

# ML with Big Data

## Assignment – 1: MapReduce and Apache Spark

- Dwivedi Jyoti Rajeshbhai  
(M25CSA010)
- Github Repo link: [https://github.com/Jyoti-Dwivedi-010/ML with Big Data](https://github.com/Jyoti-Dwivedi-010/ML_with_Big_Data)

In this assignment, I have implemented a Big Data pipeline using Apache Hadoop and Apache Spark on the project Gutenberg dataset (consisting 400+ books in .txt format). I explored foundational MapReduce paradigms, TF-IDF procedures and Network Graph Analysis. Majorly, I learnt about the hardware constraints that are possible with real-world big data applications. I encountered similar hardware constraints during execution (1.5 GB available RAM in my WSL2 environment), which caused Out-of-Memory Crashes.

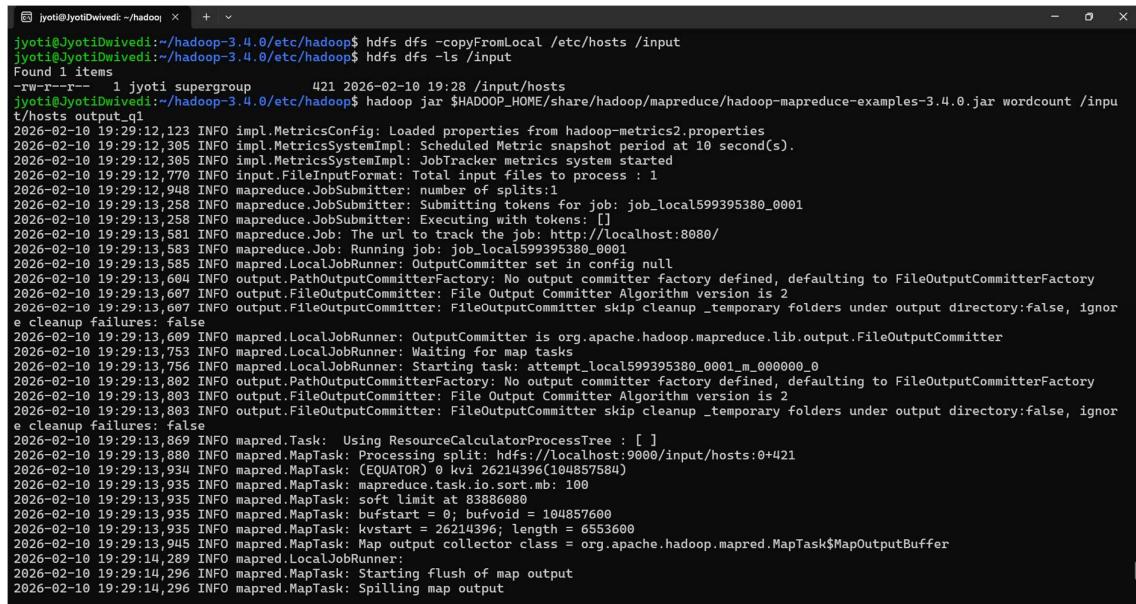
## Apache Hadoop and MapReduce

### Que -1: Working of the WordCount example on Apache Hadoop Website

- I validated the Single Node Cluster setup in Apache Hadoop system and ran the canonical WordCount MapReduce example.

```
jyoti@JyotiDwivedi:~$ start-dfs.sh
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [JyotiDwivedi]
jyoti@JyotiDwivedi:~$ start-yarn.sh
Starting resourcemanager
Starting nodemanagers
jyoti@JyotiDwivedi:~$ jps
2161 DataNode
2900 Jps
2373 SecondaryNameNode
2037 NameNode
2621 ResourceManager
2734 NodeManager
```

Figure 1- Single Node cluster



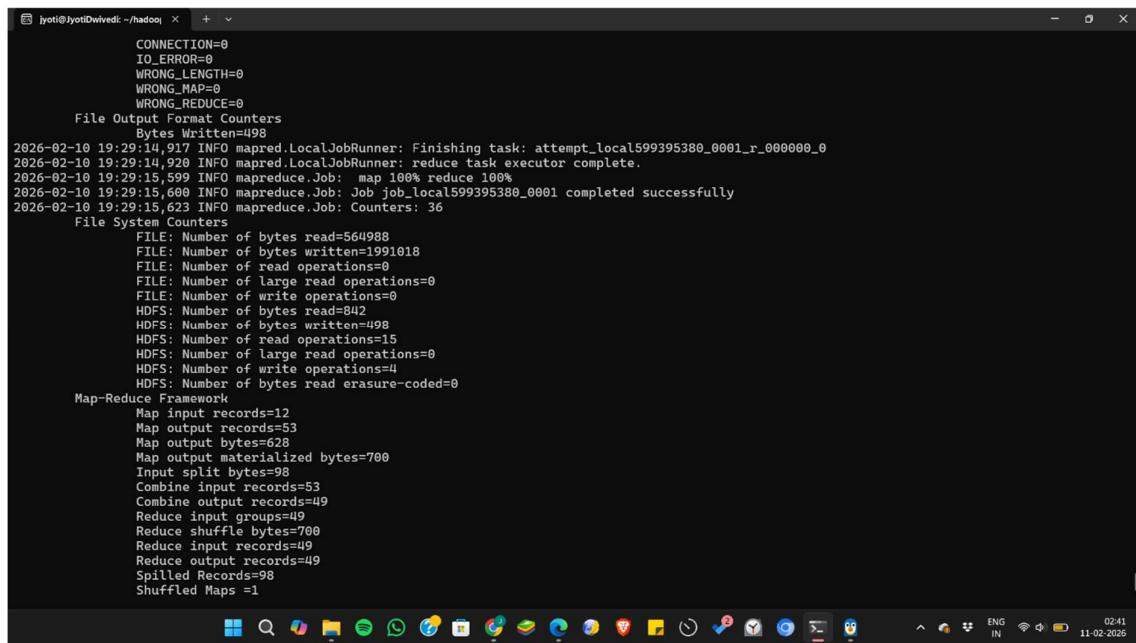
```

jyoti@JyotiDwivedi:~/hadoop$ hdfs dfs -copyFromLocal /etc/hosts /input
jyoti@JyotiDwivedi:~/hadoop$ hdfs dfs -ls /input
Found 1 items
-rw-r--r-- 1 jyoti supergroup 421 2026-02-10 19:28 /input/hosts
jyoti@JyotiDwivedi:~/hadoop$ hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.4.0.jar wordcount /input/host output_q1
2026-02-10 19:29:12,123 INFO impl.MetricsConfig: Loaded properties from hadoop-metrics2.properties
2026-02-10 19:29:12,385 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2026-02-10 19:29:12,385 INFO impl.MetricsSystemImpl: JobTracker metrics system started
2026-02-10 19:29:12,770 INFO input.FileInputFormat: Total input files to process : 1
2026-02-10 19:29:12,948 INFO mapreduce.JobSubmitter: number of splits:1
2026-02-10 19:29:13,258 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local599395380_0001
2026-02-10 19:29:13,258 INFO mapreduce.JobSubmitter: Executing with tokens: []
2026-02-10 19:29:13,581 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
2026-02-10 19:29:13,581 INFO mapreduce.Job: Running job: job_local599395380_0001
2026-02-10 19:29:13,585 INFO mapred.LocalJobRunner: OutputCommitter set in config null
2026-02-10 19:29:13,604 INFO output.PathOutputCommitterFactory: No output committer factory defined, defaulting to FileOutputCommitterFactory
2026-02-10 19:29:13,607 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2026-02-10 19:29:13,607 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failures: false
2026-02-10 19:29:13,609 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapreduce.lib.output.FileOutputCommitter
2026-02-10 19:29:13,753 INFO mapred.LocalJobRunner: Waiting for map tasks
2026-02-10 19:29:13,756 INFO mapred.LocalJobRunner: Starting task: attempt_local599395380_0001_m_000000_0
2026-02-10 19:29:13,802 INFO output.PathOutputCommitterFactory: No output committer factory defined, defaulting to FileOutputCommitterFactory
2026-02-10 19:29:13,803 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2026-02-10 19:29:13,803 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failures: false
2026-02-10 19:29:13,869 INFO mapred.Task: Using ResourceCalculatorProcessTree : []
2026-02-10 19:29:13,880 INFO mapred.MapTask: Processing split: hdfs://localhost:9000/input/hosts:0+421
2026-02-10 19:29:13,934 INFO mapred.MapTask: (EQUATOR) 0 kvi 26214396(104857584)
2026-02-10 19:29:13,935 INFO mapred.MapTask: mapreduce.task.io.sort.mb: 100
2026-02-10 19:29:13,935 INFO mapred.MapTask: soft limit at 83886080
2026-02-10 19:29:13,935 INFO mapred.MapTask: bufstart = 0; bufvoid = 104857600
2026-02-10 19:29:13,935 INFO mapred.MapTask: kvstart = 26214396; length = 6553600
2026-02-10 19:29:13,945 INFO mapred.MapTask: Map output collector class = org.apache.hadoop.mapred.MapTask$MapOutputBuffer
2026-02-10 19:29:14,289 INFO mapred.LocalJobRunner:
2026-02-10 19:29:14,296 INFO mapred.MapTask: Starting flush of map output
2026-02-10 19:29:14,296 INFO mapred.MapTask: Spilling map output

```

Figure 2- Config and setup

- The HDFS environment was successfully configured on my local WSL2 based linux subsystem. The standard WordCount MapReduce code in java was compiled into a JAR file and executed against 2 basic text files (file01.txt and file02.txt) stored in the HDFS.



```

jyoti@JyotiDwivedi:~/hadoop$ ./wordcount
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Output Format Counters
Bytes Written=98
2026-02-10 19:29:14,917 INFO mapred.LocalJobRunner: Finishing task: attempt_local599395380_0001_r_000000_0
2026-02-10 19:29:14,920 INFO mapred.LocalJobRunner: reduce task executor complete.
2026-02-10 19:29:15,599 INFO mapreduce.Job: map 100% reduce 100%
2026-02-10 19:29:15,600 INFO mapreduce.Job: Job job_local599395380_0001 completed successfully
2026-02-10 19:29:15,623 INFO mapreduce.Job: Counters: 36
File System Counters
FILE: Number of bytes read=564988
FILE: Number of bytes written=1991018
FILE: Number of read operations=0
FILE: Number of large read operations=0
FILE: Number of write operations=0
HDFS: Number of bytes read=842
HDFS: Number of bytes written=498
HDFS: Number of read operations=15
HDFS: Number of large read operations=0
HDFS: Number of write operations=4
HDFS: Number of bytes read erasure-coded=0
Map-Reduce Framework
Map input records=12
Map output records=53
Map output bytes=628
Map output materialized bytes=700
Input split bytes=98
Combine input records=53
Combine output records=49
Reduce input groups=49
Reduce shuffle bytes=700
Reduce input records=19
Reduce output records=49
Spilled Records=98
Shuffled Maps =1

