CSCI 5408
ASSIGNMENT 3

Prepared by: Sushank Saini
B00922727

Index

GitLab Link

https://git.cs.dal.ca/sushank/csci5408_s23_b00922727_sushank_saini.git

Problem 1A

Overview

The RuetRead.java program performs the extraction, transformation, and loading (ETL) function for a given file. The class has function called ETL(), that first reads a file and then Regex [1][2] and Pattern-Matcher [3][4] is used to extract the required text between the <TEXT></ TEXT> tags, and <TITLE></ TITLE > tags. Then for each news article, the required data is transformed into key-value pairs and inserted [5] into the newsArticles collection of the reuterDb database in MongoDB.

Flowchart

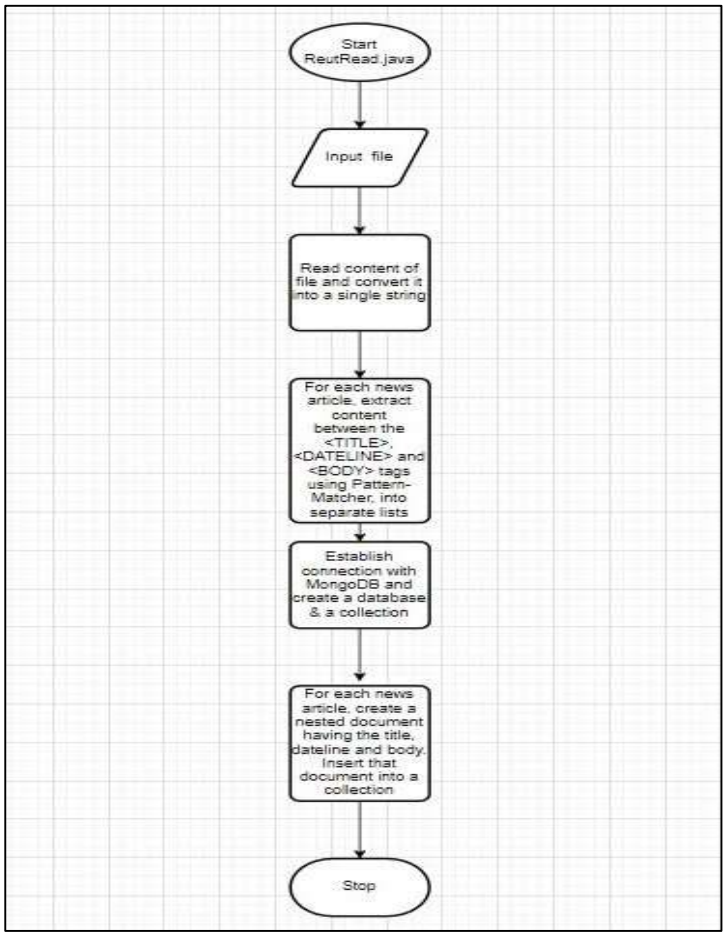

Figure 1: Flowchart for ReutRead.java program
Source: Author using [6]

Explanation of Code

The code starts by taking the file as an input. The file is read line by line using a BufferedReader object and transformed into a string using a StringBuilder object. The newline escape sequence in the string is replaced with an empty space to make the regular expression work (**Figure 2**).

```
no usages  Sushank Saini
public void ETL(String file)
{
    //read the content of a file line by line
    try(BufferedReader bufferedReader=new BufferedReader(new FileReader(file)))
    {
        String line;
        StringBuilder stringBuilder=new StringBuilder();
        //append each line to a string builder object to transform the file content into a string
        while((line=bufferedReader.readLine())!=null)
        {
            stringBuilder.append(line);
        }
        //transforming the data by replacing the newline with space
        String fileContent= stringBuilder.toString().replace( target: "\n", replacement: " ");
```

Figure 2: Reading a file and transforming it.
Source: Author

Then, using Regex [1][2] and Pattern-Matcher [3][4], the required content is extracted between the <TEXT></ TEXT> tags, and <TITLE></ TITLE > tags. The content of each title, dateline, and body is stored in the listOfTitles, listOfDateLines, and listOfBodies arraylists, respectively. (**Figure 3**).

```
//extracting the data using regular expression
String reGex="<TITLE>(.*?)<\\/TITLE>\\s*<DATELINE>(.*?)<\\/DATELINE><BODY>(.*?)<\\/BODY>";
Pattern pattern=Pattern.compile(reGex);
Matcher matcher=pattern.matcher(fileContent);
List<String> listOfTitles=new ArrayList<>();
List<String> listOfBodies=new ArrayList<>();
List<String> listOfDateLines=new ArrayList<>();
//storing list of titles, datelines and bodies in a news article
while(matcher.find())
{
    listOfTitles.add(matcher.group(1));
    listOfDateLines.add(matcher.group(2));
    listOfBodies.add(matcher.group(3));
}
```

Figure 3: Storing content in Arraylists.
Source: Author

These arraylists are iterated and a nested document for each news article is created. Simultaneously, the nested document is inserted [5] into the collection (**Figure 4**).

```
//insert a nested document in the mongodb database
for(int i=0;i<= listOfDateLines.size()-1;i++)
{
    Document mainDocument = new Document();
    Document textDocument = new Document();
    textDocument.append("dateline",listOfDateLines.get(i))
                .append("body",listOfBodies.get(i));
    mainDocument.append("title",listOfTitles.get(i))
                .append("text",textDocument);
    collection.insertOne(mainDocument);
}
```

Figure 4: Inserting nested document to the MongoDB collection.
Source: Author

The stored data looks like in **Figure 5**

.

Figure 5: Stored data in MongoDB collection.
Source: Author

Problem 1B

Steps performed to set up Apache Spark cluster on Google Cloud Platform (GCP)

First, search for the dataproc resource on the GCP (**Figure 6**).

