



**DALHOUSIE
UNIVERSITY**

CSCI 5408

**Data Management, Warehousing and
Analytics**

Assignment 3

Name : Sagarkumar Pankajbhai Vaghasia

CSID : vaghasia

Banner ID : #B00878629

GitLab Link :

https://git.cs.dal.ca/vaghasia/csci5408_s22_sagarkumar_vaghasia_b00878629/-/tree/main/A3

Problem #1

Task – 1 : News Article Data Extraction and Transformation

I have created three classes named as Main, NewsExtraction and NewsFiltration.

The NewsExtraction class is responsible for extracting news data from the website <https://newsapi.org/> [1] and then I stored the extracted news data unfiltered in text files. I have code in such a way that each text file should contain three articles. I have searched the articles based on the keywords Canada, University, Dalhousie University, Halifax, Canada Education, Moncton, Toronto, Oil, and Inflation.

For each keyword, I have permitted to extract hundred articles. The news articles are stored in TXT files. I have named created a directory named “newsdata” and stored all the articles file in that file.

The naming convention for the text files I used is <search_keyword_lowercase>_<current_time>.txt. To exemplify, canada_1657567359840.txt and dalhousieuniversity_1657640150597.txt

Also, I have used JSON-java library [2] in the NewsExtraction to store the response from NewsAPI.

Main.java

This class do not have any methods. It is used to call the methods of NewsExtraction and NewsFiltration by using objects.

NewsExtraction.java

I have created static variables to store directory name, newsapi url, newsapi key, articles permitted, category keywords.

Methods in NewsExtraction :

- **extractNewsData()**

This method acts like a controller which I used to call different methods in the class.

I have used for loop for all the keywords to fetch and store the news articles by calling relevant methods.

- **fetchNews()**

This method is responsible to fetch news articles from newsapi website where I passed the argument as category keyword. I have used HttpRequest, HttpClient and HttpResponse. I have stored the fetched data in String variable.

- **prepareAndStoreNews()**

This function is responsible prepare and store news articles. Here, I have passed the arguments category keyword and the obtained news article string.

- **storeNews()**

This method is responsible to store news in the local disk. If the directory is not there to store files, then this function will create new directory.

The below attached figure displays all the text files containing news articles extracted for search keywords.

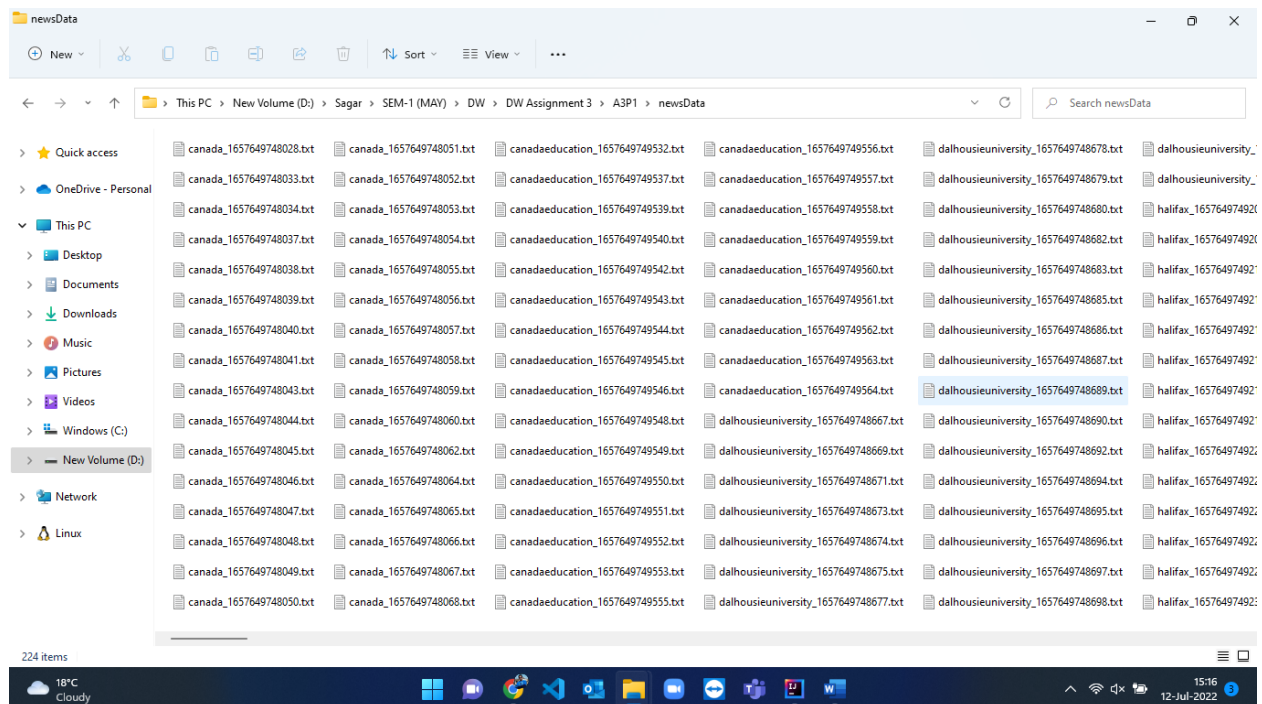


Figure 1 - Text files after extraction of news articles.

NewsFiltration.java

This java class I have created to filter the news articles that is stored in the directory and to upload the filtered data to the mongodb database [3].

Firstly, I have declared static variables such as SEPARATOR_REPLACEMENT, SEPARATOR_SPLIT, EMPTY_STRING for filtration. Then, I have made connection with the mongodb atlas by using the url for connection.

Methods in NewsFiltration :

- **filterNewsData()**

This method acts like a controller which I used to call different methods in the class. Firstly, I read all the files that are stored in a directory in the form of array. Then, I called readAndFilterArticles method to filter the articles and then I have inserted the obtained filtered data to the mongodb database. I have also created arraylist for storing the mongoNewsDocuments in the form of list.

- **readAllFileNames()**

This method is responsible for reading the files that are stored in the directory newsdata.

- **readAndFilterArticles()**

In this method, I have passed two arguments : allNewsFiles and mongoNewsDocuments. This function is responsible for calling filterArticleContent() method and store the obtained filtered content in the String filteredArticleList. In this method, I have also parsed the obtained filtered articles data and added it to the mongoNewsDocuments.