```

Figure 3- Map & Reduce operations

- As shown in the below image, the Map and Reduce tasks completed successfully with a replication of the shuffle and sort phases. The final output was verified using the hdfs dfs -cat command on the part-r-00000 output file.

```

jyoti@JyotiDwivedi: ~/CSL7111 X + v
          Reduce shuffle bytes=85
          Reduce input records=6
          Reduce output records=5
          Spilled Records=12
          Shuffled Maps =2
          Failed Shuffles=0
          Merged Map outputs=2
          GC time elapsed (ms)=16
          Total committed heap usage (bytes)=803733504
        Shuffle Errors
          BAD_ID=0
          CONNECTION=0
          IO_ERROR=0
          WRONG_LENGTH=0
          WRONG_MAP=0
          WRONG_REDUCE=0
      File Input Format Counters
        Bytes Read=50
      File Output Format Counters
        Bytes Written=41
jyoti@JyotiDwivedi:~/CSL7110_Assignment$ hdfs dfs -cat /user/jyoti/wordcount/output/part-r-00000
Bye      1
Goodbye 1
Hadoop   2
Hello    2
World    2
jyoti@JyotiDwivedi:~/CSL7110_Assignment$ |

```

*Figure 4: WordCount result*

### **Que -2: Output of the Map phase for the following Lyrics:**

We're up all night till the sun

We're up all night to get some

We're up all night for good fun

We're up all night to get lucky

Ans:

#### 1.Output of Map phase

The Mapper reads the input text line by line and tokenize it into individual words. For each word, it emits a key-value pair where the key is the word and the value is the integer. The input/output key-value pairs and their corresponding Hadoop datatype during the Map phase for a given text snippet is as follows:

(“We’re”, 1), (“up”, 1), (“all”, 1), (“night”, 1), (“till”, 1), (“the”, 1), (“sun”, 1)  
 (“We’re”, 1), (“up”, 1”), (“all”, 1), (“night”, 1), (“to”, 1), (“get”, 1), (“some”, 1)  
 (“We’re”, 1), (“up”, 1”), (“all”, 1), (“night”, 1), (“for”, 1), (“good”, 1), (“fun”, 1)  
 (“We’re”, 1), (“up”, 1”), (“all”, 1), (“night”, 1), (“to”, 1), (“get”, 1), (“lucky”, 1)

#### 2. Types of Keys and Values in the Map Phase

Hadoop does not use standard Java primitives (like int or String) for its MapReduce data flow because it requires classes that implement the Writable interface for efficient network serialization across the distributed cluster.

These are the datatypes in Map phase:

- Input Key (LongWritable): The byte offset of the line from the beginning of the file. As noted, this number can grow extremely large for massive files, so Hadoop uses a LongWritable rather than a standard integer.
- Input Value (Text): The actual line of text (the lyrics). Hadoop uses Text as its equivalent to Java's String.
- Output Key (Text): The individual parsed word (e.g., "night").
- Output Value (IntWritable): The count for each emitted word, which is always 1 during the Map phase. Hadoop uses IntWritable as its equivalent to Java's Integer.

### **Que -3: Output of the Reduce phase for the following:**

("up", 4)  
("to", 3)  
("get", 2)  
("lucky", 1)

Ans:

#### 1.Inputs of the Reduce Phase

Between the Map phase and the Reduce phase, Hadoop performs a hidden "Shuffle and Sort" operation. During this step, the framework collects all the key-value pairs emitted by the Mappers, sorts them alphabetically by key, and groups all the values associated with the exact same key into an iterable list.

Therefore, to produce the final output pairs like ("up", 4) and ("to", 3), the input pairs fed into the reduce() function will look like this:

("up", [1, 1, 1, 1])  
("to", [1, 1, 1])  
("get", [1, 1])  
("lucky", [1])

The Reducer's job is simply to iterate through that list of 1s and sum them together.

#### 2.Types of Keys and Values in Reduce phase

- Input Key (Text): The grouped word (e.g., "up"). This perfectly matches the Output Key of the Map phase.
- Input Value (Iterable<IntWritable>): A Java Iterable containing the grouped integer counts (the list of 1s) emitted by the Mappers for that specific word.
- Output Key (Text): The word being counted.
- Output Value (IntWritable): The final, aggregated sum of all the occurrences of that word across the entire dataset.

**Que -4: Improvised Map() function and Reduce() function and Arguments in main function**

```
public class WordCount {  
    public static class TokenizerMapper extends Mapper<LongWritable, Text, Text, IntWritable> {  
        private final static IntWritable one = new IntWritable(1);  
        private Text word = new Text();  
  
        public void map(LongWritable key, Text value, Context context) throws IOException,  
InterruptedException {  
            String cleanLine = value.toString().replaceAll("[^a-zA-Z\\s]", " ").toUpperCase();  
  
            StringTokenizer itr = new StringTokenizer(cleanLine);  
            while (itr.hasMoreTokens()) {  
                word.set(itr.nextToken());  
                context.write(word, one);  
            }  
        }  
    }  
}
```

Figure 5 - Map Class with Correct Datatypes

```
public static class IntSumReducer  
    | extends Reducer<Text, IntWritable, Text, IntWritable> {  
    private IntWritable result = new IntWritable();  
  
    public void reduce(Text key, Iterable<IntWritable> values,  
                      | Context context  
                      | ) throws IOException, InterruptedException {  
        int sum = 0;  
        for (IntWritable val : values) {  
            sum += val.get();  
        }  
        result.set(sum);  
        context.write(key, result);  
    }  
}
```

Figure 6 - Reduce class with correct datatypes

```

job.setJarByClass(WordCount.class);
job.setMapperClass(TokenizerMapper.class);
job.setCombinerClass(IntSumReducer.class);
job.setReducerClass(IntSumReducer.class);
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(IntWritable.class);
FileInputFormat.addInputPath(job, new Path(args[0]));
FileOutputFormat.setOutputPath(job, new Path(args[1]));

```

*Figure 7 - Correct Arguments for setOutputKeyClass() and setOutputValueClass() of main function*

### Que -5: Map() function for disregarding punctuation

To implement the logic inside the map() function, ensuring that punctuation is removed using String.replaceAll() and words are split using StringTokenizer.

Implementation Logic:

As shown previously in the TokenizerMapper class implementation, the raw input line (passed as a Text value) is first converted to a standard Java String. To strictly disregard punctuation and standardize the text, a Regular Expression is applied:

1. String cleanLine = value . toString () . replaceAll ( " [^ a - zA - Z \\s ] " , " " ) .toUpperCase () ;

This Regex replaces any character that is not a letter (a-z, A-Z) or whitespace (\s) with a space, and then converts the entire string to uppercase to ensure case-insensitive counting.

Following the cleanup, a StringTokenizer is instantiated with the cleanLine. A while loop iterates through the tokens, assigning each word to the Text key and emitting it alongside the IntWritable value of 1 to the Context.

### Que -6: Writing the Reduce function and Compilation

To implement the logic inside the reduce() function to aggregate the wordcounts and ensure successful project compilation.

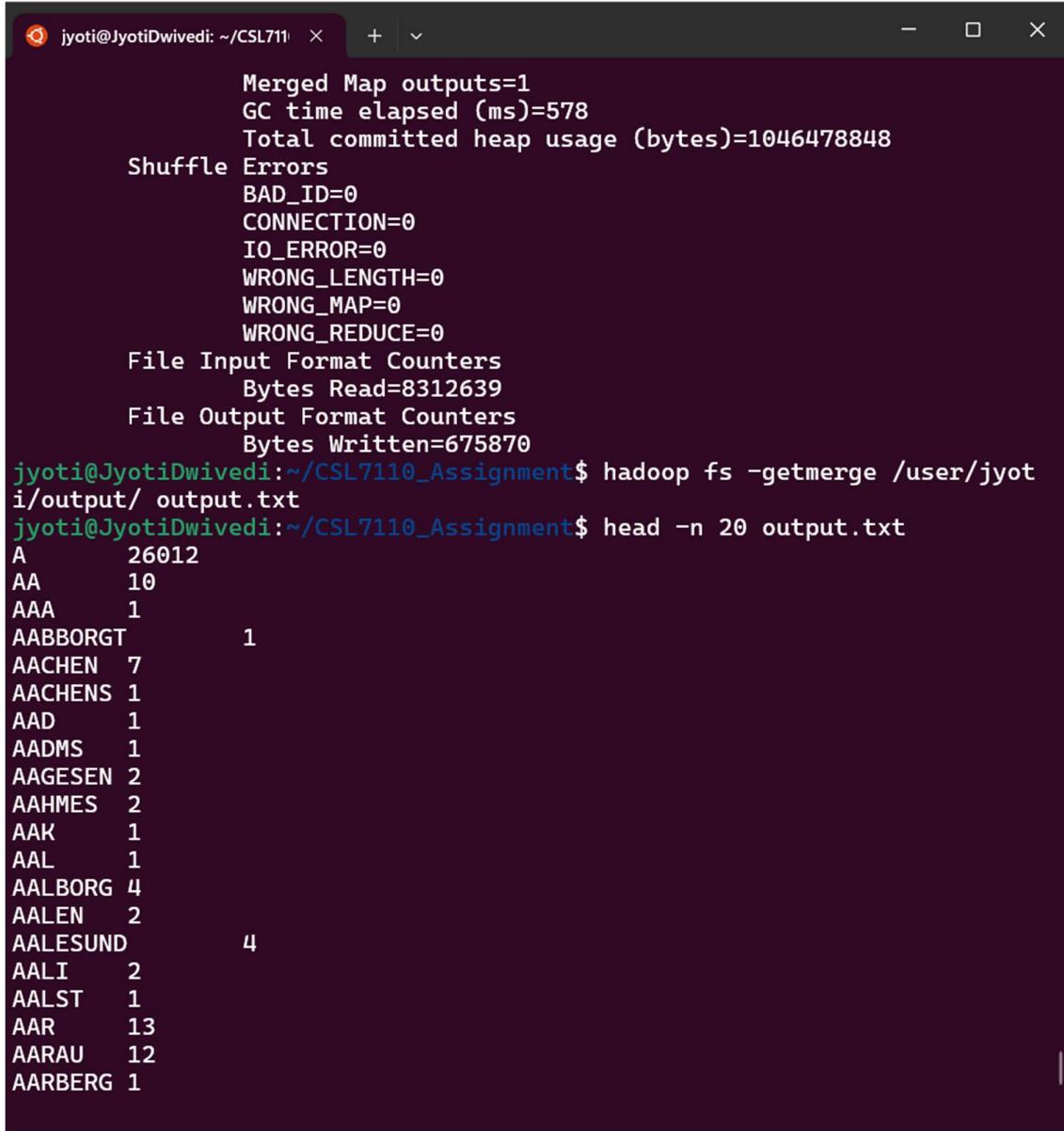
Implementation Logic: As shown previously in the IntSumReducer class implementation, the reduce() function receives a unique word (Text key) and an iterable collection of its counts (Iterable<IntWritable>values).