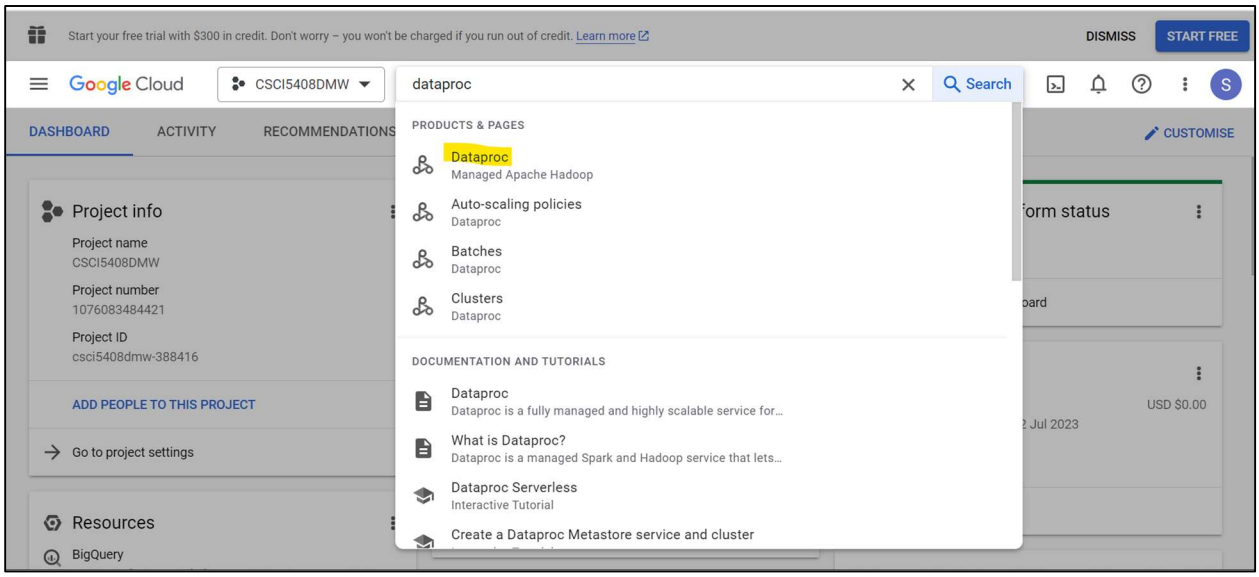

Figure 6: Searching for dataproc resource.
Source: Author

Second, click on "Create Cluster" to create a Apache Spark Cluster (**Figure 7**).

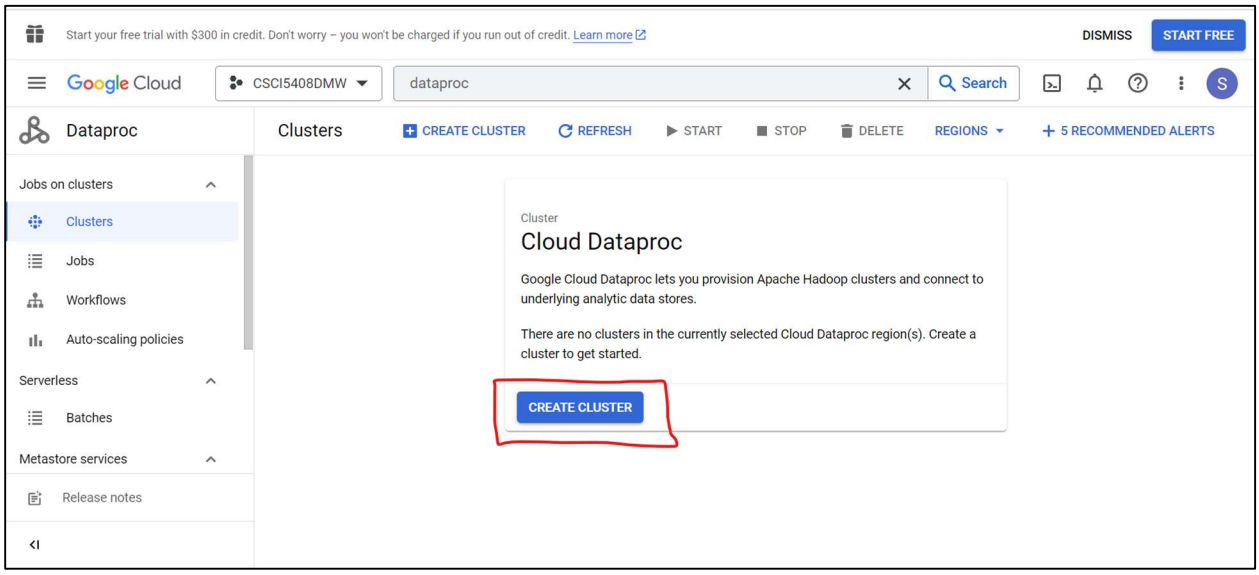

Figure 7: Click on Create Cluster.
Source: Author

Third, choose “Cluster on Compute Engine” option and click on “Create” (**Figure 8**).

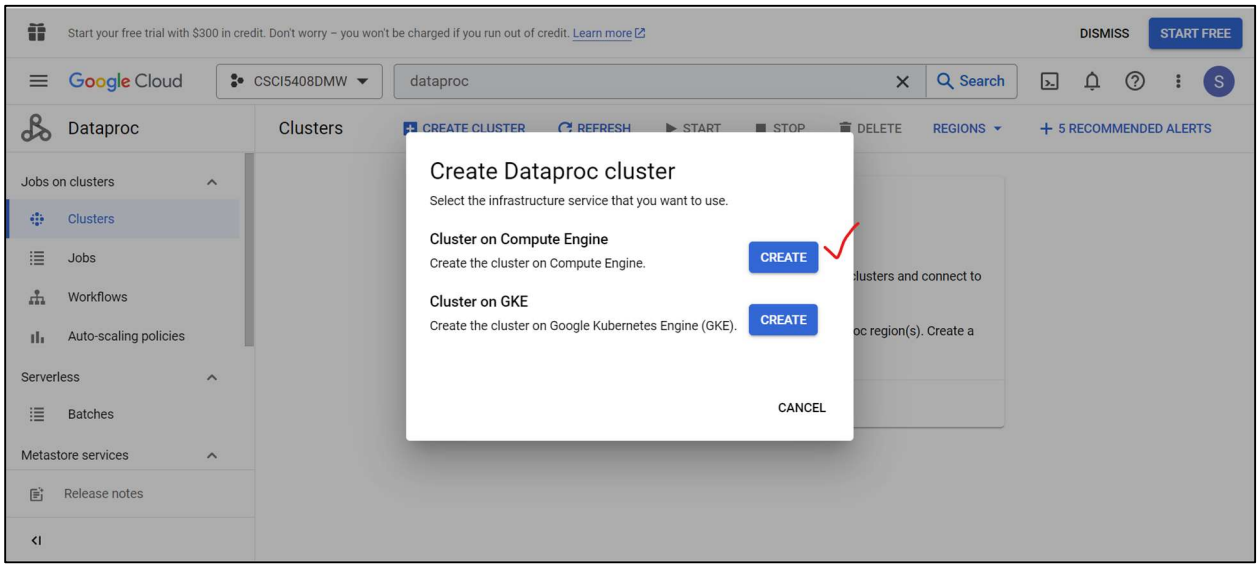

Figure 8: Choose Cluster on Compute Engine option.
Source: Author

Fourth, fill in the required information (Name and Location) and click on create (**Figure 9**).

