

# Hadoop MapReduce program to count occurrences of a word and give Top N words.

## Dataset Overview

Please download one dataset of your choice from Amazon review data using the following URL:

- <http://jmcauley.ucsd.edu/data/amazon/Links to an external site.>

[Links to an external site.](#)

## Data source:

I choose All beauty product category from link below

[https://cseweb.ucsd.edu/~jmcauley/datasets/amazon\\_v2/#complete-data](https://cseweb.ucsd.edu/~jmcauley/datasets/amazon_v2/#complete-data)

## Data description:

It is the reviews provided by different customers for different beauty products on amazon platform. I am using 2018 version of this dataset. It has 371345 records i.e reviews. The reviews are stored in json file and the file size is 167 MB.

It has following attributes:

- overall : overall rating of product. Range is 1 to 5
- vote: number of votes
- verified: verified or not(values-True/False)
- reviewTime: time of the review
- reviewerID: reviewer ID
- asin: product ID
- reviewerName : reviewer name
- reviewText: review provided by the reviewer.
- summary: summary of review
- unixReviewTime: time of the review in unix time

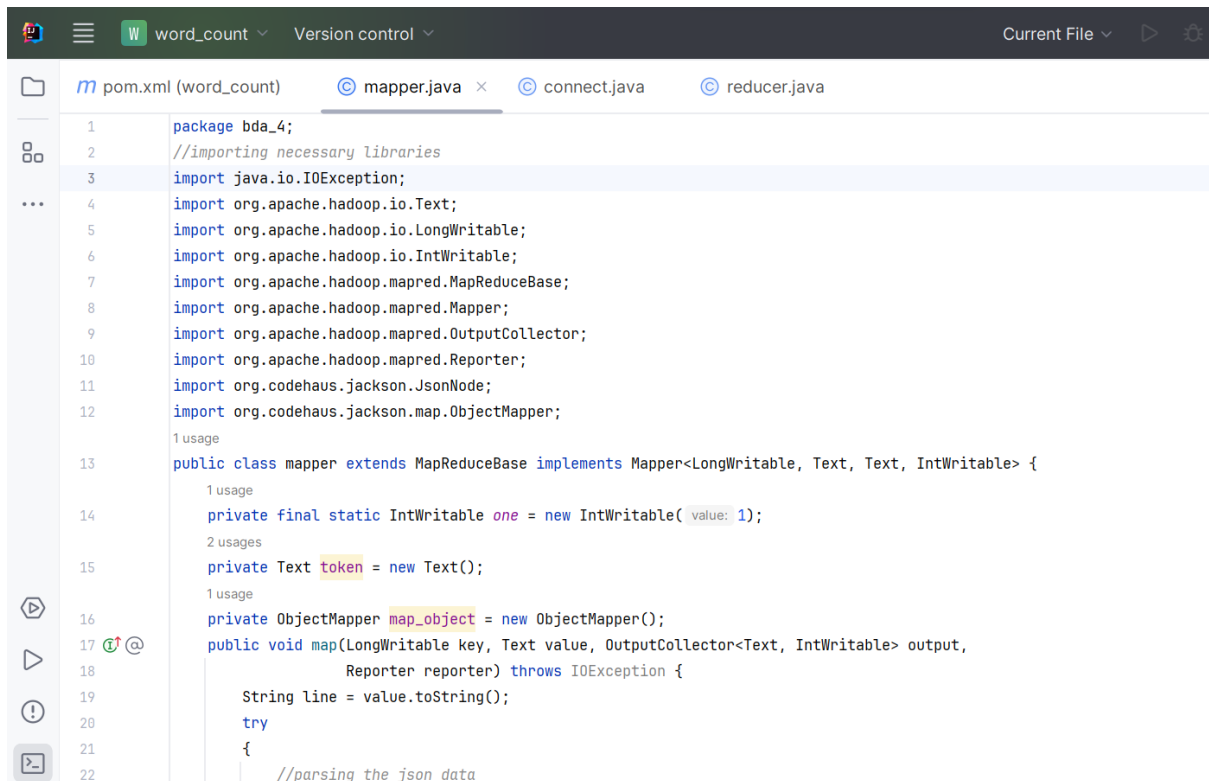
## Task 1: Word Count

1. Implementing a MapReduce program to count the occurrences of each word in the dataset.
2. Ensuring that the program removes punctuation and converts all words to lowercase.
3. Output the word count in the format: <word> <count>.
4. And a brief explanation of the MapReduce workflow for the Word Count program.

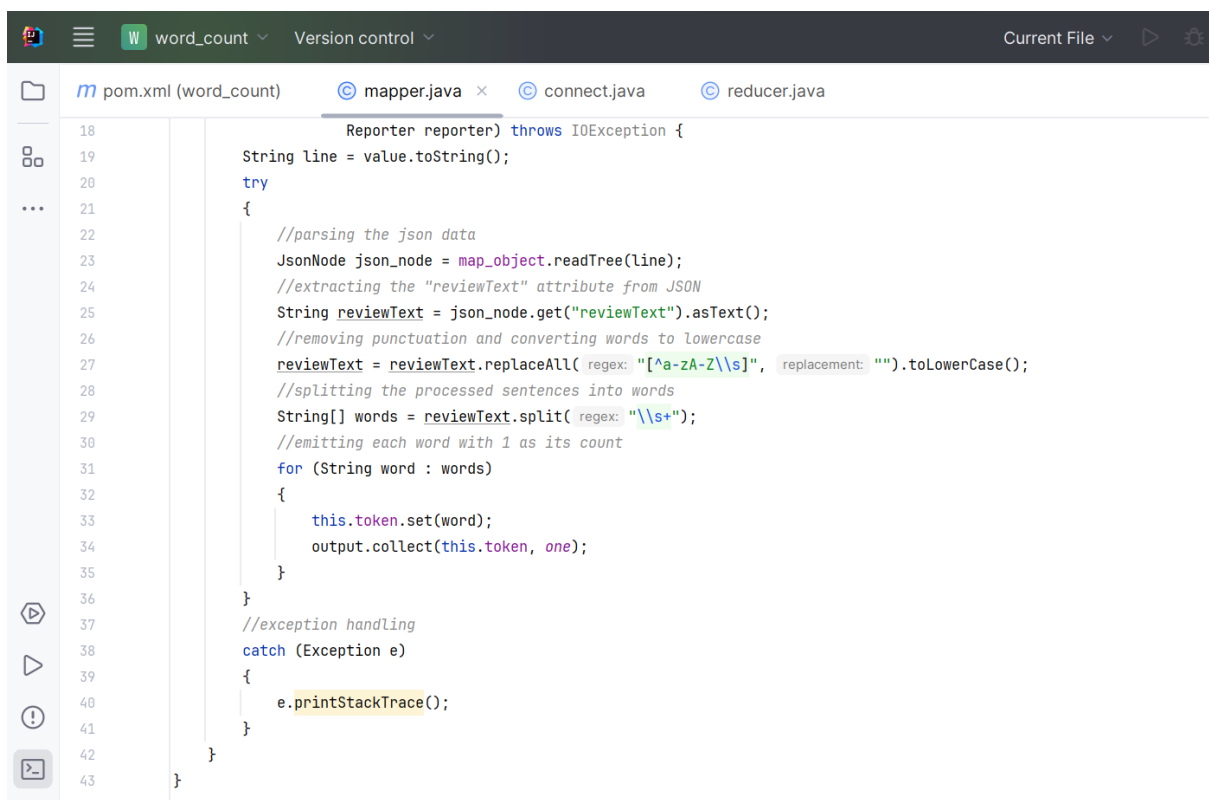
I used IntelliJ IDEA IDE to achieve task 1 and task 2 as we can add dependencies directly using Maven.