- **filterArticleContent()**

This method contains one argument of String for the content of the article (articleContent). This method is responsible for actual filtration where I have used regular expressions [4][5][6][7] to filter the articles.

In this method, I have replaced emojis, urlToImage, url, author, id, and HTML tags from all articles with empty string.

After filtering, the obtained String is parsed into MongoDB document and stored as a list. At the end, after filtering all the articles a connection is made with MongoDB by using a connection string. Then, using insertMany() command, I have uploaded the documents into the myMongoNews database.

Algorithm for NewsFiltration

- 1) Start
- 2) Read name of all the text file that is extracted.
- 3) Initialize an empty list of MongoDB documents (mongoNewsDocuments).
- 4) Initialize counter variable to count the total number of articles read totalArticlesRead.
- 5) Iterate through all the text news articles file names.
- 6) All files reading finish? If yes, then go to step 11 else go to step 7
- 7) Read content of the current news article file.
- 8) Filter the content of the current news article file using the custom regexes.
- 9) Split the filtered news articles file content to separate news articles.
- 10) Parse each news article as a MongoDB document and store it in a list of MongoDB document and increment totalArticlesRead
- 11) Display total articles read and filtered(totalArticlesRead).
- 12) Connect to MongoDB atlas using mongo connection uri
- 13) Get the reference to the mongo database myMongoNews
- 14) Insert the created list of mongo news articles documents.
- 15) Display article stored successfully.
- 16) Stop

The attached figure displays the news articles uploaded to MongoDB database myMongoNews.

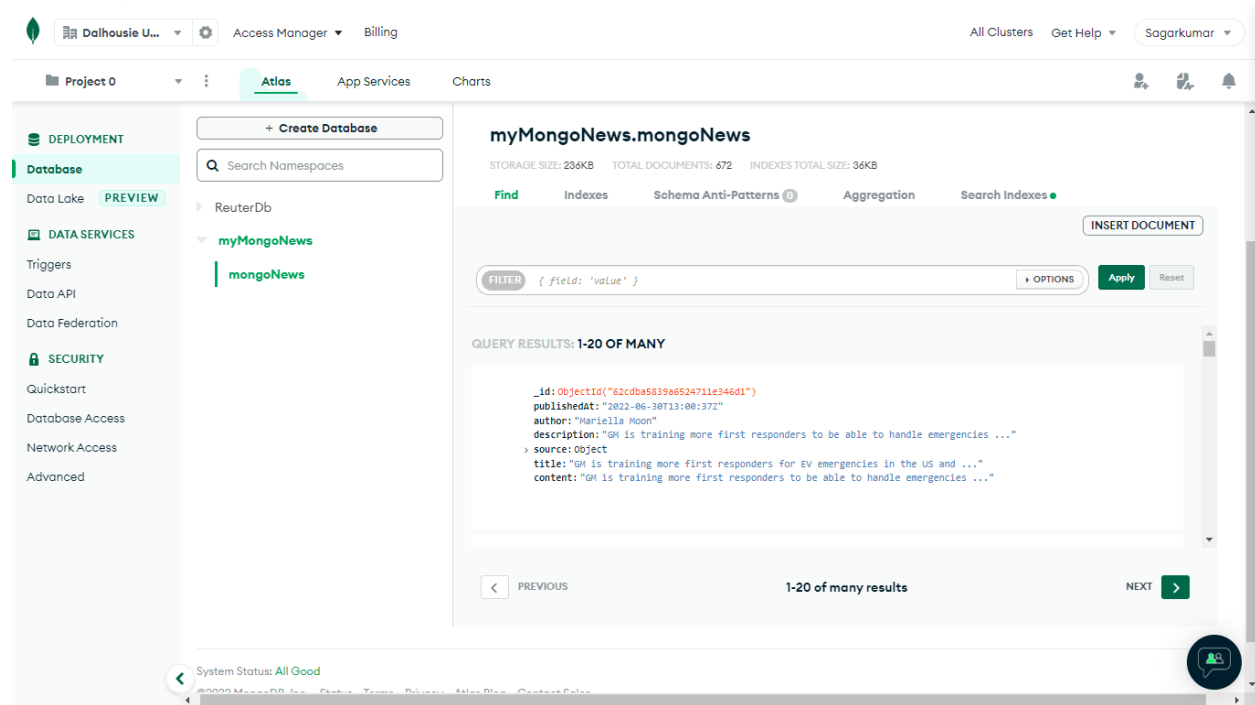


Figure 2 – NewsData uploaded to MongoDB database myMongoNews

Task – 2 : Movie Data Extraction and Transformation

I have created three classes named as Main, MovieDBExtraction and MovieFiltration.

The MovieDBExtraction class is responsible for extracting movies data from the website <https://www.omdbapi.com/> [8] and then I stored the extracted movies data unfiltered in text files. I have code in such a way that each text file should contain three articles. I have searched the articles based on the keywords Canada, University, Moncton, Halifax, Toronto, Vancouver, Alberta, and Niagara.

For each keyword, I have permitted to extract hundred articles. The movie articles are stored in TXT files. I have named created a directory named “moviesdata” and stored all the articles file in that file.

The naming convention for the text files I used is <search_keyword_lowercase>_<current_time>.txt. To exemplify, university_1657650860528.txt and niagara_1657650862012.txt

Also, I have used JSON-java library [2] in the MovieDBExtraction to store the response from omdbapi.

Main.java

This class do not have any methods. It is used to call the methods of MovieDBExtraction and MovieFiltration by using objects.

MovieDBExtraction.java

I have created static variables to store directory name, moviesapi url, moviesapi key, articles permitted, category keywords.

Methods in MovieExtraction :

- **extractMoviesData()**

This method acts like a controller which I used to call different methods in the class.

I have used for loop for all the keywords to fetch and store the movies articles by calling relevant methods.

- **fetchMovies()**

This function is responsible to fetch movies articles from omdbapi website where I passed the argument as category keyword. I have used HttpRequest, HttpClient and HttpResponse. I have stored the fetched data in String variable.

- **prepareAndStoreMovies()**

This method is responsible prepare and store movie articles. Here, I have passed the arguments category keyword and the obtained movie article string.

- **storeMovies()**

This method is responsible to store movies in the local disk. If the directory is not there to store files, then this function will create new directory.

The below attached figure displays all the text files containing movie articles extracted for search keywords.

