CSCI 5408
DATA MANAGEMENT AND WAREHOUSING

ASSIGNMENT – 2

Banner ID: B00959783

GitLab Assignment Link:
https://git.cs.dal.ca/hudka/csci5408_f23_b00959783_nikunj_hudka

Table Of Contents:

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

This problem can be divided into 2 steps. One is Reading and Transforming the data, and the second is storing it in the database.

Following the SOLID principles, I have created one class ReutRead to read the raw the data from the files and transform that data based on our need, and a second class MongoDatabase to initialize the MongoDB connection and store the transformed data in the database.

ReutRead class has methods to read both files and then extract the title and body from the data that we read from the files, clean the extracted title and body which later will be stored in the MongoDB database ReuterDb.

Algorithm:

```
Initialize collection Collection
For input files F1, F2
    Read F1 and F2
    Combine data into F
    From F, chunk REUTER inside tag <REUTERS> and </REUTERS>
    For each chunk REUTER
        From REUTER, get text title REUTER_TITLE inside tag <TITLE> and </TITLE>
        From REUTER, get text title REUTER_BODY inside tag <BODY > and </BODY>
        Cleanup REUTER_TITLE and REUTER_BODY using regex and trim
        Create a document with Title and Body
        Insert the document into MongoDB collection
Close the MongoDB connection
```

Flowchart:

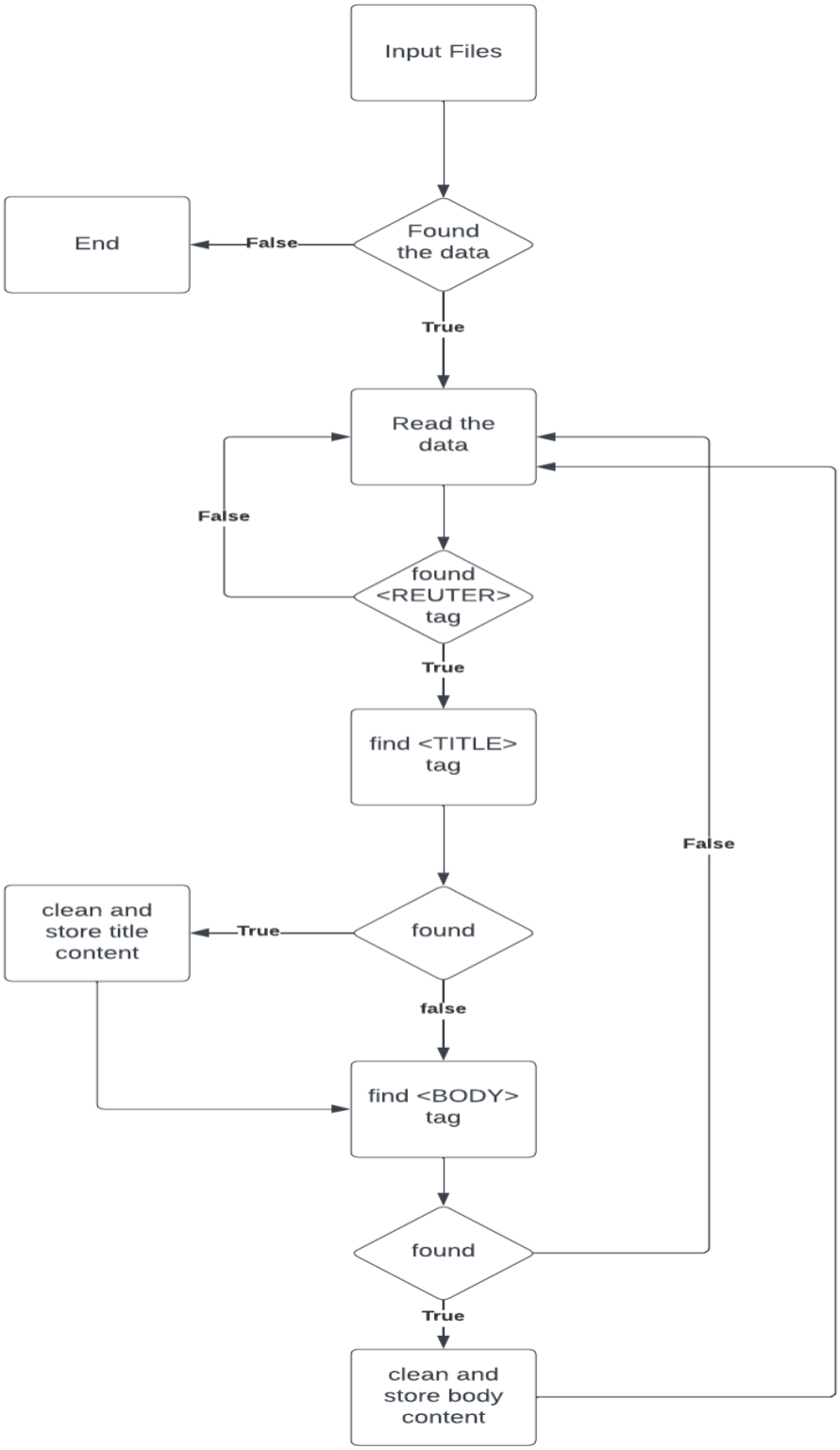


Figure 1: Flowchart to read, transform and store reut news data

Execution:

Here, I have attached the screenshot of the MongoDb collection reuter inside the ReuterDb database which has 1694 documents each have the title and body that I have extracted and transformed from the given files.