## Implementation

### Mapper Class

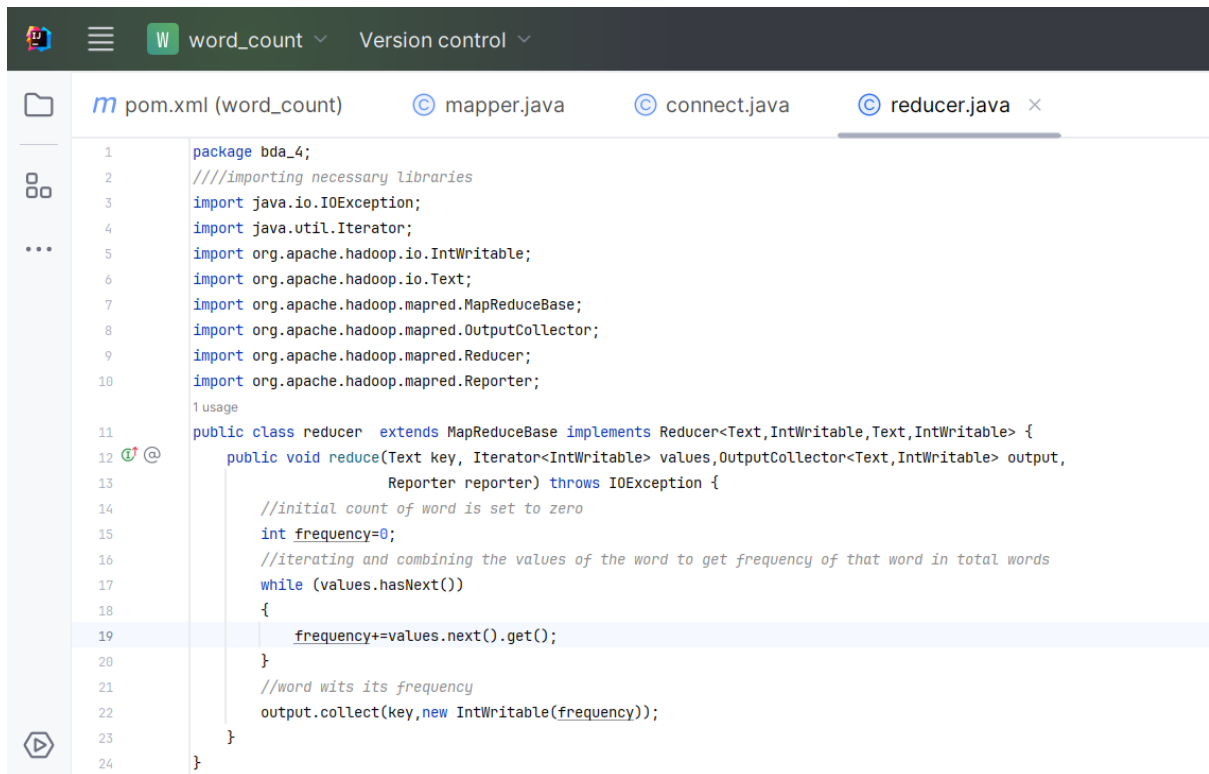


```
1 package bda_4;
2 //importing necessary libraries
3 import java.io.IOException;
4 import org.apache.hadoop.io.Text;
5 import org.apache.hadoop.io.LongWritable;
6 import org.apache.hadoop.io.IntWritable;
7 import org.apache.hadoop.mapred.MapReduceBase;
8 import org.apache.hadoop.mapred.Mapper;
9 import org.apache.hadoop.mapred.OutputCollector;
10 import org.apache.hadoop.mapred.Reporter;
11 import org.codehaus.jackson.JsonNode;
12 import org.codehaus.jackson.map.ObjectMapper;
13
14 1 usage
15 public class mapper extends MapReduceBase implements Mapper<LongWritable, Text, Text, IntWritable> {
16     1 usage
17     private final static IntWritable one = new IntWritable( value: 1);
18     2 usages
19     private Text token = new Text();
20     1 usage
21     private ObjectMapper map_object = new ObjectMapper();
22     public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable> output,
23         Reporter reporter) throws IOException {
24         String line = value.toString();
25         try
26         {
27             //parsing the json data
```



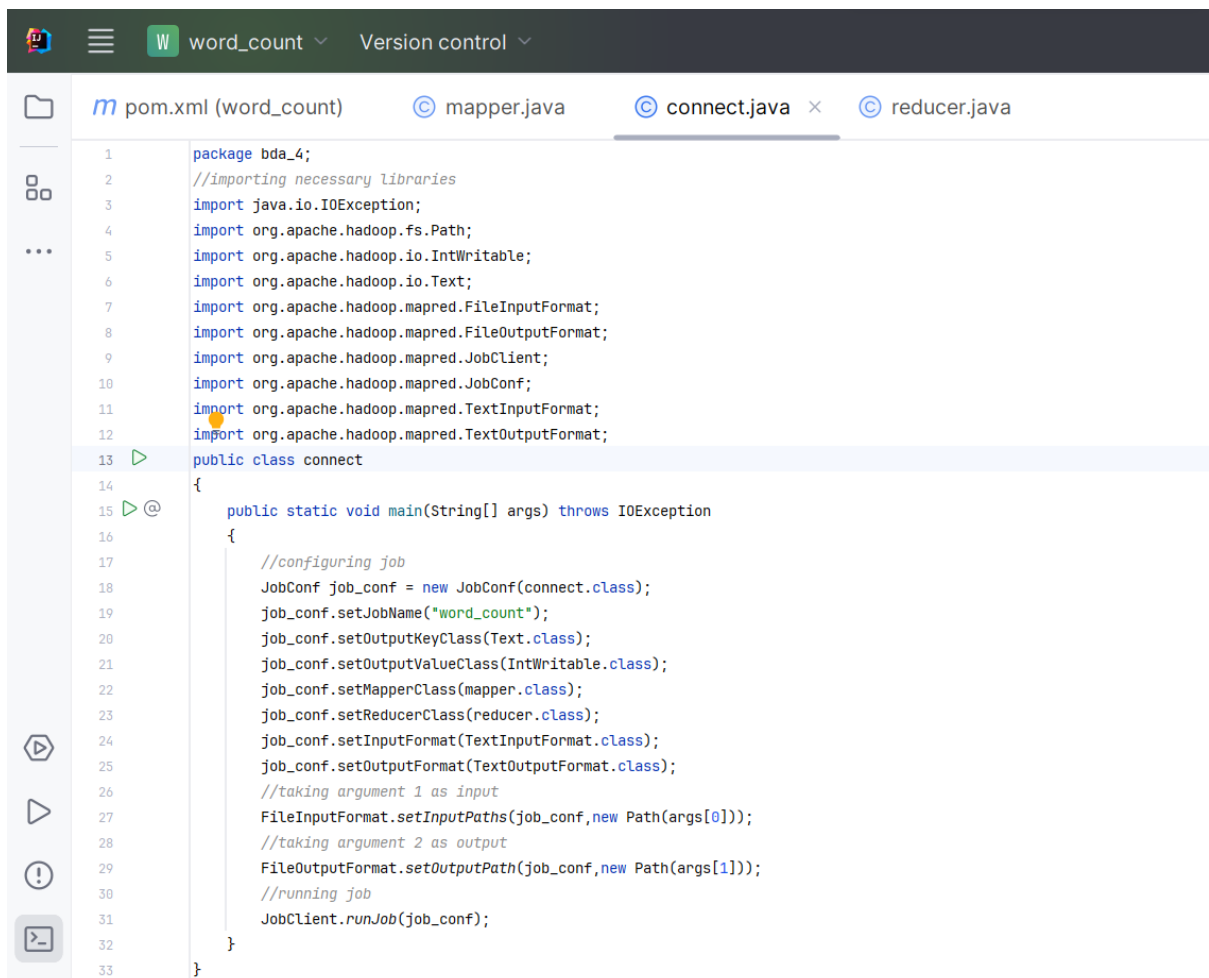
```
28         Reporter reporter) throws IOException {
29         String line = value.toString();
30         try
31         {
32             //parsing the json data
33             JsonNode json_node = map_object.readTree(line);
34             //extracting the "reviewText" attribute from JSON
35             String reviewText = json_node.get("reviewText").asText();
36             //removing punctuation and converting words to lowercase
37             reviewText = reviewText.replaceAll( regex: "[^a-zA-Z\\s]", replacement: "").toLowerCase();
38             //splitting the processed sentences into words
39             String[] words = reviewText.split( regex: "\\s+");
40             //emitting each word with 1 as its count
41             for (String word : words)
42             {
43                 this.token.set(word);
44                 output.collect(this.token, one);
45             }
46         }
47         //exception handling
48         catch (Exception e)
49         {
50             e.printStackTrace();
51         }
52     }
53 }
```