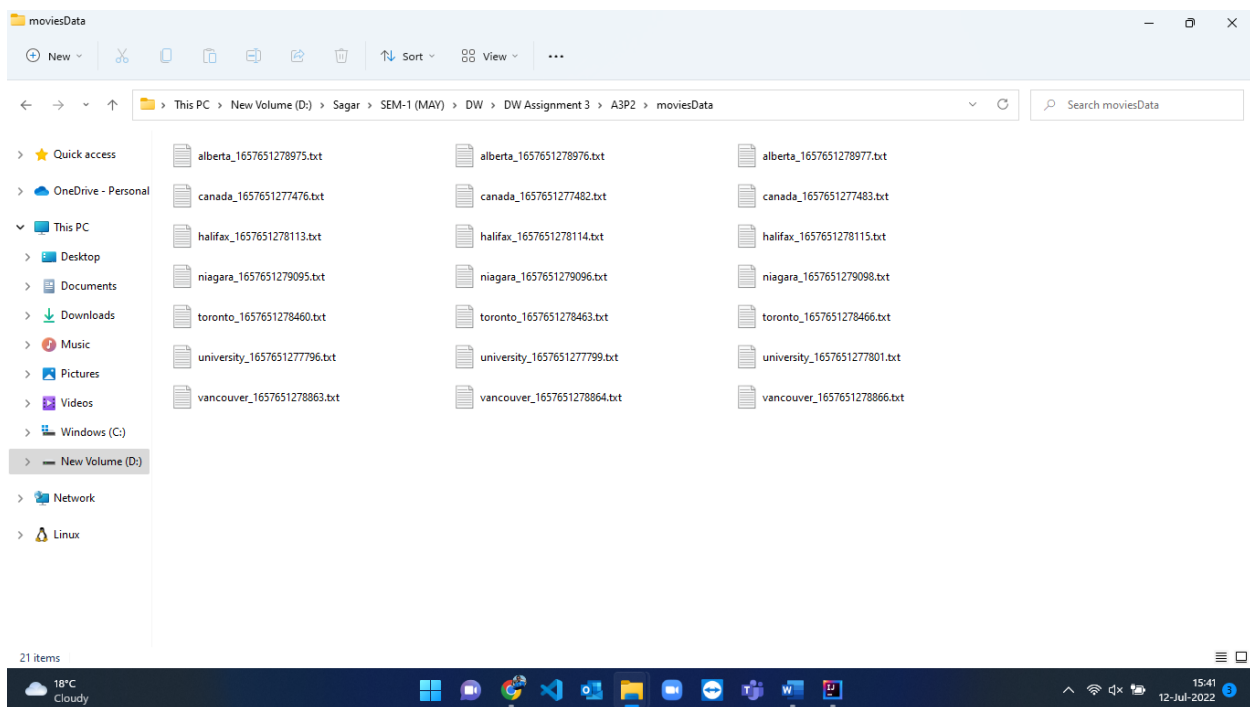


Figure 3 - Text files after extraction of movie articles.

MoviesFiltration.java

This java class I have created to filter the movies articles that is stored in the directory and to upload the filtered data to the mongodb database [3].

Firstly, I have declared static variables such as SEPARATOR_REPLACEMENT, SEPARATOR_SPLIT, EMPTY_STRING for filtration. Then, I have made connection with the mongodb atlas by using the url for connection.

Methods in MoviesFiltration :

- **filterMovieData()**

This method acts like a controller which I used to call different methods in the class.

Firstly, I read all the files that are stored in a directory in the form of array. Then, I called readAndFilterArticles method to filter the articles and then I have inserted the obtained filtered data to the mongodb database. I have also created arraylist for storing the mongoMoviesDocuments in the form of list.

- **readAllFileNames()**

This method is responsible for reading the files that are stored in the directory moviesdata.

- **readAndFilterArticles()**

In this method, I have passed two arguments : allmoviesFiles and mongoMoviesDocuments. This function is responsible for calling filterArticleContent() method and store the obtained filtered content in the String filteredArticleList. In this method, I have also parsed the obtained filtered articles data and added it to the mongoMoviesDocuments.

- **filterArticleContent()**

This method contains one argument of String for the content of the article (articleContent). This method is responsible for actual filtration where I have used regular expressions [4][5][6][7] to filter the articles.

In this method, I have replaced emojis, urlToImage, url, author, id, and HTML tags from all articles with empty string.

After filtering, the obtained String is parsed into Mongodb document and stored as a list. At the end, after filtering all the articles a connection is made with Mongodb by using a connection string. Then, using insertMany() command, I have uploaded the documents into the myMongoMovie database.

Algorithm for NewsFiltration

- 1) Start
- 2) Read name of all the text file that is extracted.
- 3) Initialize an empty list of Mongodb documents (mongoMoviesDocuments).
- 4) Initialize counter variable to count the total number of articles read
totalArticlesRead.
- 5) Iterate through all the text movie articles file names.
- 6) All files reading finish? If yes, then go to step 11 else go to step 7
- 7) Read content of the current movie article file.
- 8) Filter the content of the current movie article file using the custom regexes.
- 9) Split the filtered movie articles file content to separate movie articles.
- 10) Parse each movie article as a Mongodb document and store it in a list of
Mongodb document and increment totalArticlesRead
- 11) Display total articles read and filtered(totalArticlesRead).
- 12) Connect to Mongodb atlas using mongo connection uri
- 13) Get the reference to the mongo database myMongoMovie
- 14) Insert the created list of mongo movie articles documents.
- 15) Display article stored successfully.
- 16) Stop

The below attached figure displays the movie articles uploaded to MongoDB database myMongoMovie.

