CSCI 5408

DATA MANAGEMENT AND WAREHOUSING

ASSIGNMENT 2

Banner ID: B00957587

GitLab Assignment Link:
https://git.cs.dal.ca/shreyak/csci5408_f23_b00957587_shreya_kapoor/

Table of Contents

Problem 1: Perform research on NoSQL and data processing

Task 1A: Reuter News Data Reading & Transformation and storing in MogoDb

As part of this task, I needed to write a Java Program to scan the required texts between the Title Tag and the Body Tag which are present inside the Reuter Tag.

1. To create the Mongo Database ReuterDb, I have followed the following steps:

```
String encodedPassword = URLEncoder.encode("password", "UTF-8");

String databaseConnection = "mongodb+srv://username:" + encodedPassword +
"@database5408.7fxwzu9.mongodb.net/?retryWrites=true&w=majority";
ServerApi api = ServerApi.builder().version(ServerApiVersion.V1).build();
MongoClientSettings clientSettings =
MongoClientSettings.builder().applyConnectionString(new
ConnectionString(databaseConnection)).serverApi(api).build();
try (MongoClient mongoClient = MongoClients.create(clientSettings)) {
    System.out.println("Connection Established");
try {
    MongoDatabase db = mongoClient.getDatabase("ReutersDb");
}
}
catch (MongoException) {
    e.printStackTrace();
}
```

Here, I have used the URLEncoder because my password contained special symbols. With this my connection was successfully established and I could see the Database created by the name ReutersDB.

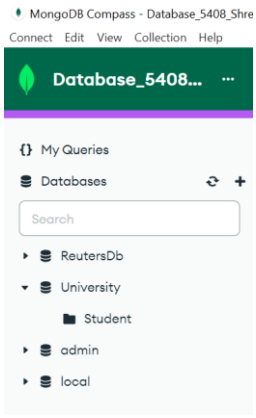


Figure 1: Mongo Database Creation

2. Once the connection was successfully established, I moved forward and created a function named **processSgmFileAndInsert.**
   This function takes in both the given SGM file and find the BODY and the TITLE tag within the REUTERS tag and then pass it to the other function for further processing.

```
private static void processSgmFileAndInsert(String fileName, MongoDatabase
database) throws IOException {
    try (BufferedReader fileReader = new BufferedReader(new
FileReader(fileName))) {
        StringBuilder fileContent = new StringBuilder();
        String line;
        while ((line = fileReader.readLine()) != null) {
            fileContent.append(line).append("");
        }
        String[] reutersData = fileContent.toString().split("</REUTERS>");
        List<Document> mongoDocs = new ArrayList<>();
        for (String reuterData : reutersData) {
            String newsTitle = cleanText(extractTagContent(reuterData,
"TITLE"));
            String newsBody = cleanText(extractTagContent(reuterData,
"BODY"));
            Document newsDocument = new Document();
            if (!newsTitle.isEmpty()) {
                newsDocument.append("title", newsTitle);
            }
            if (!newsBody.isEmpty()) {
                newsDocument.append("body", newsBody);
            }

            if (!newsTitle.isEmpty() || !newsBody.isEmpty()) {
                mongoDocs.add(newsDocument);
            }
        }

        if (!mongoDocs.isEmpty()) {
            MongoCollection<Document> newsCollection =
database.getCollection("news");
            newsCollection.insertMany(mongoDocs);
            System.out.println("Data successfully inserted into MongoDB!");
        }
    }
}
```

As seen in the code, it takes into account the title and the body tag which are present inside the reuters tag and append it into our mongo db after cleaning the data.

The **extractTagContent** is responsible for fetching the content between both the tags

```
private static String extractTagContent(String input, String tagName) {
    Pattern pattern = Pattern.compile("<" + tagName + ">(.*?)</" + tagName
+ ">", Pattern.DOTALL);
    Matcher matcher = pattern.matcher(input);
    if (matcher.find()) {
        return matcher.group(1).trim();
    } else {
        return "";
    }
}
```

Once we have the data which is present inside the tags, the next step is to perform the data cleaning.

For that, I have created a separate function named **cleanText.** All of my modules are following the Single Responsibility Principle.