## Reducer Class



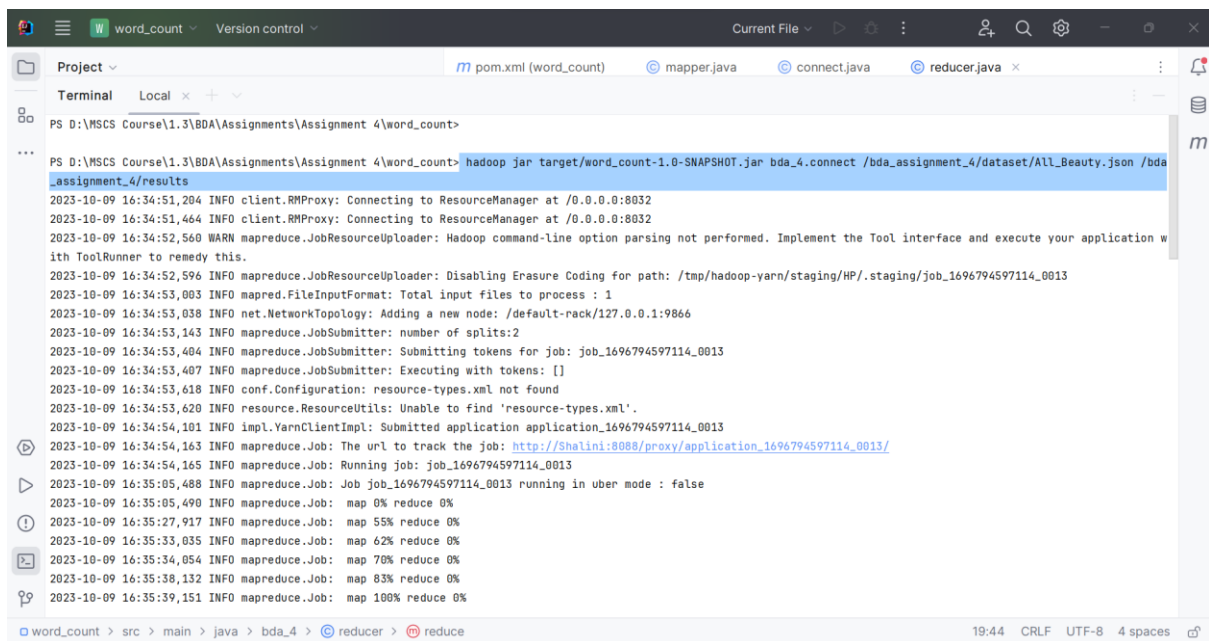
```
1 package bda_4;
2 ///importing necessary libraries
3 import java.io.IOException;
4 import java.util.Iterator;
5 import org.apache.hadoop.io.IntWritable;
6 import org.apache.hadoop.io.Text;
7 import org.apache.hadoop.mapred.MapReduceBase;
8 import org.apache.hadoop.mapred.OutputCollector;
9 import org.apache.hadoop.mapred.Reducer;
10 import org.apache.hadoop.mapred.Reporter;
11
12 1 usage
13 public class reducer extends MapReduceBase implements Reducer<Text,IntWritable,Text,IntWritable> {
14     public void reduce(Text key, Iterator<IntWritable> values,OutputCollector<Text,IntWritable> output,
15         Reporter reporter) throws IOException {
16         //initial count of word is set to zero
17         int frequency=0;
18         //iterating and combining the values of the word to get frequency of that word in total words
19         while (values.hasNext())
20         {
21             frequency+=values.next().get();
22         }
23         //word wits its frequency
24         output.collect(key,new IntWritable(frequency));
25     }
26 }
```