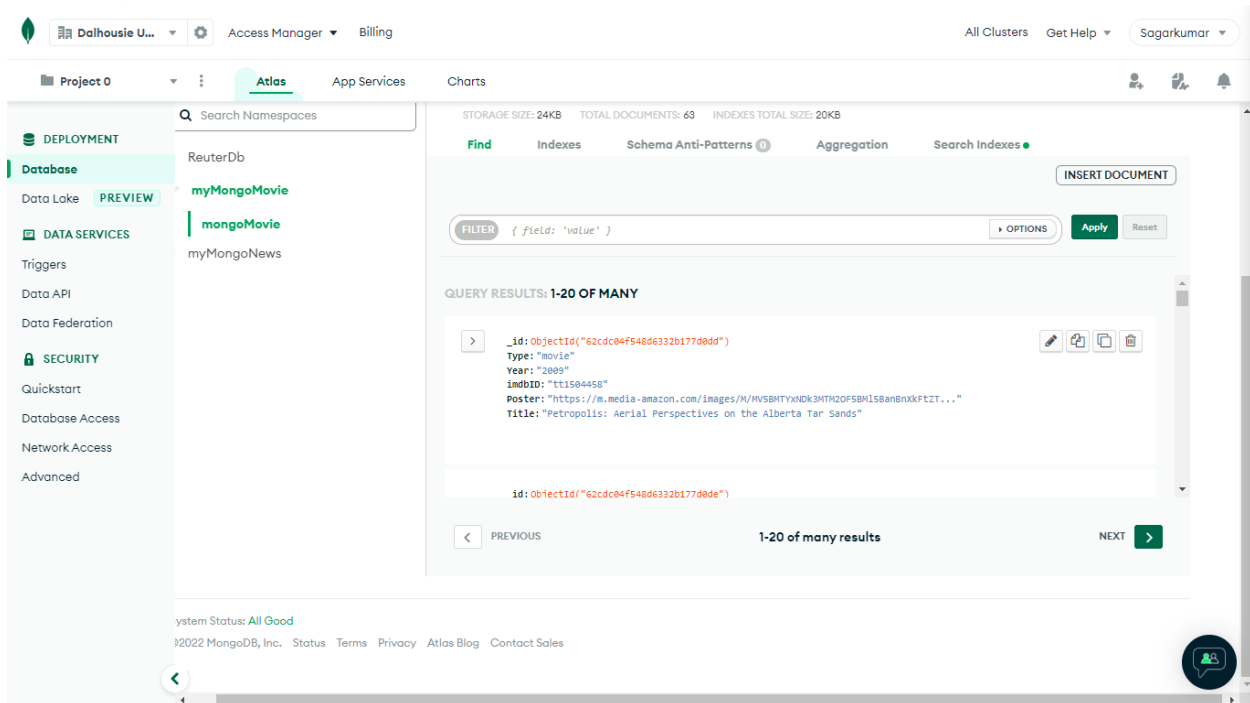


Figure 4 – MovieData uploaded to MongoDB database myMongoMovie.

Task – 3 : Reuter News Data Reading and Transformation

I have created five classes named as Main, Reuter, ReuterParse, ReuterStore and MongoDBConnection.

Main.java

This class do not have any methods. It is used to call the methods of ReuterParse and ReuterStore by using objects. I have also assigned connection url for Mongoddb to databaseString

ReuterParse.java

I have created a static string array to store file names. This class is responsible for extraction of reuter data and then filtering that extracted data.

Methods in ReuterParse :

- **parseAllFiles()**

This method iterates through all the file names and call parseSGM() to extract the reuter data from the file.

- **parseSGM()**

This method takes one parameter which is file name. This method is responsible for reading file into string by using filterReuter() [9]. After filtration, it passes the string from the regular expression to obtain the reuter data.

- **filterReuter()**

This method is responsible for filtering the reuter data. I have replaced url, special word, white space, special character and emojis by empty string.

MongoDBConnection.java

This class is responsible for connection with the Mongoddb Atlas. The constructor of this class creates the default objects for the connection. This class also contains the objects of MongoClient, MongoDBDatabase and ConnectionString.

Methods in MongoDBConnection:

- **getCollection()**

In this method, I passed one parameter of collection name and returns the mongodb collection object.

- **setDatabase()**

In this method, I passed one parameter for the database name and returns the mongodb database object

Reuter.java

This class has two String variables: title and text. The constructor of this class sets the defines the values. This class only contains getters and setters for title and text.

Methods in Reuter: setTitle(), getTitle(), setText(), getText().

ReuterStore.java

This class is responsible for storing the extracted and filtered data to mongodb database. This has MongoDBConnection and MongoCollection objects.

Methods in ReuterStore:

- **storeAll()**

This method takes HashMap as a parameter and it iterates through all map entry in HashMap and get the key(file name) and value pairs(list of reuter objects). After that, it store() method.

- **store()**

This method is responsible for storing the document to the mongodb database. This method takes reuter object as a parameter. After that, it creates a document object and then it calls insertOne() method to upload that document to the database.

Algorithm for Reuter Data Cleaning and Transformation.

- 1) Start
- 2) Read the reuter files.
- 3) Filter the content by using the custom regexes.
- 4) Write the filtered string to the text file.
- 5) Parse the clean string from the regular expression to extract information between <REUTERS></REUTERS> tags.
- 6) The information extracted between the REUTERS tags will be parsed to extract the information between <TEXT></TEXT> tags.
- 7) The information extracted between the TEXT tags will be parsed to extract the information between <TITLE></TITLE> tags.
- 8) The information extracted between the TITLE tags will be parsed to extract the information between <BODY></BODY> tags.
- 9) Create the reuter class object for each text and title .
- 10) Create connection to the MongoDB database and upload the data to the MongoDB database.
- 11) Stop

The below attached figures displays the reuter documents uploaded to MongoDB database ReuterDb.