Following is the function which took me a lot of time, as I had to dig deep into the data in order to clean the data properly.

```
private static String cleanText(String text) {
    text = text.replaceAll("&lt;", "").replaceAll("&gt;", "");
    text = text.replaceAll("[^a-zA-Z0-9\\s,.]", "");
    text = text.replace(" Reuter3","");
    return text.trim();
}
```

Firstly, I have removed all the stop words such as &lt and &gt.
Then, if there are any additional characters apart from alphabets or the numbers in that case also it needs to be removed.

After doing this I had checked the output, and it came to my attention that each body tag content is ending with the word “Reuter3” which again is redundant.
So, I have went on and removed those as well.

In the **processSgmFileAndInsert** function, earlier I did not handled the condition where if the body or title tag is empty, it was still getting inserted into the mongoDB, due to which I was getting empty records into my mongoDB and the count of them was over 2000.

Then I had handled that condition into my function after which it has been resolved.

After performing all the above mentioned steps, I was able to successfully clean the data and insert them into my ReuterDB Database.

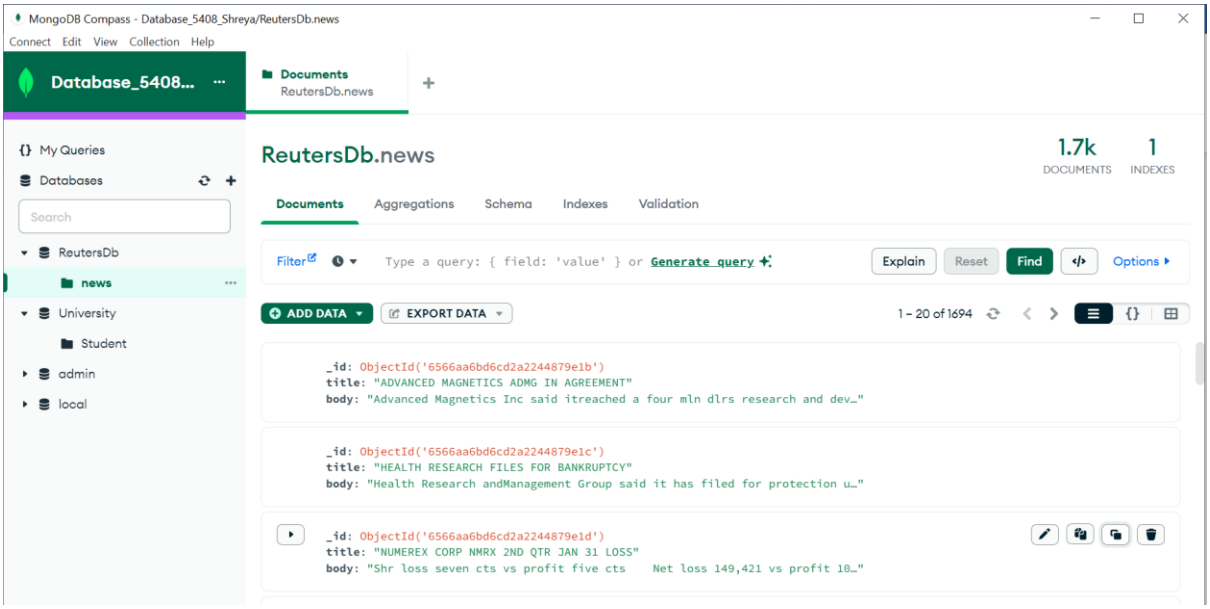


Figure 2: Output after inserting data into MongoDB

Algorithm of the data cleaning

Certainly! Let's break down the algorithm of the provided Java program:

1. **Main Method (`public static void main(String[] args) throws UnsupportedEncodingException`):**
    - Encode the MongoDB password using URL encoding.
    - Build the MongoDB connection string with the encoded password.
    - Establish a connection to the MongoDB server.
    - Access the "ReutersDb" database.
    - Process and insert data from two Reuters files ("reut2-009.sgm" and "reut2-014.sgm").
    - Retrieve news titles from the processed files.
    - Save the retrieved titles to a file named "titles.txt."
2. **Processing Reuters Files (`processSgmFileAndInsert` and `processSgmFileAndGetTitles` methods):**
    - Read each line from the specified Reuters file.
    - Concatenate the lines to form a single string representing the file's content.
    - Split the content into individual Reuters data entries using "</REUTERS>" as the delimiter.
    - For each Reuters data entry:
        - Extract the news title and clean the text using the `extractTagContent` and `cleanText` methods.
        - For `processSgmFileAndInsert`, also extract the news body.
        - Create a MongoDB document with the extracted title and body (if available).
        - Add the document to a list of MongoDB documents.
    - If the list of documents is not empty, insert them into the "news" collection of the MongoDB database.
3. **Saving Titles to File (`saveTitlesToFile` method):**
    - Write each news title to a file named "titles.txt."
4. **Text Extraction (`extractTagContent` method):**
    - Extract the content between specified XML tags in a given input string using regular expressions.
5. **Text Cleaning (`cleanText` method):**
    - Remove specific XML entities ("<" and ">").
    - Remove non-alphanumeric characters, except spaces, commas, and periods.
    - Remove the substring " Reuter3".
    - Trim leading and trailing whitespaces.

6. **MongoDB Setup:**
    - The MongoDB connection string is constructed using the provided username, encoded password, and database details.
    - A connection to the MongoDB server is established using the constructed connection string.
    - The MongoDB database "ReutersDb" is accessed.
7. **Exception Handling:**
    - Catch `MongoException` and `IOException` during database operations and file reading, respectively.
    - Print the stack trace for any caught exceptions.

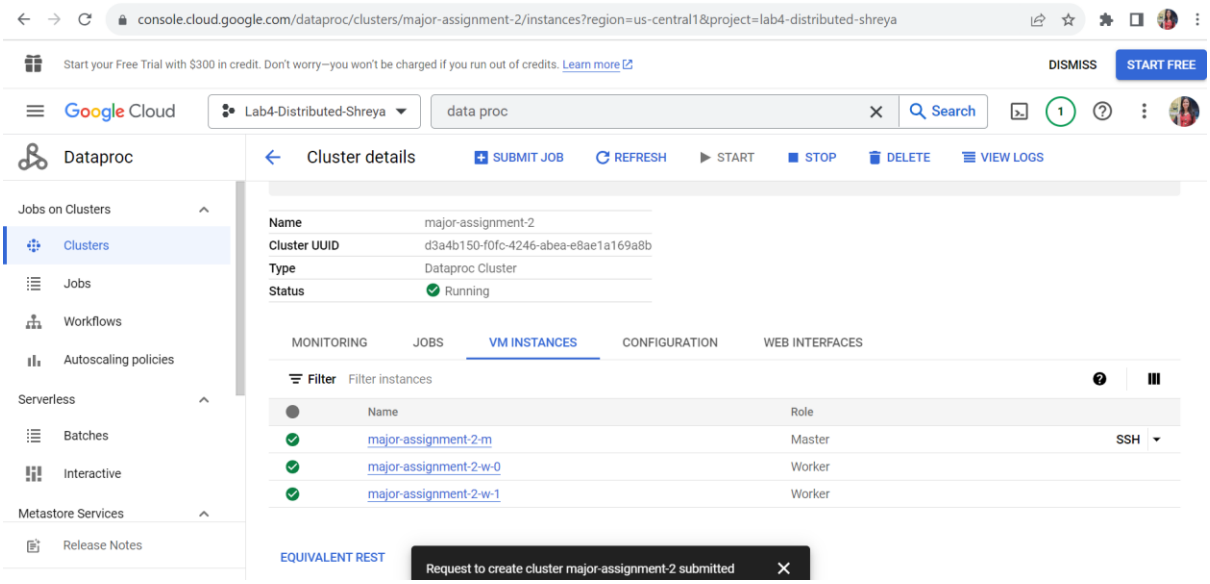
Task 1B: Reuter News Data Processing using Spark

As part of this task, we have to count the frequency of the words in the reuter file. For that, I have configured my GCP with the following steps:

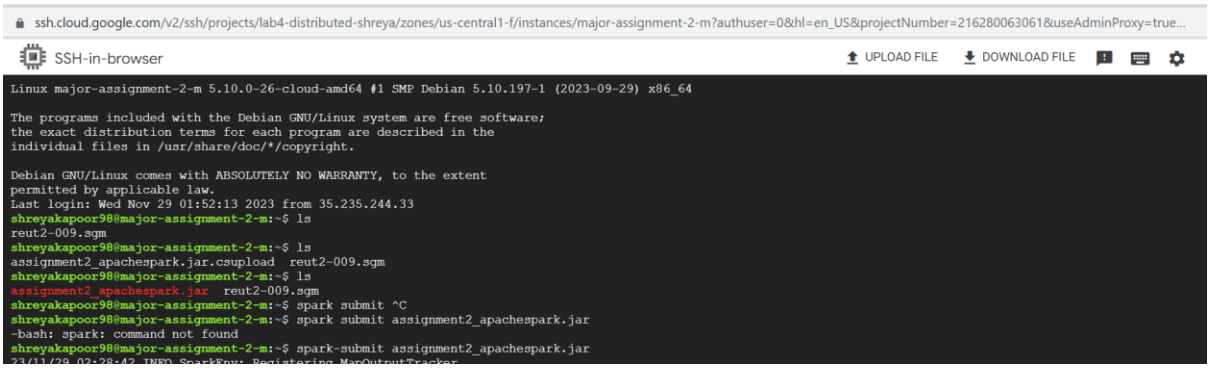
To setup the apache spark, I logged in to my Google Cloud Platform [1] and followed the following steps:

1. Searched for Dataproc in it, which is basically a GCP managed service for spark [2].

2. Then I created a Cluster on computer engine.



3. After creating the cluster, I selected the VM instances and opened the SSH terminal in the browser.



Problem 2: Sentiment Analysis using BOW model on title of Reuters News Article

For this task, we had to perform the processing on the titles content which was present inside the SGM file.

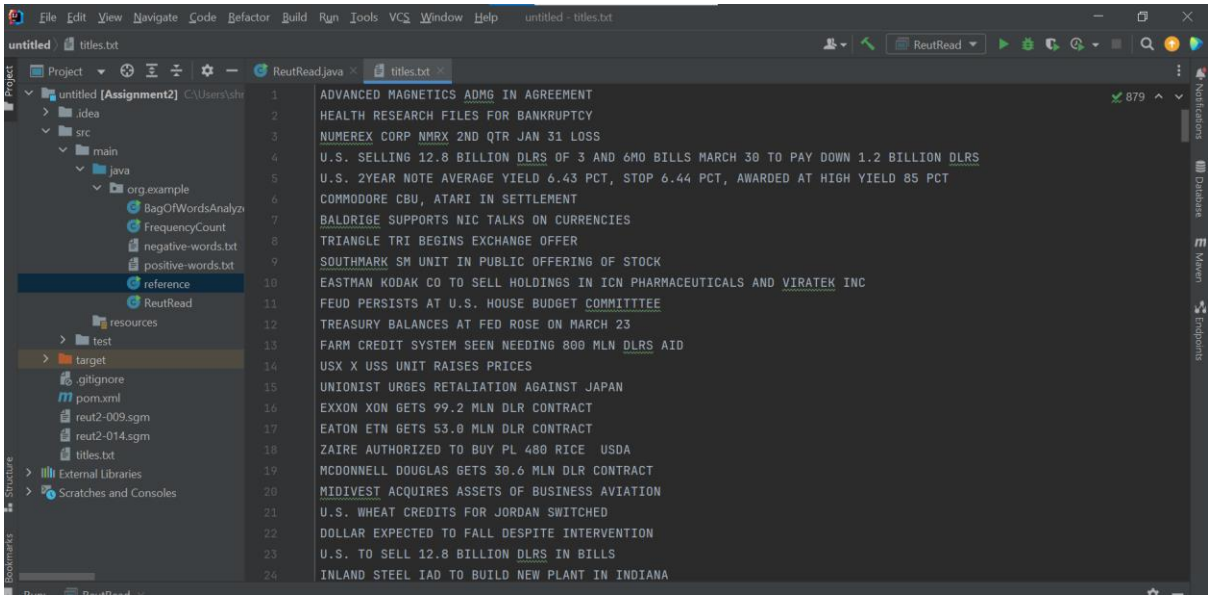
So, I created a function saveTitlesToFile to create a file named titles.txt which contains all the titles which we get after performing the data cleaning.

```
private static void saveTitlesToFile(List<String> titles, String filePath)
{
    try (Writer writer = new FileWriter(filePath)) {
        for (String title : titles) {
            writer.write(title + "\n");
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