## Connect Class



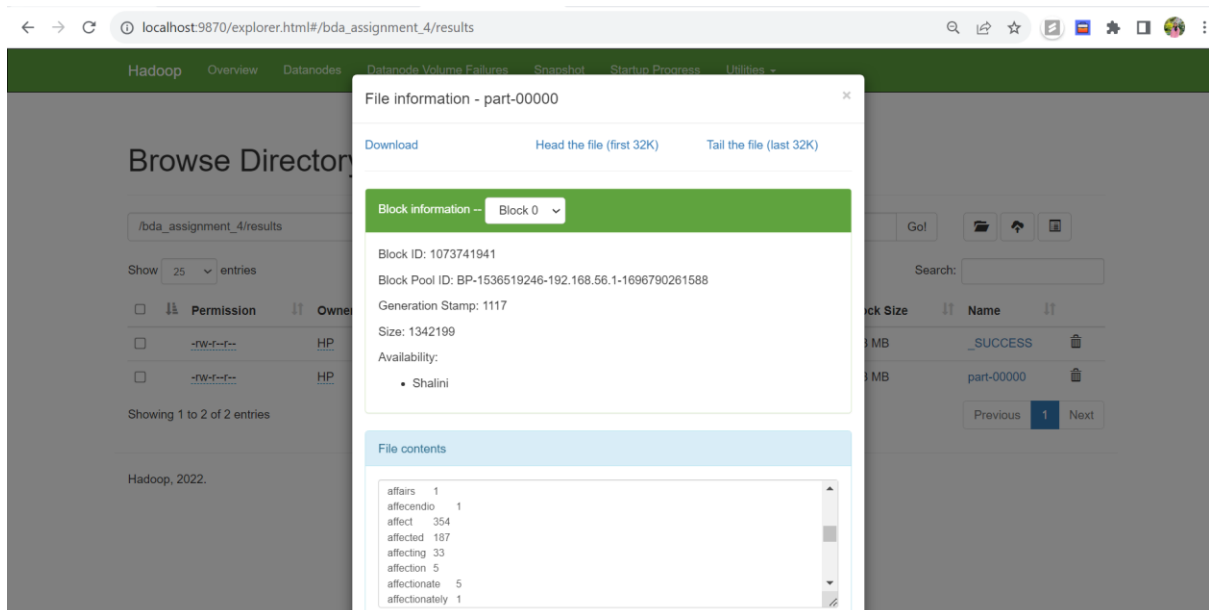
```
1 package bda_4;
2 ///importing necessary libraries
3 import java.io.IOException;
4 import org.apache.hadoop.fs.Path;
5 import org.apache.hadoop.io.IntWritable;
6 import org.apache.hadoop.io.Text;
7 import org.apache.hadoop.mapred.FileInputFormat;
8 import org.apache.hadoop.mapred.FileOutputFormat;
9 import org.apache.hadoop.mapred.JobClient;
10 import org.apache.hadoop.mapred.JobConf;
11 import org.apache.hadoop.mapred.TextInputFormat;
12 import org.apache.hadoop.mapred.TextOutputFormat;
13
14 public class connect
15 {
16     public static void main(String[] args) throws IOException
17     {
18         //configuring job
19         JobConf job_conf = new JobConf(connect.class);
20         job_conf.setJobName("word_count");
21         job_conf.setOutputKeyClass(Text.class);
22         job_conf.setOutputValueClass(IntWritable.class);
23         job_conf.setMapperClass mapper.class);
24         job_conf.setReducerClass(reducer.class);
25         job_conf.setInputFormat(TextInputFormat.class);
26         job_conf.setOutputFormat(TextOutputFormat.class);
27         //taking argument 1 as input
28         FileInputFormat.setInputPaths(job_conf,new Path(args[0]));
29         //taking argument 2 as output
30         FileOutputFormat.setOutputPath(job_conf,new Path(args[1]));
31         //running job
32         JobClient.runJob(job_conf);
33     }
34 }
```

## Output:



```
PS D:\MSCS Course\1.3\BDA\Assignments\Assignment 4\word_count>
...
PS D:\MSCS Course\1.3\BDA\Assignments\Assignment 4\word_count> hadoop jar target/word_count-1.0-SNAPSHOT.jar bda_4.connect /bda_assignment_4/dataset/All_Beauty.json /bda_assignment_4/results
2023-10-09 16:34:51,204 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
2023-10-09 16:34:51,464 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
2023-10-09 16:34:52,560 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
2023-10-09 16:34:52,596 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/HP/.staging/job_1696794597114_0013
2023-10-09 16:34:53,003 INFO mapred.FileInputFormat: Total input files to process : 1
2023-10-09 16:34:53,038 INFO net.NetworkTopology: Adding a new node: /default-rack/127.0.0.1:9866
2023-10-09 16:34:53,143 INFO mapreduce.JobSubmitter: number of splits:2
2023-10-09 16:34:53,404 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1696794597114_0013
2023-10-09 16:34:53,407 INFO mapreduce.JobSubmitter: Executing with tokens: []
2023-10-09 16:34:53,618 INFO conf.Configuration: resource-types.xml not found
2023-10-09 16:34:53,620 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2023-10-09 16:34:54,101 INFO impl.YarnClientImpl: Submitted application application_1696794597114_0013
2023-10-09 16:34:54,163 INFO mapreduce.Job: The url to track the job: http://Shalini:8088/proxy/application_1696794597114_0013/
2023-10-09 16:34:54,165 INFO mapreduce.Job: Running job: job_1696794597114_0013
2023-10-09 16:35:05,488 INFO mapreduce.Job: Job job_1696794597114_0013 running in uber mode : false
2023-10-09 16:35:05,490 INFO mapreduce.Job: map 0% reduce 0%
2023-10-09 16:35:27,917 INFO mapreduce.Job: map 55% reduce 0%
2023-10-09 16:35:33,035 INFO mapreduce.Job: map 62% reduce 0%
2023-10-09 16:35:34,054 INFO mapreduce.Job: map 70% reduce 0%
2023-10-09 16:35:38,132 INFO mapreduce.Job: map 83% reduce 0%
2023-10-09 16:35:39,151 INFO mapreduce.Job: map 100% reduce 0%
```

## Seeing results in UI



Seeing results through cmd: I returned only first 50 due to huge size

```
C:\hadoop-3.2.4\sbin>hadoop fs -cat /bda_assignment_4/results/part-00000 | more +50
aathank 1
aawesome 1
aawkward 1
aazon 2
ab 39
aback 14
abad 1
abag 1
abaility 1
abaj 1
abandon 8
abandoned 18
abandoning 3
abanoned 1
abate 6
abated 4
abating 2
abberration 1
abbey 1
abbrasions 1
abbreviated 3
abbreviations 1
abc 16
abcderm 1
abck 1
abcs 1
abcsell 1
abd 21
abdomen 44
abdominal 33
abducted 1
abdul 1
abdulaziz 2
abe 3
abel 1
aber 2
abercornbie 1
abercornbie 9
```

### Program workflow:

Prior to executing program: I have added dependencies in xml file. I have uploaded All\_Beauty.json file to hdfs using commands.

Coming to program's workflow

**Mapping phase**(mapper java class): It takes All\_Beauty.json file as input and extracts reviewText attribute from all records. We then process sentences in reviewText by removing punctuation and convert it lower cases. We then split sentences based on spaces. After splitting we have tokens. Each token is printed with its count as 1.

**Shuffling phase:** Hadoop groups and sort the data obtained from mappers and send it to reducer class.

**Reducer phase**(reducer class file). This program takes sorted tokens as input and combines the sum of all the values which has same key and result in key value pairs where word is the key and corresponding frequency is the value.

**Connect class:** In this class, I defined input to the program and the output it should obtain. It also can be seen as center of all classes. It specifies mapper and reducer class, formats. Basically, it configures the job to be executed by Hadoop.

**Executing program:**

Once all the class files are coded, I generated the jar file for my program and executed the project in terminal using following command:

