CSCI 5408
DATA MANAGEMENT AND WAREHOUSING

DALHOUSIE UNIVERSITY

ASSIGNMENT - 2

Submitted By:	Kenil Shaileshkumar Patel (kenil.patel@dal.ca)
Banner ID:	B00954251
Submitted On:	December 1, 2023

Gitlab Repository Link

https://git.cs.dal.ca/kenil/csci5408_f23_b00954251_kenil_patel/-/tree/main/A2

Table of Contents

1. Problem Statement 1A

From the two given news files (reut2-009.sgm, and reut2-014.sgm), create MongoDb Database – ReuterDb, where each Document contains a news article. The task must be done using a Java Program “ReutRead.java”.

Created the MongoDB database named NewsDb and with collection news where all the news files documents will be created by the java code.

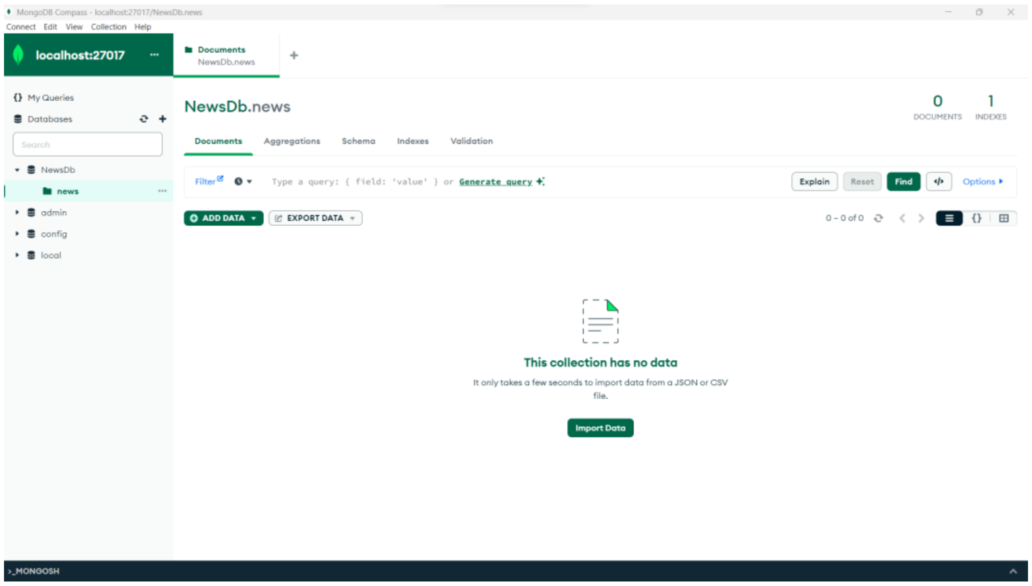
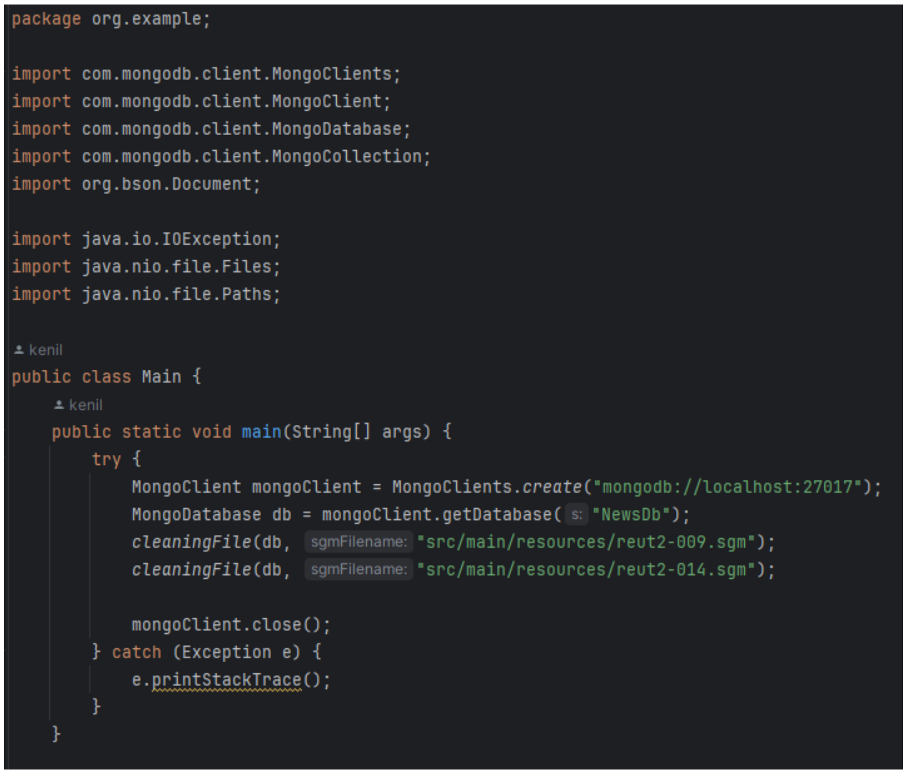

MongoDB Compass - localhost:27017/NewsDb.news
Connect Edit View Collection Help
localhost:27017
Documents NewsDb.news
My Queries
Databases
Search
NewsDb
news
admin
config
local
NewsDb.news
0 DOCUMENTS
1 INDEXES
Documents Aggregations Schema Indexes Validation
Filter
Type a query: { field: 'value' } or Generate query
Explain Reset Find Options
ADD DATA
EXPORT DATA
0 – 0 of 0
This collection has no data
It only takes a few seconds to import data from a JSON or CSV file.
Import Data
>_MONGOSH

Figure 1: MongoDB Database NewsDb is created.



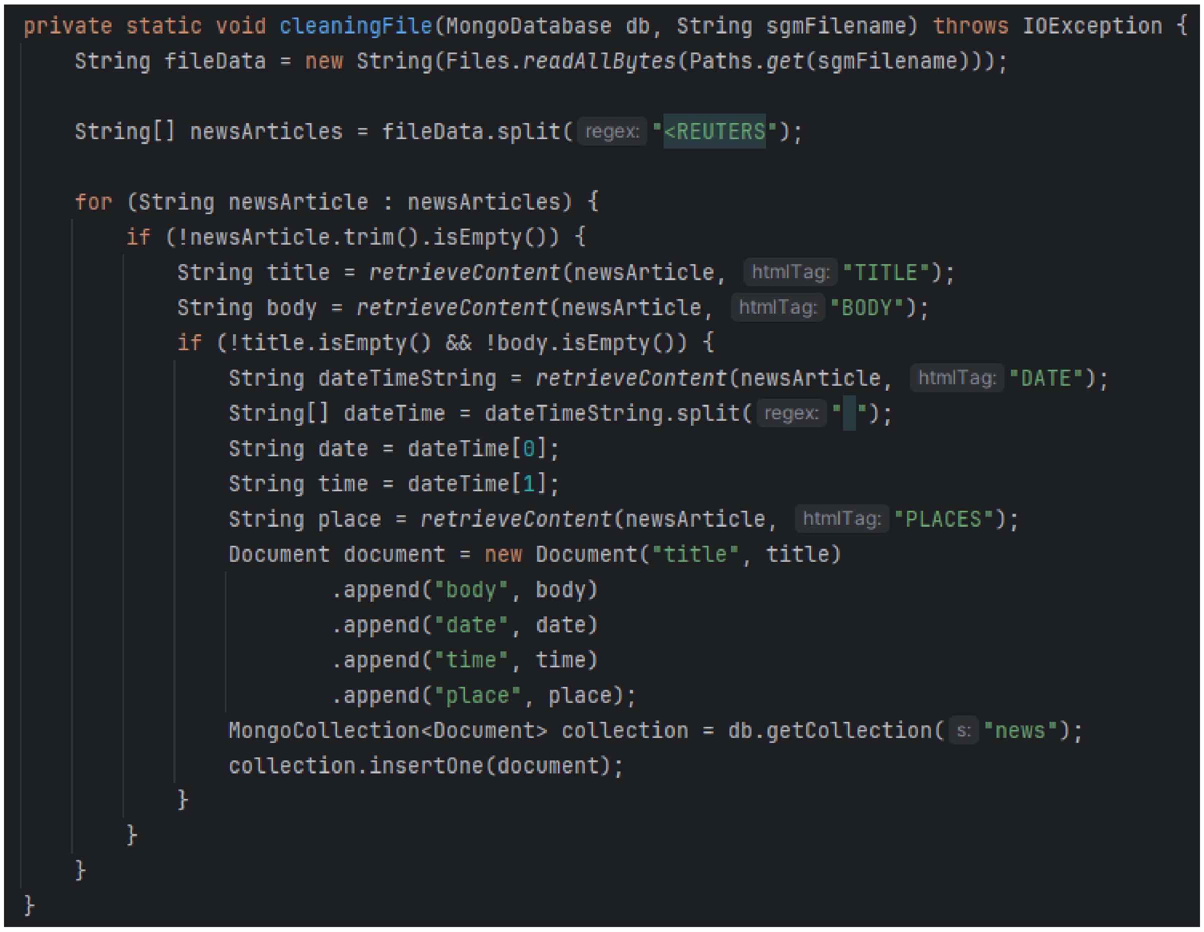
```
package org.example;

import com.mongodb.client.MongoClients;
import com.mongodb.client.MongoClient;
import com.mongodb.client.MongoDatabase;
import com.mongodb.client.MongoCollection;
import org.bson.Document;

import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;

kenil
public class Main {
    kenil
    public static void main(String[] args) {
        try {
            MongoClient mongoClient = MongoClients.create("mongodb://localhost:27017");
            MongoDatabase db = mongoClient.getDatabase( s: "NewsDb");
            cleaningFile(db,  sgmFilename: "src/main/resources/reut2-009.sgm");
            cleaningFile(db,  sgmFilename: "src/main/resources/reut2-014.sgm");

            mongoClient.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

Figure 2: Main function of the Class

1.1 To perform this operation, you need to write a Java code to scan the required texts between <TITLE></TITLE>tags, and <BODY></BODY>tags which are inside each <REUTER></REUTER>tag.

The Java code comprises of the two function cleaningFile and retrieveContent. These functions are made to parse a file containing news items in a particular format and retrieve relevant information, including the article's title, body, date, time, and location. Reading the file, dividing it according to <REUTERS> tags, and parsing each news story to extract particular text contained in HTML-like tags like <TITLE>, <BODY>, <DATE>, and <PLACES> are the steps in the process.

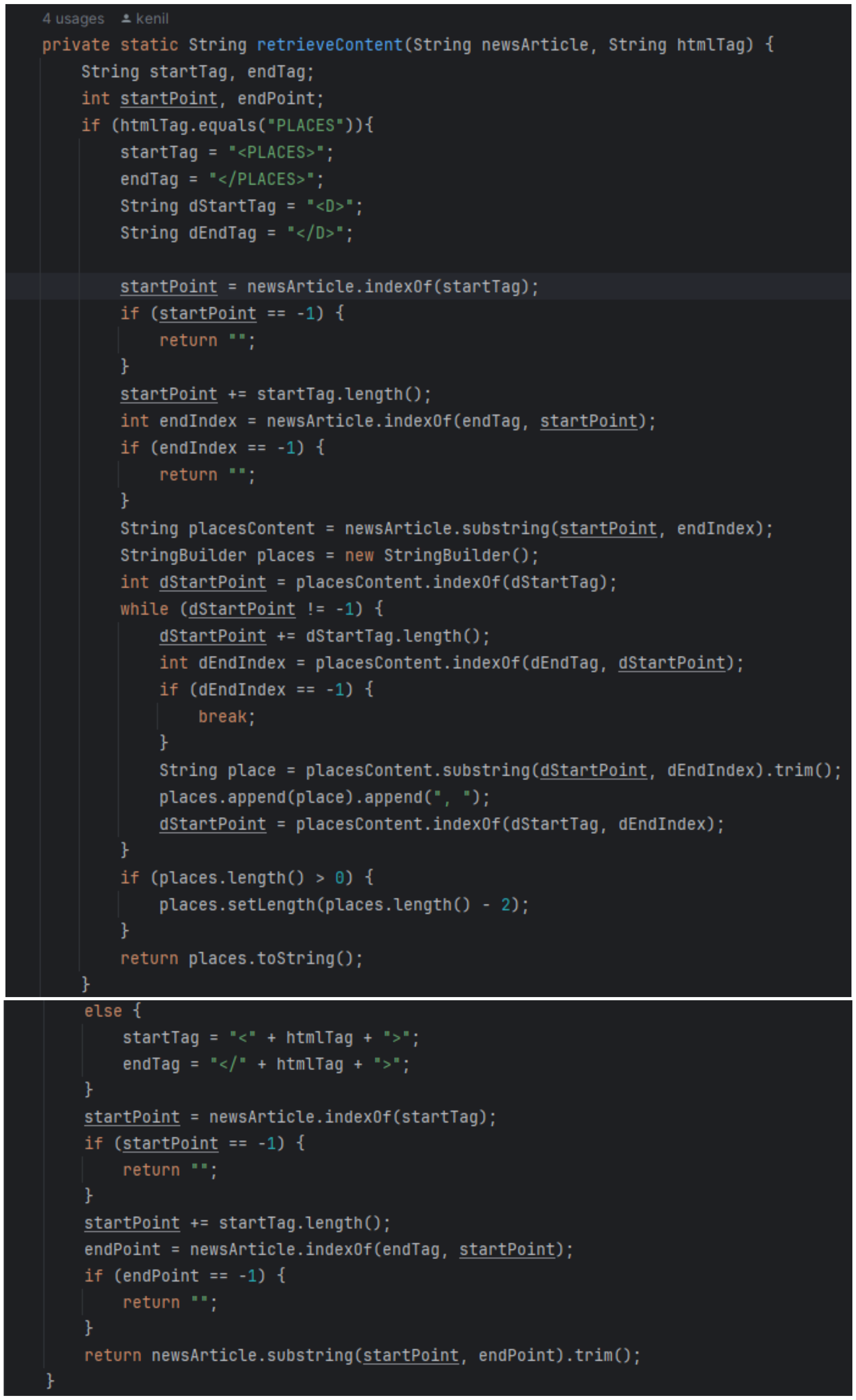
cleaningFile: This function manages the file's parsing. After reading the file, it divides the content into separate news pieces according to the <REUTERS> tag. It calls the retrieveContent method to obtain the title, body, date, time, and location for every news article. It creates a MongoDB Document with this data and puts it into the designated news collection in the NewsDb database if the title and body are not empty.