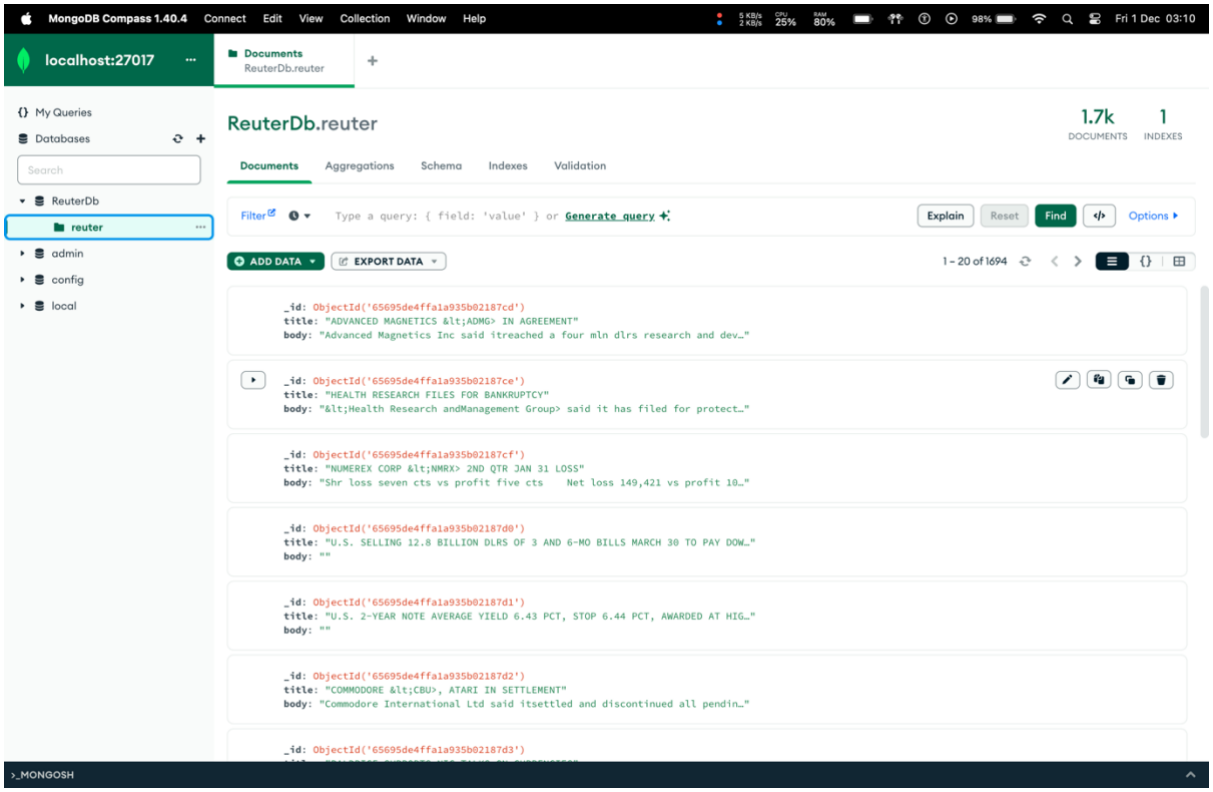


Figure 2: MongoDb ReuterDb database collection

Problem 1B: Reuter News Data Processing using Spark

The program will first create a spark session and after that, it will read the data from the reuter file and it clean and transform the data. Afterward, the processed data is sent to the spark context in which it is reduced to a key-value pair in which the key is the word and the value is its frequency count. This frequency count will be saved in a file. Also, it gave the unique words in the files whose frequency count is 1 as I am filtering the frequency count by a count equal to 1. Those unique words will also be saved in another file.

To run the program in GCP, firstly I have set the DataProc cluster and moved my JAR file and other files like reut2-009.sgm that which will be used in spark execution

To run the GCP I have used another person's gmail account because my gmail account in which I have credits is deactivated for a while. So I have login through their gcp account and executed the spark job.

Algorithm:

Set STOP_WORDS as a HashSet containing common English stop words.

Initialize the reuterFile variable with the path to the input file (e.g., "./reut2-009.sgm").

Try the following block:

- Create a SparkSession named "FrequencyCount."
- Create a JavaSparkContext using the SparkSession's SparkContext.
- Read the content of the input file into a StringBuilder.

Transform the content by:

- Removing "<" characters.
- Removing non-alphabetic characters.
- Removing single characters.
- Replacing multiple whitespaces with a single space.
- Trimming leading and trailing whitespaces.
- Converting the text to lowercase.

Create a JavaRDD of strings from the transformed text by splitting on spaces.

Perform word count:

- FlatMap each line into key-value pairs, filtering out stop words.
- Reduce by key (word) by summing up the counts.

Filter out words with a count of 1 to get single-count words.

Collect the results and print the number of unique words.

Save word counts and unique words to specified output file paths.

Print information about saved files and the highest/lowest frequency words.

Stop the SparkContext and SparkSession.

Flowchart:

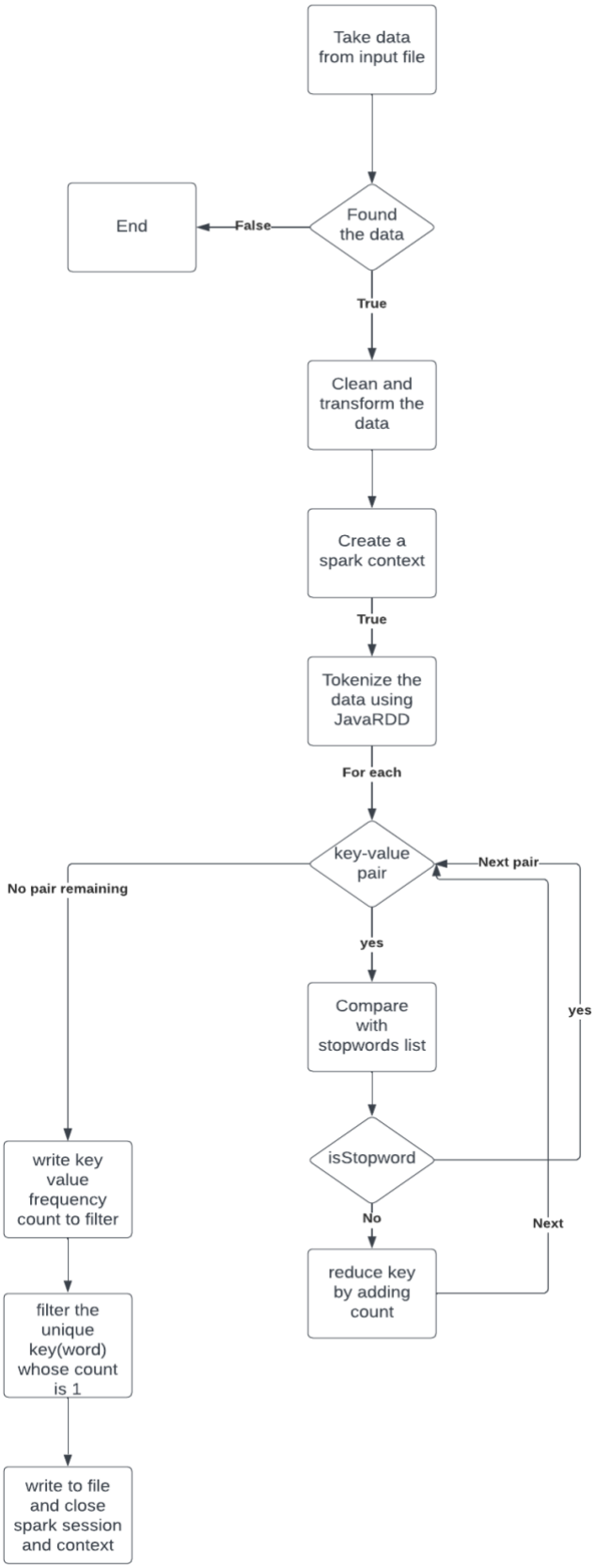


Figure 3: Flow chart for frequency count in spark

Execution:

I set up the Dataproc cluster on GCP. Connect the SSH terminal and uploaded the jar file of the program, reut data file.

```
mayursiinh@assignment-2-nikunj-m:~$ ls
Assignment2_probl_1b-1.0-SNAPSHOT.jar  reut2-009.sgm
mayursiinh@assignment-2-nikunj-m:~$ rm Assignment2_probl_1b-1.0-SNAPSHOT.jar
mayursiinh@assignment-2-nikunj-m:~$ ls
'Assignment2_probl_1b-1.0-SNAPSHOT 1.jar'   reut2-009.sgm
mayursiinh@assignment-2-nikunj-m:~$ spark-submit --class com.example.FrequencyCount Assignment2_probl_1b-1.0-SNAPSHOT\ 1.jar hdfs://assignment-2-nikunj-m/frequency-count hdfs:
ignment-2-nikunj-m/unique-words
Error: Failed to load class com.example.FrequencyCount.
```

Figure 4: Uploaded JAR file and reut data file

Created the result directory in HDFS where our spark program will write output in partitions.

```
SSH-in-browser
mayursiinh@assignment-2-nikunj-m:~$ ls
Assignment2_probl_1b-1.0-SNAPSHOT.jar  reut2-009.sgm
mayursiinh@assignment-2-nikunj-m:~$ hadoop fs -ls hdfs://assignment-2-nikunj-m/result
Found 2 items
-rw-r--r--   2 mayursiinh hadoop          0 2023-12-01 17:25 hdfs://assignment-2-nikunj-m/result/_SUCCESS
-rw-r--r--   2 mayursiinh hadoop     185634 2023-12-01 17:25 hdfs://assignment-2-nikunj-m/result/part-00000
```

