**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.](http://jmcauley.ucsd.edu/data/amazon/)

[Links to an external site.](http://jmcauley.ucsd.edu/data/amazon/)

**Data source:**

**I choose All beauty product category from link below** <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 Intelli J IDEA IDE to achieve task 1 and task 2 as we can add dependencies directly using Maven.**

**Implementation**

**Mapper Class**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**Reducer Class**

**A screenshot of a computer

Description automatically generated**

**Connect Class**

**A screenshot of a computer

Description automatically generated**

**Output:**

**A screenshot of a computer

Description automatically generated**

**Seeing results in UI**

**A screenshot of a computer

Description automatically generated**

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

**A screenshot of a computer

Description automatically generated**

**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.**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**Reducer Class**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**Connect Class**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**Output:**

**A screenshot of a computer

Description automatically generated**

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

**A screenshot of a computer

Description automatically generated**

**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 3rd argument from the user input(input - 1st argument, output-2nd argument, N – 3rd 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);**

**A screenshot of a computer program

Description automatically generated**