```java
private static void cleaningFile(MongoDatabase db, String sgmFilename) throws IOException {
    String fileData = new String(Files.readAllBytes(Paths.get(sgmFilename)));

    String[] newsArticles = fileData.split("<REUTERS");

    for (String newsArticle : newsArticles) {
        if (!newsArticle.trim().isEmpty()) {
            String title = retrieveContent(newsArticle, "TITLE");
            String body = retrieveContent(newsArticle, "BODY");
            if (!title.isEmpty() && !body.isEmpty()) {
                String dateTimeString = retrieveContent(newsArticle, "DATE");
                String[] dateTime = dateTimeString.split(" ");
                String date = dateTime[0];
                String time = dateTime[1];
                String place = retrieveContent(newsArticle, "PLACES");
                Document document = new Document("title", title)
                        .append("body", body)
                        .append("date", date)
                        .append("time", time)
                        .append("place", place);
                MongoCollection<Document> collection = db.getCollection("news");
                collection.insertOne(document);
            }
        }
    }
}
```

Figure 3: cleaningFile function.

retrieveContent: An HTML element and a news article string are the inputs for this method. The material encased between these tags is extracted after identifying the start and end tags linked to the given HTML element. When the HTML element is "PLACES," it handles nested <D> tags to create a string of locations by retrieving the place data in a particular format. For other HTML tags (like "TITLE," "BODY," and so on), it pulls out the text contained in between the designated tags.



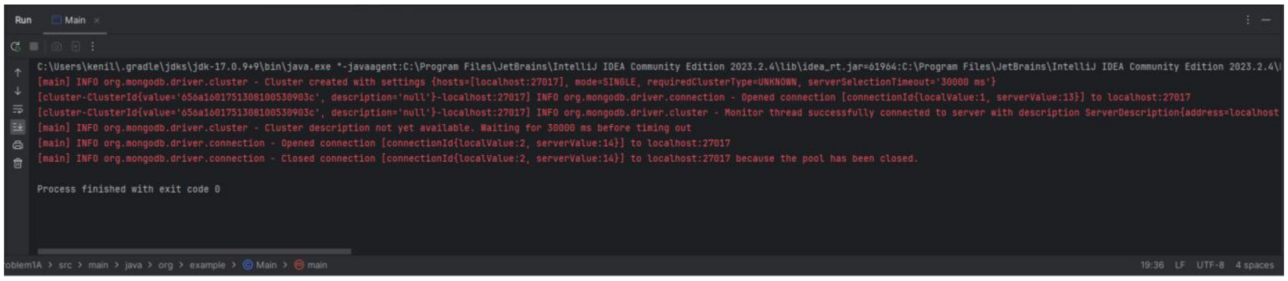
```
4 usages  kenil
private static String retrieveContent(String newsArticle, String htmlTag) {
    String startTag, endTag;
    int startPoint, endPoint;
    if (htmlTag.equals("PLACES")){
        startTag = "<PLACES>";
        endTag = "</PLACES>";
        String dStartTag = "<D>";
        String dEndTag = "</D>";

        startPoint = newsArticle.indexOf(startTag);
        if (startPoint == -1) {
            return "";
        }
        startPoint += startTag.length();
        int endIndex = newsArticle.indexOf(endTag, startPoint);
        if (endIndex == -1) {
            return "";
        }
        String placesContent = newsArticle.substring(startPoint, endIndex);
        StringBuilder places = new StringBuilder();
        int dStartPoint = placesContent.indexOf(dStartTag);
        while (dStartPoint != -1) {
            dStartPoint += dStartTag.length();
            int dEndIndex = placesContent.indexOf(dEndTag, dStartPoint);
            if (dEndIndex == -1) {
                break;
            }
            String place = placesContent.substring(dStartPoint, dEndIndex).trim();
            places.append(place).append(", ");
            dStartPoint = placesContent.indexOf(dStartTag, dEndIndex);
        }
        if (places.length() > 0) {
            places.setLength(places.length() - 2);
        }
        return places.toString();
    }
    else {
        startTag = "<" + htmlTag + ">";
        endTag = "</" + htmlTag + ">";
    }
    startPoint = newsArticle.indexOf(startTag);
    if (startPoint == -1) {
        return "";
    }
    startPoint += startTag.length();
    endPoint = newsArticle.indexOf(endTag, startPoint);
    if (endPoint == -1) {
        return "";
    }
    return newsArticle.substring(startPoint, endPoint).trim();
}
```

Figure 4: retrieveContent Function

1.2 In the ReuterDb, you may consider each news as a document. You can also include nested or subdocument.

The two files, "reut2-009.sgm" and "reut2-014.sgm," are processed in order by the program when the main file is executed by sending each separately to the cleaningFile method. Each file's contents are read and parsed within this process to retrieve news articles. Relevant data, including the title, body, date, time, and location, are retrieved for each article that can be located in these files. After that, these data are organized into MongoDB Document objects.

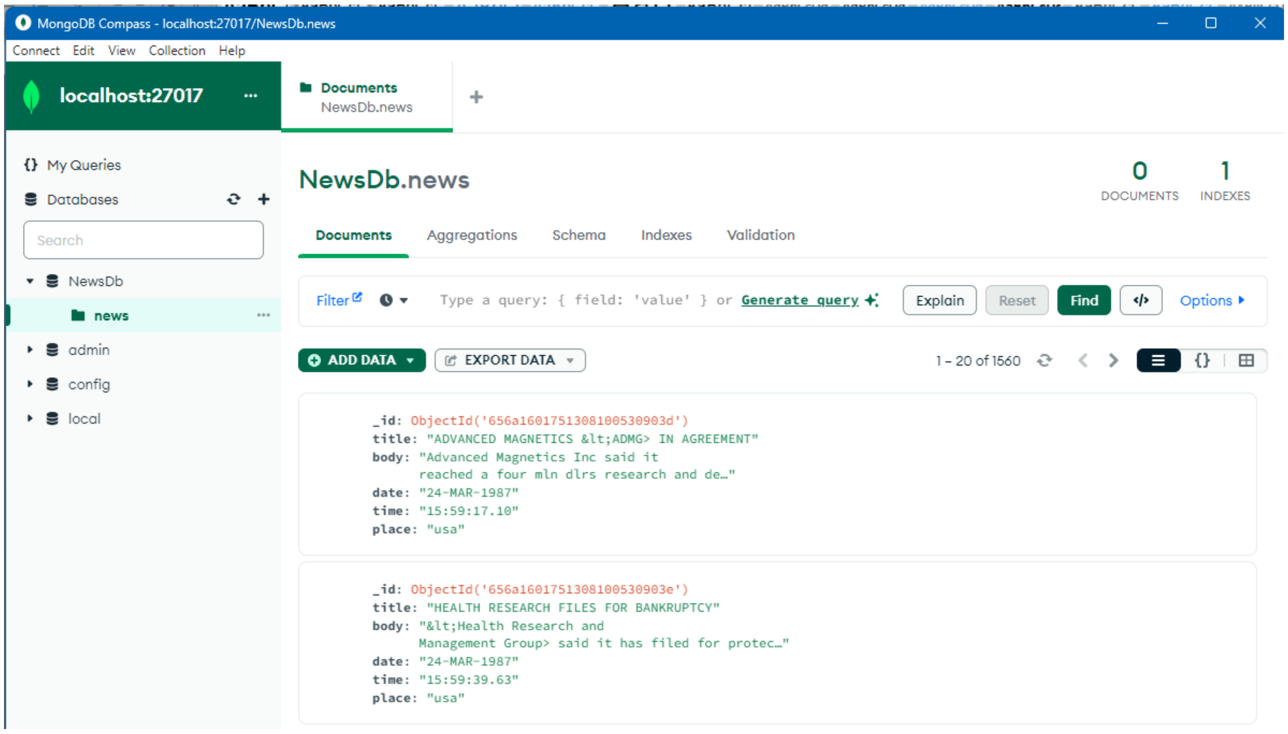
After that, the created documents - each representing a news story - are added to the MongoDB database. The program creates a collection of news items in the MongoDB storage.



```
C:\Users\kenil\.gradle\jdks\jdk-17.0.9+9\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2023.2.4\lib\idea_rt.jar=61964:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2023.2.4\
[main] INFO org.mongodb.driver.cluster - Cluster created with settings {hosts=[localhost:27017], mode=SINGLE, requiredClusterType=UNKNOWN, serverSelectionTimeout='30000 ms'}
[cluster-ClusterId{value='656a16017513081005309 03c', description='null'}-localhost:27017] INFO org.mongodb.driver.connection - Opened connection [connectionId{localValue:1, serverValue:13}] to localhost:27017
[cluster-ClusterId{value='656a160175130810053090 3c', description='null'}-localhost:27017] INFO org.mongodb.driver.cluster - Monitor thread successfully connected to server with description ServerDescription{address=localhost
[main] INFO org.mongodb.driver.cluster - Cluster description not yet available. Waiting for 30000 ms before timing out
[main] INFO org.mongodb.driver.connection - Opened connection [connectionId{localValue:2, serverValue:14}] to localhost:27017
[main] INFO org.mongodb.driver.connection - Closed connection [connectionId{localValue:2, serverValue:14}] to localhost:27017 because the pool has been closed.

Process finished with exit code 0
```

Figure 5: Execution of the Main Function.

There are a total of 1560 news documents stored in the MongoDB storage.



MongoDB Compass - localhost:27017/NewsDb.news
Connect Edit View Collection Help
localhost:27017
Documents
NewsDb.news
My Queries
Databases
Search
NewsDb
news
admin
config
local
NewsDb.news
0 DOCUMENTS
1 INDEXES
Documents Aggregations Schema Indexes Validation
Filter
Type a query: { field: 'value' } or Generate query
Explain Reset Find Options
ADD DATA
EXPORT DATA
1 – 20 of 1560

```
_id: ObjectId('656a160175130810053090 3d')
title: "ADVANCED MAGNETICS &lt;ADMG> IN AGREEMENT"
body: "Advanced Magnetics Inc said it
      reached a four mln dlrs research and de…"
date: "24-MAR-1987"
time: "15:59:17.10"
place: "usa"