```
private static List<String> processSgmFileAndGetTitles(String fileName,
MongoDatabase database) throws IOException {
    List<String> titles = new ArrayList<>();
    try (BufferedReader fileReader = new BufferedReader(new
FileReader(fileName))) {
        StringBuilder fileContent = new StringBuilder();
        String line;
        while ((line = fileReader.readLine()) != null) {
            fileContent.append(line).append("");
        }
        String[] reutersData = fileContent.toString().split("</REUTERS>");
        for (String reuterData : reutersData) {
            String newsTitle = cleanText(extractTagContent(reuterData,
"TITLE"));
            if (!newsTitle.isEmpty()) {
                titles.add(newsTitle);
            }
        }
    }
```

```
    return titles;
}
```

By doing this, I have got the output with a titles.txt file which looks like below:



As seen, it has nothing redundant and just contains the content of the titles tag.

Now, as per the problem statement, we have to create a bag of words with a list of positive and negative words and showcase them into a table.

We also need to mark the polarity of it on the basis of its overall score.

To perform this, I have created a new class named BagOfWordsAnalyzer, and followed the following steps into it.

1. Got the list of positive and negative words **mkulakowski2** github repository and saved it into my working directory into 2 separate txt files.
2. Then, I had setup my JDBC connection

```
try {
    Map<String, Boolean> positiveWords =
readWordsFromFile("./src/main/java/org/example/positive-words.txt", true);
    Map<String, Boolean> negativeWords =
readWordsFromFile("./src/main/java/org/example/negative-words.txt", false);
try{
    Class.forName("com.mysql.cj.jdbc.Driver");

    Connection connection = DriverManager.getConnection(
            "jdbc:mysql://localhost:3306/word_analyser" , "root", "8898");

    System.out.println("Connection Established Successfully\n");
```

Once, the connection was successful, I have created a table using a createTable function

```
private static void createTable(Connection connection) throws SQLException
{
```

```
    String createTableSQL = "CREATE TABLE IF NOT EXISTS news_titles (" +
            "id INT AUTO_INCREMENT PRIMARY KEY," +
            "title VARCHAR(255)," +
            "tag VARCHAR(255)," +
            "matched_words TEXT," +
            "score INT" +
            ")";
    try (PreparedStatement preparedStatement =
connection.prepareStatement(createTableSQL)) {
        preparedStatement.executeUpdate();
    }
}
```

With this, I have a table with the above mentioned columns.

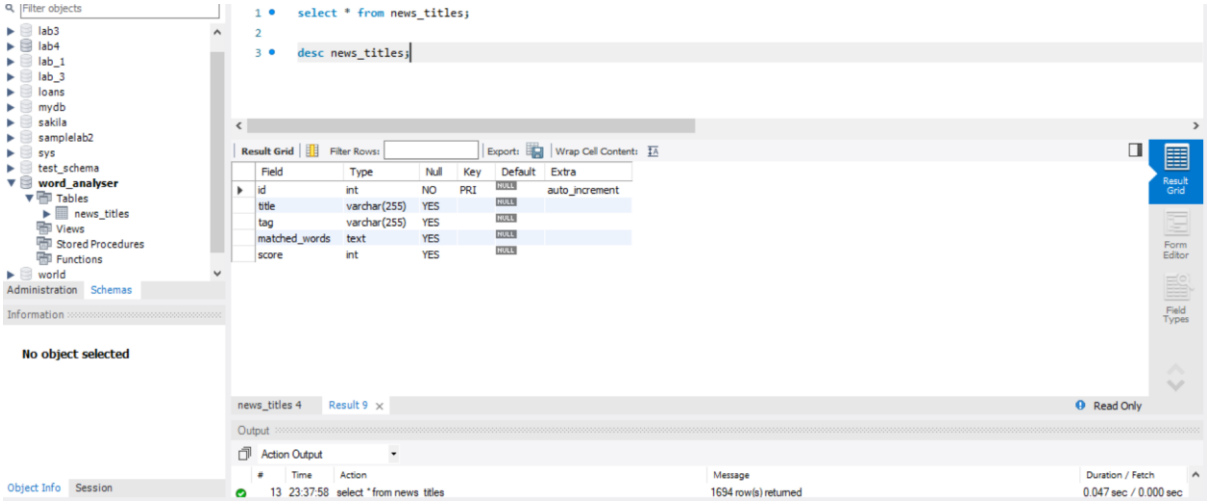


Figure 3: MySQL Workbench Schema

Now, from my titles.txt file, I had to match the records with the positive-words.txt and negative-words.txt file.

For that I have created the following function:

```
private static Map<String, Integer> countMatches(Map<String, Integer>
bagOfWords, Map<String, Boolean> wordsToMatch) {
    Map<String, Integer> matchedWords = new HashMap<>();
    for (String word : bagOfWords.keySet()) {
        if (wordsToMatch.containsKey(word)) {
            matchedWords.put(word, matchedWords.getOrDefault(word, 0) + 1);
        }
    }
    return matchedWords;
}
```

With this I have got a hashmap of matchedWords which is either a positive or a negative word with their count.

```
int positiveCount =
matchedPositiveWords.values().stream().mapToInt(Integer::intValue).sum();
int negativeCount =
matchedNegativeWords.values().stream().mapToInt(Integer::intValue).sum();
```

```
int score = positiveCount - negativeCount;
String tag = determineTag(score);
```

With this, I could find out the total score of a particular title content.

The next step is to determine, whether it will be a Positive, Negative or the Neutral tag on the basis of their score for that I have created a **determineTag** function.

```
private static String determineTag(int score) {
    if (score > 0) {
        return "positive";
    } else if (score < 0) {
        return "negative";
    } else {
        return "neutral";
    }
}
```

It takes the argument as the score and provide an appropriate tag to it.

After this, we are good to store our data into our table. For that I have created the following function.

```
private static void insertIntoDatabase(Connection connection, String title,
String tag, Map<String, Integer> matchedPositiveWords, Map<String, Integer>
matchedNegativeWords, int score) throws SQLException {
    String insertDataSQL = "INSERT INTO news_titles (title, tag,
matched_words, score) VALUES (?, ?, ?, ?)";
    try (PreparedStatement preparedStatement =
connection.prepareStatement(insertDataSQL)) {
        preparedStatement.setString(1, title);
        preparedStatement.setString(2, tag);
        preparedStatement.setString(3,
buildMatchedWordsString(matchedPositiveWords, matchedNegativeWords));
        preparedStatement.setInt(4, score);
        preparedStatement.executeUpdate();
    }
}
```

After inserting the data into the table, my table looks like below:

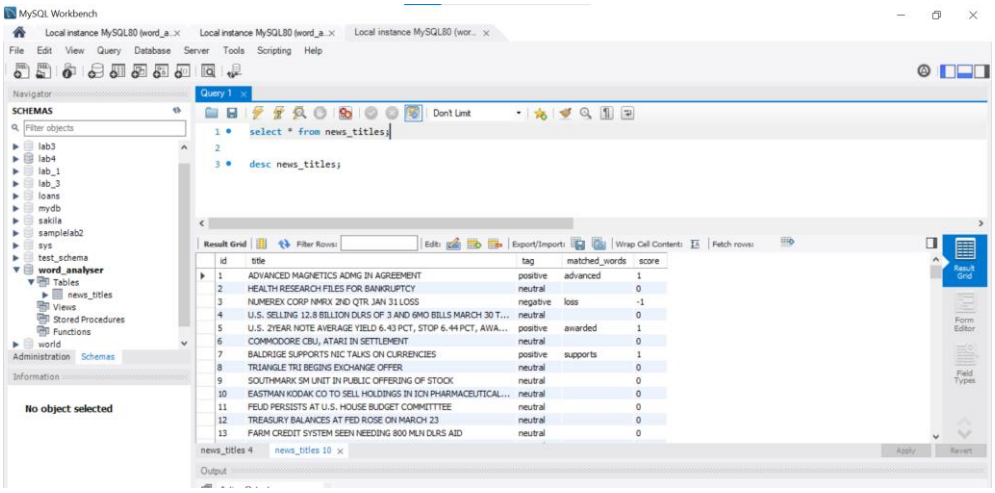


Figure 4: Data inserted into SQL

References:

[1] “MySQL :: MySQL Community Downloads”. Available: https://dev.mysql.com/downloads/.