**Cmd to execute the word\_count algorithm:**

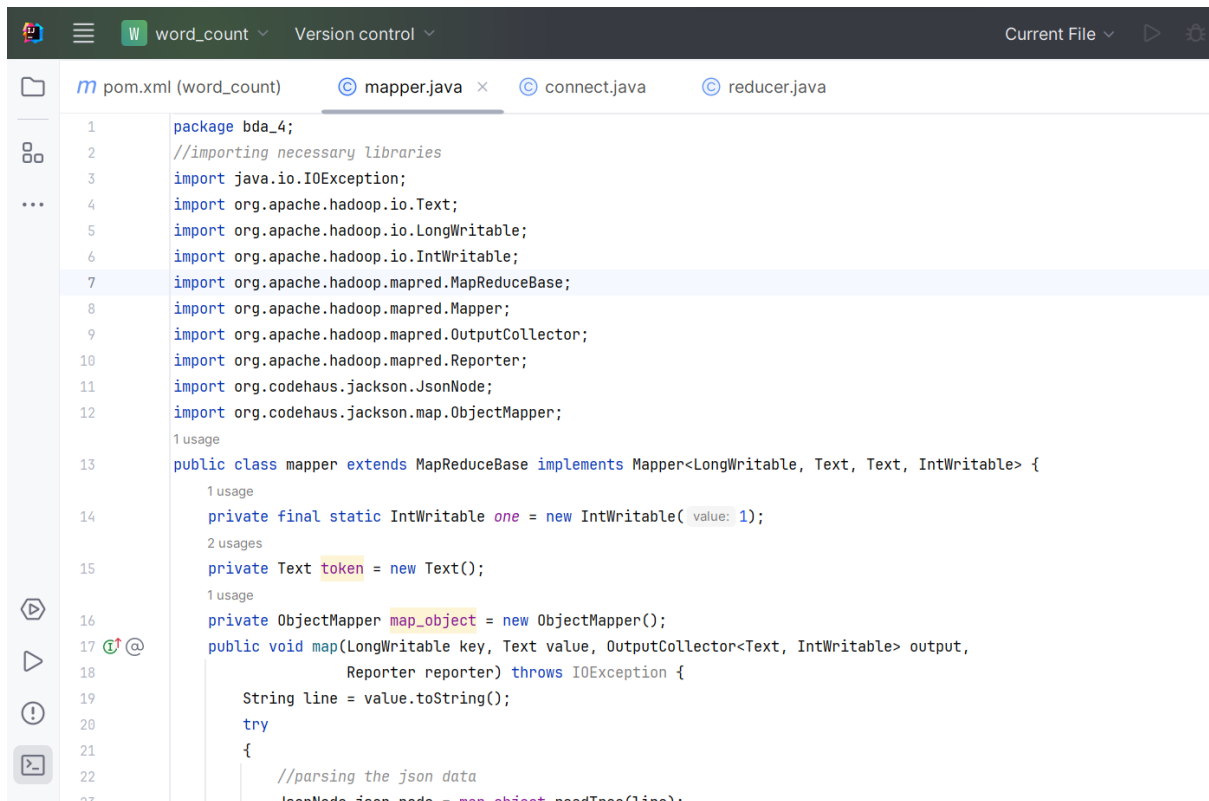
```
hadoop jar target/word_count-1.0-SNAPSHOT.jar bda_4.connect  
/bda_assignment_4/dataset/All_Beauty.json /bda_assignment_4/results
```

## Task 2: Top N Words

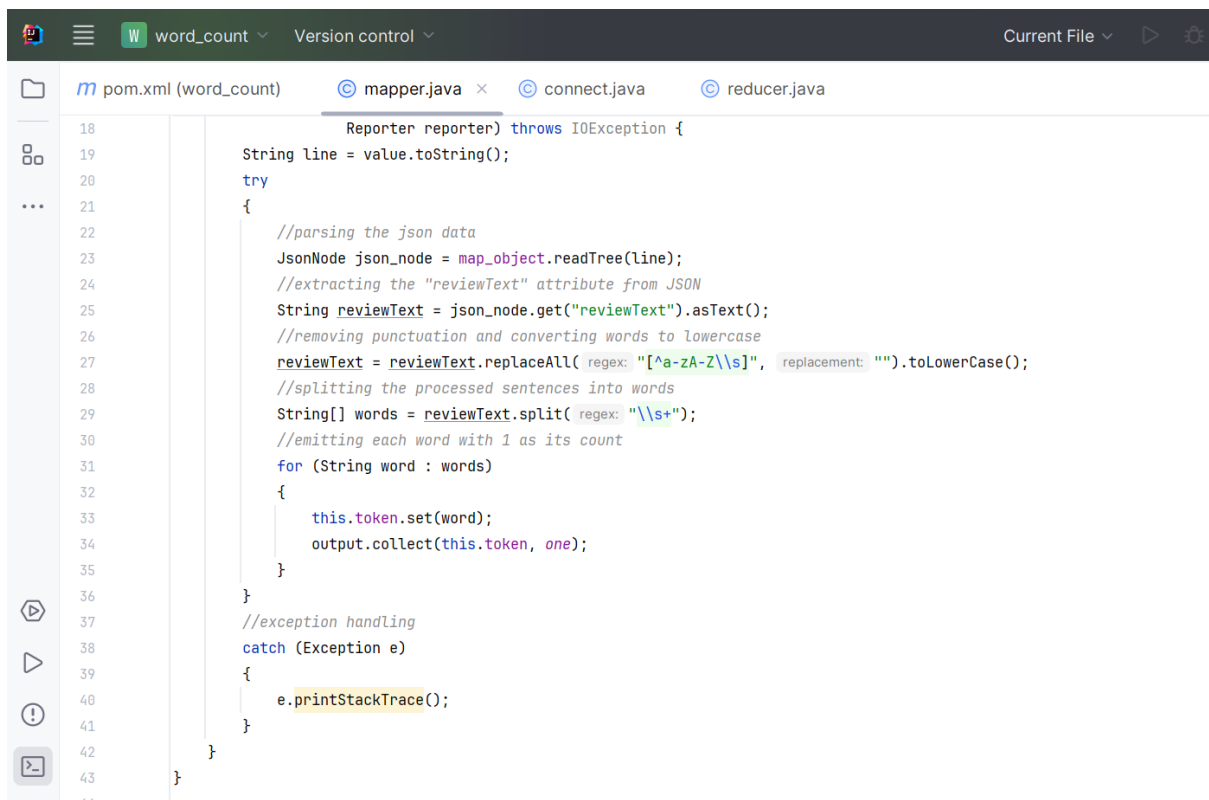
1. Extending the Word Count program to find the top N most frequently occurring words in the dataset.
2. Allowing the user to specify the value of N.
3. Output the top N words and their counts in descending order.
4. Repeat section 1 while using "combiner" in reducer and compare the results with the results in section 1. Does the "combiner" change the results in this case?
5. And a brief explanation of how you modified the Word Count program to accomplish this task.

### Implementation

**Mapper Class:** The mapper class is same as task 1 and not altered.



```
1 package bda_4;
2 //importing necessary libraries
3 import java.io.IOException;
4 import org.apache.hadoop.io.Text;
5 import org.apache.hadoop.io.LongWritable;
6 import org.apache.hadoop.io.IntWritable;
7 import org.apache.hadoop.mapred.MapReduceBase;
8 import org.apache.hadoop.mapred.Mapper;
9 import org.apache.hadoop.mapred.OutputCollector;
10 import org.apache.hadoop.mapred.Reporter;
11 import org.codehaus.jackson.JsonNode;
12 import org.codehaus.jackson.map.ObjectMapper;
13
14 1 usage
15 public class mapper extends MapReduceBase implements Mapper<LongWritable, Text, Text, IntWritable> {
16     1 usage
17     private final static IntWritable one = new IntWritable( value: 1);
18     2 usages
19     private Text token = new Text();
20     1 usage
21     private ObjectMapper map_object = new ObjectMapper();
22     public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable> output,
23         Reporter reporter) throws IOException {
24         String line = value.toString();
25         try
26         {
27             //parsing the json data
28             JsonNode json_node = map_object.readTree(line);
```



```
29         Reporter reporter) throws IOException {
30         String line = value.toString();
31         try
32         {
33             //parsing the json data
34             JsonNode json_node = map_object.readTree(line);
35             //extracting the "reviewText" attribute from JSON
36             String reviewText = json_node.get("reviewText").asText();
37             //removing punctuation and converting words to lowercase
38             reviewText = reviewText.replaceAll( regex: "[^a-zA-Z\\s]", replacement: "").toLowerCase();
39             //splitting the processed sentences into words
40             String[] words = reviewText.split( regex: "\\s+");
41             //emitting each word with 1 as its count
42             for (String word : words)
43             {
44                 this.token.set(word);
45                 output.collect(this.token, one);
46             }
47         }
48         //exception handling
49         catch (Exception e)
50         {
51             e.printStackTrace();
52         }
53     }
54 }
```