A for loop is utilized to iterate over these values:

1. int sum = 0;
   
for ( IntWritable val : values ) {
   
sum += val . get () ;}

The .get() method converts the Hadoop IntWritable back to a standard Java int for arithmetic summation. Once the loop finishes, the final sum is wrapped back into an IntWritable and written to the output context alongside the word key.

### Que -7: Running Wordcount on 200.txt



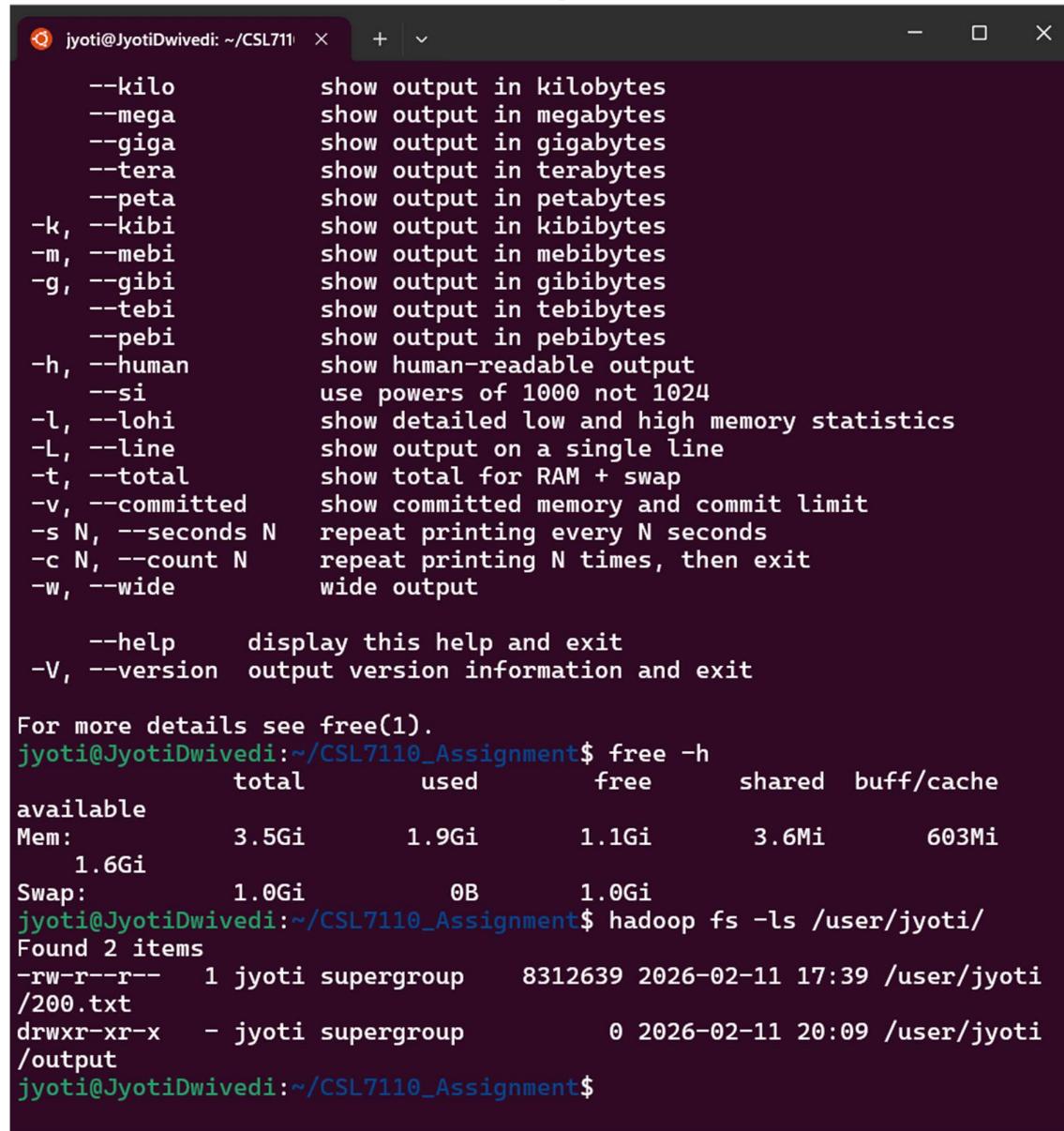
```
Merged Map outputs=1
GC time elapsed (ms)=578
Total committed heap usage (bytes)=1046478848
Shuffle Errors
  BAD_ID=0
  CONNECTION=0
  IO_ERROR=0
  WRONG_LENGTH=0
  WRONG_MAP=0
  WRONG_REDUCE=0
File Input Format Counters
  Bytes Read=8312639
File Output Format Counters
  Bytes Written=675870
jyoti@JyotiDwivedi:~/CSL7110_Assignment$ hadoop fs -getmerge /user/jyoti/output/ output.txt
jyoti@JyotiDwivedi:~/CSL7110_Assignment$ head -n 20 output.txt
A          26012
AA         10
AAA        1
AABBORGT    1
AACHEN      7
AACHENS     1
AAD         1
AADMS       1
AAGESEN     2
AAHMES      2
AAK         1
AAL         1
AALBORG     4
AALEN       2
AALESUND    4
AALI         2
AALST       1
AAR         13
AARAU      12
AARBERG     1
```

Figure 8- output without punctuation

### Que -8: HDFS Operations and Directory Replication

In the Hadoop Distributed File System (HDFS), the replication factor applies exclusively to physical data blocks. When a large text file is uploaded, HDFS splits it into physical blocks (e.g., 128 MB chunks) and replicates those specific blocks across multiple DataNodes (typically 3 times) to ensure fault tolerance against hardware failure. A directory, however, is not a physical file and contains no data blocks. It is strictly logical namespace metadata, a structural record of file hierarchies, permissions, and ownership. This metadata is managed and stored entirely within the NameNode's memory. Because a directory has no physical blocks distributed across the DataNodes, there is nothing for HDFS to apply a block replication factor to. Fault tolerance for directories and other metadata is handled separately at

the NameNode level (e.g., through High Availability Active/Standby NameNodes or Secondary NameNodes), rather than through DataNode block replication.



```

jyoti@JyotiDwivedi: ~/CSL7110 ~
--kilo      show output in kilobytes
--mega      show output in megabytes
--giga      show output in gigabytes
--tera      show output in terabytes
--peta      show output in petabytes
-k, --kibi   show output in kibibytes
-m, --mebi   show output in mebibytes
-g, --gibi   show output in gibibytes
--tebi     show output in tebibytes
--pebi     show output in pebibytes
-h, --human  show human-readable output
--si        use powers of 1000 not 1024
-l, --lohi   show detailed low and high memory statistics
-L, --line   show output on a single line
-t, --total  show total for RAM + swap
-v, --committed  show committed memory and commit limit
-s N, --seconds N repeat printing every N seconds
-c N, --count N repeat printing N times, then exit
-w, --wide    wide output

--help      display this help and exit
-V, --version output version information and exit

For more details see free(1).
jyoti@JyotiDwivedi:~/CSL7110_Assignment$ free -h
              total        used        free      shared  buff/cache available
Mem:       3.5Gi       1.9Gi       1.1Gi      3.6Mi       603Mi
          1.6Gi
Swap:      1.0Gi         0B       1.0Gi
jyoti@JyotiDwivedi:~/CSL7110_Assignment$ hadoop fs -ls /user/jyoti/
Found 2 items
-rw-r--r--  1 jyoti supergroup  8312639 2026-02-11 17:39 /user/jyoti/
/200.txt
drwxr-xr-x  - jyoti supergroup         0 2026-02-11 20:09 /user/jyoti/
/output
jyoti@JyotiDwivedi:~/CSL7110_Assignment$
```

Figure 9 - showing directory replication output

### Que -9: Parameter Impact Performance{ Execution time and Split size impact}

```

50 | public static void main(String[] args) throws Exception {
51 | | Configuration conf = new Configuration();
52 | | // the config setting below leads to number of split = 9, if we comment that line then we'll have
| | number of split = 1
53 | | conf.setLong("mapreduce.input.fileinputformat.split.maxsize", 1000000L);
```

Figure 10 - Code to implement split for parameter impact understanding

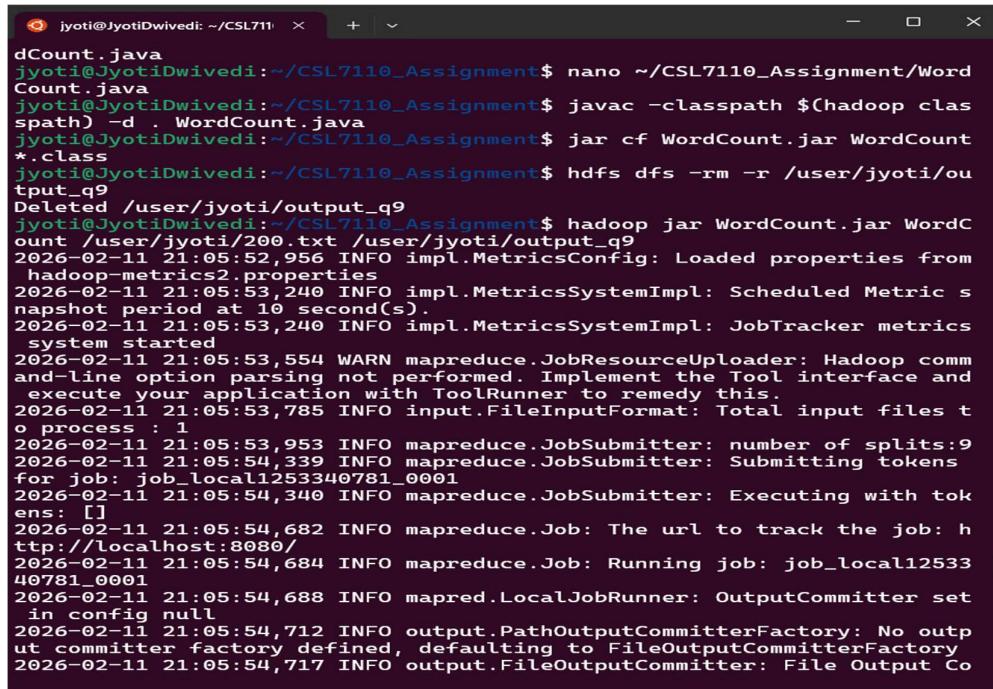
The parameter `mapreduce.input.fileinputformat.split.maxsize` explicitly defines the maximum allowable byte size for a single logical InputSplit. Because the Hadoop framework strictly assigns

exactly **one Mapper task per InputSplit**, changing this value directly controls the degree of parallelism in the Map phase.