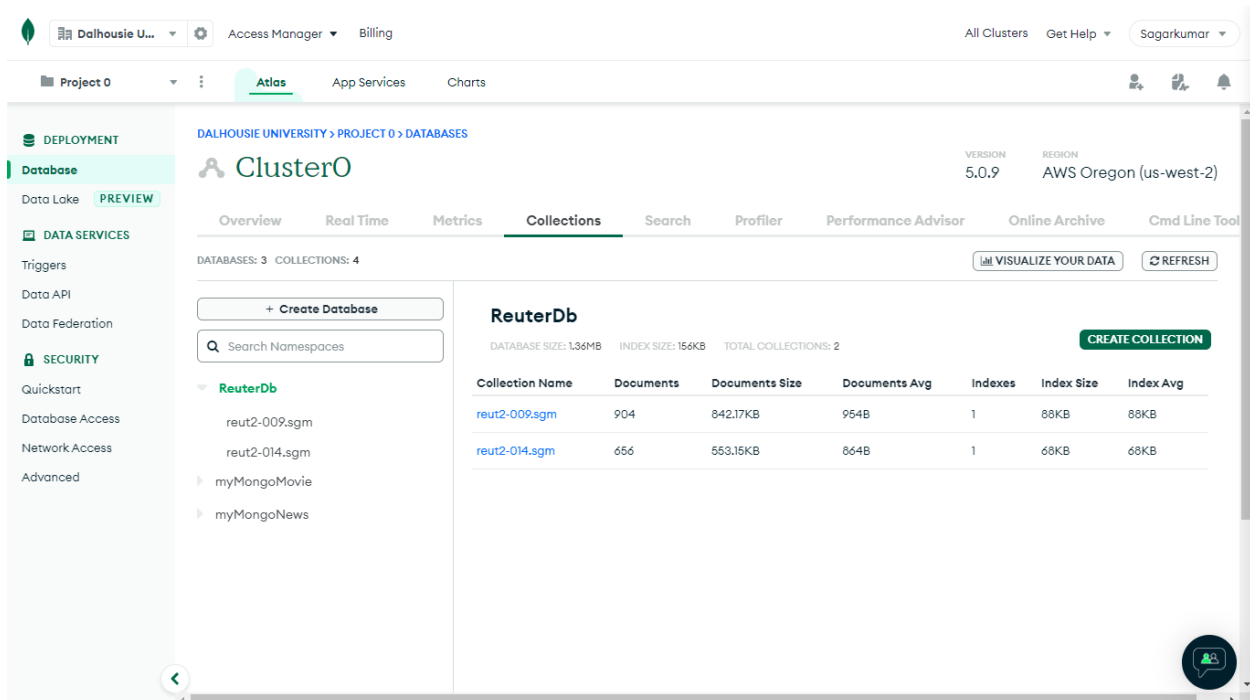


Figure 5 – Reuter Documents uploaded to MongoDB database ReuterDb.

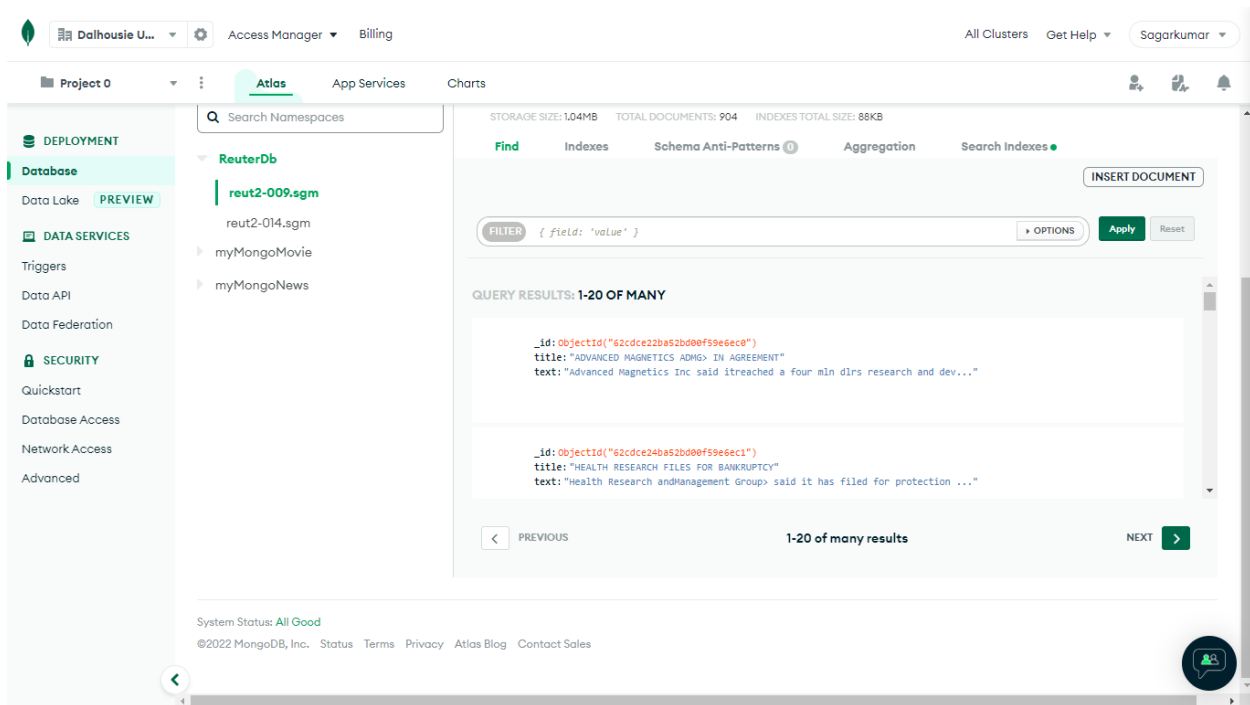


Figure 6 – ReuterData uploaded to Mongodb database ReuterDb.

Problem #2

Task – 1 : Data Processing Using Spark – MapReduce to perform count

Firstly, I have created the vm instance named as vminstance1 on GCP [10]. Then, on the left side panel, below the VPC network, click Firewall option. Edit settings for default-allow-http and default-allow-https to make it allow all. The below attached figure displays the created virtual machine (VM) instance vminstance1.