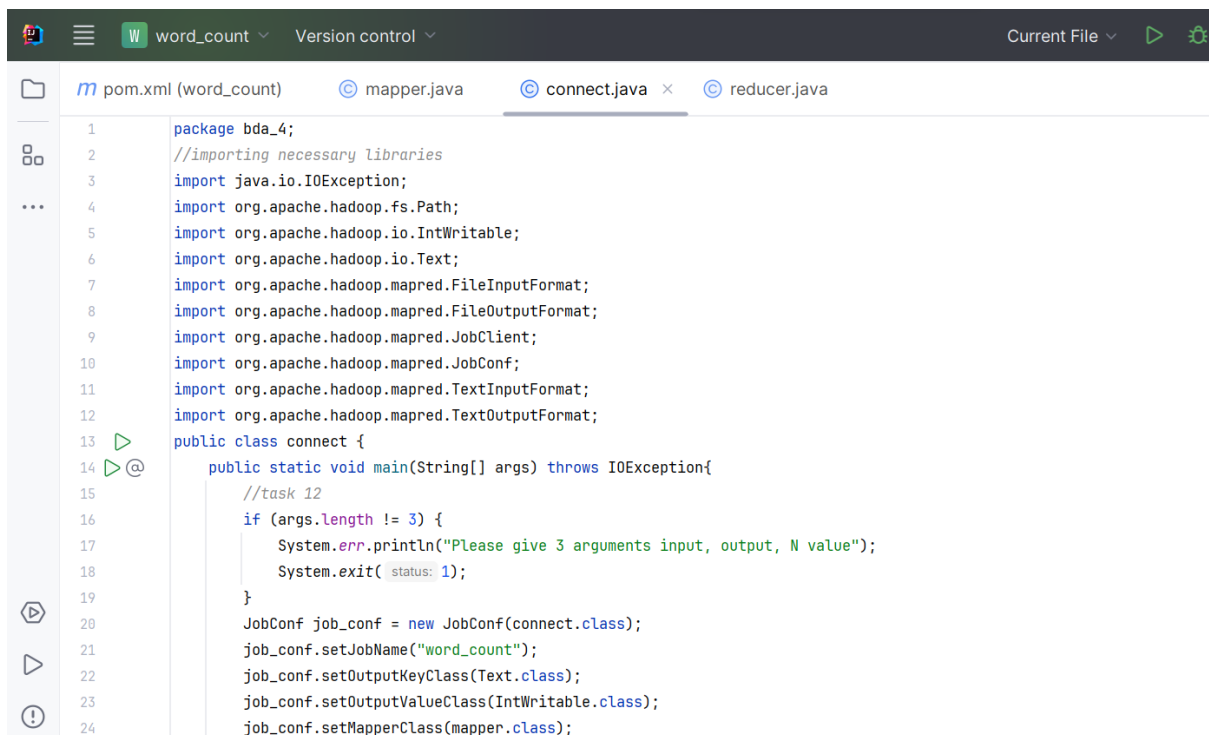
## Reducer Class

```
1 package bda_4;
2 import java.io.IOException;
3 import java.util.Comparator;
4 import java.util.Iterator;
5 import java.util.Map;
6 import java.util.TreeMap;
7 import org.apache.hadoop.io.IntWritable;
8 import org.apache.hadoop.io.Text;
9 import org.apache.hadoop.mapred.MapReduceBase;
10 import org.apache.hadoop.mapred.OutputCollector;
11 import org.apache.hadoop.mapred.Reducer;
12 import org.apache.hadoop.mapred.Reporter;
13 import org.apache.hadoop.mapred.JobConf;
14 public class reducer extends MapReduceBase implements Reducer<Text, IntWritable, Text, IntWritable> {
15     private int N;
16     //TreeMap to store word and its frequency
17     private TreeMap<Integer, Text> top_N_words;
18     private OutputCollector<Text, IntWritable> result;
19     public void configure(JobConf job)
20     {
21         //getting n value from configuration, I gave default value as 3
22         N = job.getInt( name: "top_N", defaultValue: 3);
23         // new TreeMap for the tokens
24         top_N_words = new TreeMap<>(Comparator.reverseOrder());
25     }
26
27     public void reduce(Text key, Iterator<IntWritable> values, OutputCollector<Text, IntWritable> output,
```

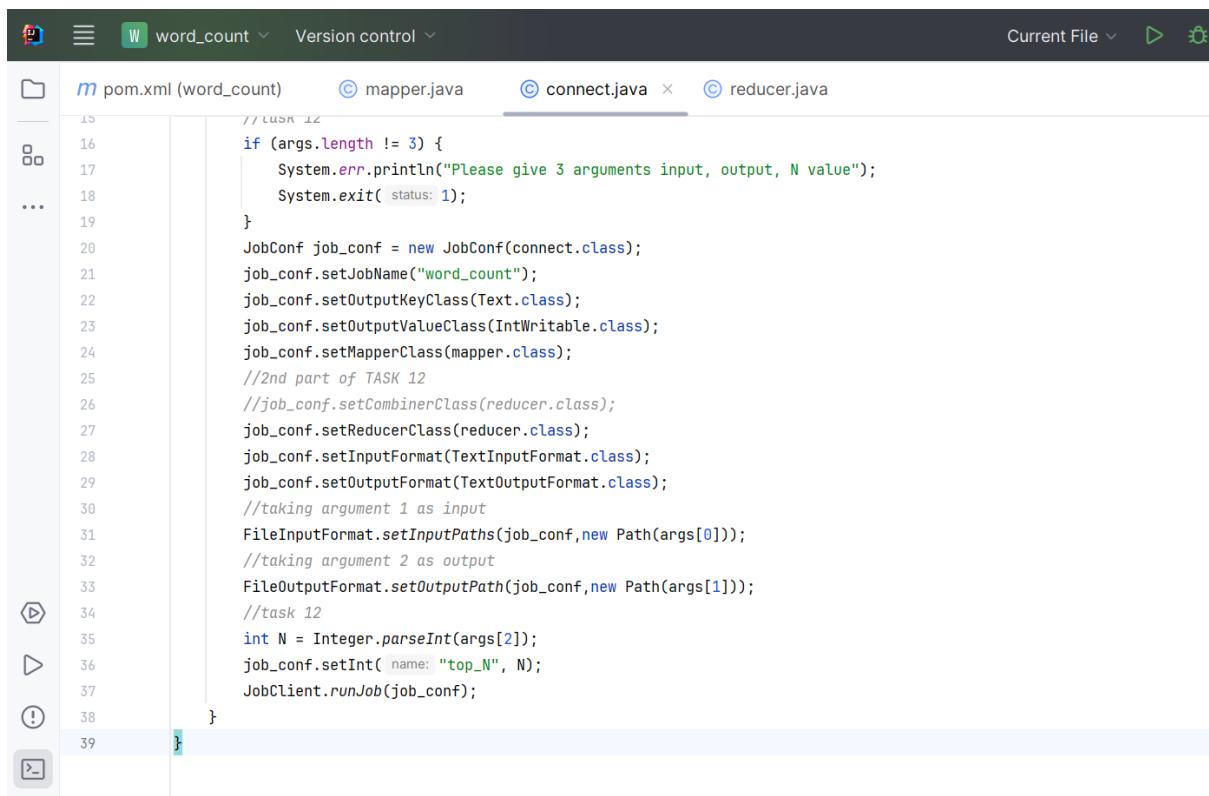
```
21 //getting n value from configuration, I gave default value as 3
22 N = job.getInt( name: "top_N", defaultValue: 3);
23 // new TreeMap for the tokens
24 top_N_words = new TreeMap<>(Comparator.reverseOrder());
25 }
26
27 public void reduce(Text key, Iterator<IntWritable> values, OutputCollector<Text, IntWritable> output,
28     Reporter reporter) throws IOException {
29     int frequency = 0;
30     while (values.hasNext()) {
31         frequency += values.next().get();
32     }
33     //adding word and its frequency to the TreeMap
34     top_N_words.put(frequency, new Text(key.toString()));
35     //keeping only top N words
36     if (top_N_words.size() > N)
37     {
38         top_N_words.pollLastEntry();
39     }
40     //storing the OutputCollector for later use in the close method as I can't use output directly in close method
41     result = output;
42 }
43 public void close() throws IOException
44 {
45     //emitting the top N words in descending order
46     for (Map.Entry<Integer, Text> entry : top_N_words.entrySet())
47     {
48         result.collect(entry.getValue(), new IntWritable(entry.getKey()));
49     }
50 }
51 }
```