As observed in my implementation, setting `split.maxsize` to 1000000 (approx. 1 MB) on the 200.txt file forced Hadoop to divide the file into 9 separate splits, thereby spawning 9 parallel Mapper tasks. When the parameter was commented out, Hadoop defaulted to using the HDFS block size (typically 128 MB), resulting in only 1 split and 1 Mapper task.

This creates a critical performance trade-off:

- **Decreasing the Max Size (More Splits):** Forces Hadoop to create smaller splits, increasing the number of Mappers. While this increases parallel processing (beneficial for massive datasets), setting it too low severely degrades performance. The system becomes bottlenecked by Task Overhead—the time spent by the ApplicationMaster scheduling tasks and spinning up/tearing down individual Java Virtual Machines (JVMs) for each tiny Mapper outpaces the actual data processing time.
- **Increasing the Max Size (Fewer Splits):** Forces Hadoop to create larger splits, decreasing the number of Mappers. While this minimizes JVM startup overhead, setting it too high severely limits parallelism. The cluster's compute resources remain idle while a single Mapper slowly processes a massive chunk of data sequentially. Therefore, altering this parameter impacts performance by shifting the balance between parallel execution capacity and JVM task-management overhead. The optimal performance is typically achieved when the split size perfectly matches the underlying HDFS physical block size (e.g., 128 MB), ensuring data locality and minimizing network I/O.



The screenshot shows a terminal window titled "jyoti@JyotiDwivedi: ~/CSL7110\_Assignment". The session starts with the command "nano ~/CSL7110\_Assignment/WordCount.java", followed by "javac -classpath \$(hadoop classpath) -d . WordCount.java". Then, "jar cf WordCount.jar WordCount\*.class" is run to create the JAR file. Finally, "hdfs dfs -rm -r /user/jyoti/output\_q9" is used to delete the previous output directory. The log output shows the job starting at 2026-02-11 21:05:52,956 INFO and running until 2026-02-11 21:05:53,240 INFO, with various logs from the JobTracker and TaskTrackers.

```
jyoti@JyotiDwivedi:~/CSL7110_Assignment$ nano ~/CSL7110_Assignment/WordCount.java
jyoti@JyotiDwivedi:~/CSL7110_Assignment$ javac -classpath $(hadoop classpath) -d . WordCount.java
jyoti@JyotiDwivedi:~/CSL7110_Assignment$ jar cf WordCount.jar WordCount*.class
jyoti@JyotiDwivedi:~/CSL7110_Assignment$ hdfs dfs -rm -r /user/jyoti/output_q9
Deleted /user/jyoti/output_q9
jyoti@JyotiDwivedi:~/CSL7110_Assignment$ hadoop jar WordCount.jar WordCount /user/jyoti/200.txt /user/jyoti/output_q9
2026-02-11 21:05:52,956 INFO impl.MetricsConfig: Loaded properties from hadoop-metrics2.properties
2026-02-11 21:05:53,240 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2026-02-11 21:05:53,240 INFO impl.MetricsSystemImpl: JobTracker metrics system started
2026-02-11 21:05:53,554 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
2026-02-11 21:05:53,785 INFO input.FileInputFormat: Total input files to process : 1
2026-02-11 21:05:53,953 INFO mapreduce.JobSubmitter: number of splits:9
2026-02-11 21:05:54,339 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local1253340781_0001
2026-02-11 21:05:54,340 INFO mapreduce.JobSubmitter: Executing with tokens: []
2026-02-11 21:05:54,682 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
2026-02-11 21:05:54,684 INFO mapreduce.Job: Running job: job_local1253340781_0001
2026-02-11 21:05:54,688 INFO mapred.LocalJobRunner: OutputCommitter set in config null
2026-02-11 21:05:54,712 INFO output.PathOutputCommitterFactory: No output committer factory defined, defaulting to FileOutputCommitterFactory
2026-02-11 21:05:54,717 INFO output.FileOutputCommitter: File Output Co
```

Figure 11- process running with parameter impact

```

jyoti@JyotiDwivedi: ~/CSL7110_Assignment$ hdfs dfs -report
HDFS: Number of bytes read=52838270
HDFS: Number of bytes written=675870
HDFS: Number of read operations=143
HDFS: Number of large read operations=0
HDFS: Number of write operations=12
HDFS: Number of bytes read erasure-coded=0
Map-Reduce Framework
  Map input records=146933
  Map output records=1318770
  Map output bytes=12867576
  Map output materialized bytes=2134575
  Input split bytes=945
  Combine input records=1318770
  Combine output records=147544
  Reduce input groups=62006
  Reduce shuffle bytes=2134575
  Reduce input records=147544
  Reduce output records=62006
  Spilled Records=295088
  Shuffled Maps =9
  Failed Shuffles=0
  Merged Map outputs=9
  GC time elapsed (ms)=1218
  Total committed heap usage (bytes)=5819596800
Shuffle Errors
  BAD_ID=0
  CONNECTION=0
  IO_ERROR=0
  WRONG_LENGTH=0
  WRONG_MAP=0
  WRONG_REDUCE=0
File Input Format Counters
  Bytes Read=8345407
File Output Format Counters
  Bytes Written=675870
Job Finished in: 11330 ms
jyoti@JyotiDwivedi: ~/CSL7110_Assignment$ 

```

Figure 12 - Time taken with parameter impact

```

jyoti@JyotiDwivedi: ~/CSL7110_Assignment$ hdfs dfs -rm -r /user/jyoti/output_q9
Deleted /user/jyoti/output_q9
jyoti@JyotiDwivedi: ~/CSL7110_Assignment$ hadoop jar WordCount.jar WordCount /user/jyoti/200.txt /user/jyoti/output_q9
2026-02-11 21:11:53,547 INFO impl.MetricsConfig: Loaded properties from hadoop-metrics2.properties
2026-02-11 21:11:53,753 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2026-02-11 21:11:53,753 INFO impl.MetricsSystemImpl: JobTracker metrics system started
2026-02-11 21:11:54,022 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
2026-02-11 21:11:54,233 INFO input.FileInputFormat: Total input files to process : 1
2026-02-11 21:11:54,392 INFO mapreduce.JobSubmitter: number of splits:1
2026-02-11 21:11:54,694 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local730291730_0001
2026-02-11 21:11:54,695 INFO mapreduce.JobSubmitter: Executing with tokens: []
2026-02-11 21:11:55,016 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
2026-02-11 21:11:55,019 INFO mapreduce.Job: Running job: job_local730291730_0001
2026-02-11 21:11:55,022 INFO mapred.LocalJobRunner: OutputCommitter set in config null
2026-02-11 21:11:55,041 INFO output.PathOutputCommitterFactory: No output committer factory defined, defaulting to FileOutputCommitterFactory
2026-02-11 21:11:55,045 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2026-02-11 21:11:55,045 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failures: false
2026-02-11 21:11:55,047 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapreduce.lib.output.FileOutputCommitter

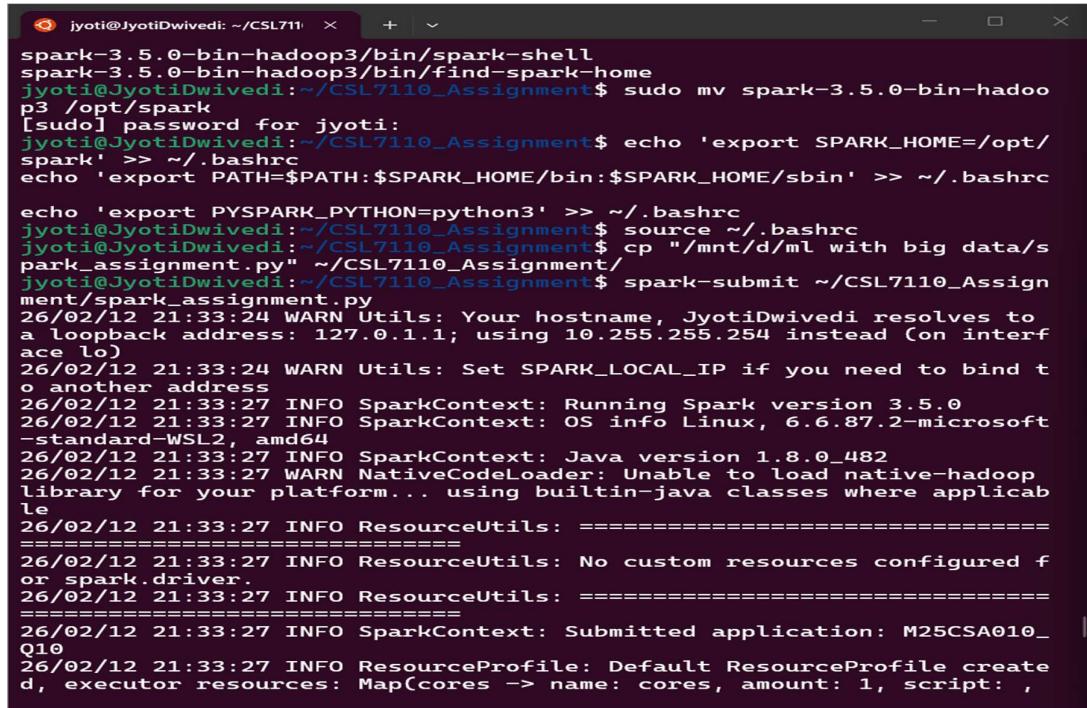
```