_id: ObjectId('656a160175130810053090 3e')
title: "HEALTH RESEARCH FILES FOR BANKRUPTCY"
body: "&lt;Health Research and
      Management Group> said it has filed for protec…"
date: "24-MAR-1987"
time: "15:59:39.63"
place: "usa"
```

Figure 6: News Data stored in MongoDB.

1.3 You need to include a flowchart and algorithm of your Reuters Data cleaning / transformation program on the PDF file.

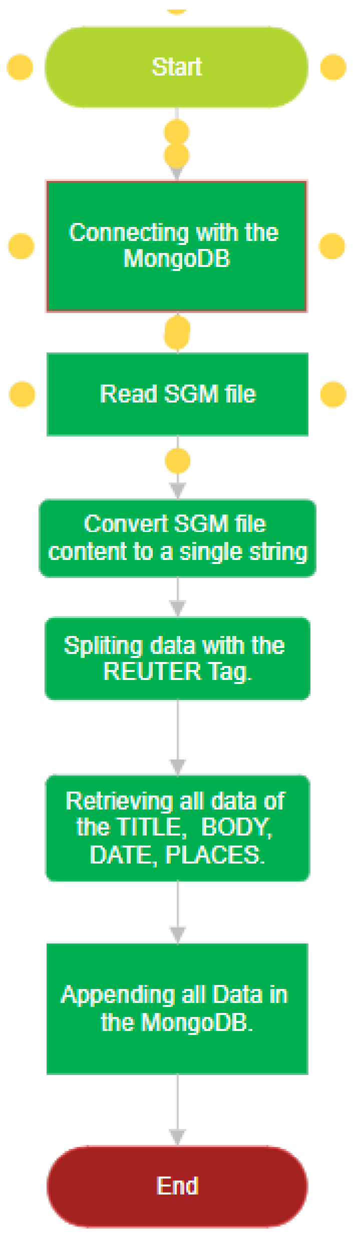
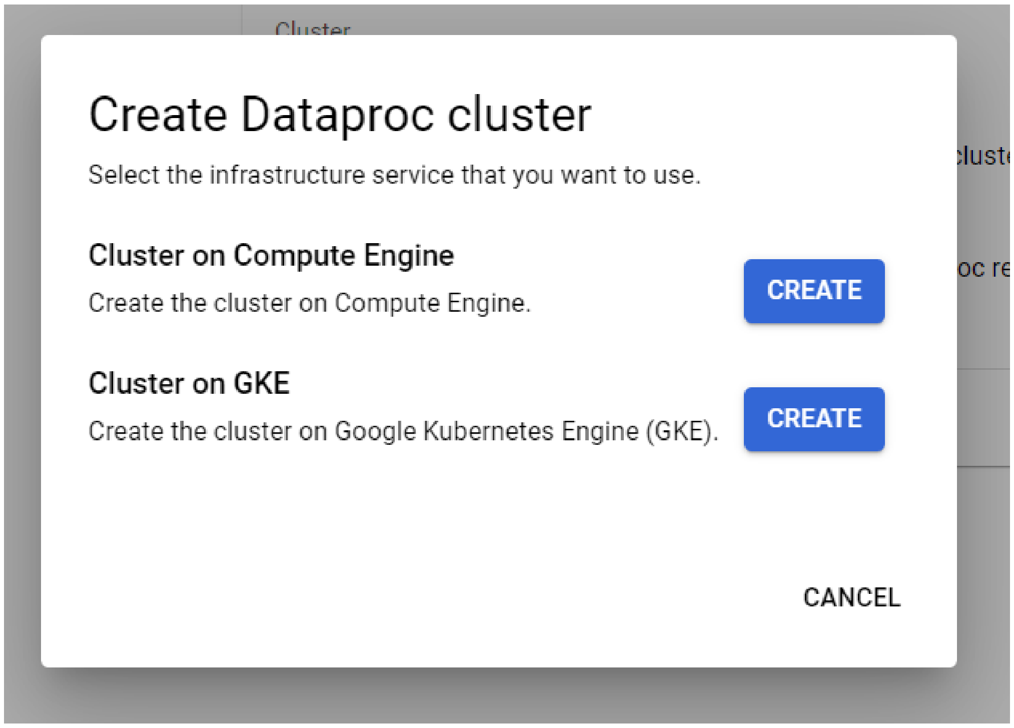

Start
Connecting with the MongoDB
Read SGM file
Convert SGM file content to a single string
Spliting data with the REUTER Tag.
Retrieving all data of the TITLE, BODY, DATE, PLACES.
Appending all Data in the MongoDB.
End

Figure 7: Flowchart of cleaning and transforming Data.

2. Problem Statement 1B

2.1 Using your GCP cloud account, configure and initialize Apache Spark cluster.

Creating Cluster named problem1b on GCP.



Create Dataproc cluster

Select the infrastructure service that you want to use.

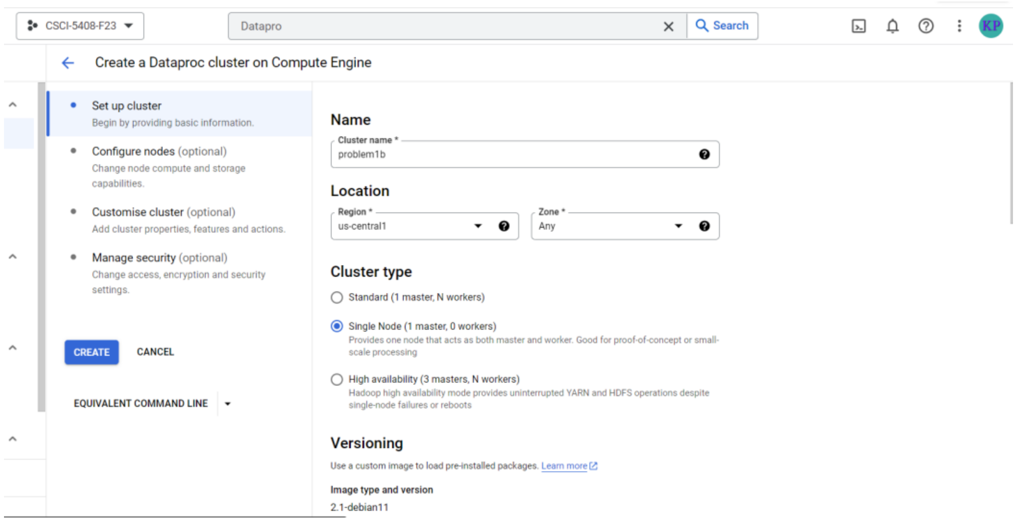
Cluster on Compute Engine

Create the cluster on Compute Engine.

Cluster on GKE

Create the cluster on Google Kubernetes Engine (GKE).

Figure 8: Creating Dataproc Cluster on the Compute Engine.



Create a Dataproc cluster on Compute Engine

- Set up cluster — Begin by providing basic information.
- Configure nodes (optional) — Change node compute and storage capabilities.
- Customise cluster (optional) — Add cluster properties, features and actions.
- Manage security (optional) — Change access, encryption and security settings.

Name

Cluster name *: problem1b

Location

Region *: us-central1

Zone *: Any

Cluster type

( ) Standard (1 master, N workers)

[✓ Single Node (1 master, 0 workers)] Provides one node that acts as both master and worker. Good for proof-of-concept or small-scale processing

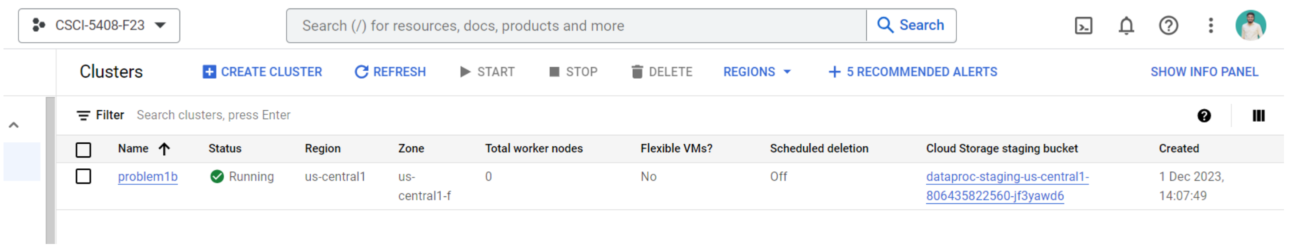
( ) High availability (3 masters, N workers) Hadoop high availability mode provides uninterrupted YARN and HDFS operations despite single-node failures or reboots

Versioning

Use a custom image to load pre-installed packages. Learn more

Image type and version: 2.1-debian11

Figure 9: Creating Single Node Cluster.


CSCI-5408-F23

Clusters

CREATE CLUSTER · REFRESH · START · STOP · DELETE · REGIONS · 5 RECOMMENDED ALERTS · SHOW INFO PANEL

Name	Status	Region	Zone	Total worker nodes	Flexible VMs?	Scheduled deletion	Cloud Storage staging bucket	Created
problem1b	Running	us-central1	us-central1-f	0	No	Off	dataproc-staging-us-central1-806435822560-jf3yawd6	1 Dec 2023, 14:07:49

Figure 10: Cluster is running.

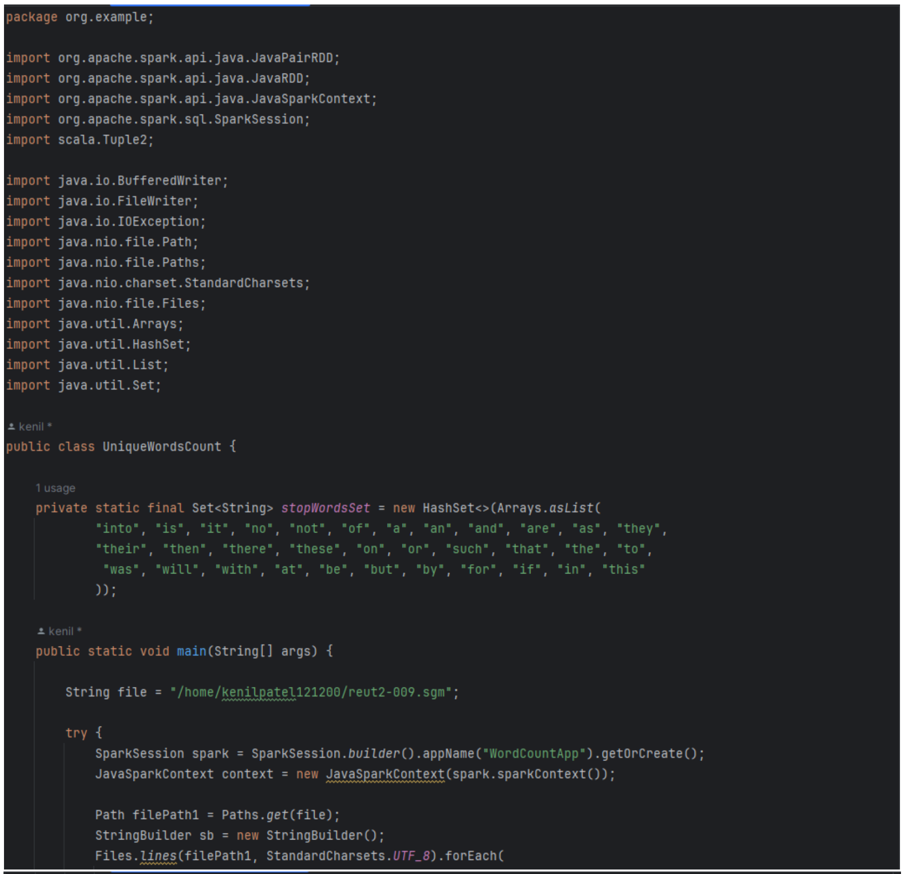
2.2 Write ½ page explanation on how you completed the task.

Counting unique words, identifying words that occur only once, calculating the total amount of unique words, identifying the word with the highest frequency, and storing the unique words in a file are all completed using Java code.

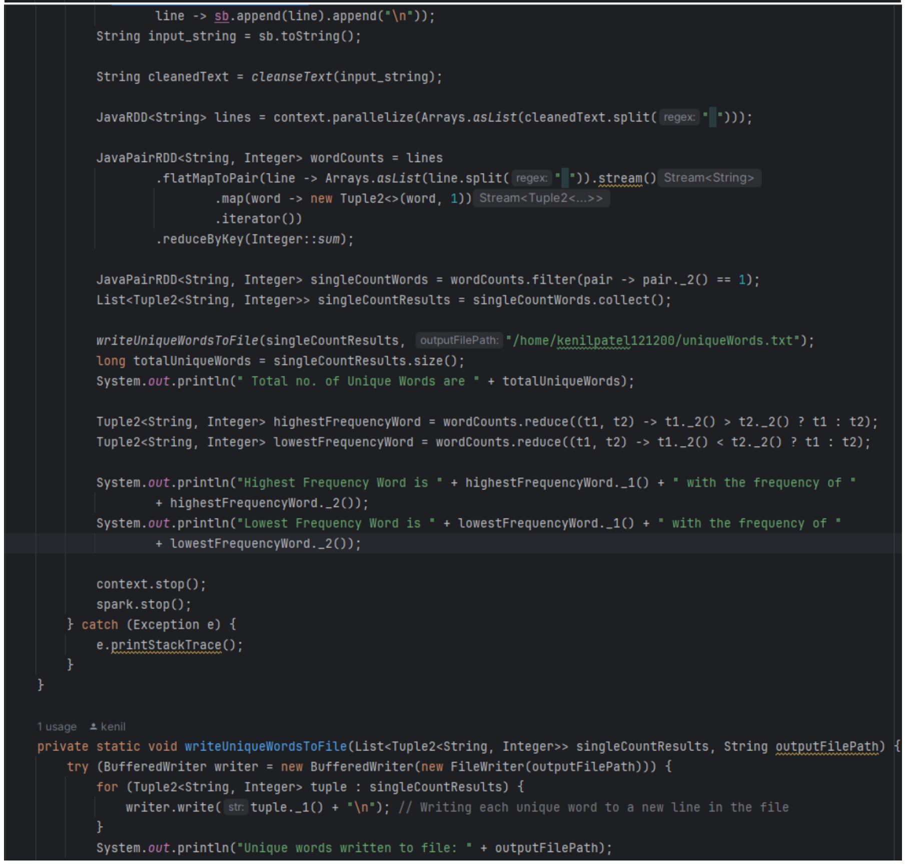
Flow of the Code:

1. **Spark Setup and Initialization:**
   - The code uses Apache Spark to process the text data.
   - It creates a SparkSession and JavaSparkContext.
2. **Text Pre-Processing:**
   - Reads the contents of a file (**reut2-009.sgm** in this case) into a single string.
   - Cleanses the text by removing HTML tags, non-alphabetic characters, single-letter words, and stop words which are given in a HashSet.
3. **Word Count and Analysis:**
   - Splits the cleaned text into words and creates a Spark RDD (Resilient Distributed Dataset) of words.
   - Maps each word to a key-value pair (word, 1) and reduces by key to get word counts.
   - Filters words with a count of 1 to find unique words that occur only once.
   - Collects the unique words and writes them to a file (**uniqueWords.txt**).
4. **Output:**
   - Calculates and displays the count of unique words.
   - Determines the word with the highest frequency (maximum count) and the word with the lowest frequency (minimum count).
5. **Spark Context Shutdown:**
   - Stops the Spark context and closes the Spark session.

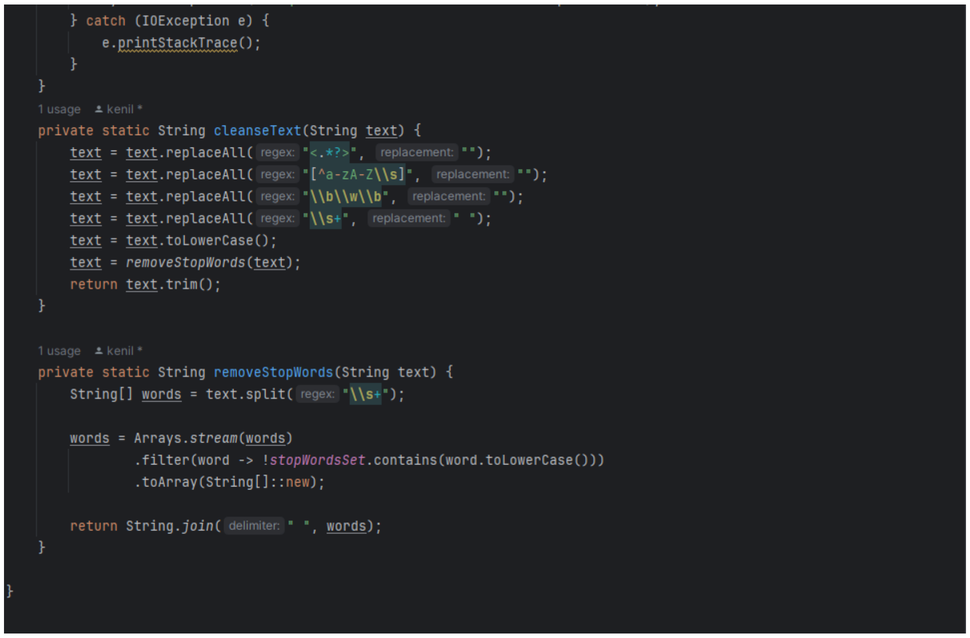
Below is the java code