Figure 5: result directory created in HDFS

Submit the program to Spark for execution. Upon completion, the program will provide information on the word with the maximum frequency ("said") occurring 2473 times and the word with the minimum frequency ("asinterseted") occurring only once. It's worth noting that there are multiple words with a frequency of 1. I have also added all the unique words in separate file.

```
mayursiinh@assignment-2-nikunj-m:~$ spark-submit --class org.example.FrequencyCount Assignment2_probl_1b-1.0-SNAPSHOT\ 1.jar hdfs://assignment-2-nikunj-m/frequency-count hdfs://ass
ignment-2-nikunj-m/unique-words
23/12/01 17:39:29 INFO SparkEnv: Registering MapOutputTracker
23/12/01 17:39:29 INFO SparkEnv: Registering BlockManagerMaster
23/12/01 17:39:29 INFO SparkEnv: Registering BlockManagerMasterHeartbeat
23/12/01 17:39:29 INFO SparkEnv: Registering OutputCommitCoordinator
23/12/01 17:39:30 INFO DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at assignment-2-nikunj-m.us-central1-f.c.csci-5408-f23.internal./10.128.0.17:8032
23/12/01 17:39:30 INFO AHSProxy: Connecting to Application History server at assignment-2-nikunj-m.us-central1-f.c.csci-5408-f23.internal./10.128.0.17:10200
23/12/01 17:39:32 INFO Configuration: resource-types.xml not found
23/12/01 17:39:32 INFO ResourceUtils: Unable to find 'resource-types.xml'.
23/12/01 17:39:33 INFO YarnClientImpl: Submitted application application_1701451185652_0003
23/12/01 17:39:34 INFO DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at assignment-2-nikunj-m.us-central1-f.c.csci-5408-f23.internal./10.128.0.17:8030
23/12/01 17:39:36 INFO GhfsStorageStatistics: Detected potential high latency for operation op_get_file_status. latencyMs=391; previousMaxLatencyMs=0; operationCount=1; context=gs:
//dataproc-temp-us-central1-246429513520-u8jlo6yg/f6657a70-3bbd-418f-afd9-25cda9d112c6/spark-job-history
23/12/01 17:39:36 INFO GoogleCloudStorageImpl: Ignoring exception of type GoogleJsonResponseException; verified object already exists with desired state.
23/12/01 17:39:36 INFO GhfsStorageStatistics: Detected potential high latency for operation op_mkdirs. latencyMs=218; previousMaxLatencyMs=0; operationCount=1; context=gs://datapro
c-temp-us-central1-246429513520-u8jlo6yg/f6657a70-3bbd-418f-afd9-25cda9d112c6/spark-job-history

Unique Words in the reuter file given is :
Number of Unique Words: 12531
Unique words with frequency 1 is saved in the file: hdfs://assignment-2-nikunj-m/unique-words
Words frequency count is saved in the file: hdfs://assignment-2-nikunj-m/frequency-count
Highest Frequency Word: said, Count: 2473
Lowest Frequency Word: asinterested, Count: 1
23/12/01 17:39:47 INFO GhfsStorageStatistics: Detected potential high latency for operation op_rename. latencyMs=259; previousMaxLatencyMs=0; operationCount=1; context=rename(gs://
dataproc-temp-us-central1-246429513520-u8jlo6yg/f6657a70-3bbd-418f-afd9-25cda9d112c6/spark-job-history/application_1701451185652_0003.inprogress -> gs://dataproc-temp-us-central1-2
46429513520-u8jlo6yg/f6657a70-3bbd-418f-afd9-25cda9d112c6/spark-job-history/application_1701451185652_0003)
```

Figure 6: successfully execution of program to spark

```
(unknowntexttitlewd,1)
(swissbased,1)
(unknowntexttitleborgwarner,1)
(bankviolated,1)
(bcreaganopposesnewt,1)
(kongisland,1)
(datelinebodypanteras,1)
(topped,1)
(businessacquisitions,1)
(saidhoward,1)
(wla,1)
(linotype,8)
(excludingexchange,1)
(customerswas,1)
(brazilscrisisladen,1)
(andvenezuelan,1)
(dole,4)
(cocaine,2)
(companmy,1)
(preventing,3)
(been,198)
(localdemand,1)
(pig,1)
(fourforeign,1)
(clients,7)
(pacifictitleblah,1)
(tn,2)
(discussionstriggered,1)
(agrowing,1)
```

Figure 7: output of frequency count file

```
(unknowntexttitlewd,1)
(swissbased,1)
(unknowntexttitleborgwarner,1)
(bankviolated,1)
(bcreaganopposesnewt,1)
(kongisland,1)
(datelinebodypanteras,1)
(topped,1)
(businessacquisitions,1)
(saidhoward,1)
(wla,1)
(linotype,8)
(excludingexchange,1)
(customerswas,1)
(brazilscrisisladen,1)
(andvenezuelan,1)
(dole,4)
(cocaine,2)
(companmy,1)
(preventing,3)
(been,198)
(localdemand,1)
(pig,1)
(fourforeign,1)
(clients,7)
(pacifictitleblah,1)
(tn,2)
(discussionstriggered,1)
(agrowing,1)
```

Figure 8: output of unique words file

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

Program will perform the sentiment analysis using the BOW model on the title of Reuters New Articles. To extract the title from the .smg files I used the same logic as I did in problem 1A. Firstly I will extract the title from each reuters tag and for each title I will create a bag of word which has a word and its frequency. Now to do the sentiment analysis of that title I will used the positive and negative words which are downloaded from the verified source. Afterwards, I will compare the BOW of that title with the positive and negative words dataset and calculate the score of that title. If positive word is matched then I will add that word to the matched words list and increment the score and if negative word is then also I will add that word to the matched words list and decrement the score. Now based on the score I have determined the polarity i.e positive, negative or neutral. If score is above zero then its polarity is positive, if less than zero then its polarity is negative, and if it is zero then its polarity is neutral.