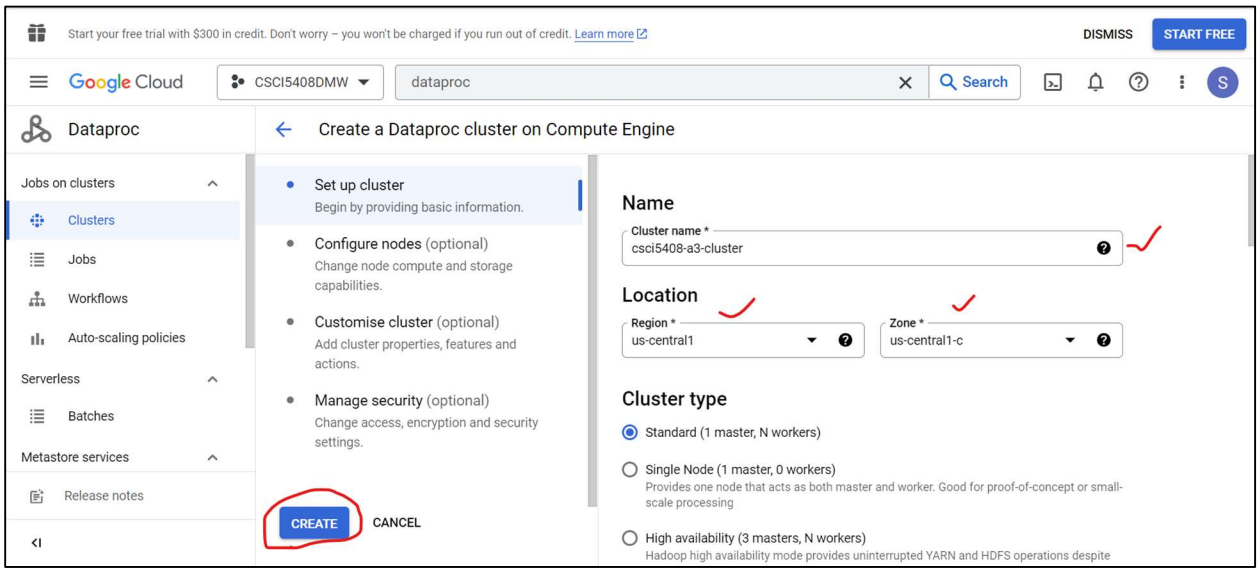

Figure 9: Fill required information.
Source: Author

Once the instance is created, the status will be shown as “Running” (**Figure 10**).

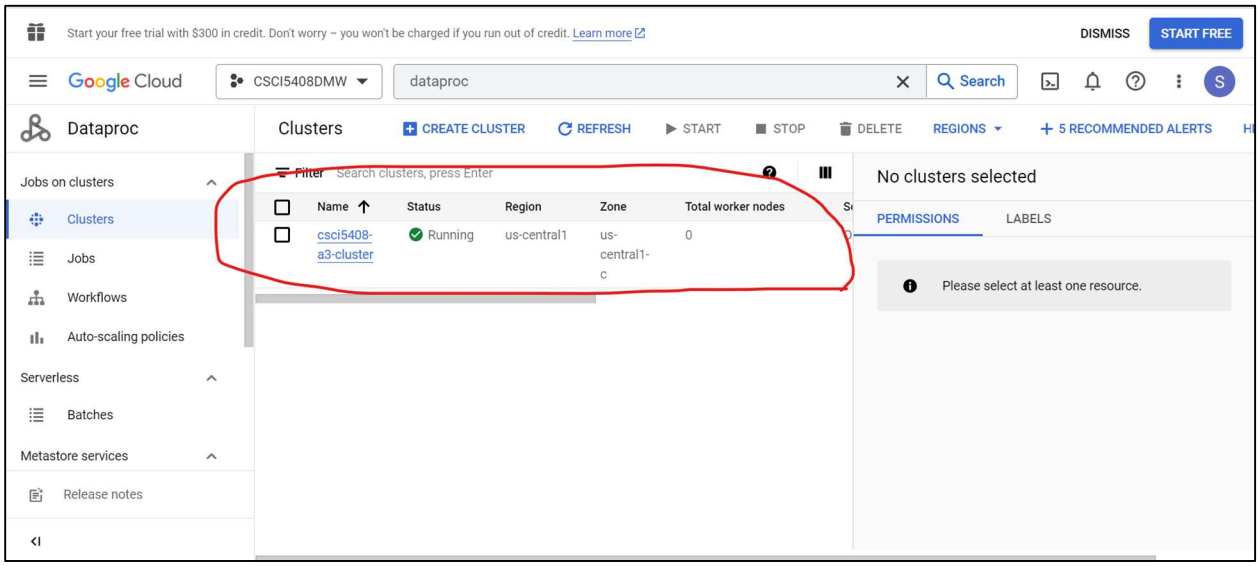

Figure 10: Apache Spark instance is created.
Source: Author

Fifth, click on the cluster name. A new window opens. Click on “VM instances” (**Figure 11**).