```java
package org.example;

import org.apache.spark.api.java.JavaPairRDD;
import org.apache.spark.api.java.JavaRDD;
import org.apache.spark.api.java.JavaSparkContext;
import org.apache.spark.sql.SparkSession;
import scala.Tuple2;

import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.util.Arrays;
import java.util.HashSet;
import java.util.List;
import java.util.Set;

public class UniqueWordsCount {

    private static final Set<String> stopWordsSet = new HashSet<>(Arrays.asList(
            "into", "is", "it", "no", "not", "of", "a", "an", "and", "are", "as", "they",
            "their", "then", "there", "these", "on", "or", "such", "that", "the", "to",
             "was", "will", "with", "at", "be", "but", "by", "for", "if", "in", "this"
            ));

    public static void main(String[] args) {

        String file = "/home/kenilpatel121200/reut2-009.sgm";

        try {
            SparkSession spark = SparkSession.builder().appName("WordCountApp").getOrCreate();
            JavaSparkContext context = new JavaSparkContext(spark.sparkContext());

            Path filePath1 = Paths.get(file);
            StringBuilder sb = new StringBuilder();
            Files.lines(filePath1, StandardCharsets.UTF_8).forEach(
```



```java
                    line -> sb.append(line).append("\n"));
            String input_string = sb.toString();

            String cleanedText = cleanseText(input_string);

            JavaRDD<String> lines = context.parallelize(Arrays.asList(cleanedText.split(" ")));

            JavaPairRDD<String, Integer> wordCounts = lines
                    .flatMapToPair(line -> Arrays.asList(line.split(" ")).stream()
                            .map(word -> new Tuple2<>(word, 1))
                            .iterator())
                    .reduceByKey(Integer::sum);

            JavaPairRDD<String, Integer> singleCountWords = wordCounts.filter(pair -> pair._2() == 1);
            List<Tuple2<String, Integer>> singleCountResults = singleCountWords.collect();

            writeUniqueWordsToFile(singleCountResults, "/home/kenilpatel121200/uniqueWords.txt");
            long totalUniqueWords = singleCountResults.size();
            System.out.println(" Total no. of Unique Words are " + totalUniqueWords);

            Tuple2<String, Integer> highestFrequencyWord = wordCounts.reduce((t1, t2) -> t1._2() > t2._2() ? t1 : t2);
            Tuple2<String, Integer> lowestFrequencyWord = wordCounts.reduce((t1, t2) -> t1._2() < t2._2() ? t1 : t2);

            System.out.println("Highest Frequency Word is " + highestFrequencyWord._1() + " with the frequency of "
                    + highestFrequencyWord._2());
            System.out.println("Lowest Frequency Word is " + lowestFrequencyWord._1() + " with the frequency of "
                    + lowestFrequencyWord._2());

            context.stop();
            spark.stop();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    private static void writeUniqueWordsToFile(List<Tuple2<String, Integer>> singleCountResults, String outputFilePath) {
        try (BufferedWriter writer = new BufferedWriter(new FileWriter(outputFilePath))) {
            for (Tuple2<String, Integer> tuple : singleCountResults) {
                writer.write(tuple._1() + "\n"); // Writing each unique word to a new line in the file
            }
            System.out.println("Unique words written to file: " + outputFilePath);
```

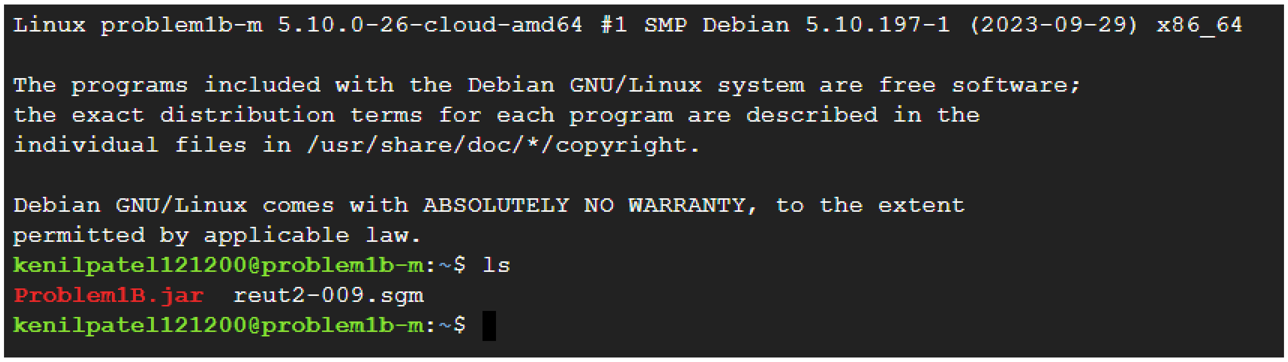


```java
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    1 usage  kenil *
    private static String cleanseText(String text) {
        text = text.replaceAll( regex: "<.*?>",  replacement: "");
        text = text.replaceAll( regex: "[^a-zA-Z\\s]",  replacement: "");
        text = text.replaceAll( regex: "\\b\\w\\b",  replacement: "");
        text = text.replaceAll( regex: "\\s+",  replacement: " ");
        text = text.toLowerCase();
        text = removeStopWords(text);
        return text.trim();
    }

    1 usage  kenil *
    private static String removeStopWords(String text) {
        String[] words = text.split( regex: "\\s+");

        words = Arrays.stream(words)
                .filter(word -> !stopWordsSet.contains(word.toLowerCase()))
                .toArray(String[]::new);

        return String.join( delimiter: " ", words);
    }

}
```

Figure 11: Java code of unique word count.

2.3 Using Apache Spark count (frequency count) the unique words found in “reut2 009.sgm”.

Uploaded the Jar file and reut2-009.sgm file on the Cluster.



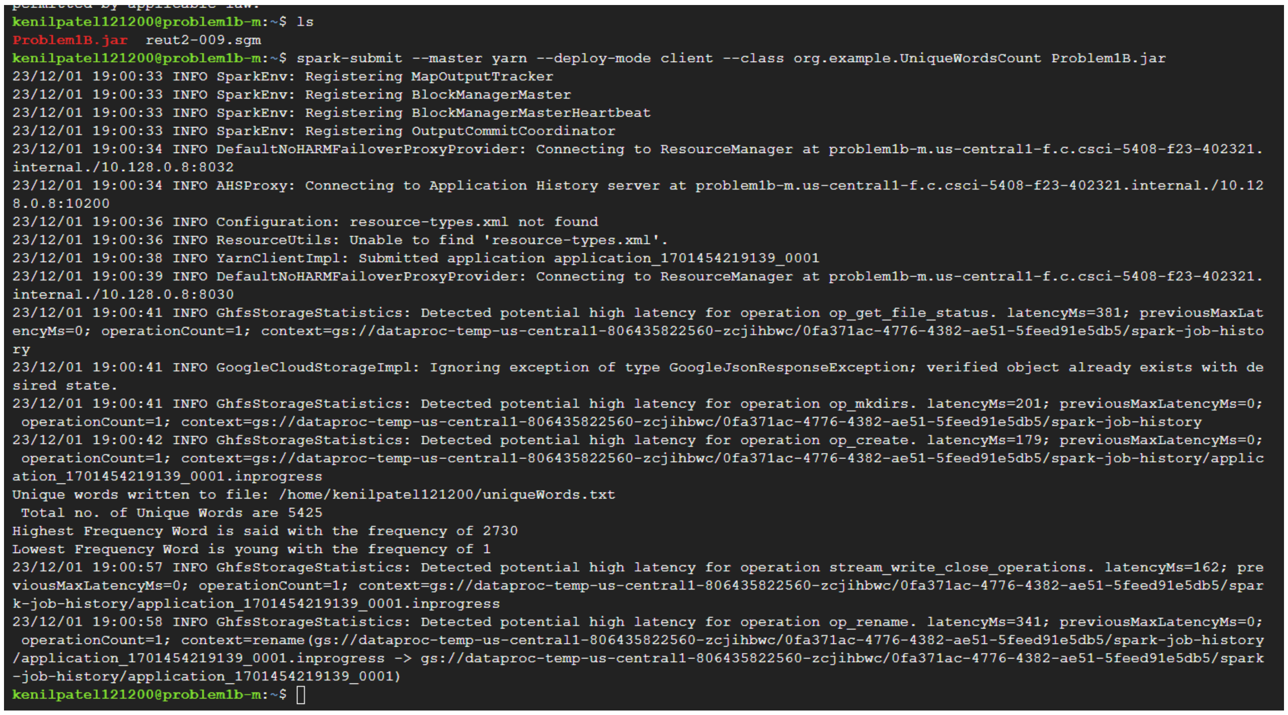
```
Linux problem1b-m 5.10.0-26-cloud-amd64 #1 SMP Debian 5.10.197-1 (2023-09-29) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
kenilpatel121200@problem1b-m:~$ ls
Problem1B.jar  reut2-009.sgm
kenilpatel121200@problem1b-m:~$
```

Figure 12: Jar and Sgm file is on cluster

After that running the uploaded Jar file using the below command.

“spark-submit --master yarn --deploy-mode client --class org.example.UniqueWordsCount Problem1B.jar”

On running this command jar file will run and generate the output on console and create a new file in the VM instance named uniqueWords.txt. The file contains all the unique words that are in the sgm file.