Figure 13 - Process without parameter impact

```
jyoti@JyotiDwivedi: ~/CSL7110_Assignment$ hadoop jar wordcount.jar wordcount /user/jyoti/input /user/jyoti/output  
HDFS: Number of bytes read=16625278  
HDFS: Number of bytes written=675870  
HDFS: Number of read operations=15  
HDFS: Number of large read operations=0  
HDFS: Number of write operations=4  
HDFS: Number of bytes read erasure-coded=0  
Map-Reduce Framework  
    Map input records=146933  
    Map output records=1318770  
    Map output bytes=12867576  
    Map output materialized bytes=912475  
    Input split bytes=105  
    Combine input records=1318770  
    Combine output records=62006  
    Reduce input groups=62006  
    Reduce shuffle bytes=912475  
    Reduce input records=62006  
    Reduce output records=62006  
    Spilled Records=124012  
    Shuffled Maps =1  
    Failed Shuffles=0  
    Merged Map outputs=1  
    GC time elapsed (ms)=445  
    Total committed heap usage (bytes)=1048576000  
Shuffle Errors  
    BAD_ID=0  
    CONNECTION=0  
    IO_ERROR=0  
    WRONG_LENGTH=0  
    WRONG_MAP=0  
    WRONG_REDUCE=0  
File Input Format Counters  
    Bytes Read=8312639  
File Output Format Counters  
    Bytes Written=675870  
Job Finished in: 9962 ms  
jyoti@JyotiDwivedi:~/CSL7110_Assignment$
```

Figure 14 - Time for process without parameter impact

## Que -10: Book Metadata Extraction and Analysis

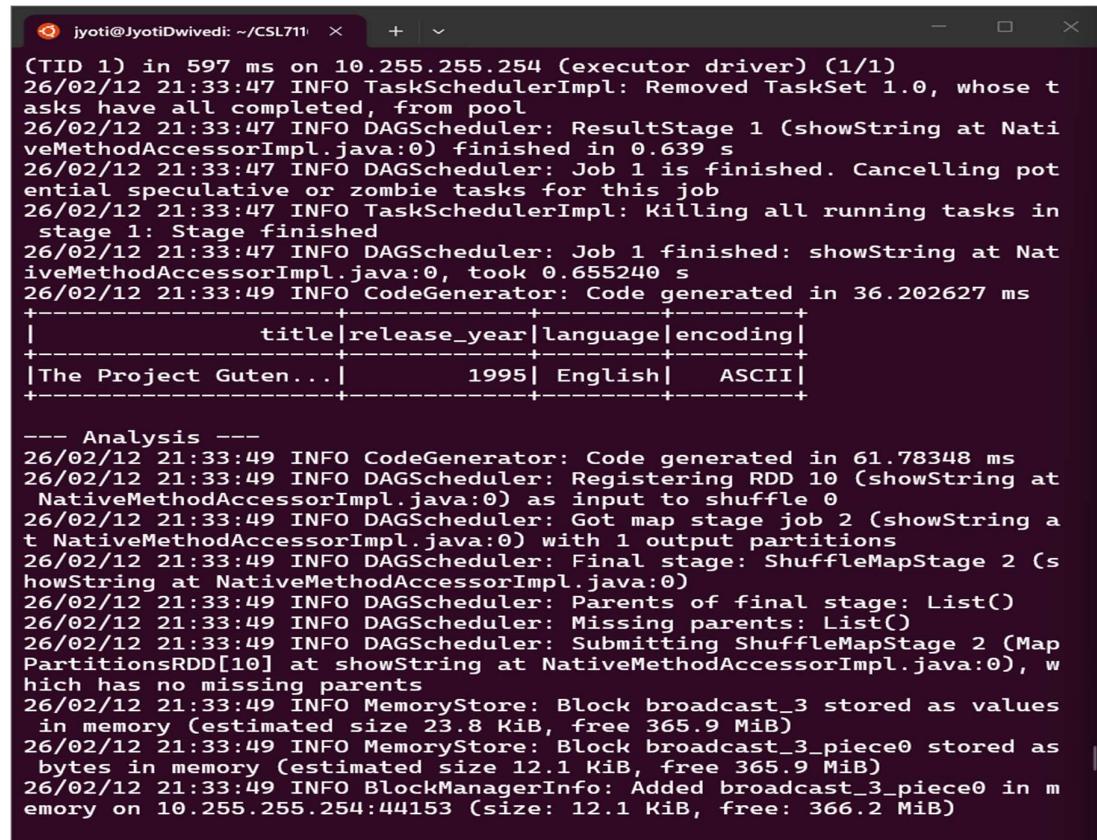


```

jyoti@JyotiDwivedi: ~/CSL7110  + ~
spark-3.5.0-bin-hadoop3/bin/spark-shell
spark-3.5.0-bin-hadoop3/bin/find-spark-home
jyoti@JyotiDwivedi:~/CSL7110_Assignment$ sudo mv spark-3.5.0-bin-hadoop3 /opt/spark
[jsudo] password for jyoti:
jyoti@JyotiDwivedi:~/CSL7110_Assignment$ echo 'export SPARK_HOME=/opt/spark' >> ~/.bashrc
echo 'export PATH=$PATH:$SPARK_HOME/bin:$SPARK_HOME/sbin' >> ~/.bashrc
echo 'export PYSPARK_PYTHON=python3' >> ~/.bashrc
jyoti@JyotiDwivedi:~/CSL7110_Assignment$ source ~/.bashrc
jyoti@JyotiDwivedi:~/CSL7110_Assignment$ cp "/mnt/d/ml with big data/spark_assignment.py" ~/CSL7110_Assignment/
jyoti@JyotiDwivedi:~/CSL7110_Assignment$ spark-submit ~/CSL7110_Assignment/spark_assignment.py
26/02/12 21:33:24 WARN Utils: Your hostname, JyotiDwivedi resolves to a loopback address: 127.0.1.1; using 10.255.255.254 instead (on interface lo)
26/02/12 21:33:24 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
26/02/12 21:33:27 INFO SparkContext: Running Spark version 3.5.0
26/02/12 21:33:27 INFO SparkContext: OS info Linux, 6.6.87.2-microsoft-standard-WSL2, amd64
26/02/12 21:33:27 INFO SparkContext: Java version 1.8.0_482
26/02/12 21:33:27 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
26/02/12 21:33:27 INFO ResourceUtils: =====
=====
26/02/12 21:33:27 INFO ResourceUtils: No custom resources configured for spark.driver.
26/02/12 21:33:27 INFO ResourceUtils: =====
=====
26/02/12 21:33:27 INFO SparkContext: Submitted application: M25CSA010_Q10
26/02/12 21:33:27 INFO ResourceProfile: Default ResourceProfile created, executor resources: Map(cores -> name: cores, amount: 1, script: ,

```

Figure 15 spark setup



title	release_year	language	encoding
The Project Guten...	1995	English	ASCII

```

--- Analysis ---
26/02/12 21:33:49 INFO CodeGenerator: Code generated in 61.78348 ms
26/02/12 21:33:49 INFO DAGScheduler: Registering RDD 10 (showString at NativeMethodAccessorImpl.java:0) as input to shuffle 0
26/02/12 21:33:49 INFO DAGScheduler: Got map stage job 2 (showString at NativeMethodAccessorImpl.java:0) with 1 output partitions
26/02/12 21:33:49 INFO DAGScheduler: Final stage: ShuffleMapStage 2 (showString at NativeMethodAccessorImpl.java:0)
26/02/12 21:33:49 INFO DAGScheduler: Parents of final stage: List()
26/02/12 21:33:49 INFO DAGScheduler: Missing parents: List()
26/02/12 21:33:49 INFO DAGScheduler: Submitting ShuffleMapStage 2 (MapPartitionsRDD[10] at showString at NativeMethodAccessorImpl.java:0), which has no missing parents
26/02/12 21:33:49 INFO MemoryStore: Block broadcast_3 stored as values in memory (estimated size 23.8 KiB, free 365.9 MiB)
26/02/12 21:33:49 INFO MemoryStore: Block broadcast_3_piece0 stored as bytes in memory (estimated size 12.1 KiB, free 365.9 MiB)
26/02/12 21:33:49 INFO BlockManagerInfo: Added broadcast_3_piece0 in memory on 10.255.255.254:44153 (size: 12.1 KiB, free: 366.2 MiB)

```

Figure 16 - metadata extraction

```
jyoti@JyotiDwivedi: ~/CSL711 × + ▾
5 bytes)
26/02/12 21:33:50 INFO Executor: Running task 0.0 in stage 4.0 (TID 3)
26/02/12 21:33:50 INFO ShuffleBlockFetcherIterator: Getting 1 (66.0 B)
non-empty blocks including 1 (66.0 B) local and 0 (0.0 B) host-local
and 0 (0.0 B) push-merged-local and 0 (0.0 B) remote blocks
26/02/12 21:33:50 INFO ShuffleBlockFetcherIterator: Started 0 remote f
etches in 27 ms
26/02/12 21:33:50 INFO CodeGenerator: Code generated in 31.335737 ms
26/02/12 21:33:50 INFO Executor: Finished task 0.0 in stage 4.0 (TID 3
). 4047 bytes result sent to driver
26/02/12 21:33:50 INFO TaskSetManager: Finished task 0.0 in stage 4.0
(TID 3) in 164 ms on 10.255.255.254 (executor driver) (1/1)
26/02/12 21:33:50 INFO TaskSchedulerImpl: Removed TaskSet 4.0, whose t
asks have all completed, from pool
26/02/12 21:33:50 INFO DAGScheduler: ResultStage 4 (showString at Nati
veMethodAccessorImpl.java:0) finished in 0.196 s
26/02/12 21:33:50 INFO DAGScheduler: Job 3 is finished. Cancelling pot
ential speculative or zombie tasks for this job
26/02/12 21:33:50 INFO TaskSchedulerImpl: Killing all running tasks in
stage 4: Stage finished
26/02/12 21:33:50 INFO DAGScheduler: Job 3 finished: showString at Nat
iveMethodAccessorImpl.java:0, took 0.236050 s
26/02/12 21:33:50 INFO CodeGenerator: Code generated in 15.640851 ms
+-----+
|AvgTitleLength|
+-----+
|      51.0|
+-----+
26/02/12 21:33:51 INFO CodeGenerator: Code generated in 112.925583 ms
26/02/12 21:33:51 INFO DAGScheduler: Registering RDD 15 (showString at
NativeMethodAccessorImpl.java:0) as input to shuffle 1
26/02/12 21:33:51 INFO DAGScheduler: Got map stage job 4 (showString a
t NativeMethodAccessorImpl.java:0) with 1 output partitions
26/02/12 21:33:51 INFO DAGScheduler: Final stage: ShuffleMapStage 5 (s
howString at NativeMethodAccessorImpl.java:0)
26/02/12 21:33:51 INFO DAGScheduler: Parents of final stage: List()
```