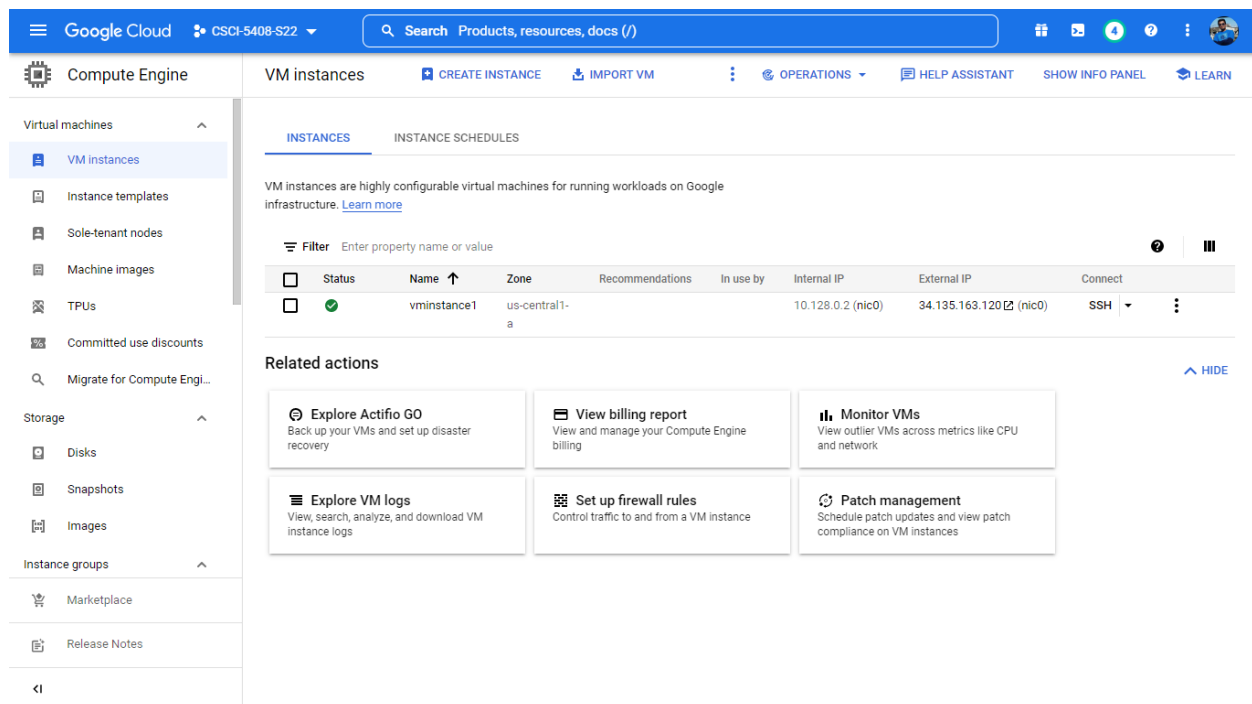


Figure 7 –vminstance1 on GCP.

Then, I connect to the vminstance1 shell by clicking SSH under connect option. After that, I downloaded the tar file of Apache spark using wget utility [11] [12]. Then, I extracted the tar file and moved extracted files to /opt/sparkfolder. After moving to that folder, I added path of the spark folder to \$SPARK_HOME environment variable and I have also added path of bin folder inside spark folder to \$PATH environment. I started master node by using the command : start-master.sh


```

https://ssh.cloud.google.com/v2/ssh/projects/csci-5408-s22-352820/zones/us-central1-a/instances/vminstance1?authuser=0&hl=en_US&projectNumber=720520269258&useAdminProxy=true&troubleshoot4005Enabled
ssh.cloud.google.com/v2/ssh/projects/csci-5408-s22-352820/zones/us-central1-a/instances/vminstance1?authuser=0&hl=en_US&projectNumber=720520269258&useAdminProxy=true&

SSH-in-browser

Welcome to Ubuntu 20.04.4 LTS (GNU/Linux 5.13.0-1031-gcp x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Tue Jul 12 20:41:27 UTC 2022

System load:  0.0               Processes:    109
Usage of /:   42.2% of 9.5GB     Users logged in: 0
Memory usage: 7%               IPv4 address for ens4: 10.128.0.2
Swap usage:  0%

19 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

*** System restart required ***
Last login: Mon Jul 11 23:40:43 2022 from 35.235.244.33
sagarvaghasia372_ss@vminstance1:~$ start-master.sh
starting org.apache.spark.deploy.master.Master, logging to /opt/spark/logs/spark-sagarvaghasia372_ss-org.apache.spark.deploy.master.Master-1-vminstance1.out
sagarvaghasia372_ss@vminstance1:~$ start-worker.sh spark://vminstance1.us-central1-a.c.csci-5408-s22-352820.internal:7077

```

I started worker node by using the command : `start-worker.sh spark://vminstance1.us-central1-a.c.csci-5408-s22-352820.internal:7077`

```

sagarvaghasia372_ss@vminstance1:~$ start-worker.sh spark://vminstance1.us-central1-a.c.csci-5408-s22-352820.internal:7077
starting org.apache.spark.deploy.worker.Worker, logging to /opt/spark/logs/spark-sagarvaghasia372_ss-org.apache.spark.deploy.worker.Worker-1-vminstance1.out
sagarvaghasia372_ss@vminstance1:~$ ||

```

The below attached screenshot is of master and worker node started in spark.

Spark Master at spark://vminstance1.us-central1-a.c.csci-5408-s22-352820.internal:7077

URL: spark://vminstance1.us-central1-a.c.csci-5408-s22-352820.internal:7077

Alive Workers: 1

Cores in use: 2 Total, 0 Used

Memory in use: 2.8 GiB Total, 0.0 B Used

Resources in use:

Applications: 0 Running, 0 Completed

Drivers: 0 Running, 0 Completed

Status: ALIVE

Workers (1)

Worker Id	Address	State	Cores	Memory	Resources
worker-20220712204409-10.128.0.2-37495	10.128.0.2:37495	ALIVE	2 (0 Used)	2.8 GiB (0.0 B Used)	

Running Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
----------------	------	-------	---------------------	------------------------	----------------	------	-------	----------

Completed Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
----------------	------	-------	---------------------	------------------------	----------------	------	-------	----------

MapReduce

I have created two classes Main and WordCounterEngine.

Main.java

This class is used for calling methods of the WordCounterEngine and performing basic operations. It holds the object of WordCounterEngine class.

I have defined two variables maxWordCount and minWordCount and assigned -1 to both variables. I have used Map for finding the frequency of the word.

At the end, I have displayed frequency of each word alongwith word with highest and lowest frequency.

WordCounterEngine.java

This class is responsible for counting the frequency of searched keywords in the news files. The search keywords are Canada, University, Dalhousie University, Halifax, Canada Education, Moncton, Toronto, Oil, and Inflation.

Methods in WordCounterEngine:

- **initWordCounterMap()**

This function creates hashmap of the keywords to be searched as a string and their frequency count as a value.

- **readAllFileNames()**

This method iterates through all files and then it reads contents of each file and performs map and reduce operations.

- **map()**

This method reads the articles of news present in each file by using the Matcher and Pattern classes. It compiles the regular expressions for extracting title and content. The matched substrings for the title and content are appended to a StringBuilder object whose string value is returned.

- **reduce()**

This method is invoked for every string returned by the map() method. It will search for the keyword in the string until the last index is reached and increase the word count of the relevant keyword in the wordCounterMap hashmap.