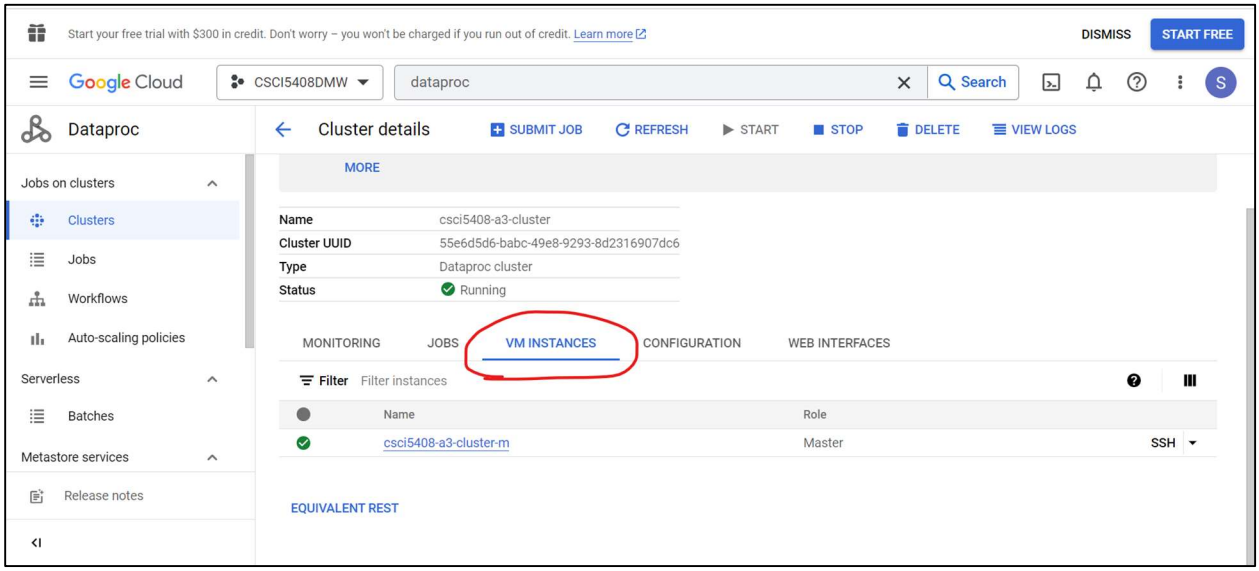

Figure 11: Click on VM instance.
Source: Author

Sixth, click on SSH and choose the “Open in browser window” option (**Figure 12**).

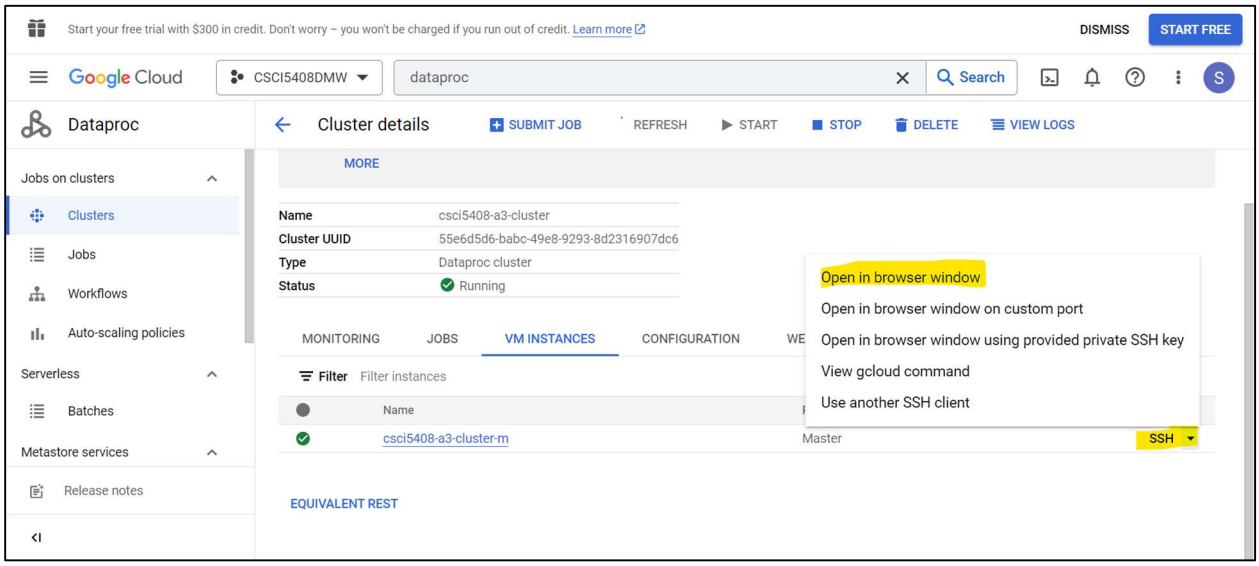

Figure 12: Open cluster through SSH in a new browser window.
Source: Author

Seventh, get the .jar file of the WordCounter.java by using the “install” option for a Maven project (**Figure 13**).

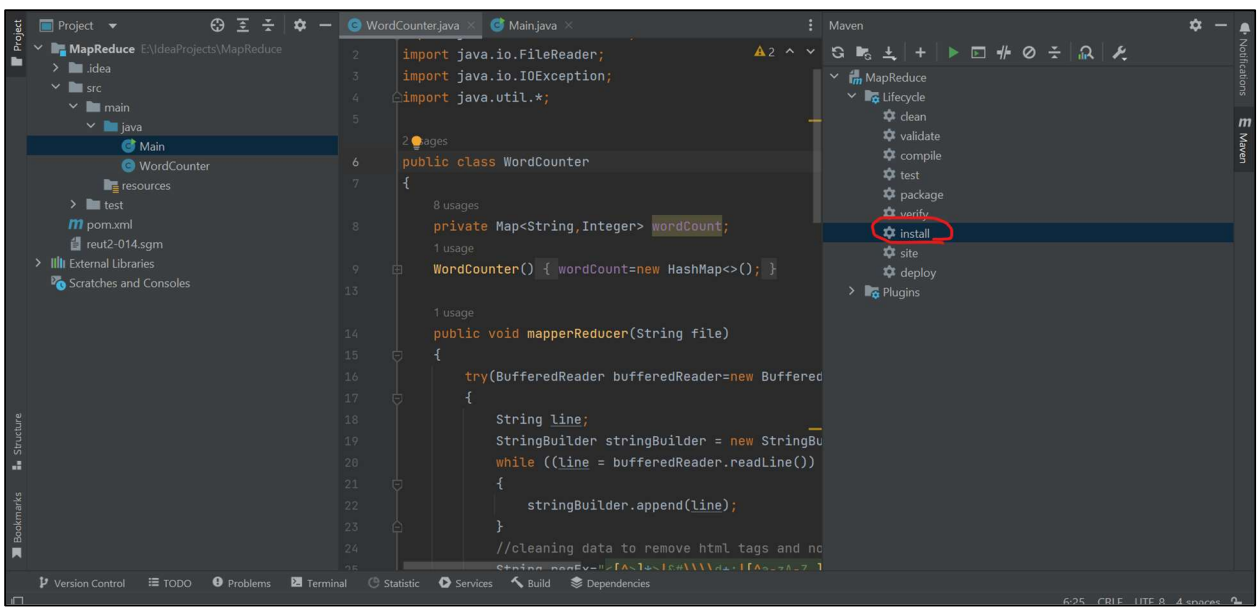

Figure 13: Build .jar file.
Source: Author

Eighth, upload the .jar file and the file whose content is to be read (**Figure 14**).