Figure 17 - Average Title length

```
jyoti@JyotiDwivedi: ~/CSL711  X  +  ^  -  □  ×
asks have all completed, from pool
26/02/12 21:33:52 INFO DAGScheduler: ResultStage 7 (showString at NativeMethodAccessorImpl.java:0) finished in 0.117 s
26/02/12 21:33:52 INFO DAGScheduler: Job 5 is finished. Cancelling potential speculative or zombie tasks for this job
26/02/12 21:33:52 INFO TaskSchedulerImpl: Killing all running tasks in stage 7: Stage finished
26/02/12 21:33:52 INFO DAGScheduler: Job 5 finished: showString at NativeMethodAccessorImpl.java:0, took 0.140148 s
26/02/12 21:33:52 INFO CodeGenerator: Code generated in 15.577256 ms
+-----+----+
|release_year|count|
+-----+----+
|      1995|      1|
+-----+----+
26/02/12 21:33:52 INFO SparkContext: SparkContext is stopping with exitCode 0.
26/02/12 21:33:52 INFO SparkUI: Stopped Spark web UI at http://10.255.255.254:4040
26/02/12 21:33:52 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
26/02/12 21:33:52 INFO MemoryStore: MemoryStore cleared
26/02/12 21:33:52 INFO BlockManager: BlockManager stopped
26/02/12 21:33:52 INFO BlockManagerMaster: BlockManagerMaster stopped
26/02/12 21:33:52 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
26/02/12 21:33:52 INFO SparkContext: Successfully stopped SparkContext
26/02/12 21:33:53 INFO ShutdownHookManager: Shutdown hook called
26/02/12 21:33:53 INFO ShutdownHookManager: Deleting directory /tmp/spark-2d02efe0-d56a-4bb8-af8d-3944ecb49690
26/02/12 21:33:53 INFO ShutdownHookManager: Deleting directory /tmp/spark-f5008003-9638-4d1e-994b-ab911ca2853b
26/02/12 21:33:53 INFO ShutdownHookManager: Deleting directory /tmp/spark-f5008003-9638-4d1e-994b-ab911ca2853b/pyspark-339cefde-06e6-4323-960f-d6ced9a55c7b
jyoti@JyotiDwivedi:~/CSL7110_Assignment$ spark-submit ~/CSL7110_Assign
```

Figure 18 - release year & count

```

Windows PowerShell  X Windows PowerShell  X jyoti@JyotiDwivedi  + - □ ×
yoti/CSL7110_Assignment/spark_novelty.py:33) finished in 11.679 s
26/02/14 12:30:32 INFO DAGScheduler: Job 0 is finished. Cancelling potential speculative or zombie tasks for this job
26/02/14 12:30:32 INFO TaskSchedulerImpl: Killing all running tasks in stage 0: Stage finished
26/02/14 12:30:32 INFO DAGScheduler: Job 0 finished: collect at /home/jyoti/CSL7110_Assignment/spark_novelty.py:33, took 11.902783 s
Book          | Vocab Richness | Avg Word Length
-----
104.txt       | 0.2476        | 4.85 characters
109.txt       | 0.2077        | 4.23 characters
112.txt       | 0.1665        | 4.39 characters
102.txt       | 0.1141        | 4.11 characters
103.txt       | 0.1047        | 4.51 characters
101.txt       | 0.1034        | 4.83 characters
11.txt        | 0.0989        | 4.04 characters
106.txt       | 0.0859        | 4.37 characters
111.txt       | 0.0803        | 4.08 characters
107.txt       | 0.0793        | 4.24 characters
110.txt       | 0.0777        | 4.25 characters
108.txt       | 0.0698        | 4.14 characters
105.txt       | 0.0690        | 4.36 characters
113.txt       | 0.0604        | 3.97 characters
10.txt        | 0.0162        | 4.08 characters
=====
26/02/14 12:30:32 INFO SparkContext: SparkContext is stopping with exitCode 0.
26/02/14 12:30:32 INFO SparkUI: Stopped Spark web UI at http://10.255.255.254:4040
26/02/14 12:30:32 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
26/02/14 12:30:32 INFO MemoryStore: MemoryStore cleared
26/02/14 12:30:32 INFO BlockManager: BlockManager stopped
26/02/14 12:30:32 INFO BlockManagerMaster: BlockManagerMaster stopped
26/02/14 12:30:32 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!

```

Figure 19 - books information

#### Explanation of Regular Expressions:

- Title, Language, and Encoding (r"Keyword:\s\*([^\n\r]+)":): Matches the literal string keyword followed by whitespace. The capture group ([^\n\r]+) greedily extracts all subsequent characters until it hits a newline or carriage return.
- Release Date (r"Release Date:.\*?\b(\d{4})\b":): Matches the literal "Release Date:", skips intermediate text non-greedily (.\*?), and looks for the first 4-digit number (\d{4}) wrapped in word boundaries (\b) to isolate the historical year.

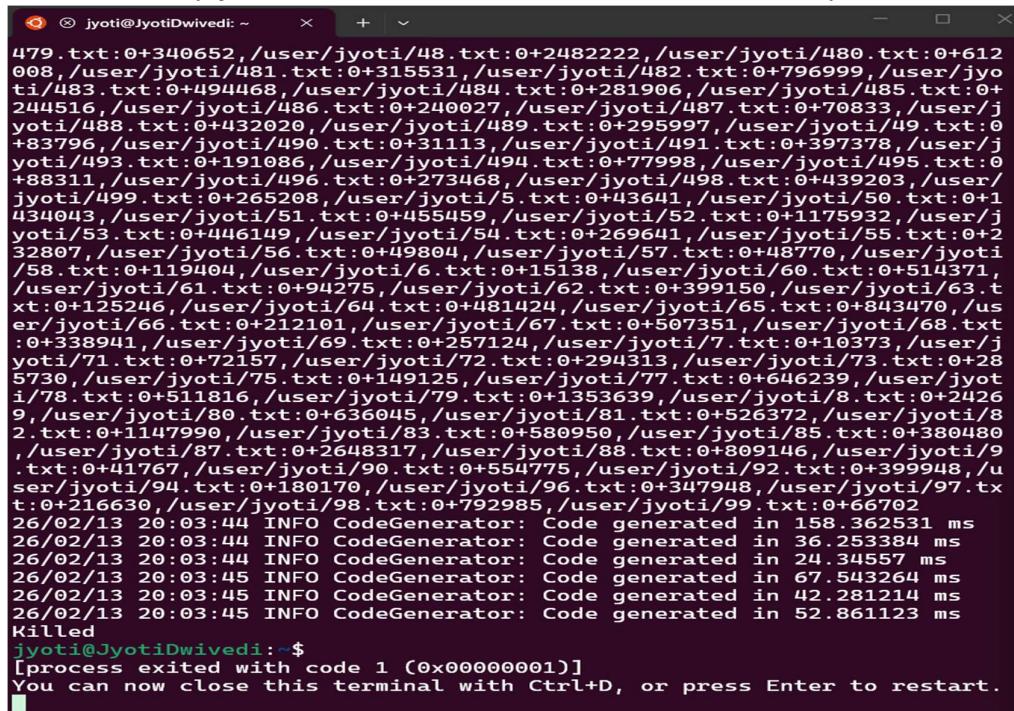
Challenges and Real-World Handling: Regex relies on strict, predictable structural patterns and is notoriously brittle when applied to unstructured text. Typographical errors by transcribers

(e.g., "Book Title:" instead of "Title:") cause Regex to fail, resulting in missing or malformed data. In a real-world scenario, I would address this by implementing a multi-tiered pipeline:

1. Data Quality Checks: Enforce strict schema validation on read (e.g., ensuring release\_year does not contain alphabetical characters).
2. NLP Pipelines: Replace fragile Regex with Machine Learning-based Named Entity Recognition (NER) models to extract dates and titles contextually.
3. Imputation: Use PySpark's fillna() for missing values and filter severely malformed records into a Dead Letter Queue for manual review.

### Que -11: TF-IDF and Book Similarity

- TF and IDF: Term Frequency (TF) measures raw word occurrence, while Inverse Document Frequency (IDF) penalizes common words (like "the") and boosts rare, topic-specific words. TF-IDF effectively vectorizes a document's core thematic meaning.
- Cosine Similarity: Calculates the angle between these high-dimensional vectors, ignoring document length. This is ideal for determining thematic overlap between books of differing sizes.
- Scalability Challenges: Calculating  $O(N^2)$  pairwise similarities requires a massive Cartesian product shuffle. In my environment (1.5GB RAM), this triggered a java.lang.OutOfMemoryError GC Overhead crash. To solve this, I tried using various inputs of books like 2, 5, 10, 15 ,so on and found that my system was able to handle the cosine similarity job of 25 books at once, so I did cosine similarity check for them.



```
jyoti@JyotiDwivedi:~ % + ~ - x
479.txt:0+340652,/user/jyoti/48.txt:0+2482222,/user/jyoti/480.txt:0+612
008,/user/jyoti/481.txt:0+315531,/user/jyoti/482.txt:0+796999,/user/jyo
ti/483.txt:0+494468,/user/jyoti/484.txt:0+281906,/user/jyoti/485.txt:0+
244516,/user/jyoti/486.txt:0+240027,/user/jyoti/487.txt:0+70833,/user/j
yoti/488.txt:0+432020,/user/jyoti/489.txt:0+295997,/user/jyoti/49.txt:0
+83796,/user/jyoti/490.txt:0+31113,/user/jyoti/491.txt:0+397378,/user/j
yoti/493.txt:0+191086,/user/jyoti/494.txt:0+77998,/user/jyoti/495.txt:0
+88311,/user/jyoti/496.txt:0+273468,/user/jyoti/498.txt:0+439203,/user/
jyoti/499.txt:0+265208,/user/jyoti/5.txt:0+43641,/user/jyoti/50.txt:0+1
434043,/user/jyoti/51.txt:0+455459,/user/jyoti/52.txt:0+1175932,/user/j
yoti/53.txt:0+446149,/user/jyoti/54.txt:0+269641,/user/jyoti/55.txt:0+2
32807,/user/jyoti/56.txt:0+49804,/user/jyoti/57.txt:0+48770,/user/jyoti
/58.txt:0+119404,/user/jyoti/6.txt:0+15138,/user/jyoti/60.txt:0+514371,
/user/jyoti/61.txt:0+94275,/user/jyoti/62.txt:0+399150,/user/jyoti/63.t
xt:0+125246,/user/jyoti/64.txt:0+481424,/user/jyoti/65.txt:0+843470,/us
er/jyoti/66.txt:0+212101,/user/jyoti/67.txt:0+507351,/user/jyoti/68.txt
:0+338941,/user/jyoti/69.txt:0+257124,/user/jyoti/7.txt:0+10373,/user/j
yoti/71.txt:0+72157,/user/jyoti/72.txt:0+294313,/user/jyoti/73.txt:0+28
5730,/user/jyoti/75.txt:0+149125,/user/jyoti/77.txt:0+646239,/user/jyot
i/78.txt:0+511816,/user/jyoti/79.txt:0+1353639,/user/jyoti/8.txt:0+2426
9,/user/jyoti/80.txt:0+636045,/user/jyoti/81.txt:0+526372,/user/jyoti/8
2.txt:0+1147990,/user/jyoti/83.txt:0+580950,/user/jyoti/85.txt:0+380480
,/user/jyoti/87.txt:0+2648317,/user/jyoti/88.txt:0+809146,/user/jyoti/9
.txt:0+41767,/user/jyoti/90.txt:0+554775,/user/jyoti/92.txt:0+399948,/u
ser/jyoti/94.txt:0+180170,/user/jyoti/96.txt:0+347948,/user/jyoti/97.tx
t:0+216630,/user/jyoti/98.txt:0+792985,/user/jyoti/99.txt:0+66702
26/02/13 20:03:44 INFO CodeGenerator: Code generated in 158.362531 ms
26/02/13 20:03:44 INFO CodeGenerator: Code generated in 36.253384 ms
26/02/13 20:03:44 INFO CodeGenerator: Code generated in 24.34557 ms
26/02/13 20:03:45 INFO CodeGenerator: Code generated in 67.543264 ms
26/02/13 20:03:45 INFO CodeGenerator: Code generated in 42.281214 ms
26/02/13 20:03:45 INFO CodeGenerator: Code generated in 52.861123 ms
Killed
jyoti@JyotiDwivedi:~$ [process exited with code 1 (0x00000001)]
You can now close this terminal with Ctrl+D, or press Enter to restart.
```