```
permitted by applicable law.
kenilpatel121200@problem1b-m:~$ ls
Problem1B.jar  reut2-009.sgm
kenilpatel121200@problem1b-m:~$ spark-submit --master yarn --deploy-mode client --class org.example.UniqueWordsCount Problem1B.jar
23/12/01 19:00:33 INFO SparkEnv: Registering MapOutputTracker
23/12/01 19:00:33 INFO SparkEnv: Registering BlockManagerMaster
23/12/01 19:00:33 INFO SparkEnv: Registering BlockManagerMasterHeartbeat
23/12/01 19:00:33 INFO SparkEnv: Registering OutputCommitCoordinator
23/12/01 19:00:34 INFO DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at problem1b-m.us-central1-f.c.csci-5408-f23-402321.
internal./10.128.0.8:8032
23/12/01 19:00:34 INFO AHSProxy: Connecting to Application History server at problem1b-m.us-central1-f.c.csci-5408-f23-402321.internal./10.12
8.0.8:10200
23/12/01 19:00:36 INFO Configuration: resource-types.xml not found
23/12/01 19:00:36 INFO ResourceUtils: Unable to find 'resource-types.xml'.
23/12/01 19:00:38 INFO YarnClientImpl: Submitted application application_1701454219139_0001
23/12/01 19:00:39 INFO DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at problem1b-m.us-central1-f.c.csci-5408-f23-402321.
internal./10.128.0.8:8030
23/12/01 19:00:41 INFO GhfsStorageStatistics: Detected potential high latency for operation op_get_file_status. latencyMs=381; previousMaxLat
encyMs=0; operationCount=1; context=gs://dataproc-temp-us-central1-806435822560-zcjihbwc/0fa371ac-4776-4382-ae51-5feed91e5db5/spark-job-histo
ry
23/12/01 19:00:41 INFO GoogleCloudStorageImpl: Ignoring exception of type GoogleJsonResponseException; verified object already exists with de
sired state.
23/12/01 19:00:41 INFO GhfsStorageStatistics: Detected potential high latency for operation op_mkdirs. latencyMs=201; previousMaxLatencyMs=0;
 operationCount=1; context=gs://dataproc-temp-us-central1-806435822560-zcjihbwc/0fa371ac-4776-4382-ae51-5feed91e5db5/spark-job-history
23/12/01 19:00:42 INFO GhfsStorageStatistics: Detected potential high latency for operation op_create. latencyMs=179; previousMaxLatencyMs=0;
 operationCount=1; context=gs://dataproc-temp-us-central1-806435822560-zcjihbwc/0fa371ac-4776-4382-ae51-5feed91e5db5/spark-job-history/applic
ation_1701454219139_0001.inprogress
Unique words written to file: /home/kenilpatel121200/uniqueWords.txt
 Total no. of Unique Words are 5425
Highest Frequency Word is said with the frequency of 2730
Lowest Frequency Word is young with the frequency of 1
23/12/01 19:00:57 INFO GhfsStorageStatistics: Detected potential high latency for operation stream_write_close_operations. latencyMs=162; pre
viousMaxLatencyMs=0; operationCount=1; context=gs://dataproc-temp-us-central1-806435822560-zcjihbwc/0fa371ac-4776-4382-ae51-5feed91e5db5/spar
k-job-history/application_1701454219139_0001.inprogress
23/12/01 19:00:58 INFO GhfsStorageStatistics: Detected potential high latency for operation op_rename. latencyMs=341; previousMaxLatencyMs=0;
 operationCount=1; context=rename(gs://dataproc-temp-us-central1-806435822560-zcjihbwc/0fa371ac-4776-4382-ae51-5feed91e5db5/spark-job-history
/application_1701454219139_0001.inprogress -> gs://dataproc-temp-us-central1-806435822560-zcjihbwc/0fa371ac-4776-4382-ae51-5feed91e5db5/spark
-job-history/application_1701454219139_0001)
kenilpatel121200@problem1b-m:~$
```

Figure 13: Console output on running Jar file

2.4 You need to include a flowchart/algorithm of your Spark framework frequency count operation.

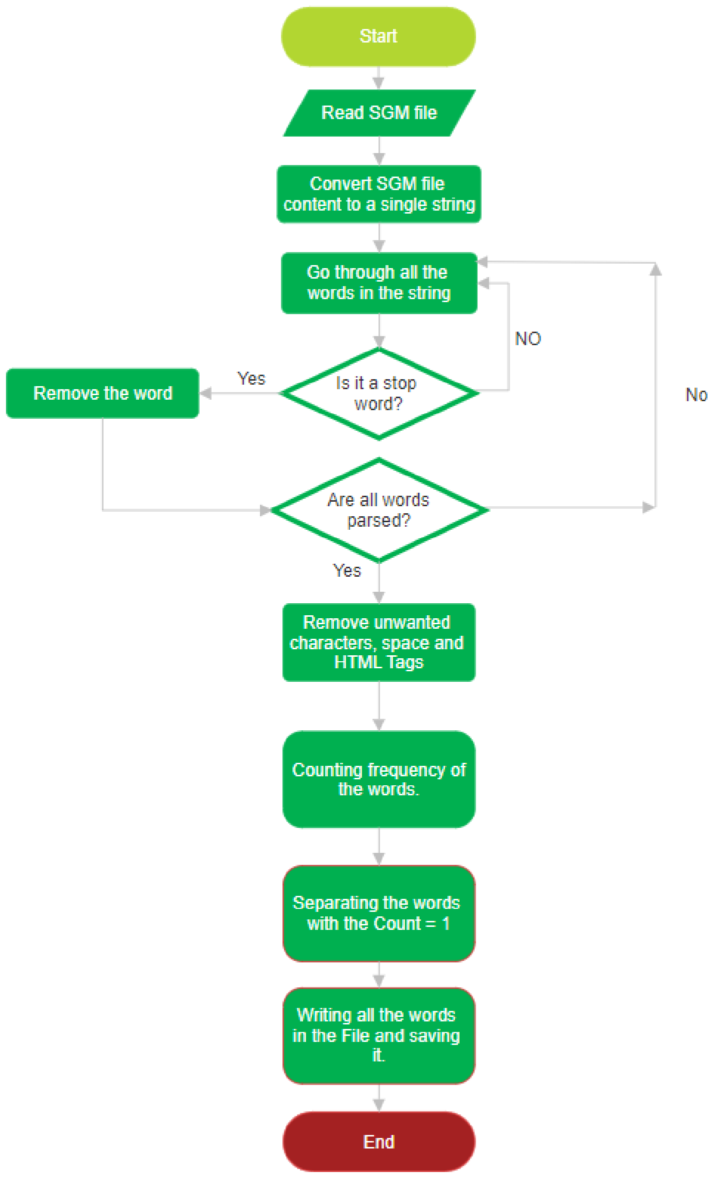
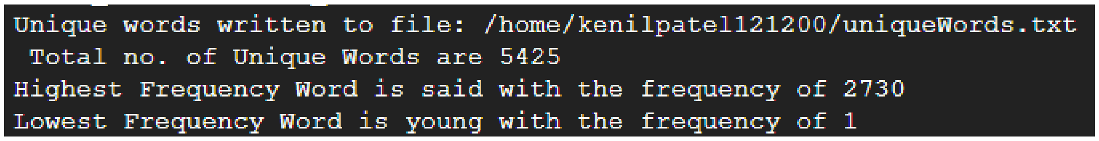

Start
Read SGM file
Convert SGM file content to a single string
Go through all the words in the string
Is it a stop word?
Yes
Remove the word
NO
Are all words parsed?
No
Yes
Remove unwanted characters, space and HTML Tags
Counting frequency of the words.
Separating the words with the Count = 1
Writing all the words in the File and saving it.
End

Figure 14: Flowchart of the frequency count program.

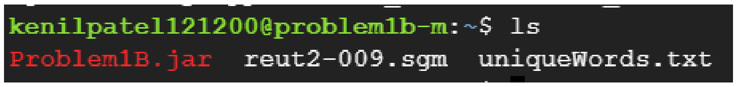
2.5 Mention the words that have highest and lowest frequencies.

The word with the highest frequency word is “said” and frequency count is 2730. And lowest frequency word is young with frequency count is 1.



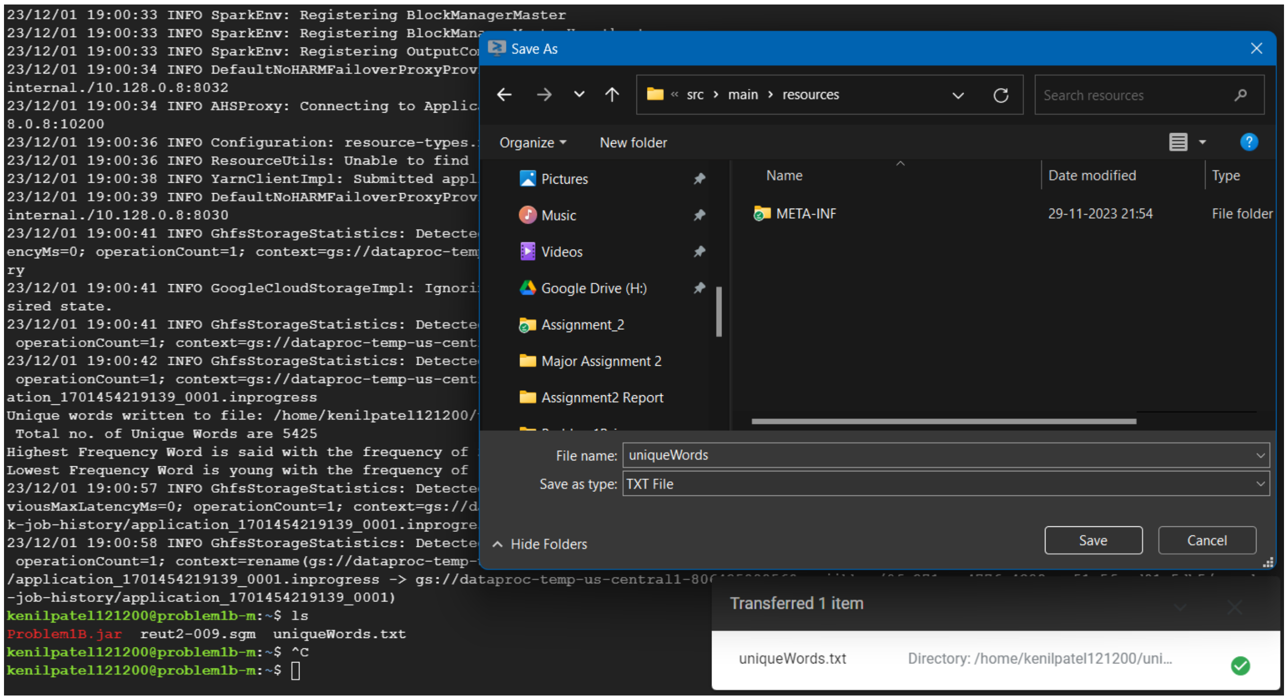
```
Unique words written to file: /home/kenilpatel121200/uniqueWords.txt
 Total no. of Unique Words are 5425
Highest Frequency Word is said with the frequency of 2730
Lowest Frequency Word is young with the frequency of 1
```

Figure 15: Console output of the highest and lowest frequency word.



```
kenilpatel121200@problem1b-m:~$ ls
Problem1B.jar  reut2-009.sgm  uniqueWords.txt
```

Figure 16: Unique word is stored in txt.



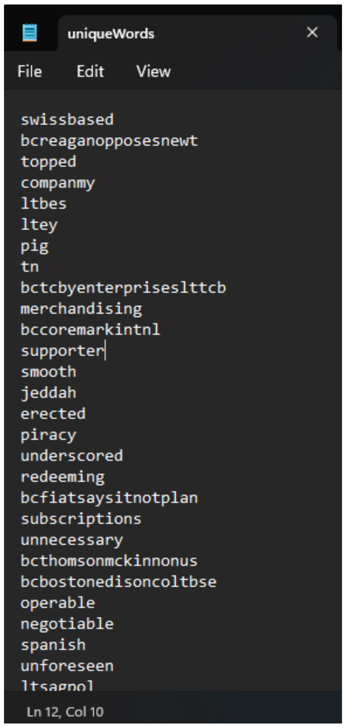
```
23/12/01 19:00:33 INFO SparkEnv: Registering BlockManagerMaster
23/12/01 19:00:33 INFO SparkEnv: Registering BlockMana
23/12/01 19:00:33 INFO SparkEnv: Registering OutputCo
23/12/01 19:00:34 INFO DefaultNoHARMFailoverProxyProv
internal./10.128.0.8:8032
23/12/01 19:00:34 INFO AHSProxy: Connecting to Applic
8.0.8:10200
23/12/01 19:00:36 INFO Configuration: resource-types.
23/12/01 19:00:36 INFO ResourceUtils: Unable to find
23/12/01 19:00:38 INFO YarnClientImpl: Submitted appl
23/12/01 19:00:39 INFO DefaultNoHARMFailoverProxyProv
internal./10.128.0.8:8030
23/12/01 19:00:41 INFO GhfsStorageStatistics: Detecte
encyMs=0; operationCount=1; context=gs://dataproc-tem
ry
23/12/01 19:00:41 INFO GoogleCloudStorageImpl: Ignori
sired state.
23/12/01 19:00:41 INFO GhfsStorageStatistics: Detecte
 operationCount=1; context=gs://dataproc-temp-us-cent
23/12/01 19:00:42 INFO GhfsStorageStatistics: Detecte
 operationCount=1; context=gs://dataproc-temp-us-cent
ation_1701454219139_0001.inprogress
Unique words written to file: /home/kenilpatel121200/
 Total no. of Unique Words are 5425
Highest Frequency Word is said with the frequency of
Lowest Frequency Word is young with the frequency of
23/12/01 19:00:57 INFO GhfsStorageStatistics: Detecte
viousMaxLatencyMs=0; operationCount=1; context=gs://d
k-job-history/application_1701454219139_0001.inprogre
23/12/01 19:00:58 INFO GhfsStorageStatistics: Detecte
 operationCount=1; context=rename(gs://dataproc-temp-
/application_1701454219139_0001.inprogress -> gs://dataproc-temp-us-central1-80
-job-history/application_1701454219139_0001)
kenilpatel121200@problem1b-m:~$ ls
Problem1B.jar  reut2-009.sgm  uniqueWords.txt
kenilpatel121200@problem1b-m:~$ ^C
kenilpatel121200@problem1b-m:~$ 
```

Name	Date modified	Type
META-INF	29-11-2023 21:54	File folder

File name: uniqueWords
Save as type: TXT File

uniqueWords.txt Directory: /home/kenilpatel121200/uni...

Figure 17: Downloading the txt from Vm instances to local.

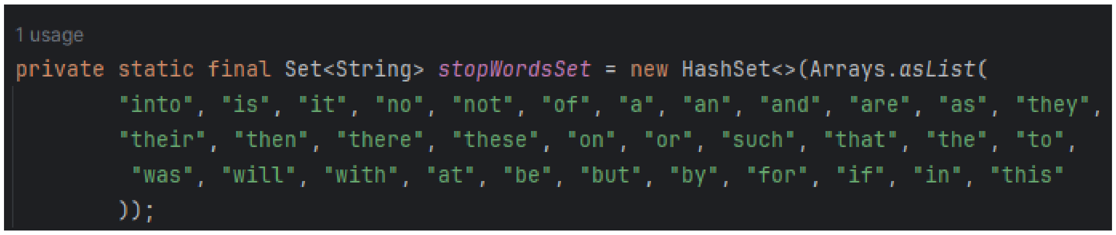


```
swissbased
bcreaganopposesnewt
topped
companmy
ltbes
ltey
pig
tn
bctcbyenterprisesltcb
merchandising
bccoremarkintnl
supporter
smooth
jeddah
erected
piracy
underscored
redeeming
bcfiatsaysitnotplan
subscriptions
unnecessary
bcthomsonmckinnonus
bcbostonedisoncoltbse
operable
negotiable
spanish
unforeseen
ltsagpol
```

Figure 18: uniqueWords.txt

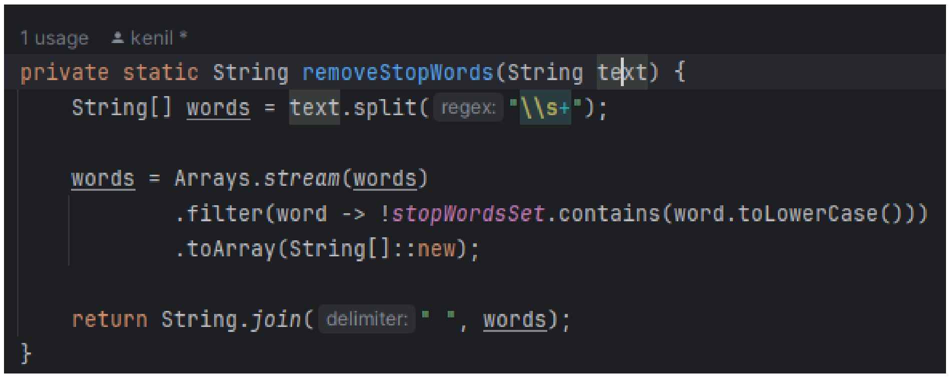
2.6 You can ignore stop words by removing them using stop words library before submitting the job to spark.

Created a HashSet of stop words and filtering words using the function removeStopWords.