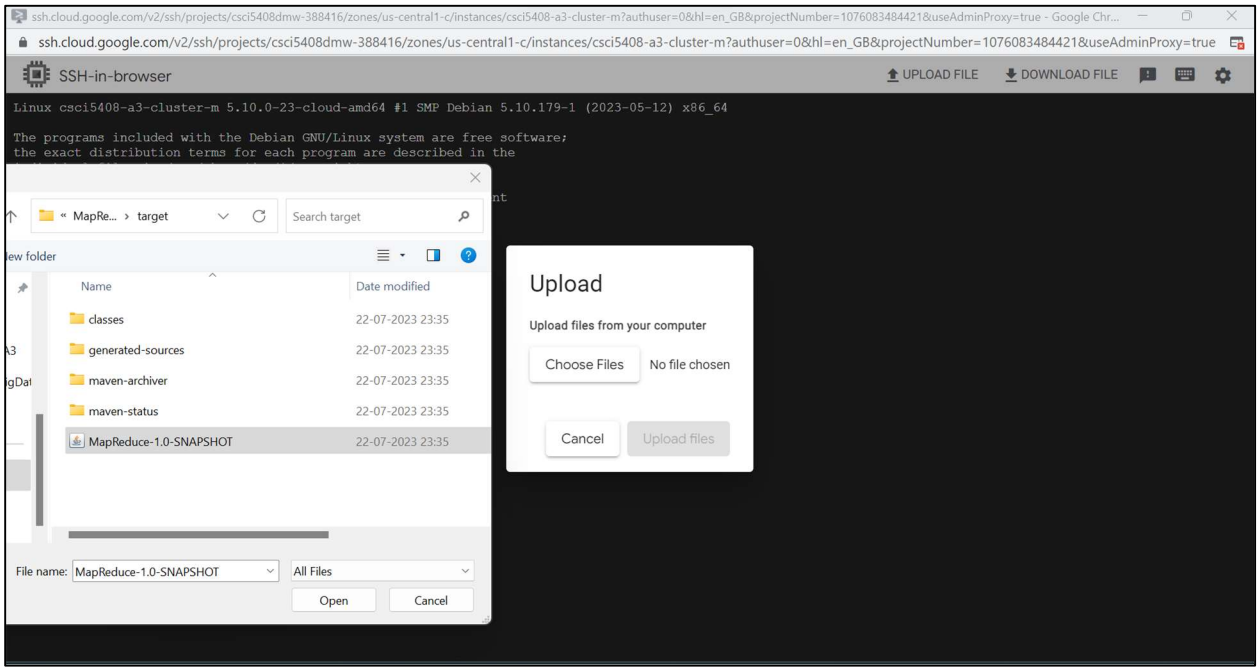

Figure 14: Uploading required files to the Spark Cluster.
Source: Author

Lastly, run the command “spark-submit <jar file name>” (**Figure 15**).

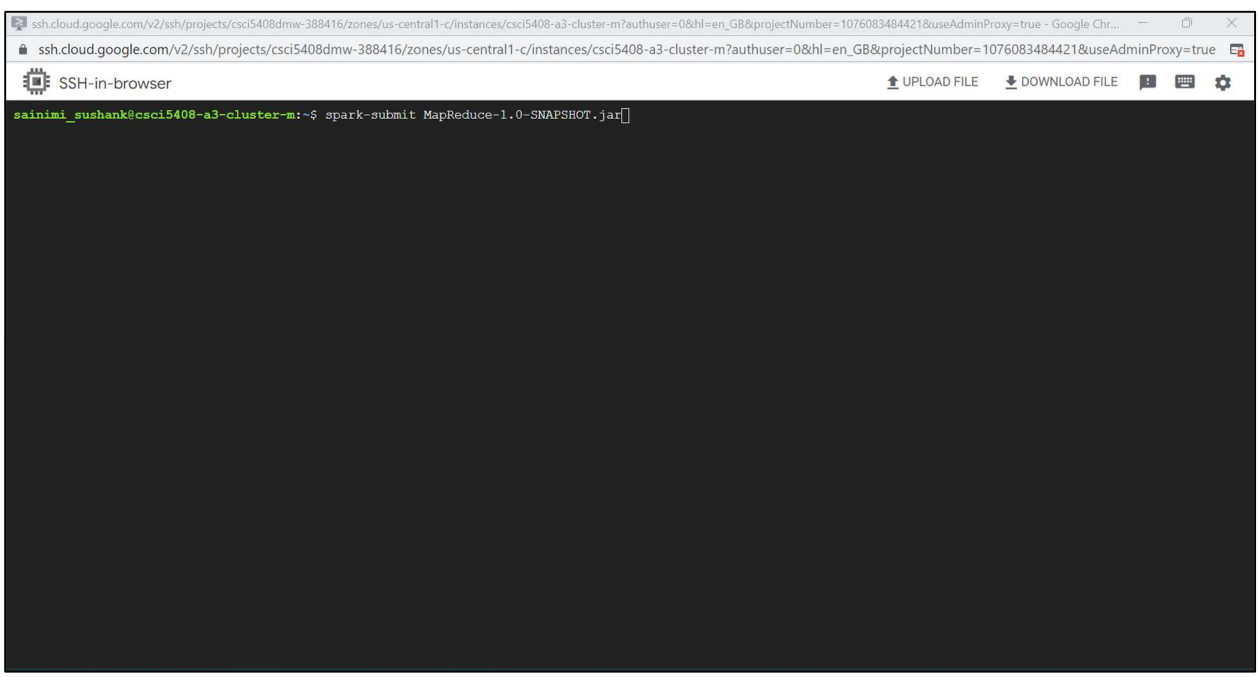

Figure 15: Submitting the job.
Source: Author

Overview of MapReduce Program

The MapReduce is implemented in java through the WordCounter.java program. The program first reads the content from the file and then extracts the unique words from the file. These unique words are then stored in a HashMap as key-value pairs with the word as the key and the count as its value. Lastly, the program reports the words with the highest and lowest frequencies.

