**Assignment Three**

Leslie Manrique

Predictive Analytics

Special Topics Section 011

Fall 2016

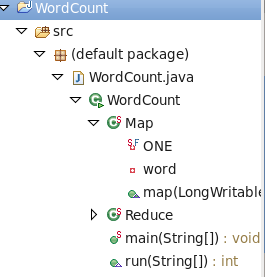
Professor Anasse Bari, Ph.D

November 28, 2016

**0. Word Count in Eclipse under Hadoop**

1. Snapshot of the WordCount.java in Eclipse Project

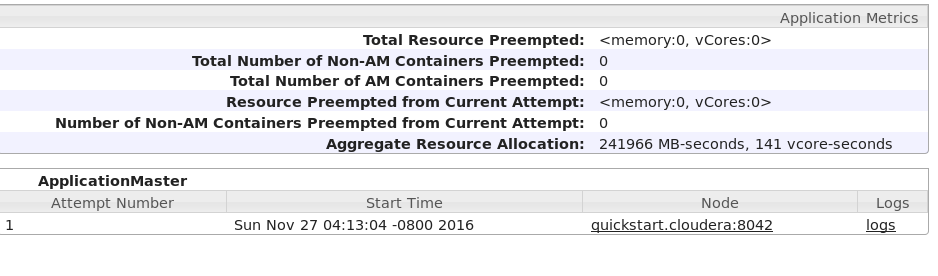
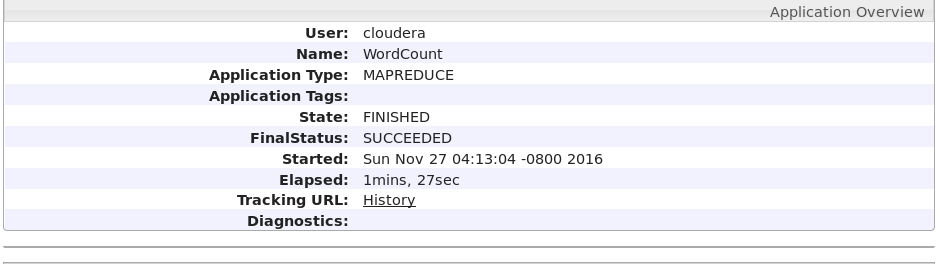
I followed the tutorial provided in order to refresh myself on using the eclipse environment. It was pretty straight forward but I was not sure how to view my mappers and reducers on <http://localhost:8080>.



1. Snapshot of the Mappers and Reducers on <http://localhost:8080>

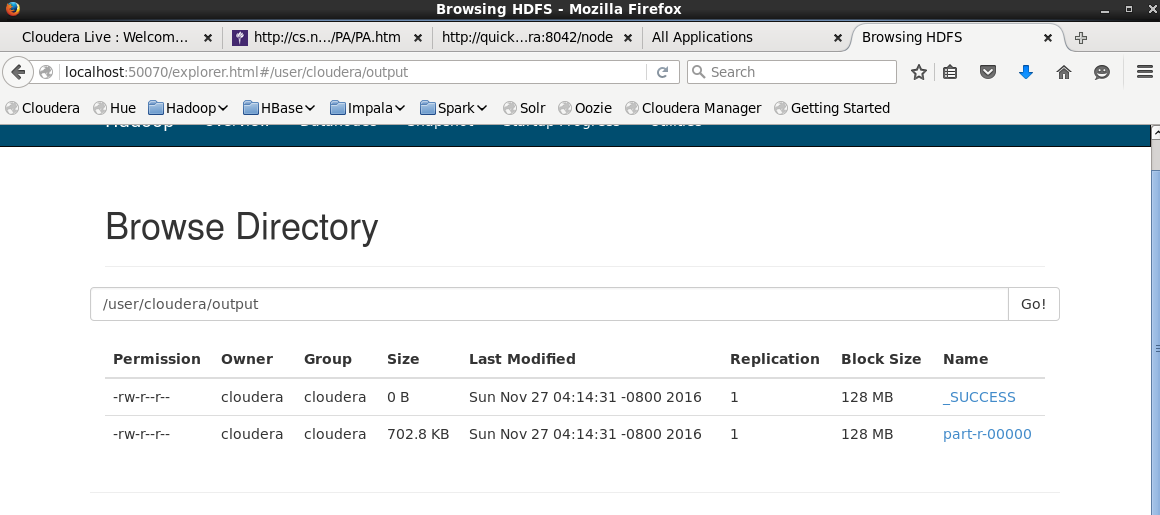
In Order to accomplish this I had to make my eclipse project into a jar. I copied my document.txt into the hadoop file system using terminal and running the following command