```
1 usage
private static final Set<String> stopWordsSet = new HashSet<>(Arrays.asList(
        "into", "is", "it", "no", "not", "of", "a", "an", "and", "are", "as", "they",
        "their", "then", "there", "these", "on", "or", "such", "that", "the", "to",
         "was", "will", "with", "at", "be", "but", "by", "for", "if", "in", "this"
        ));
```

Figure 19: HashSet of Stop Words.



```
1 usage   kenil *
private static String removeStopWords(String text) {
    String[] words = text.split( regex: "\\s+");

    words = Arrays.stream(words)
            .filter(word -> !stopWordsSet.contains(word.toLowerCase()))
            .toArray(String[]::new);

    return String.join( delimiter: " ", words);
}
```

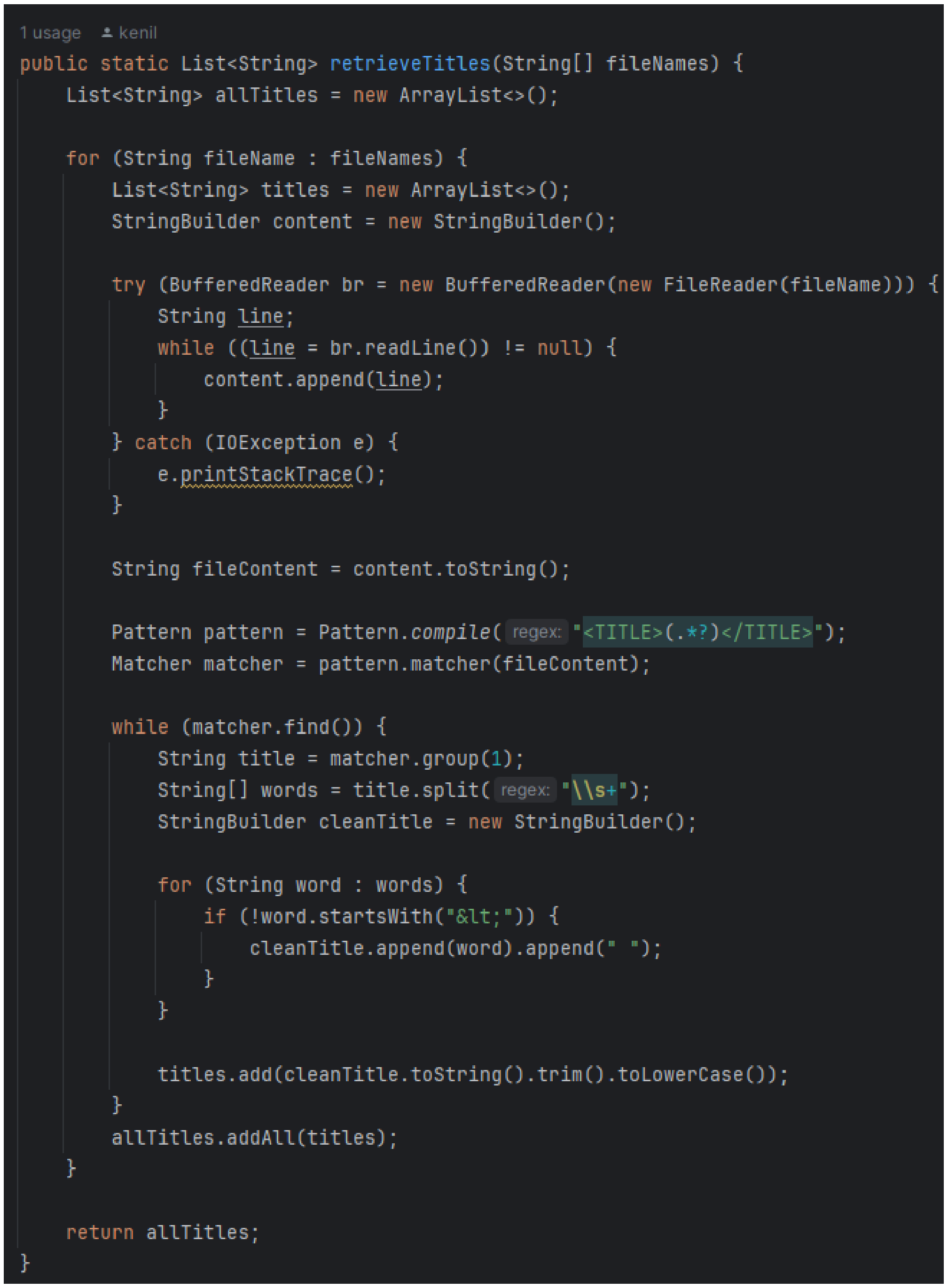
Figure 20: Function that removes the stop words.

3. Problem Statement 2

Use Core Java Program only with no additional libraries. Use the parser (regex-based parser).

3.1 Write a Java program to create bag-of-words for each News title.

To create a bag-of- words for each news title. First, I have retrieved all the titles of the news and then created the word list. Below is the java function to get all the titles for all the news.



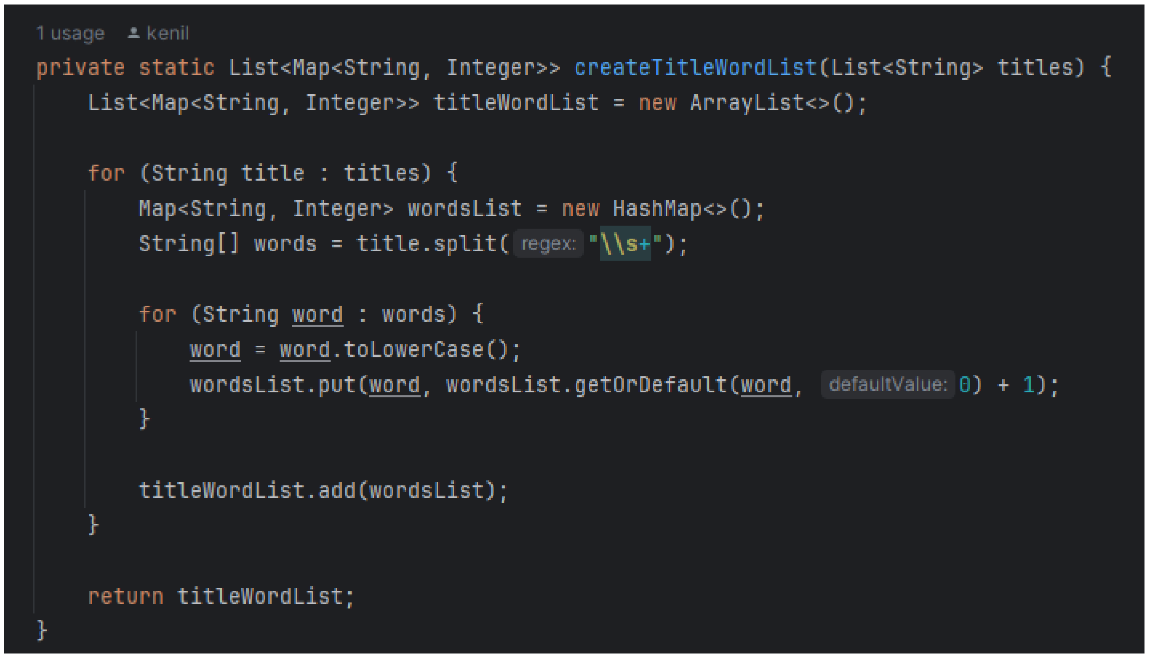
```
1 usage   kenil
public static List<String> retrieveTitles(String[] fileNames) {
    List<String> allTitles = new ArrayList<>();

    for (String fileName : fileNames) {
        List<String> titles = new ArrayList<>();
        StringBuilder content = new StringBuilder();

        try (BufferedReader br = new BufferedReader(new FileReader(fileName))) {
            String line;
            while ((line = br.readLine()) != null) {
                content.append(line);
            }
        } catch (IOException e) {
            e.printStackTrace();
        }

        String fileContent = content.toString();

        Pattern pattern = Pattern.compile( regex: "<TITLE>(.*?)</TITLE>");
        Matcher matcher = pattern.matcher(fileContent);

        while (matcher.find()) {
            String title = matcher.group(1);
            String[] words = title.split( regex: "\\s+");
            StringBuilder cleanTitle = new StringBuilder();

            for (String word : words) {
                if (!word.startsWith("&lt;")) {
                    cleanTitle.append(word).append(" ");
                }
            }

            titles.add(cleanTitle.toString().trim().toLowerCase());
        }
        allTitles.addAll(titles);
    }

    return allTitles;
}
```

Figure 21: retrieveTitles function.

retrieveTitles Function:

- Reads content from specified files.
- Extracts text within <TITLE> tags using regex pattern matching.
- Cleans titles by removing unwanted characters, discarding HTML entities, and converting to lowercase.
- Collect clean titles in a list and return the concatenated list of all titles.

After all the titles are retrieved we will create a word list i.e. a bag of words using the function **createTitleWordList.**



```
1 usage  kenil
private static List<Map<String, Integer>> createTitleWordList(List<String> titles) {
    List<Map<String, Integer>> titleWordList = new ArrayList<>();

    for (String title : titles) {
        Map<String, Integer> wordsList = new HashMap<>();
        String[] words = title.split( regex: "\\s+");

        for (String word : words) {
            word = word.toLowerCase();
            wordsList.put(word, wordsList.getOrDefault(word,  defaultValue: 0) + 1);
        }

        titleWordList.add(wordsList);
    }

    return titleWordList;
}
```

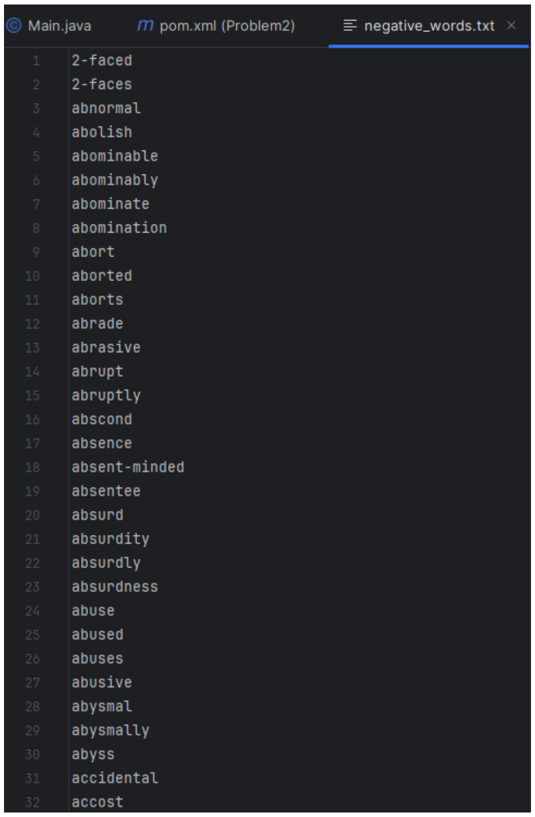
Figure 22: createTitleWordList Function.

createTitleWordList Function:

- Processes each title from the list of titles.
- Splits each title into words.
- Counts occurrences of words and stores them in a map (word as key, count as value).
- Compile these maps for all titles into a list and return it.

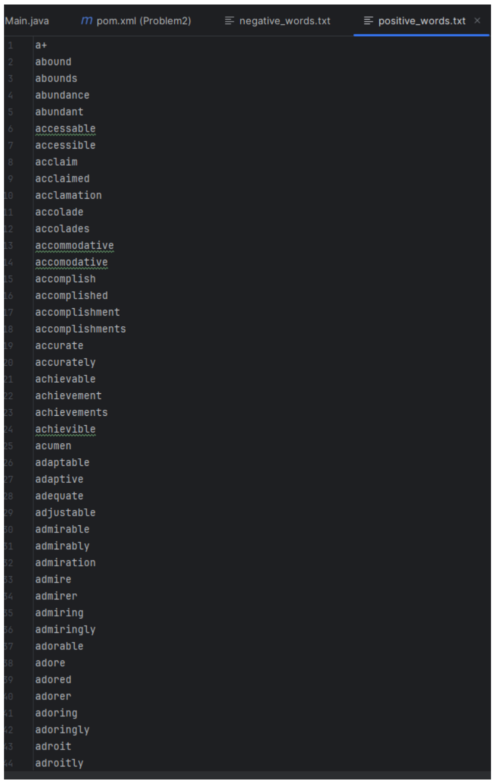
3.2 Compare each bag of words with a list of positive and negative words.

Downloaded the two word lists one is positive words and the other is negative words. There are 4833 negative words and 2054 positive words in the list.