Flowchart

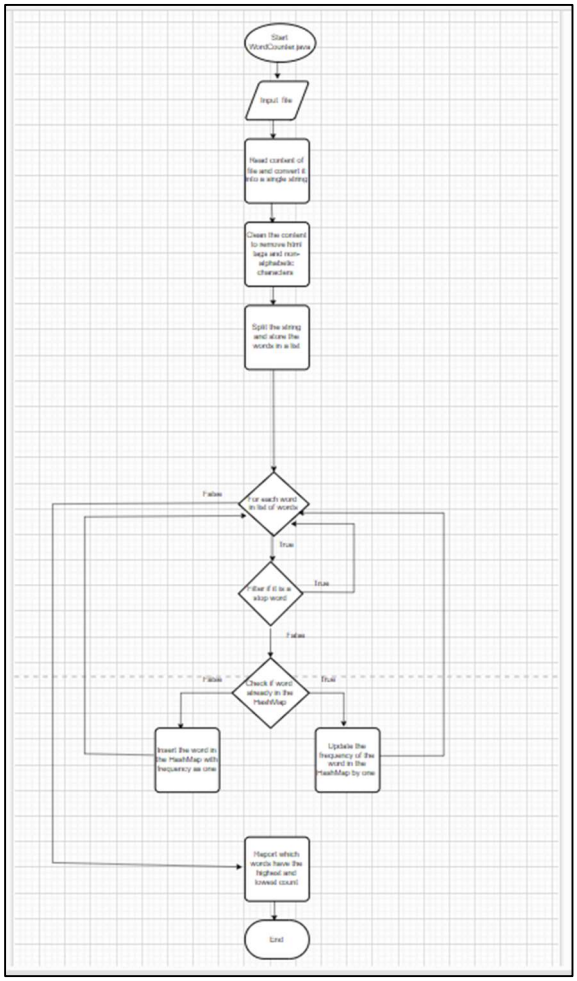

Figure 16: Flowchart for WordCounter.java program.
Source: Author using [6]

Explanation of the code

The program starts by reading the given file using a BufferedReader object and converts the content into a string using StringBuilder (**Figure 17**).

```
public void mapperReducer(String file)
{
    try(BufferedReader bufferedReader=new BufferedReader(new FileReader(file)))
    {
        String line;
        StringBuilder stringBuilder = new StringBuilder();
        while ((line = bufferedReader.readLine()) != null)
        {
            stringBuilder.append(line);
        }
```

Figure 17: Program reads the file content.
Source: Author

The content is then cleaned to remove the tags and non-alphabetic characters using regex [1][2] and then split to store the words in an arraylist (**Figure 18**).

```
//cleaning data to remove html tags and non-alphabetic characters
String regEx="<[^>]*>|&#\\\\d+;|[^a-zA-Z ]";
String fileContent= stringBuilder.toString().toLowerCase().replaceAll(regEx, replacement: "").trim();
List<String> listOfUniqueWords=new ArrayList<>(Arrays.asList(fileContent.split( regex: " ")));
```

Figure 18: Cleaning the content of the file and splitting it to store in an arraylist.
Source: Author

Each word in list of unique words is stored in a HashMap with its frequency. However, prior to that some common stop words [7] are filtered out (**Figure 19**).

```java
//mapping the unique words with their frequencies
for(String word:listOfUniqueWords) {
    //filtering the stop words and ""
    boolean isStopWord = word.equals("") || word.equals("a") || word.equals("an") || word.equals("the")
            || word.equals("and") || word.equals("it") || word.equals("for") || word.equals("or")
            ||word.equals("but")||word.equals("in")||word.equals("my")||word.equals("your")
            ||word.equals("our")||word.equals("they")||word.equals("to")||word.equals("of");
    if(!isStopWord)
    {
        final int one = 1;
        //reducing the unique words and combining their frequencies
        if (wordCount.containsKey(word))
        {
            wordCount.replace(word, wordCount.get(word) + one);
        }
        else
        {
            wordCount.put(word, one);
        }
    }
}
```

Figure 19: Filtering stop words and storing the unique words with their frequencies.
Source: Author

The program then calculates the highest and lowest count in the HashMap using Collections [8] (**Figure 20**).

```java
int maximumFrequency=Collections.max(wordCount.values());
int minimumFrequency=Collections.min(wordCount.values());
```

Figure 20: Finding highest and lowest value in a HashMap.
Source: Author

Then, all the words having the maximum frequency are stored in a list (**Figure 21 and 22**).

```
//getting all words with the maximum count
List<String> wordsWithMaxCount=getWordsWithMaxCount(maximumFrequency);
```

Figure 21: Function call to get all words with maximum count.
Source: Author

```
1 usage   Sushank Saini
private List<String> getWordsWithMaxCount(int maximumFrequency)
{
    List<String> wordsWithMaxCount= new ArrayList<>();
    for(String word: wordCount.keySet())
    {
        if(maximumFrequency==wordCount.get(word))
        {
            String wordAndCount=word+":"+maximumFrequency;
            wordsWithMaxCount.add(wordAndCount);
        }
    }
    return wordsWithMaxCount;
}
```

Figure 22: Function that stores all the words with maximum count.
Source: Author

Similarly, all the words having the minimum frequency are stored in a list (**Figure 23 and 24**).

```
//getting all words with the minimum count
List<String> wordsWithMinCount=getWordsWithMinCount(minimumFrequency);
```

Figure 23: Function call to get all words with minimum count.
Source: Author

```
1 usage   Sushank Saini
private List<String> getWordsWithMinCount(int minimumFrequency)
{
    List<String> wordsWithMinCount= new ArrayList<>();
    for(String word: wordCount.keySet())
    {
        if(minimumFrequency==wordCount.get(word))
        {
            String wordAndCount=word+":"+minimumFrequency;
            wordsWithMinCount.add(wordAndCount);
        }
    }
    return wordsWithMinCount;
}
```

Figure 24: Function that stores all the words with minimum count.
Source: Author

Lastly, the output is printed by calling the printOutput() function (**Figure 25,26 and 27**).

```
//print the output
printOutput(wordsWithMaxCount,wordsWithMinCount);
```

Figure 25: Print function called.
Source: Author

```
private void printOutput(List<String> wordsWithMaxCount, List<String> wordsWithMinCount)
{
    for(String wordWithMaxCount:  wordsWithMaxCount)
    {
        System.out.println(wordWithMaxCount+"\n");
    }
    for(String wordWithMinCount:  wordsWithMinCount)
    {
        System.out.println(wordWithMinCount+"\n");
    }
}
```