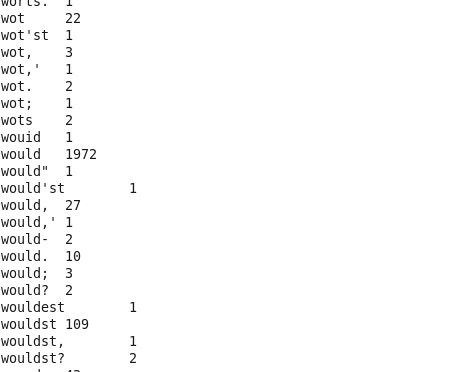
hadoop fs -put document.txt document.txt . On terminal, again, I went to my eclipse workspace and found my wordcount.jar file. I ran the command hadoop jar wordcount.jar WordCount document.txt output. This created an output folder in the hadoop fs, which can be viewed using hadoop fs -ls. I could then go onto <http://localhost:8080> to view my dashboard that includes the Word Count task.



1. Snapshot of the full output of the WordCount

I was able to browse the hdfs by typing on the browser <http://localhost:50070> then I downloaded the file called part-r-00000, which are the results of the wordcount. This can also be found locally on eclipse.

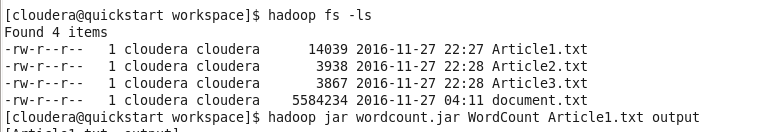
 Terminal output: $ hdfs dfs -cat /user/cloudera/output/part-r-00000



1. I repeated the process for Article1.txt, Article2.txt and Article3.txt

Below are my results.

**Article1.txt**

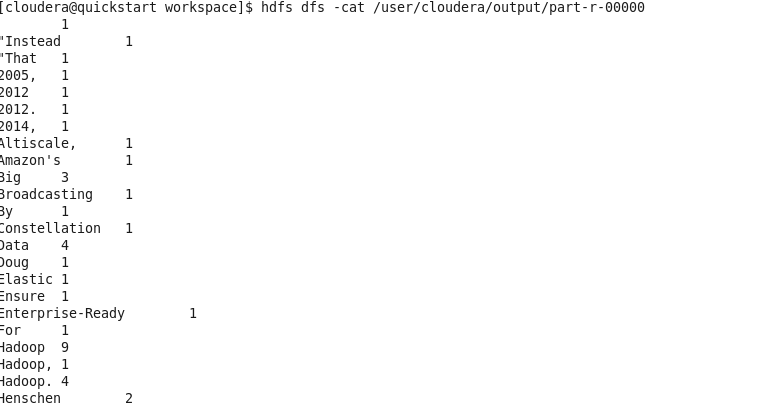
Run:

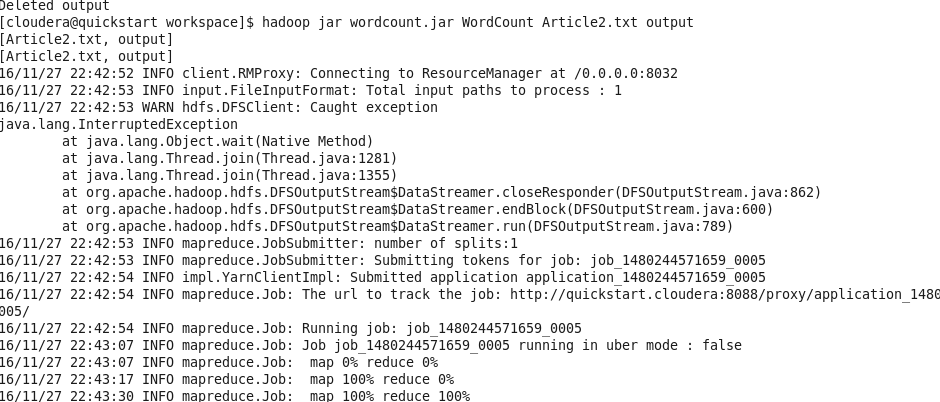
Output:

Note: This input seems to be an encoding that cannot be read and therefore the results are the following.