Figure 20- Execution killed due to hardware constraints

```
jyoti@JyotiDwivedi: ~          jyoti@JyotiDwivedi: ~          Windows PowerShell          - + v          X
or zombie tasks for this job
26/02/13 22:02:42 INFO TaskSchedulerImpl: Killing all running tasks in stage 3: Stage finished
26/02/13 22:02:42 INFO DAGScheduler: Job 3 finished: takeOrdered at /home/jyoti/CSL7110_Assignment/spark_q11.py:403, took 4.165867 s
=====
SIMILARITY SCORES TO 10.txt
=====
Book: 109.txt | Score: 0.4218
Book: 108.txt | Score: 0.0689
Book: 102.txt | Score: 0.0561
Book: 107.txt | Score: 0.0538
Book: 101.txt | Score: 0.0452
Book: 11.txt | Score: 0.0403
Book: 106.txt | Score: 0.0275
Book: 105.txt | Score: 0.0244
Book: 104.txt | Score: 0.0233
Book: 103.txt | Score: 0.0205
26/02/13 22:02:42 INFO SparkContext: SparkContext is stopping with exitCode 0.
26/02/13 22:02:42 INFO SparkUI: Stopped Spark web UI at http://10.255.255.254:4040
26/02/13 22:02:42 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
26/02/13 22:02:42 INFO MemoryStore: MemoryStore cleared
26/02/13 22:02:42 INFO BlockManager: BlockManager stopped
26/02/13 22:02:42 INFO BlockManagerMaster: BlockManagerMaster stopped
26/02/13 22:02:42 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
26/02/13 22:02:42 INFO SparkContext: Successfully stopped SparkContext
26/02/13 22:02:43 INFO ShutdownHookManager: Shutdown hook called
26/02/13 22:02:43 INFO ShutdownHookManager: Deleting directory /tmp/spark-1cda3ee7-c057-4009-a9c1-78e5bf5eb32b/pyspark-4a754121-e4aa-4de4-9712-b5f9302d231c
26/02/13 22:02:43 INFO ShutdownHookManager: Deleting directory /tmp/spark-1cda3ee7-c057-4009-a9c1-78e5bf5eb32b
26/02/13 22:02:43 INFO ShutdownHookManager: Deleting directory /tmp/spark-bab61d85-833c-400b-b282-7bfbef2eae42
jyoti@JyotiDwivedi:~$
```

Figure 21- cosine similarity for 10 books

```
jyoti@JyotiDwivedi: ~      jyoti@JyotiDwivedi: ~      Windows PowerShell      -      +      -      X  
=====SIMILARITY SCORES TO 10.txt=====  
Book: 109.txt | Score: 0.3794  
Book: 110.txt | Score: 0.0889  
Book: 108.txt | Score: 0.0838  
Book: 113.txt | Score: 0.0608  
Book: 107.txt | Score: 0.0594  
Book: 102.txt | Score: 0.0587  
Book: 112.txt | Score: 0.0524  
Book: 101.txt | Score: 0.0501  
Book: 11.txt | Score: 0.0426  
Book: 104.txt | Score: 0.0338  
Book: 105.txt | Score: 0.0317  
Book: 106.txt | Score: 0.0305  
Book: 103.txt | Score: 0.0274  
Book: 111.txt | Score: 0.0200  
Book: 114.txt | Score: 0.0073  
26/02/13 22:05:26 INFO SparkContext: SparkContext is stopping with exitCode 0.  
26/02/13 22:05:26 INFO SparkUI: Stopped Spark web UI at http://10.255.255.254:4040  
26/02/13 22:05:26 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!  
26/02/13 22:05:26 INFO MemoryStore: MemoryStore cleared  
26/02/13 22:05:26 INFO BlockManager: BlockManager stopped  
26/02/13 22:05:26 INFO BlockManagerMaster: BlockManagerMaster stopped  
26/02/13 22:05:26 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!  
26/02/13 22:05:26 INFO SparkContext: Successfully stopped SparkContext  
26/02/13 22:05:27 INFO ShutdownHookManager: Shutdown hook called  
26/02/13 22:05:27 INFO ShutdownHookManager: Deleting directory /tmp/spark-9a5f9925-bcf6-4d6b-85ec-fef8a9ae38a4/pyspark-4ccb846-7267-400d-8bc9-fd8c5a9f7bc5  
26/02/13 22:05:27 INFO ShutdownHookManager: Deleting directory /tmp/spark-9a5f9925-bcf6-4d6b-85ec-fef8a9ae38a4  
26/02/13 22:05:27 INFO ShutdownHookManager: Deleting directory /tmp/spark-720f687e-6951-47ed-b523-9949d50743d0  
jyoti@JyotiDwivedi:~$
```

Figure 22 - Cosine similarity for 15 books

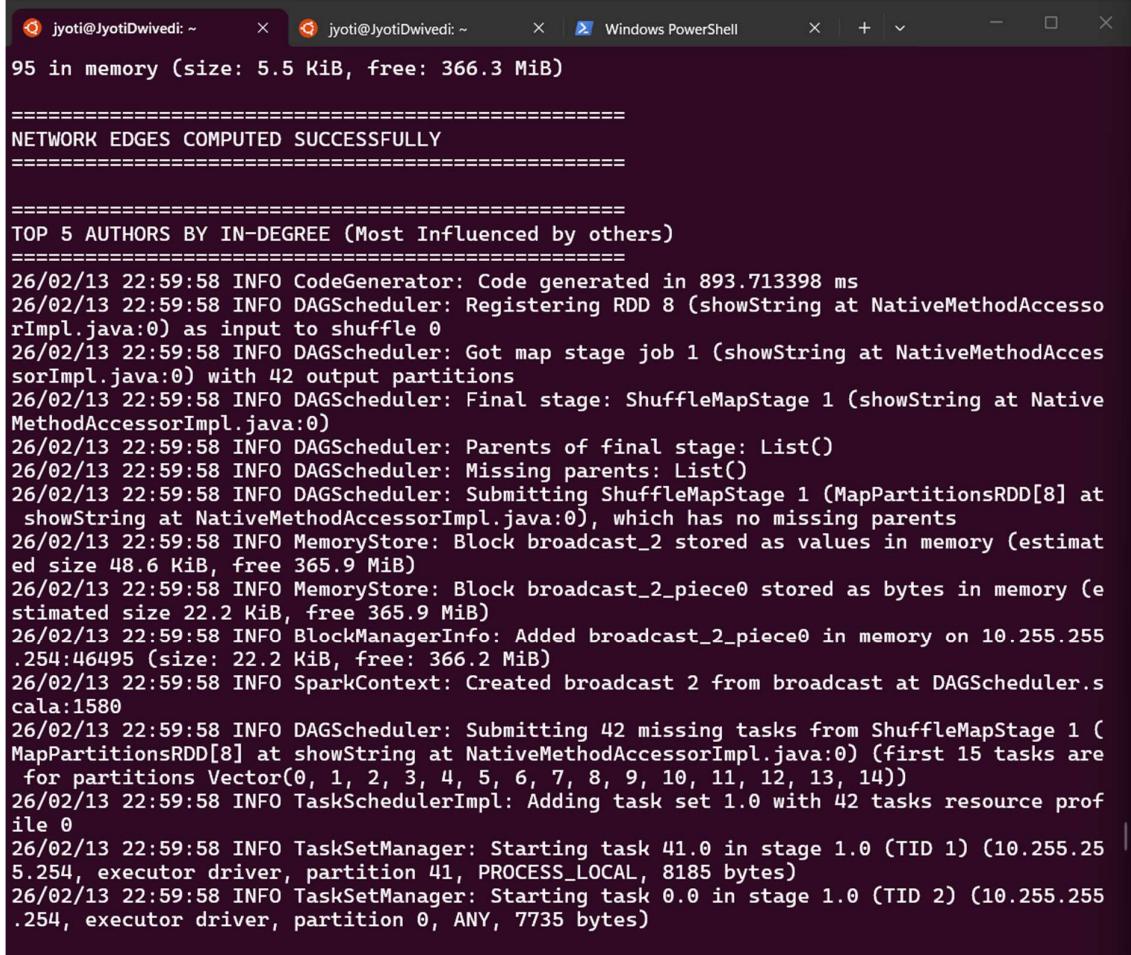
```
jyoti@JyotiDwivedi: ~          jyoti@JyotiDwivedi: ~          Windows PowerShell          - + v
ished
26/02/13 22:11:58 INFO DAGScheduler: Job 3 finished: takeOrdered at /home/jyoti/CSL7110_A
ssignment/spark_q11.py:403, took 31.908853 s
=====
SIMILARITY SCORES TO 10.txt
=====
Book: 109.txt | Score: 0.3453
Book: 124.txt | Score: 0.3279
Book: 110.txt | Score: 0.0979
Book: 108.txt | Score: 0.0876
Book: 123.txt | Score: 0.0667
Book: 122.txt | Score: 0.0654
Book: 125.txt | Score: 0.0619
Book: 102.txt | Score: 0.0581
Book: 107.txt | Score: 0.0580
Book: 113.txt | Score: 0.0572
Book: 112.txt | Score: 0.0567
Book: 101.txt | Score: 0.0522
Book: 126.txt | Score: 0.0514
Book: 121.txt | Score: 0.0476
Book: 12.txt | Score: 0.0457
Book: 11.txt | Score: 0.0440
Book: 104.txt | Score: 0.0409
Book: 106.txt | Score: 0.0342
Book: 105.txt | Score: 0.0331
Book: 118.txt | Score: 0.0312
Book: 103.txt | Score: 0.0273
Book: 111.txt | Score: 0.0257
Book: 114.txt | Score: 0.0082
Book: 117.txt | Score: 0.0032
Book: 115.txt | Score: 0.0009
26/02/13 22:11:58 INFO SparkContext: SparkContext is stopping with exitCode 0.
26/02/13 22:11:58 INFO SparkUI: Stopped Spark web UI at http://10.255.255.254:4040
26/02/13 22:11:58 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint sto
pped!
26/02/13 22:11:58 INFO MemoryStore: MemoryStore cleared
```