Figure 26: Method that prints the output.
Source: Author

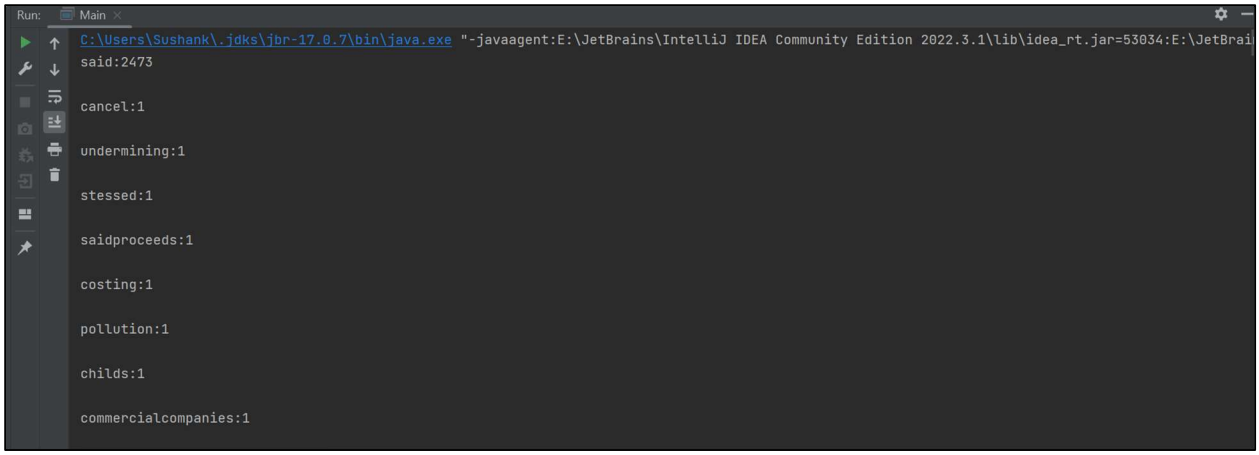

Figure 27: Output.
Source: Author

Problem 2

Overview

The BOWModel.java is program that does the sentiment analysis using bag of words(bow). To create the bow for each title, the program fetches the titles from the MongoDB collection, newsArticles. Each word in title is stored in a map as a key-value pair where the key is the word itself and value is the frequency/count of the word. The words in the bow are then compared with the list of positive [9] and negative words [10] to calculate the overall polarity of the news. This sentiment analysis is represented in a tabular form using the Table.java class that builds a JTable [11].

Explanation of the code

The program begins with the constructor loading the JTable and the list of positive and negative words (**Figures 28,29,30**).

```
public class BOWModel
{
    2 usages
    private Table table;
    2 usages
    private List<String> listOfPositiveWords;
    2 usages
    private List<String> listOfNegativeWords;
    no usages   Sushank Saini
    BOWModel()
    {
        table=new Table();
        listOfPositiveWords=getPositiveWords();
        listOfNegativeWords=getNegativeWords();
    }
```

Figure 28: Constructor loads the table, positive and negative words.
Source: Author

```
1 usage   Sushank Saini
private List<String> getPositiveWords()
{
    List<String> listOfPositiveWords=new ArrayList<>();
    try(BufferedReader bufferedReader=new BufferedReader(new FileReader( fileName: "positivewords.txt")))
    {
        String line;
        while((line=bufferedReader.readLine())!=null)
        {
            String positiveWord=line.trim();
            listOfPositiveWords.add(positiveWord);
        }
        return listOfPositiveWords;
    }
    catch(IOException e)
    {
        System.out.println(e.getMessage());
    }
    return null;
}
```

Figure 29: Function that loads the positive words into an arraylist.
Source: Author

```
private List<String> getNegativeWords()
{
    List<String> listOfNegativeWords=new ArrayList<>();
    try(BufferedReader bufferedReader=new BufferedReader(new FileReader( fileName: "negativewords.txt")))
    {
        String line;
        while((line=bufferedReader.readLine())!=null)
        {
            String negativeWord=line.trim();
            listOfNegativeWords.add(negativeWord);
        }
        return listOfNegativeWords;
    }
    catch(IOException e)
    {
        System.out.println(e.getMessage());
    }
    return null;
}
```

Figure 30: Function that loads the negative words into an arraylist.
Source: Author

Then the titles are fetched [12] from the MongoDB collection, and stored in an arraylist (**Figure 31**).

```
List<String> listOfTitles=new ArrayList<>();
//creating connection with MongoDb to fetch the documents
String uri = "mongodb+srv://sainimisushank:Mongodb13%23@cluster0.qo0raya.mongodb.net/?retryWr
MongoClient mongoClient = MongoClients.create(uri);
MongoDatabase database = mongoClient.getDatabase( s: "reuterDb");
MongoCollection<Document> collection = database.getCollection( s: "newsArticles");
//get all documents from the collection newsArticles
MongoCursor<Document> cursor = collection.find().cursor();
while (cursor.hasNext())
{
    Document document = cursor.next();
    listOfTitles.add(document.get("title").toString().toLowerCase());
}
```

Figure 31: Fetching titles of news articles from the MongoDB collection.
Source: Author

Next, for each title, the words in it are split to form a bow using a HashMap (**Figure 32**).

```
List<String> wordsInTitle = new ArrayList<>(Arrays.asList(title.split( regex: " ")));
//creating bag of words for the title in current iteration
for (String word : wordsInTitle)
{
    final int one = 1;
    if (bagOfWords.containsKey(word))
    {
        bagOfWords.replace(word, bagOfWords.get(word) + one);
    }
    else
    {
        bagOfWords.put(word, one);
    }
}
```

Figure 32: Creating bow for a title.
Source: Author

Then, each word in bow is compared to the list of positive and negative words and accordingly, the polarity of the news is determined (**Figure 33**)

```
//comparing words in bag of words with positive and negative words
for(String word:bagOfWords.keySet())
{
    for(String positiveWord: listOfPositiveWords)
    {
        if(word.equals(positiveWord))
        {
            polarity=polarity+bagOfWords.get(word);
            matches.append(word).append(" ");
            break;
        }
    }
    for(String negativeWord:listOfNegativeWords)
    {
        if(word.equals(negativeWord))
        {
            polarity=polarity-bagOfWords.get(word);
            matches.append(word).append(" ");
            break;
        }
    }
}
```

Figure 33: Comparing each word in bow with list of negative and positive words.
Source: Author

Once the overall polarity is calculated, the title of the news, the matches and the polarity are stored in a 2-D array (**Figure 34)**.

```
if(polarity>0)
{
    data[newsNumber][0]= String.valueOf(newsNumber);
    data[newsNumber][1]=title;
    data[newsNumber][2]=matches.toString();
    data[newsNumber][3]="Positive";

}
else if(polarity<0)
{
    data[newsNumber][0]= String.valueOf(newsNumber);
    data[newsNumber][1]=title;
    data[newsNumber][2]=matches.toString();
    data[newsNumber][3]="Negative";
}
else
{
    data[newsNumber][0]= String.valueOf(newsNumber);
    data[newsNumber][1]=title;
    data[newsNumber][2]=matches.toString();
    data[newsNumber][3]="Neutral";
}
```

