35	WEFUYI PI FCTRIC OUTLIT UP 24 PCT FROM 1999			0	Neutral

Figure 35 Starting sentimental analysis row from csv file

The screenshot shows a Microsoft Excel-like application window titled "sentiment-analysis". The spreadsheet contains data from rows 963 to 998. Column A lists various news headlines. Column C contains labels such as "qualified", "loss", "cleared", "good", and "wins". Column D contains labels like "neutral", "positive", "negative", and "limit". Column E contains numerical sentiment scores ranging from -1 (Negative) to 1 (Positive), with most entries being 0 Neutral.

	A	B	C	D	E
963	962 FIRST MEDICAL FMDC GETS ORDER		qualified	0 Neutral	1 Positive
964	963 HEALTHMATE HMTE EXPECTS QUALIFIED OPINION			0 Neutral	0 Neutral
965	964 BALLY BLY NAMES NEW CHIEF FINANCIAL OFFICER			0 Neutral	0 Neutral
966	965 BRAZIL ASKS 60DAY EXTENSION SHORTTERM CREDIT				-1 Negative
967	966 HEALTHMATE INC HMTE 4TH QTR LOSS		loss		
968	967 USAIR U CLEARED TO BUY REDMONT P/E SHARES		cleared	1 Positive	
969	FED SETS TWO BILLION DLR CUSTOMER REPURCHASE FED SAYS				
970	968 HOUSE OF FABRICS INC HF 4TH QTR JAN 31 NET			0 Neutral	0 Neutral
971	970 GERMAN RETAILERS EXPECT GOOD 1987		good	1 Positive	1 Positive
972	971 GERMAN WAGE ROUND SAID TO LIMIT MONETARY OPTIONS		limit	-1 Negative	-1 Negative
973	972 FOREIGN FIRMS HOPE TO JOIN JAPAN TELECOM COMPANY			0 Neutral	0 Neutral
974	973 SIEMENS RAISES STAKE IN TELECOM PLUS OF US			0 Neutral	0 Neutral
975	974 FIRST INTERSTATE SEEKS ACQUISITION			0 Neutral	0 Neutral
976	975 ATLANTIC RESEARCH ATRC WINS MOTOR CONTRACT		wins	1 Positive	1 Positive
977	976 THOMAS AND BETTS TNB FILES PATENT SUIT			0 Neutral	0 Neutral
978	977 HOUSES OF FABRICS HF SEES RESULTS IMPROVING		improving	1 Positive	1 Positive
979	978 US SENATE HITS EC OILS TAX VOWS RETALIATION			0 Neutral	0 Neutral
980	979 CROP GENETICS INITIAL OFFERING UNDERWAY			0 Neutral	0 Neutral
981	980 KODAK EK BUYS STAKE IN BIOTECHNOLOGY COMPANY			0 Neutral	0 Neutral
982	981 AVERY AVY 1ST QTR FEB 28 NET			0 Neutral	0 Neutral
983	982 FORD F CREDIT UNIT SELLS NEW ZEALAND DLR FRNS			0 Neutral	0 Neutral
984	983 IONICS ION WINS 20YEAR DESALINATION CONTRACT		wins	1 Positive	1 Positive
985	984 FED SETS TWO BILLION DLR CUSTOMER REPURCHASE			0 Neutral	0 Neutral
986	985 SENATE SEEKS US PROBE OF CANADIAN CORN LEVY			0 Neutral	0 Neutral
987	986 RICOH REORGANIZES US UNITS			0 Neutral	0 Neutral
988	987 UNITED WATER UWR BASE TEN BASE SET VENTURE			0 Neutral	0 Neutral
989	988 KNIGHTRIDER INC KRN SETS QUARTERLY			0 Neutral	0 Neutral
990	989 TECHNITROL INC TNL SETS QUARTERLY			0 Neutral	0 Neutral
991	990 NATIONWIDE CELLULAR SERVICE INC NCSEL 4TH QTR			0 Neutral	0 Neutral
992	991 AHA AUTOMOTIVE TECHNOLOGIES CORP YEAR NET			0 Neutral	0 Neutral
993	992 BRAZIL SUGGESTS 60DAY SHORTTERM CREDIT EXTENSION			0 Neutral	0 Neutral
994	993 EHART EMH WINS FOUR COMMUNICATIONS CONTRACTS		wins	1 Positive	1 Positive
995	994 FLUOR FLR AND UNITS DOWNGRADED BY SP			0 Neutral	0 Neutral
996	995 BRUNSWICK CORP BC SELLS NOTES AT 8199 PCT			0 Neutral	0 Neutral
997	996 ROGERS ROG SEES 1ST QTR NET UP SIGNIFICANTLY			0 Neutral	0 Neutral
998					

Figure 36 Ending sentimental analysis row from csv file

References

- [1] “Quick Start - Java Sync Driver,” MongoDB, [Online]. Available: <https://www.mongodb.com/docs/drivers/java-sync/current/quick-start/> [Accessed: March 21, 2024].
- [2] Naveen (NNK), “Spark Read Text File: RDD: DataFrame,” *Spark By Examples*, [Online], Feb 8, 2023. Available: <https://sparkbyexamples.com/spark/spark-read-text-file-rdd-dataframe/> [Accessed: March 21, 2024].
- [3] “Query a neo4j database using Cypher - Getting Started,” Neo4j Graph Data Platform, [Online]. Available: <https://neo4j.com/docs/getting-started/cypher-intro/> [Accessed: March 22, 2024].