Algorithm for WordCounter

- 1) Start
- 2) Initialize variables wordCounterMap and initialWordCount.
- 3) Initialize File array allNewsFiles to store the name of all the news articles files.
- 4) If wordCounterMap is empty or if allNewsFiles is null or empty, then go to step 16
- 5) Read the contents of the file.
- 6) Initialize titleMatcher and contentMatcher to match the pattern compiling a regex.
- 7) Initialize mappedStringBuilder to an empty StringBuilder object.
- 8) Append all the matches found by titleMatcher and contentMatcher to mappedStringBuilder.
- 9) Iterate through all the keywords.
- 10) Initialize the variable lastEncounterIndex to 0 for the keyword to be searched.
- 11) Iterate till lastEncounterIndex is not equal to -1.
- 12) Find the index of the occurrence of the current keyword to be searched from the lastEncounterIndex and assign it to lastEncounterIndex.
- 13) If the lastEncounterIndex is not equal to -1, then go to step 14.
- 14) Increment the count of the current keyword.
- 15) Continue till all files are read.
- 16) Stop

Steps to submit job to spark cluster:

- Using mvn install command in the IDE, I created the jar file.
- Then, I have uploaded the code for mapreduce, jar file and text files to git repository and cloned that git repository to the cloud instance.
- After that, I submitted my jar file to the spark master node to run using the spark-submit command [13].

```
spark-submit --master spark://34.135.163.120:7077 --deploy-mode cluster --class
org.example.Main target/A3P3-1.0-SNAPSHOT.jar
```

- After executing the above command, the driver will be in a finished state, means that the jar file successfully run on the Spark Cluster.

Output screenshot on spark's worker node's log page.

Showing 102400 Bytes: 533820 - 636220 of 636220

```
peak summer consumption, after a U.S. rate hike sparked fears of slower economic growth and less fuel demand.", "source": {"name": "Reuters", "id": "reuters"}, "title": "Oil rebounds after steep drop, underpinned by tight supplies - Reuters", "url": "https://www.reuters.com/markets/europe/oil-rebounds-after-steep-drop-underpinned-by-tight-supplies-2022-06-16/", "content": "SINGAPORE, June 16 (Reuters) - Oil prices recovered on Thursday from a steep drop in the previous session, supported by tight oil supply and peak summer consumption, after a U.S. rate hike sparked fears of slower economic growth and less fuel demand.", "publishedAt": "2022-06-24T15:26:00Z", "author": null, "urlToImage": "https://www.reuters.com/resizer/tHju020xPHKdIn8P7Qkx27YPZoV=1200x628/smart/filters:quality(80)/cloudfront-us-east-2-images.amazonaws.com/reuters/UEX8BMYDUJ3HLBEFPMWKTFFVWQ.jpg", "description": "Nigeria's oil minister said on Friday that after meeting with oil companies this week he expects to see some improvement in security in the sector, enabling Africa's top producer meet its OPEC production quota by the end of August.", "source": {"name": "Reuters", "id": "reuters"}, "title": "Nigeria expects improved security to boost oil production - minister - Reuters.com", "url": "https://www.reuters.com/world/africa/nigeria-expects-improved-security-boost-oil-production-minister-2022-06-24/", "content": "ABUJA, June 24 (Reuters) - Nigeria's oil minister said on Friday that after meeting with oil companies this week he expects to see some improvement in security in the sector, enabling Africa's top producer meet its OPEC production quota by the end of August."}]
```

Word	Frequency
Canada	288
Oil	27
Inflation	5
University	54
Dalhousie University	4
Moncton	50
Canada Education	0
Halifax	40
Toronto	119

The word having the highest frequency is "Canada", which is "288".
The word having the lowest frequency is "Canada Education", which is "0".
22/07/11 20:29:33 INFO ShutdownHookManager: Shutdown hook called
22/07/11 20:29:33 INFO ShutdownHookManager: Deleting directory /tmp/spark-db6358c1-a1a0-4f46-a693-583aebdc3113

Load New

Task – 2 : Neo4j Database

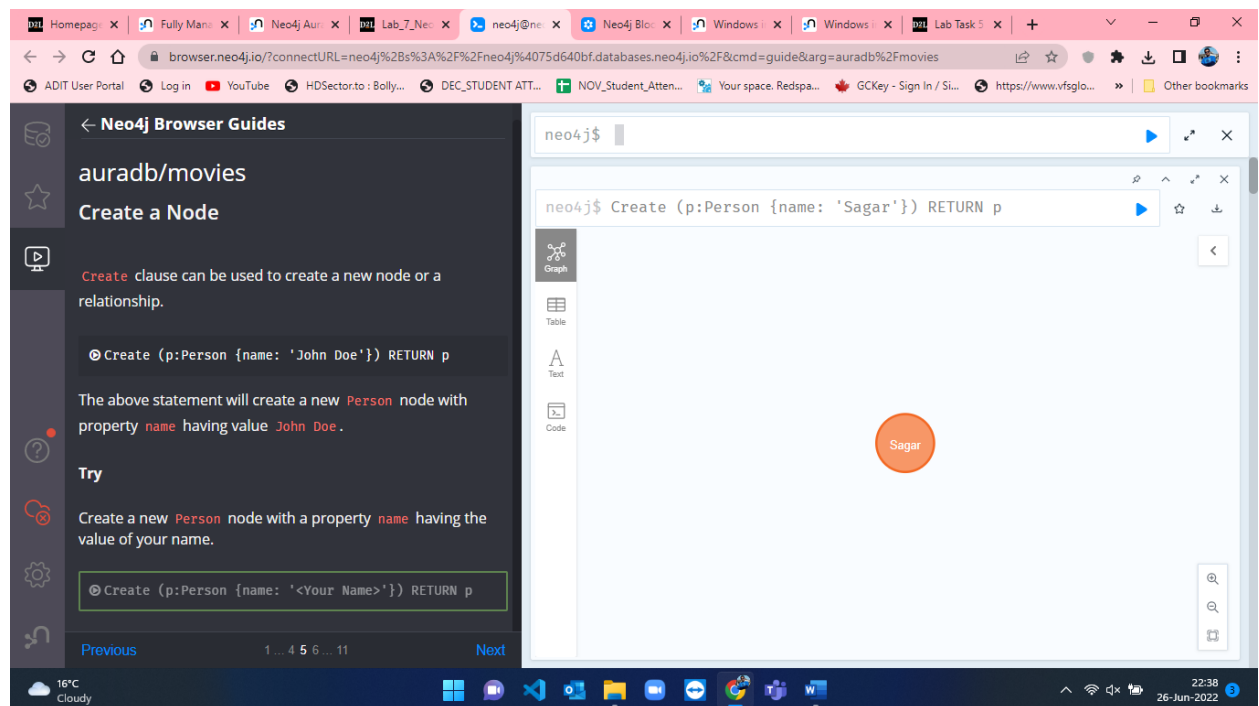
Neo4j[14] is a graph database which has nodes edges and properties. It is more suitable for certain big data and analytics applications. A graph database is used to represent relationships. It uses Cypher Query Language (CQL) [15][16].