Figure 23 - Cosine similarity for 25 books

## Que -12: Author Influence Network

Output:

- Top In-Degree: Robert Louis Stevenson (584), Thomas Hardy (461), Arthur Conan Doyle (449)
- Top Out-Degree: Thomas Hardy (644), Robert Louis Stevenson (504), Edgar Rice Burroughs (470)



```
95 in memory (size: 5.5 KiB, free: 366.3 MiB)
=====
NETWORK EDGES COMPUTED SUCCESSFULLY
=====

=====
TOP 5 AUTHORS BY IN-DEGREE (Most Influenced by others)
=====

26/02/13 22:59:58 INFO CodeGenerator: Code generated in 893.713398 ms
26/02/13 22:59:58 INFO DAGScheduler: Registering RDD 8 (showString at NativeMethodAccesso
rImpl.java:0) as input to shuffle 0
26/02/13 22:59:58 INFO DAGScheduler: Got map stage job 1 (showString at NativeMethodAcces
sorImpl.java:0) with 42 output partitions
26/02/13 22:59:58 INFO DAGScheduler: Final stage: ShuffleMapStage 1 (showString at Native
MethodAccessorImpl.java:0)
26/02/13 22:59:58 INFO DAGScheduler: Parents of final stage: List()
26/02/13 22:59:58 INFO DAGScheduler: Missing parents: List()
26/02/13 22:59:58 INFO DAGScheduler: Submitting ShuffleMapStage 1 (MapPartitionsRDD[8] at
showString at NativeMethodAccessorImpl.java:0), which has no missing parents
26/02/13 22:59:58 INFO MemoryStore: Block broadcast_2 stored as values in memory (estimat
ed size 48.6 KiB, free 365.9 MiB)
26/02/13 22:59:58 INFO MemoryStore: Block broadcast_2_piece0 stored as bytes in memory (e
stimated size 22.2 KiB, free 365.9 MiB)
26/02/13 22:59:58 INFO BlockManagerInfo: Added broadcast_2_piece0 in memory on 10.255.255
.254:46495 (size: 22.2 KiB, free: 366.2 MiB)
26/02/13 22:59:58 INFO SparkContext: Created broadcast 2 from broadcast at DAGScheduler.s
cala:1580
26/02/13 22:59:58 INFO DAGScheduler: Submitting 42 missing tasks from ShuffleMapStage 1 (
MapPartitionsRDD[8] at showString at NativeMethodAccessorImpl.java:0) (first 15 tasks are
for partitions Vector(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14))
26/02/13 22:59:58 INFO TaskSchedulerImpl: Adding task set 1.0 with 42 tasks resource prof
ile 0
26/02/13 22:59:58 INFO TaskSetManager: Starting task 41.0 in stage 1.0 (TID 1) (10.255.25
5.254, executor driver, partition 41, PROCESS_LOCAL, 8185 bytes)
26/02/13 22:59:58 INFO TaskSetManager: Starting task 0.0 in stage 1.0 (TID 2) (10.255.255
.254, executor driver, partition 0, ANY, 7735 bytes)
```

Figure 24- Network Edges

```
jyoti@JyotiDwivedi: ~ jyoti@JyotiDwivedi: ~ × | Windows PowerShell × + - □ ×
leted, from pool
26/02/13 23:00:10 INFO DAGScheduler: ResultStage 10 (showString at NativeMethodAccessorImpl.java:0) finished in 0.136 s
26/02/13 23:00:10 INFO DAGScheduler: Job 5 is finished. Cancelling potential speculative or zombie tasks for this job
26/02/13 23:00:10 INFO TaskSchedulerImpl: Killing all running tasks in stage 10: Stage finished
26/02/13 23:00:10 INFO DAGScheduler: Job 5 finished: showString at NativeMethodAccessorImpl.java:0, took 0.158589 s
26/02/13 23:00:10 INFO CodeGenerator: Code generated in 14.897923 ms
26/02/13 23:00:11 INFO BlockManagerInfo: Removed broadcast_6_piece0 on 10.255.255.254:46495 in memory (size: 34.6 KiB, free: 366.1 MiB)
26/02/13 23:00:11 INFO BlockManagerInfo: Removed broadcast_5_piece0 on 10.255.255.254:46495 in memory (size: 27.6 KiB, free: 366.1 MiB)
26/02/13 23:00:11 INFO BlockManagerInfo: Removed broadcast_7_piece0 on 10.255.255.254:46495 in memory (size: 34.8 KiB, free: 366.2 MiB)
26/02/13 23:00:16 INFO CodeGenerator: Code generated in 10.586723 ms
+-----+-----+
|influenced |in_degree|
+-----+-----+
|Robert Louis Stevenson |584
|Thomas Hardy |461
|Arthur Conan Doyle |449
|Henry James |438
|Frances Hodgson Burnett|341
+-----+-----+
only showing top 5 rows

=====
TOP 5 AUTHORS BY OUT-DEGREE (Most Influential to others)
=====
26/02/13 23:00:16 INFO DAGScheduler: Registering RDD 29 (showString at NativeMethodAccesso
rImpl.java:0) as input to shuffle 2
26/02/13 23:00:16 INFO DAGScheduler: Got map stage job 6 (showString at NativeMethodAccesso
rImpl.java:0) with 1 output partitions
26/02/13 23:00:16 INFO DAGScheduler: Final stage: ShuffleMapStage 12 (showString at Nativ
```

Figure 25- in degree of authors

```

jyoti@JyotiDwivedi: ~      jyoti@JyotiDwivedi: ~      Windows PowerShell
26/02/13 23:00:17 INFO ShuffleBlockFetcherIterator: Started 0 remote fetches in 1 ms
26/02/13 23:00:17 INFO Executor: Finished task 0.0 in stage 15.0 (TID 48). 8594 bytes res-
ult sent to driver
26/02/13 23:00:17 INFO TaskSetManager: Finished task 0.0 in stage 15.0 (TID 48) in 39 ms
on 10.255.255.254 (executor driver) (1/1)
26/02/13 23:00:17 INFO TaskSchedulerImpl: Removed TaskSet 15.0, whose tasks have all comp-
leted, from pool
26/02/13 23:00:17 INFO DAGScheduler: ResultStage 15 (showString at NativeMethodAccessorIm-
pl.java:0) finished in 0.062 s
26/02/13 23:00:17 INFO DAGScheduler: Job 7 is finished. Cancelling potential speculative
or zombie tasks for this job
26/02/13 23:00:17 INFO TaskSchedulerImpl: Killing all running tasks in stage 15: Stage fi-
nished
26/02/13 23:00:17 INFO DAGScheduler: Job 7 finished: showString at NativeMethodAccessorIm-
pl.java:0, took 0.081778 s
+-----+-----+
|influencer          |out_degree|
+-----+-----+
|Thomas Hardy          |644   |
|Robert Louis Stevenson|504   |
|Edgar Rice Burroughs |470   |
|John Milton           |466   |
|Lucy Maud Montgomery |387   |
+-----+-----+
only showing top 5 rows

26/02/13 23:00:17 INFO SparkContext: SparkContext is stopping with exitCode 0.
26/02/13 23:00:17 INFO SparkUI: Stopped Spark web UI at http://10.255.255.254:4040
26/02/13 23:00:17 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint sto-
pped!
26/02/13 23:00:17 INFO MemoryStore: MemoryStore cleared
26/02/13 23:00:17 INFO BlockManager: BlockManager stopped
26/02/13 23:00:17 INFO BlockManagerMaster: BlockManagerMaster stopped
26/02/13 23:00:17 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCom-
mitCoordinator stopped!
26/02/13 23:00:17 INFO SparkContext: Successfully stopped SparkContext
26/02/13 23:00:19 INFO ShutdownHookManager: Shutdown hook called

```

Figure 26- outdegree of authors

#### Analysis:

- **Spark Representation:** The network was represented as a DataFrame edge-list. DataFrames leverage the Catalyst Optimizer, making relational operations (the Theta self-join) incredibly fast. However, DataFrames lack native graph traversal capabilities (like PageRank).
- **Time Window (X):** A small X yields a sparse graph of strict contemporaries, while a large X creates a dense graph capturing generational influence, albeit risking spurious edges.
- **Limitations:** Temporal correlation does not equal causation. True influence requires semantic overlap, and Gutenberg "Release Dates" generally reflect digitization metadata rather than historical publication reality.
- **Scaling to Millions of Books:** The  $(a2.year - a1.year) \leq 5$  logic creates a Theta-Join. In a dataset of millions, this demands a full Cartesian-like shuffle, causing memory crashes. To optimize, Time Bucketing (Windowing) must be introduced. By assigning authors to a "Decade" bucket, Spark partitions the data to join only adjacent buckets, eliminating the  $O(N^2)$  global cluster shuffle.