Next, I have stored the newsId , title, matched words, score and polarity of each news in the database. Data from the database is exported in the csv file.

I have also used the SOLID principle like the Single Responsibility Principle by creating separate classes for performing reading, sentiment analysis, and extracting data in tabular format.

Execution:

Below is the code snippet of the core logic in doing sentiment analysis of the title in each reuter news article.

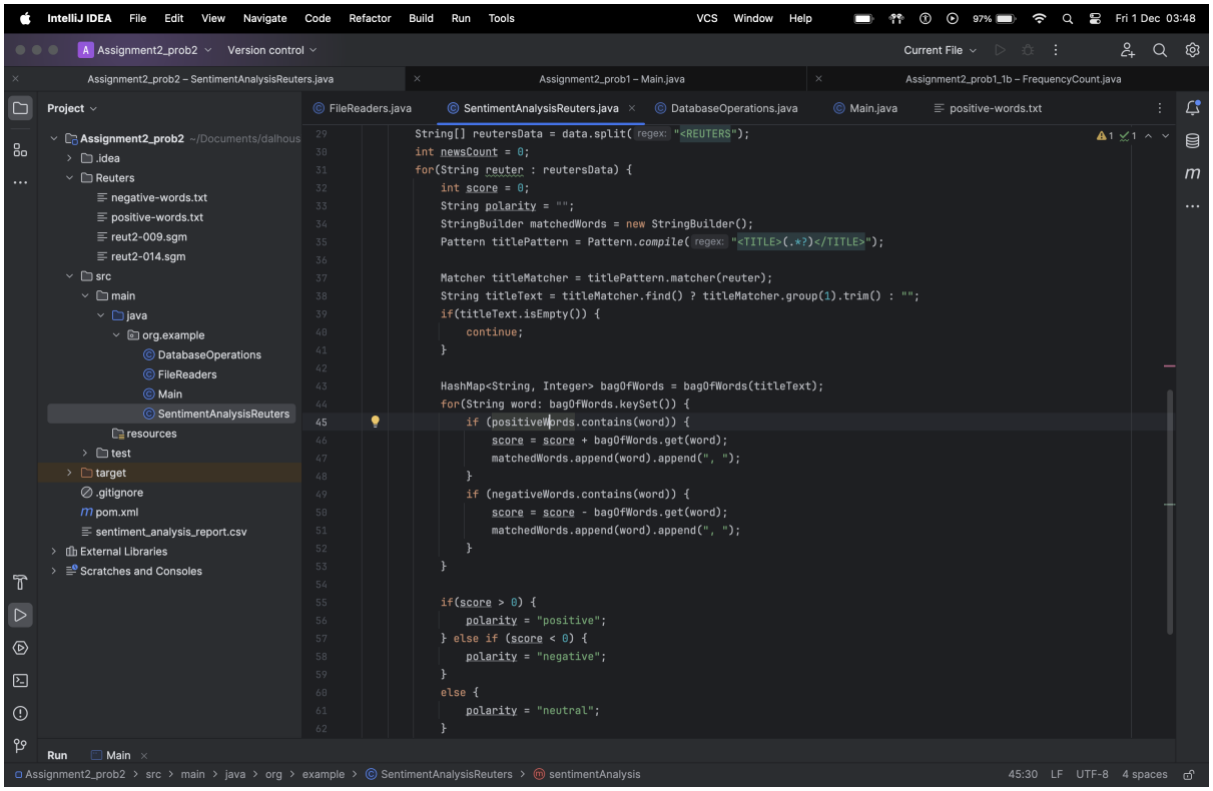


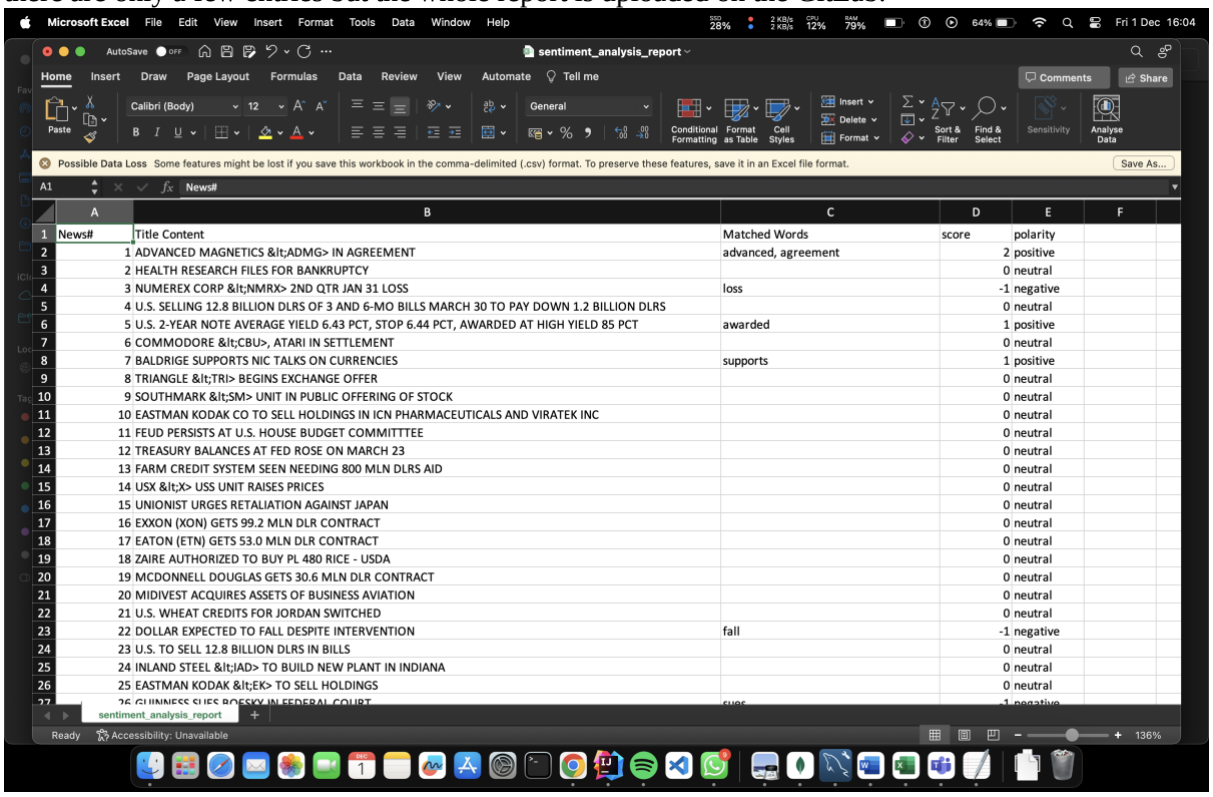
Figure 9: Code snippet of core logic for sentiment analysis

I have created a bag of words for each title.

```
{corp=1, dixons=1, reforms=1, get=1, of=1, cyclops=2, majority=1, after=1, to=1, stock=1, board=1, fails=1, group=1}
{uk=1, against=1, retain=1, unitary=1, to=1, u.s=1, powers=1, taxation=1}
{works=1, treasury's=1, baker=1, says=1, economic=1, cooperation=1}
{atlantic=1, re=1, &lt;pnre>=1, 4th=1, pan=1, net=1, qtr=1, inc=1}
{corp=1, sets=1, chemical=1, &lt;qchm>=1, quaker=1, quarterly=1}
{raytheon=1, director=1, resigns=1, &lt;rtn>=1}
{middle=1, south=1, dividend=1, to=1, consider=1, &lt;msu>=1}
{capital=1, calls=1, &lt;mon>=1, preferred=1, monarch=1}
{extract=1, sets=1, carotene=1, cyanotech=1, to=1, &lt;cyan>=1, way=1}
{scandinavia=1, sas=1, upgrade=1, in=1, service=1, cabin=1, to=1}
{trading=1, outlaw=1, plans=1, to=1, belgium=1, insider=1}
```

Figure 10: Bag of words for all titles

Below I have attached the tabular format report of sentiment analysis done on each news article. Here there are only a few entries but the whole report is uploaded on the GitLab.