Figure 35: Storing title, matches and polarity in a 2-D array.
Source: Author

This 2-D array is then passed to the createTable() of the Table.java class (**Figure 36**)which builds the JTable to display the output of the sentiment analysis (**Figure 37**).

```
1 usage   Sushank Saini
public void createTable(String data [][])
{
    String columnNames[]={"News#","Title","Matches","Polarity"};
    jTable=new JTable(data,columnNames);
    TableColumnModel columnModel=jTable.getColumnModel();
    columnModel.getColumn( columnIndex: 0).setPreferredWidth(50);
    columnModel.getColumn( columnIndex: 1).setPreferredWidth(300);
    columnModel.getColumn( columnIndex: 2).setPreferredWidth(100);
    columnModel.getColumn( columnIndex: 3).setPreferredWidth(50);
    JScrollPane jScrollPane=new JScrollPane(jTable);
    jFrame.add(jScrollPane);
    jFrame.setSize( width: 1000, height: 500);
    jFrame.setVisible(true);
}
```

Figure 36: 2-D array passed to represent it in JTable.
Source: Author

SENTIMENT ANALYSIS

News#	Title	Matches	Polarity
1	advanced magnetics <admg> in agreement	advanced	Positive
2	health research files for bankruptcy		Neutral
3	numerex corp <nmrx> 2nd qtr jan 31 loss	loss	Negative
4	u.s. selling 12.8 billion dlrs of 3 and 6-mo bills march 30 to pay down 1.2 billion dlrs</title>blah blah blah.&#3;</text></reuters><r...	blah awarded	Negative
5	baldrige supports nic talks on currencies	supports	Positive
6	triangle <tri> begins exchange offer		Neutral
7	southmark <sm> unit in public offering of stock		Neutral
8	eastman kodak co to sell holdings in icn pharmaceuticals and viratek inc</title>blah blah blah.&#3;</text></reuters><reuters topi...	blah	Negative
9	treasury balances at fed rose on march 23		Neutral
10	farm credit system seen needing 800 mln dlrs aid		Neutral
11	usx <x> uss unit raises prices		Neutral
12	unionist urges retaliation against japan		Neutral
13	exxon (xon) gets 99.2 mln dlr contract		Neutral
14	eaton (etn) gets 53.0 mln dlr contract		Neutral
15	zaire authorized to buy pl 480 rice - usda		Neutral
16	mcdonnell douglas gets 30.6 mln dlr contract		Neutral
17	midivest acquires assets of business aviation		Neutral
18	u.s. wheat credits for jordan switched		Neutral
19	dollar expected to fall despite intervention</title><author> by claire miller, reuters</author><dateline> new york, march 24 - </...	doubts strong helped trust hard watered-down rumors break stabiliz...	Positive
20	inland steel <iad> to build new plant in indiana		Neutral
21	eastman kodak <ek> to sell holdings		Neutral
22	guinness sues boesky in federal court	sues	Negative
23	firm reduces sceptre resources <srl> holdings		Neutral
24	gencorp board withdraws proposals to stagger directors terms</title>blah blah blah.&#3;</text></reuters><reuters topics="yes" le...	blah	Negative
25	bull and bear group a <bnbga> cuts fund payouts		Neutral
26	caltex to raise bahrain oil product prices		Neutral
27	charter co <qchr> to complete reorganization		Neutral
28	nashua <nsh> to purchase private disc maker		Neutral
29	altron inc <alrn> 4th qtr jan 3		Neutral
30	collins foods <cf> moves up warrant conversion		Neutral
31	gencorp <gy> proposals withdrawn from meeting		Neutral
32	weekly electric output up 2.4 pct from 1986		Neutral
33	monolithic <mmic> to drop gate array line		Neutral
34	api says distillate stocks off 4.07 mln bbls, gasoline off 2.69 mln, crude up 8.53 mln</title>blah blah blah.&#3;</text></reuters><re...	crude blah	Negative
35	resorts international gets buyout proposal from ksz co inc</title>blah blah blah.&#3;</text></reuters><reuters topics="no" lewissp...	blah	Negative
36	corrected - att <t> forms computer sales groups		Neutral
37	great atlantic and pacific tea co inc <gap> div	great	Positive
38	gartner group <gart> acquires comtec program		Neutral

Figure 37: Sentiment Analysis output.
Source: Author

References

[1] *Regular expressions 101* [Online]. Available: https://regex101.com/. [Accessed: July 19,2023].

[2] GeeksforGeeks ,“How to write Regular Expressions?”, *GeeksforGeeks* [Online]. Available: https://www.geeksforgeeks.org/write-regular-expressions/. [Accessed: July 19,2023].

[3] Class Pattern”, *Java™ Platform Standard Ed. 8* [Online]. Available: https://docs.oracle.com/javase/8/docs/api/java/util/regex/Pattern.html. [Accessed: July 19,2023].

[4] “Java Regular Expressions”, *w3schools* [Online]. Available: https://www.w3schools.com/java/java_regex.asp. [Accessed: July 19,2023].

[5] “Insert a Document”, *MongoDB* [Online]. Available: https://www.mongodb.com/docs/drivers/java/sync/current/usage-examples/insertOne/#insert-a-document. [Accessed: July 20, 2023].

[6] *draw.io* [Online]. Available: https://app.diagrams.net/. [Accessed: July 20,2023].

[7] C.Fontanella, “75 Stop Words That Are Common in SEO & When You Should Use Them”, *HubSpot* [Online]. Available: https://blog.hubspot.com/marketing/stop-words-seo#. [Accessed: July 21, 2023].

[8] R. Maurya, “Find Smallest and Largest Value in a Map”, *HowToDoInJava* [Online]. Available: https://howtodoinjava.com/java/collections/hashmap/smallest-largest-value-in-map/. [Accessed: July 21,2023].

[9] https://ptrckprry.com/course/ssd/data/positive-words.txt. [Accessed: July 22,2023].

[10] https://ptrckprry.com/course/ssd/data/negative-words.txt. [Accessed: July 22,2023].

[11] ShivamKD, “Java Swing | JTable”, *GeeksforGeeks* [Online]. Available: https://www.geeksforgeeks.org/java-swing-jtable/. [Accessed: July 23,2023].

[12] “Access Data from a Cursor”, *MongoDB* [Online]. Available: https://www.mongodb.com/docs/drivers/java/sync/v4.3/fundamentals/crud/read-operations/cursor/. [Accessed: July 24,2023].