```
2-faced
2-faces
abnormal
abolish
abominable
abominably
abominate
abomination
abort
aborted
aborts
abrade
abrasive
abrupt
abruptly
abscond
absence
absent-minded
absentee
absurd
absurdity
absurdly
absurdness
abuse
abused
abuses
abusive
abysmal
abysmally
abyss
accidental
accost
```

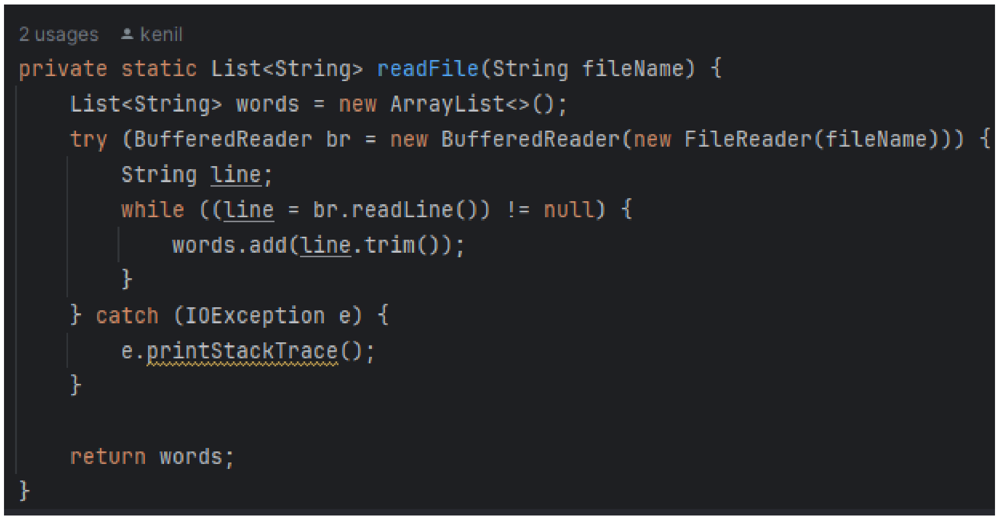
Figure 23: Negative words.



```
a+
abound
abounds
abundance
abundant
accessable
accessible
acclaim
acclaimed
acclamation
accolade
accolades
accommodative
accomodative
accomplish
accomplished
accomplishment
accomplishments
accurate
accurately
achievable
achievement
achievements
achievible
acumen
adaptable
adaptive
adequate
adjustable
admirable
admirably
admiration
admire
admirer
admiring
admiringly
adorable
adore
adored
adorer
adoring
adoringly
adroit
adroitly
```

Figure 24: Positive Words.

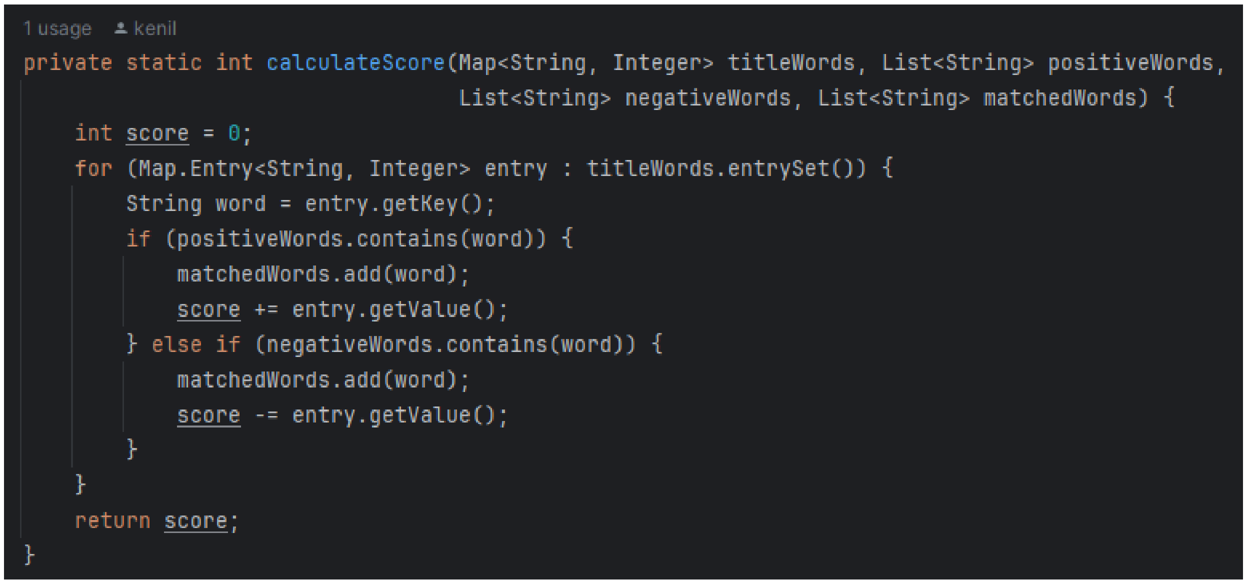
The **readFile** function reads negative and positive word files containing words and stores each word in a list. It opens the file specified by **fileName** and reads it line by line using a BufferedReader. For each line, it trims any extra spaces and adds the resulting word to the **word** list. Finally, it returns the list containing all the words read from the file.



```
2 usages  kenil
private static List<String> readFile(String fileName) {
    List<String> words = new ArrayList<>();
    try (BufferedReader br = new BufferedReader(new FileReader(fileName))) {
        String line;
        while ((line = br.readLine()) != null) {
            words.add(line.trim());
        }
    } catch (IOException e) {
        e.printStackTrace();
    }

    return words;
}
```

Figure 25: readFile function.

The **calculateScore** function checks each word in a news title to see if it's positive or negative. If a word is positive, it adds to the score, and if it's negative, it subtracts from the score. It keeps track of these words in the **matched words** list. Finally, it returns the total score based on the positive and negative words found in the title.

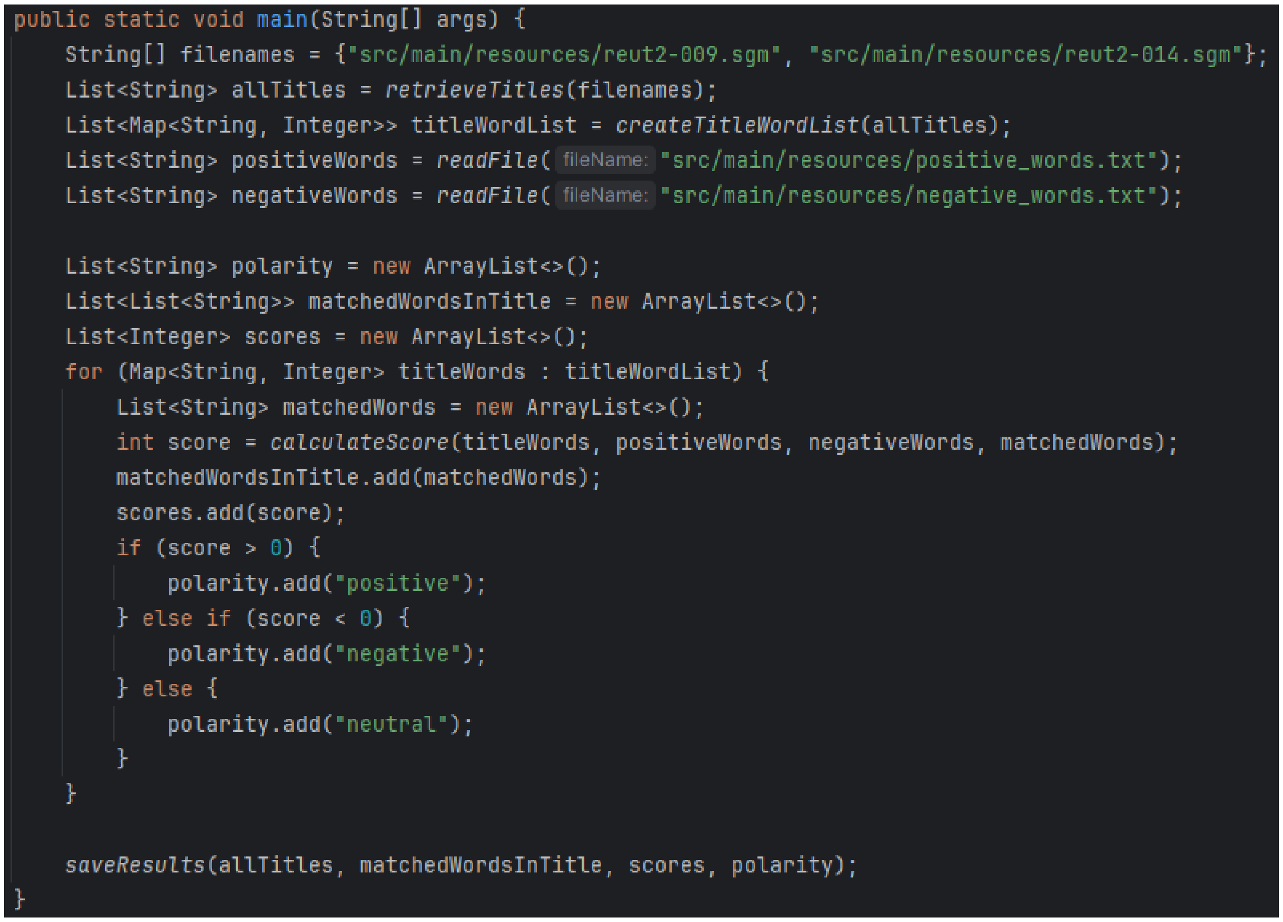


```
1 usage  kenil
private static int calculateScore(Map<String, Integer> titleWords, List<String> positiveWords,
                                  List<String> negativeWords, List<String> matchedWords) {
    int score = 0;
    for (Map.Entry<String, Integer> entry : titleWords.entrySet()) {
        String word = entry.getKey();
        if (positiveWords.contains(word)) {
            matchedWords.add(word);
            score += entry.getValue();
        } else if (negativeWords.contains(word)) {
            matchedWords.add(word);
            score -= entry.getValue();
        }
    }
    return score;
}
```

Figure 26: calculateScore function.

3.3 Tag each news title as “positive”, “negative”, or “neutral” based on the overall score.

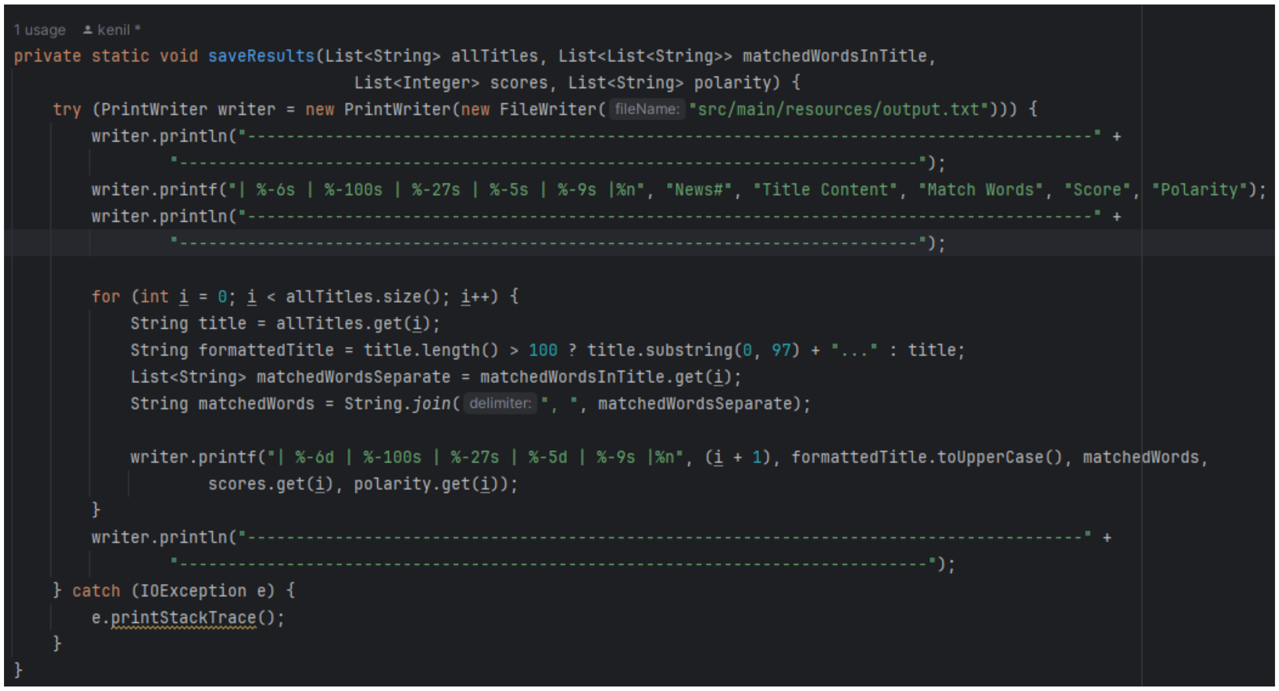
After the score is calculated then using the score it will decide the polarity. Depending on whether the score is positive (> 0), negative (< 0), or neutral (equal to 0), it categorizes the title's sentiment as 'positive', 'negative', or 'neutral', respectively. These sentiments are stored in the 'polarity' list.