News#	Title Content	Matched Words	score	polarity
1	ADVANCED MAGNETICS <ADMG> IN AGREEMENT	advanced, agreement	2	positive
2	HEALTH RESEARCH FILES FOR BANKRUPTCY		0	neutral
3	NUMEREX CORP <NMRX> 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 <CBU>, ATARI IN SETTLEMENT		0	neutral
7	BALDRIGE SUPPORTS NIC TALKS ON CURRENCIES	supports	1	positive
8	TRIANGLE <TRI> BEGINS EXCHANGE OFFER		0	neutral
9	SOUTHMARK <SM> 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 <X> 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		0	neutral
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 <IAD> TO BUILD NEW PLANT IN INDIANA		0	neutral
25	EASTMAN KODAK <EK> TO SELL HOLDINGS		0	neutral

Figure 11: Tabular report of sentiment analysis

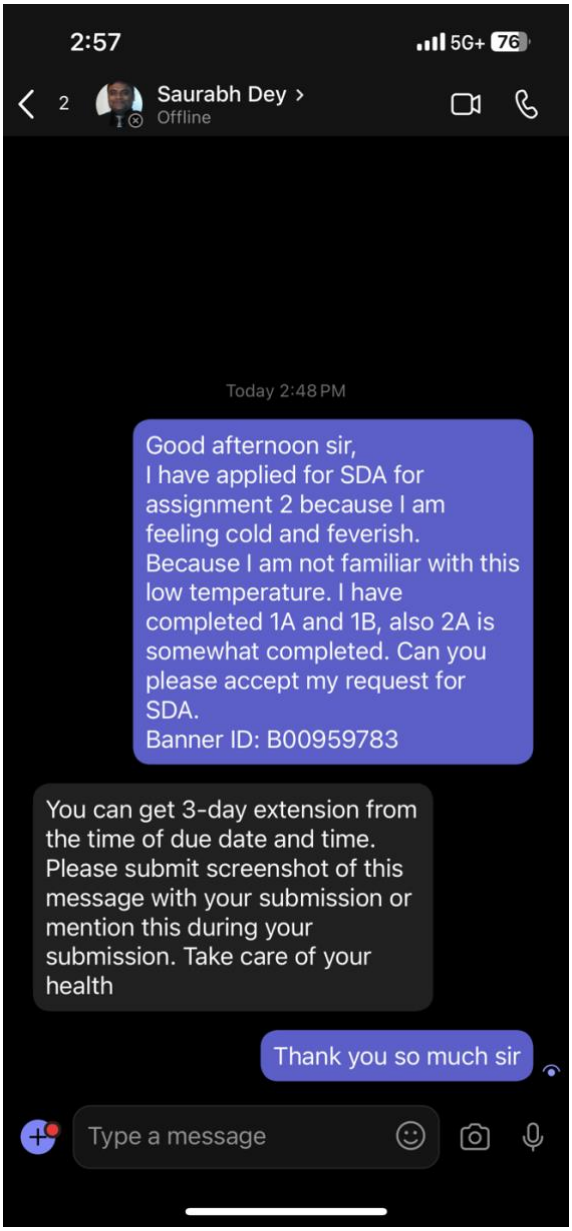


Figure 12: SDA confirmation

References:

"*MongoDB Quick Start - Java"*. mongodb.com. [Online]. Available: https://www.mongodb.com/docs/drivers/java/sync/v4.3/quick-start/. November 28th, 2023.

"*Java Regular Expressions"*. w3schools.com. [Online]. Available: https://www.w3schools.com/java/java_regex.asp. November 28th, 2023.

"negative-words.txt", gist.github.com. [Online]. Available: https://gist.github.com/mkulakowski2/4289441. November 29th, 2023.

"positive-words.txt", gist.github.com. [Online]. Available: https://gist.github.com/mkulakowski2/4289437. November 29th, 2023.

"*stopwords.txt".* princeton.edu. [Online]. Available: https://algs4.cs.princeton.edu/35applications/stopwords.txt. November 28th, 2023.

"*Apache Spark Examples"*. apache.org. [Online]. Available: https://spark.apache.org/examples.html. November 29th, 2023.

"*Lab_5 CSCI 5408"*. dal.brightspace.com. [Online]. Available: https://dal.brightspace.com/d2l/le/content/284056/viewContent/3937121/View. November 28th, 2023.