## Connect Class



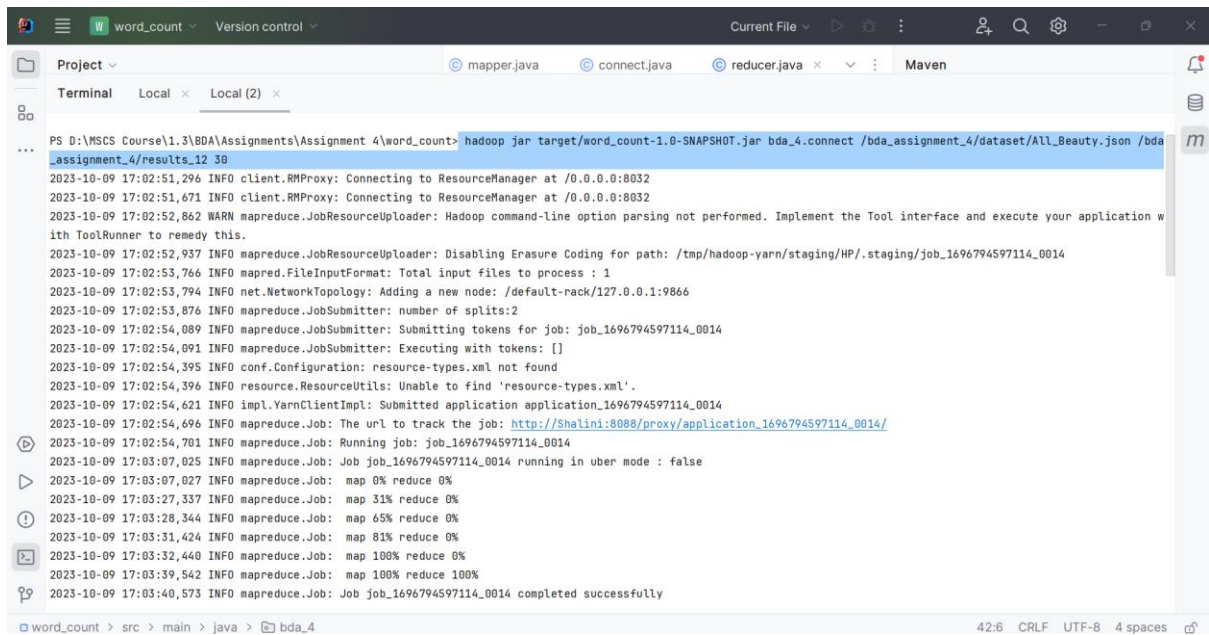


```
1 package bda_4;
2 //importing necessary libraries
3 import java.io.IOException;
4 import org.apache.hadoop.fs.Path;
5 import org.apache.hadoop.io.IntWritable;
6 import org.apache.hadoop.io.Text;
7 import org.apache.hadoop.mapred.FileInputFormat;
8 import org.apache.hadoop.mapred.FileOutputFormat;
9 import org.apache.hadoop.mapred.JobClient;
10 import org.apache.hadoop.mapred.JobConf;
11 import org.apache.hadoop.mapred.TextInputFormat;
12 import org.apache.hadoop.mapred.TextOutputFormat;
13 public class connect {
14     public static void main(String[] args) throws IOException{
15         //task 12
16         if (args.length != 3) {
17             System.err.println("Please give 3 arguments input, output, N value");
18             System.exit( status: 1);
19         }
20         JobConf job_conf = new JobConf(connect.class);
21         job_conf.setJobName("word_count");
22         job_conf.setOutputKeyClass(Text.class);
23         job_conf.setOutputValueClass(IntWritable.class);
24         job_conf.setMapperClass(mapper.class);
```

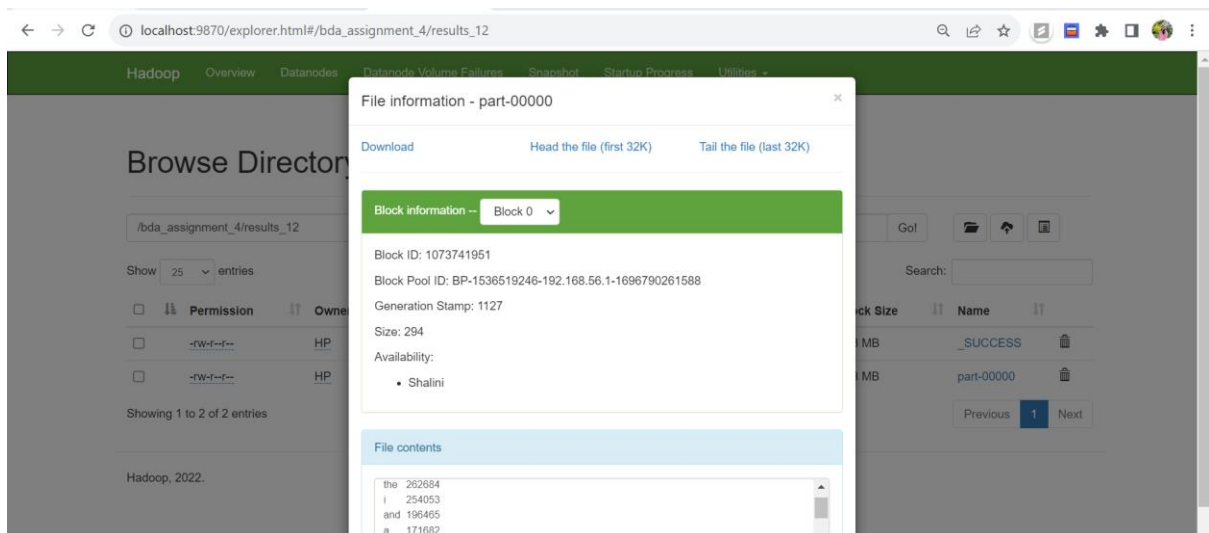


```
25 //2nd part of TASK 12
26 //job_conf.setCombinerClass(reducer.class);
27 job_conf.setReducerClass(reducer.class);
28 job_conf.setInputFormat(TextInputFormat.class);
29 job_conf.setOutputFormat(TextOutputFormat.class);
30 //taking argument 1 as input
31 FileInputFormat.setInputPaths(job_conf,new Path(args[0]));
32 //taking argument 2 as output
33 FileOutputFormat.setOutputPath(job_conf,new Path(args[1]));
34 //task 12
35 int N = Integer.parseInt(args[2]);
36 job_conf.setInt( name: "top_N", N);
37 JobClient.runJob(job_conf);
38 }
39 }
```