```java
public static void main(String[] args) {
    String[] filenames = {"src/main/resources/reut2-009.sgm", "src/main/resources/reut2-014.sgm"};
    List<String> allTitles = retrieveTitles(filenames);
    List<Map<String, Integer>> titleWordList = createTitleWordList(allTitles);
    List<String> positiveWords = readFile( fileName: "src/main/resources/positive_words.txt");
    List<String> negativeWords = readFile( fileName: "src/main/resources/negative_words.txt");

    List<String> polarity = new ArrayList<>();
    List<List<String>> matchedWordsInTitle = new ArrayList<>();
    List<Integer> scores = new ArrayList<>();
    for (Map<String, Integer> titleWords : titleWordList) {
        List<String> matchedWords = new ArrayList<>();
        int score = calculateScore(titleWords, positiveWords, negativeWords, matchedWords);
        matchedWordsInTitle.add(matchedWords);
        scores.add(score);
        if (score > 0) {
            polarity.add("positive");
        } else if (score < 0) {
            polarity.add("negative");
        } else {
            polarity.add("neutral");
        }
    }

    saveResults(allTitles, matchedWordsInTitle, scores, polarity);
}
```

Figure 27: main function to get the polarity.

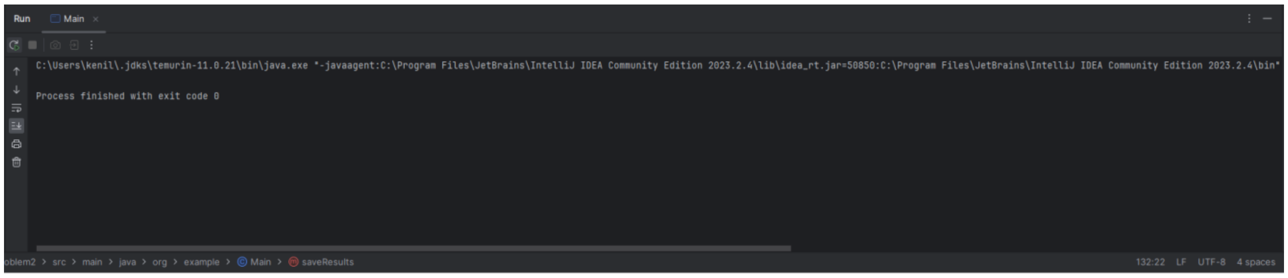
After getting all the polarity for the titles, it will save the table in the output.txt file. The result is saved in a file using the saveResults function.



```java
1 usage  kenil *
private static void saveResults(List<String> allTitles, List<List<String>> matchedWordsInTitle,
                                   List<Integer> scores, List<String> polarity) {
    try (PrintWriter writer = new PrintWriter(new FileWriter( fileName: "src/main/resources/output.txt"))) {
        writer.println("----------------------------------------------------------------------------------------" +
                "----------------------------------------------------------------------------");
        writer.printf("| %-6s | %-100s | %-27s | %-5s | %-9s |%n", "News#", "Title Content", "Match Words", "Score", "Polarity");
        writer.println("----------------------------------------------------------------------------------------" +
                "----------------------------------------------------------------------------");

        for (int i = 0; i < allTitles.size(); i++) {
            String title = allTitles.get(i);
            String formattedTitle = title.length() > 100 ? title.substring(0, 97) + "..." : title;
            List<String> matchedWordsSeparate = matchedWordsInTitle.get(i);
            String matchedWords = String.join( delimiter: ", ", matchedWordsSeparate);

            writer.printf("| %-6d | %-100s | %-27s | %-5d | %-9s |%n", (i + 1), formattedTitle.toUpperCase(), matchedWords,
                    scores.get(i), polarity.get(i));
        }
        writer.println("---------------------------------------------------------------------------------------" +
                "-----------------------------------------------------------------------------");
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

Figure 28: saveResults function.

On running the java file it generates the output.txt

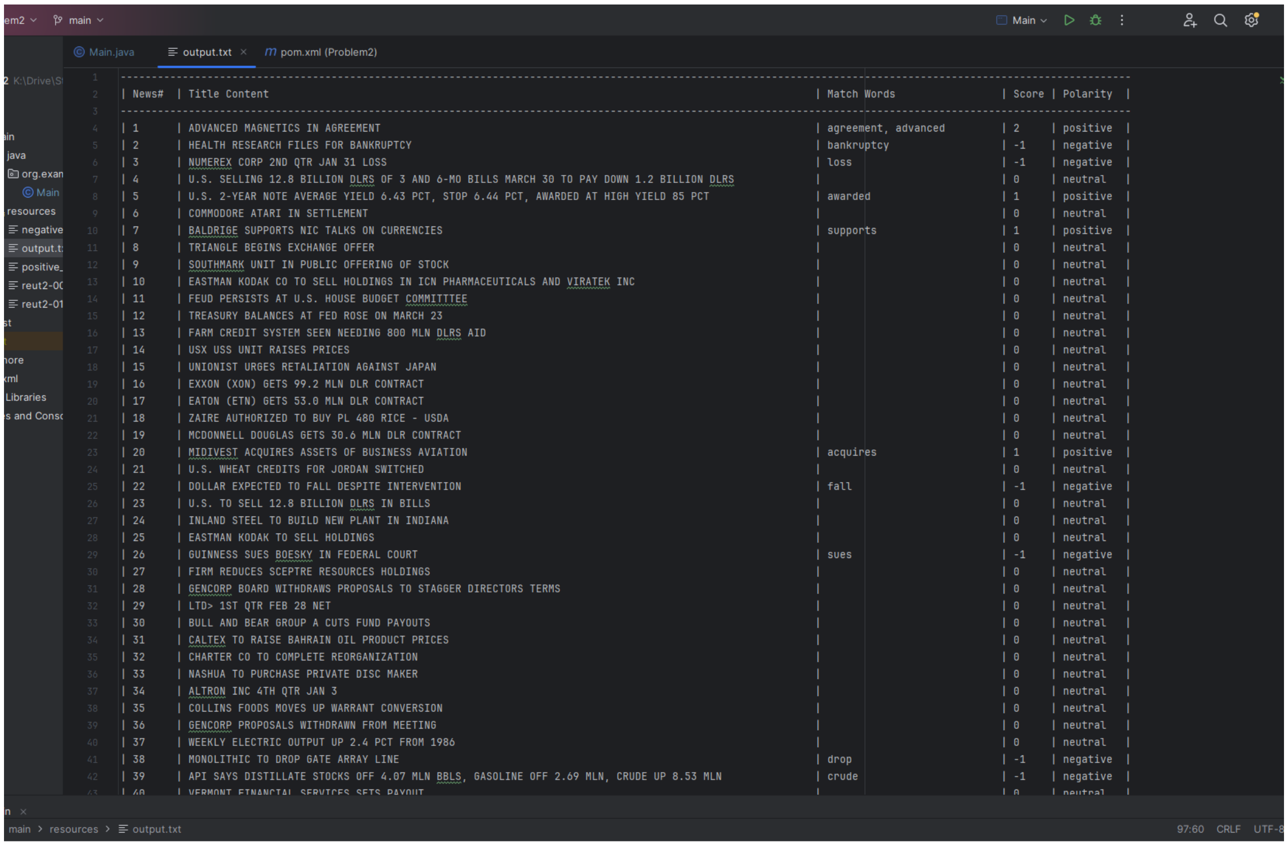


```
C:\Users\kenil\.jdks\temurin-11.0.21\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2023.2.4\lib\idea_rt.jar=50850:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2023.2.4\bin"

Process finished with exit code 0
```

Run
Main
oblem2 > src > main > java > org > example > Main > saveResults
132:22 LF UTF-8 4 spaces

Figure 29: Successful Execution.

OUTPUT:



```
-------------------------------------------------------------------------------------------------------------------------------------------------------------------
| News#  | Title Content                                                                                        | Match Words                 | Score | Polarity  |
-------------------------------------------------------------------------------------------------------------------------------------------------------------------
| 1      | ADVANCED MAGNETICS IN AGREEMENT                                                                      | agreement, advanced         | 2     | positive  |
| 2      | HEALTH RESEARCH FILES FOR BANKRUPTCY                                                                 | bankruptcy                  | -1    | negative  |
| 3      | NUMEREX CORP 2ND QTR JAN 31 LOSS                                                                     | loss                        | -1    | negative  |
| 4      | U.S. SELLING 12.8 BILLION DLRS OF 3 AND 6-MO BILLS MARCH 30 TO PAY DOWN 1.2 BILLION DLRS             |                             | 0     | neutral   |
| 5      | U.S. 2-YEAR NOTE AVERAGE YIELD 6.43 PCT, STOP 6.44 PCT, AWARDED AT HIGH YIELD 85 PCT                 | awarded                     | 1     | positive  |
| 6      | COMMODORE ATARI IN SETTLEMENT                                                                        |                             | 0     | neutral   |
| 7      | BALDRIGE SUPPORTS NIC TALKS ON CURRENCIES                                                            | supports                    | 1     | positive  |
| 8      | TRIANGLE BEGINS EXCHANGE OFFER                                                                       |                             | 0     | neutral   |
| 9      | SOUTHMARK UNIT IN PUBLIC OFFERING OF STOCK                                                           |                             | 0     | neutral   |
| 10     | EASTMAN KODAK CO TO SELL HOLDINGS IN ICN PHARMACEUTICALS AND VIRATEK INC                             |                             | 0     | neutral   |
| 11     | FEUD PERSISTS AT U.S. HOUSE BUDGET COMMITTTEE                                                        |                             | 0     | neutral   |
| 12     | TREASURY BALANCES AT FED ROSE ON MARCH 23                                                            |                             | 0     | neutral   |
| 13     | FARM CREDIT SYSTEM SEEN NEEDING 800 MLN DLRS AID                                                     |                             | 0     | neutral   |
| 14     | USX USS UNIT RAISES PRICES                                                                           |                             | 0     | neutral   |
| 15     | UNIONIST URGES RETALIATION AGAINST JAPAN                                                             |                             | 0     | neutral   |
| 16     | EXXON (XON) GETS 99.2 MLN DLR CONTRACT                                                               |                             | 0     | neutral   |
| 17     | EATON (ETN) GETS 53.0 MLN DLR CONTRACT                                                               |                             | 0     | neutral   |
| 18     | ZAIRE AUTHORIZED TO BUY PL 480 RICE - USDA                                                           |                             | 0     | neutral   |
| 19     | MCDONNELL DOUGLAS GETS 30.6 MLN DLR CONTRACT                                                         |                             | 0     | neutral   |
| 20     | MIDIVEST ACQUIRES ASSETS OF BUSINESS AVIATION                                                        | acquires                    | 1     | positive  |
| 21     | U.S. WHEAT CREDITS FOR JORDAN SWITCHED                                                               |                             | 0     | neutral   |
| 22     | DOLLAR EXPECTED TO FALL DESPITE INTERVENTION                                                         | fall                        | -1    | negative  |
| 23     | U.S. TO SELL 12.8 BILLION DLRS IN BILLS                                                              |                             | 0     | neutral   |
| 24     | INLAND STEEL TO BUILD NEW PLANT IN INDIANA                                                           |                             | 0     | neutral   |
| 25     | EASTMAN KODAK TO SELL HOLDINGS                                                                       |                             | 0     | neutral   |
| 26     | GUINNESS SUES BOESKY IN FEDERAL COURT                                                                | sues                        | -1    | negative  |
| 27     | FIRM REDUCES SCEPTRE RESOURCES HOLDINGS                                                              |                             | 0     | neutral   |
| 28     | GENCORP BOARD WITHDRAWS PROPOSALS TO STAGGER DIRECTORS TERMS                                         |                             | 0     | neutral   |
| 29     | LTD> 1ST QTR FEB 28 NET                                                                              |                             | 0     | neutral   |
| 30     | BULL AND BEAR GROUP A CUTS FUND PAYOUTS                                                              |                             | 0     | neutral   |
| 31     | CALTEX TO RAISE BAHRAIN OIL PRODUCT PRICES                                                           |                             | 0     | neutral   |
| 32     | CHARTER CO TO COMPLETE REORGANIZATION                                                                |                             | 0     | neutral   |
| 33     | NASHUA TO PURCHASE PRIVATE DISC MAKER                                                                |                             | 0     | neutral   |
| 34     | ALTRON INC 4TH QTR JAN 3                                                                             |                             | 0     | neutral   |
| 35     | COLLINS FOODS MOVES UP WARRANT CONVERSION                                                            |                             | 0     | neutral   |
| 36     | GENCORP PROPOSALS WITHDRAWN FROM MEETING                                                             |                             | 0     | neutral   |
| 37     | WEEKLY ELECTRIC OUTPUT UP 2.4 PCT FROM 1986                                                          |                             | 0     | neutral   |
| 38     | MONOLITHIC TO DROP GATE ARRAY LINE                                                                   | drop                        | -1    | negative  |
| 39     | API SAYS DISTILLATE STOCKS OFF 4.07 MLN BBLS, GASOLINE OFF 2.69 MLN, CRUDE UP 8.53 MLN               | crude                       | -1    | negative  |
| 40     | VERMONT FINANCIAL SERVICES SETS PAYOUT                                                               |                             | 0     | neutral   |
```

Figure 30: Sentiment Analysis of the News Title.

References

[1] "Download MongoDB Community Server," *MongoDB*. [Online]. Available: https://www.mongodb.com/try/download/community. [Accessed: 30-November-2023].

[2] "Google Cloud Platform," Google.com. [Online]. Available: https://cloud.google.com/?hl=en. [Accessed: 30-November-2023].

[3] Java.com. [Online]. Available: https://www.java.com/en/. [Accessed: 30-November-2023].

[4] "Apache SparkTM - Unified Engine for large-scale data analytics," Apache.org. [Online]. Available: https://spark.apache.org/. [Accessed: 30-November-2023]

[5] "Download IntelliJ IDEA – the leading java and kotlin IDE," *JetBrains*. [Online]. Available: https://www.jetbrains.com/idea/download/?section=windows. [Accessed: 20-November-2023].

[6] Marcin, "Negative-words.Txt," *github.com*. [Online]. Available: https://gist.github.com/mkulakowski2/4289441. [Accessed: 25-November-2023].

[7] Marcin, "Positive-words.Txt," *github.com*. [Online]. Available: https://gist.github.com/mkulakowski2/4289437. [Accessed: 25-November-2023].