Queries:

1) Creating a node

This query creates the Person node of name sagar

```
CREATE (p:Person {name: 'Sagar'}) RETURN p
```



2) Displaying nodes

This cypher query represents all the person nodes with limit of 10 results.

```
MATCH (p:Person) RETURN p limit 10
```

3) Finding nodes by using MATCH

This query will get all the movies that were released between year 2000 and 2020.

MATCH (m:Movie) where m.released > 2000 and m.released < 2020 RETURN m

← Neo4j Browser Guides

auradb/movies

Finding Nodes with Match and Where Clause

Match clause is used to find nodes that match a particular pattern. This is the primary way of getting data from a Neo4j database.

In most cases, a **Match** is used along with certain conditions to narrow down the result.

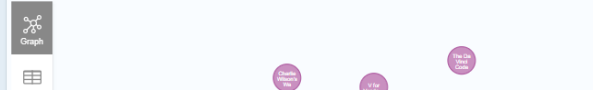
```
⌘ Match (p:Person {name: 'Tom Hanks'}) RETURN p
```

This is one way of doing it. Although you can only do basic string match based filtering this way (without using **WHERE** clause).

Previous1 ... 5 6 7 ... 11Next

neo4j\$

neo4j\$ MATCH (m:Movie) where m.released > 2000 and m.release...



4) Relationships

This cypher query represents all the Person nodes who directed a movie that was released after 2000. I have limited the number of result to 10.

MATCH (p:Person)-[a:ACTED_IN]-(m:Movie) where m.released > 2000 RETURN p,a,m limit 10

The screenshot shows the Neo4j browser interface. On the left, there is a sidebar with navigation options: Home, Star, Video, and a search icon. Below these are some tips and a hint. The main area displays a Cypher query in the editor:

```
neo4j$ MATCH (p:Person)-[a:ACTED_IN]-(m:Movie) where m.released > 2000 RETURN p,a,m limit 10
```

Below the query editor, there is a graph visualization showing several nodes connected by relationships. The nodes are labeled with names like 'The Matrix', 'Keanu Reeves', 'Carrie-Anne', 'Laurence Fishburne', 'Hugo Weaving', 'Diane Kruger', and 'The Matrix Reloaded'. The relationships are labeled 'ACTED_IN'.

At the bottom of the interface, there is a status bar showing the temperature (16°C Cloudy) and the date (26-Jun-2022).

Summary:

In this task of learning about Neo4j database, I explored through the graph databases which is more useful in big data and analytics applications when there is huge amount of data. It has many advantages, and it is more convenient than RDBMS.

I have walked through the basic tutorial of the Neo4j database and also learnt many queries and structures of Cypher Query Language.

I have learnt about creating node, adding properties, building relationships, deleting nodes, deleting relationships, finding nodes and many more.

I learnt that this database represents the nodes and relationships among them in the form of graph which is faster, efficient while searching.

Neo4j performs traversing related operations (such as spanning tree, DFS, BFS, etc.) very smoothly and fastly compared to other databases.

In future, if I will be working on machine learning related fields where I have to work related to recommendations or suggestions at that time I'll be working on graph database as it is much faster and efficient compared to RDBMS.

References :

- [1] "News API – search news and blog articles on the web," *News API*. [Online]. Available: <https://newsapi.org/>. [Accessed: 08-Jul-2022].
- [2] JSON-Java, *Mvnrepository.com*. [Online]. Available: <https://mvnrepository.com/artifact/org.json/json/20220320>. [Accessed: 08-Jul-2022].
- [3] "Atlas," *MongoDB*. [Online]. Available: <https://www.mongodb.com/atlas/database>. [Accessed: 08-Jul-2022].
- [4] "What is the regex to extract all the emojis from a string?," *Stack Overflow*. [Online]. Available: <https://stackoverflow.com/questions/24840667/what-is-the-regex-to-extract-all-the-emojis-from-a-string>. [Accessed: 08-Jul-2022].
- [5] "Regex pattern including all special characters," *Stack Overflow*. [Online]. Available: <https://stackoverflow.com/questions/18057962/regex-pattern-including-all-special-characters/18058074>. [Accessed: 08-Jul-2022].
- [6] "Regular expression to remove HTML tags from a string," *Stack Overflow*. [Online]. Available: <https://stackoverflow.com/questions/11229831/regular-expression-to-remove-html-tags-from-a-string>. [Accessed: 08-Jul-2022].
- [7] "Regular expression to find URLs within a string," *Stack Overflow*. [Online]. Available: <https://stackoverflow.com/questions/6038061/regular-expression-to-find-urls-within-a-string>. [Accessed: 08-Jul-2022].
- [8] "OMDb API - the open movie database," *Omdbapi.com*. [Online]. Available: <https://www.omdbapi.com/>. [Accessed: 08-Jul-2022].
- [9] "SGML parser in java?," *Stack Overflow*. [Online]. Available: <https://stackoverflow.com/questions/4867894/sgml-parser-in-java>. [Accessed: 08-Jul-2022].
- [10] "Cloud computing services," *Google Cloud*. [Online]. Available: <https://cloud.google.com/>. [Accessed: 08-Jul-2022].
- [11] D. Tucakov, "How to install Spark on Ubuntu," *Knowledge Base by phoenixNAP*, 13-Apr-2020. [Online]. Available: <https://phoenixnap.com/kb/install-spark-on-ubuntu>. [Accessed: 08-Jul-2022].

[12] "Downloads," *Apache.org*. [Online]. Available: <https://spark.apache.org/downloads.html>. [Accessed: 10-Jul-2022].

[13] "Submitting Applications - Spark 3.3.0 Documentation," *Apache.org*. [Online]. Available: <https://spark.apache.org/docs/latest/submitting-applications.html>. [Accessed: 10-Jul-2022].

[14] "Graph data platform," *Neo4j Graph Data Platform*, 16-May-2020. [Online]. Available: <https://neo4j.com/>. [Accessed: 11-Jul-2022].

[15] "Neo4j tutorial," *Tutorialspoint.com*. [Online]. Available: <https://www.tutorialspoint.com/neo4j/index.htm>. [Accessed: 11-Jul-2022].

[16] "Neo4j tutorial," *www.javatpoint.com*. [Online]. Available: <https://www.javatpoint.com/neo4j-tutorial>. [Accessed: 11-Jul-2022].