Output:



Since we were not asked to do complete text preprocessing of reviewText, results contain words like 'a', 'the' etc



## Modifications on Word Count to achieve Task 2 - Top N Words:

I have modified the reducer and connect java class of word count - task 1 to achieve task 2.

- Connect class: It has N integer to take 3<sup>rd</sup> argument from the user input(input - 1<sup>st</sup> argument, output-2<sup>nd</sup> argument, N – 3<sup>rd</sup> argument)
- Reducer class: It has same functionality but in addition I used a TreeMap with a custom comparator that sorts the entries based on counts in descending order. This allows to efficiently keep track of the top N words and their frequencies. The close method emits these top N words in descending order of counts.

**Cmd to execute the top N words algorithm:** (30 – top 30 words)

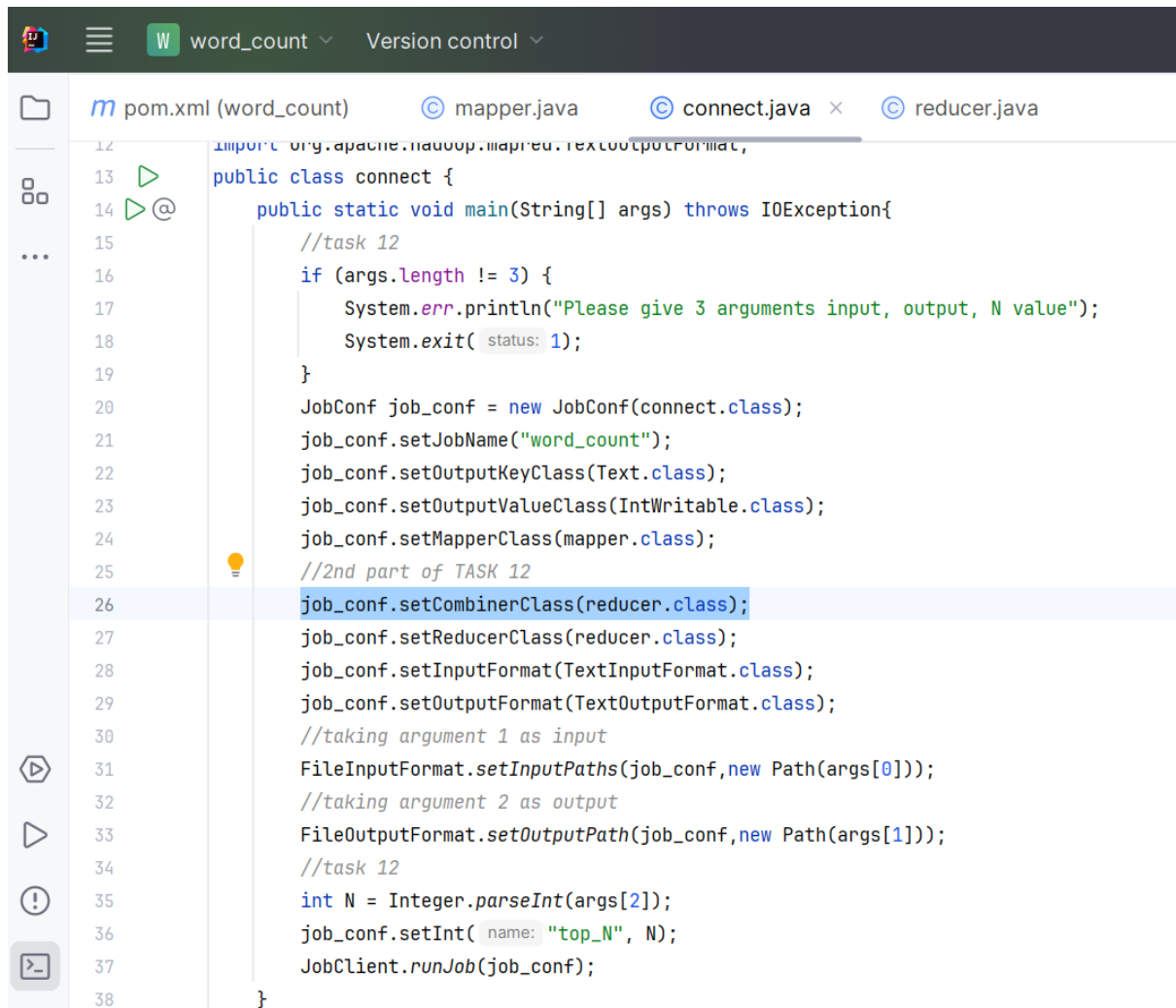
```
hadoop jar target/word_count-1.0-SNAPSHOT.jar bda_4.connect  
/bda_assignment_4/dataset/All_Beauty.json /bda_assignment_4/results_12 30
```

## Using Combiner:

The combiner runs on the output of the mapper class. It performs a local aggregation of word counts. It did not change the results but increases the efficiency of the algorithm.

We just add following line to connect java class file

```
Job_conf.setCombinerClass(reducer.class);
```



```
12 import org.apache.hadoop.mapred.TextOutputFormat;
13 public class connect {
14     public static void main(String[] args) throws IOException{
15         //task 12
16         if (args.length != 3) {
17             System.err.println("Please give 3 arguments input, output, N value");
18             System.exit( status: 1);
19         }
20         JobConf job_conf = new JobConf(connect.class);
21         job_conf.setJobName("word_count");
22         job_conf.setOutputKeyClass(Text.class);
23         job_conf.setOutputValueClass(IntWritable.class);
24         job_conf.setMapperClass(mapper.class);
25         //2nd part of TASK 12
26         job_conf.setCombinerClass(reducer.class);
27         job_conf.setReducerClass(reducer.class);
28         job_conf.setInputFormat(TextInputFormat.class);
29         job_conf.setOutputFormat(TextOutputFormat.class);
30         //taking argument 1 as input
31         FileInputFormat.setInputPaths(job_conf,new Path(args[0]));
32         //taking argument 2 as output
33         FileOutputFormat.setOutputPath(job_conf,new Path(args[1]));
34         //task 12
35         int N = Integer.parseInt(args[2]);
36         job_conf.setInt( name: "top_N", N);
37         JobClient.runJob(job_conf);
38     }
```