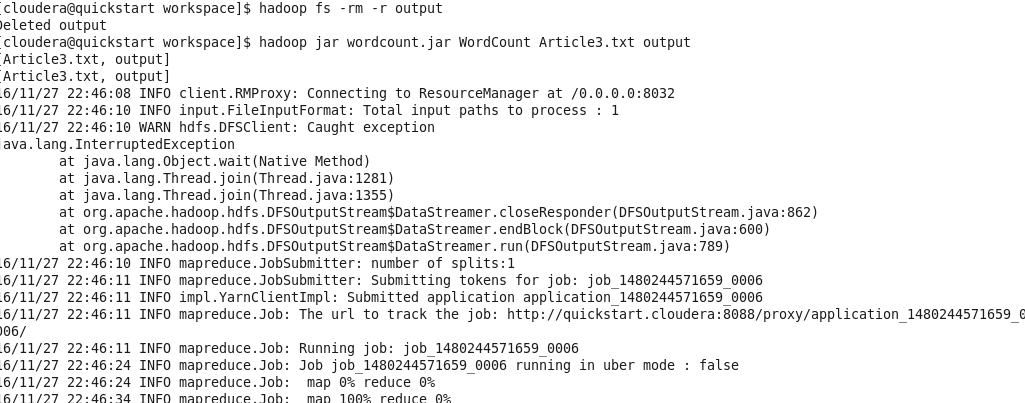


**Article2.txt**

**Run:**

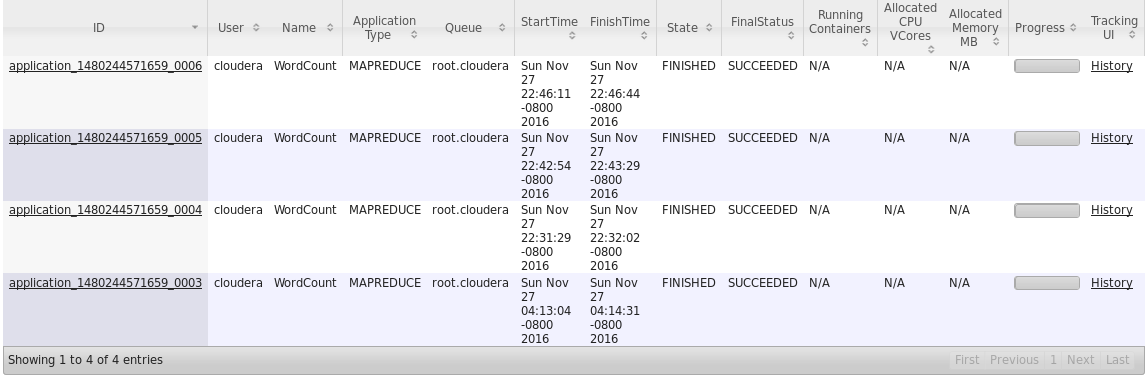
**Output:** 

**Article3.txt**

Run:

Output:

**Localhost:8080**



**1. Targeted Advertising using MapReduce**

Function Descriptions:

1. **Mapper**

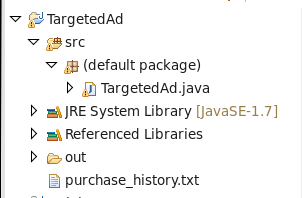
My map function splits each line into a user id and an array of categories.

For each category in the category list, I add it to a List<Object>. This now gets processed into a function getSubsetUsingBitMap(List<Object> list), that returns all possible combinations of categories, for my list. I then create a string of each subset and then transform the string into a Text object as my key. The uid will be the value: context.write(cat, new Text(uid)).

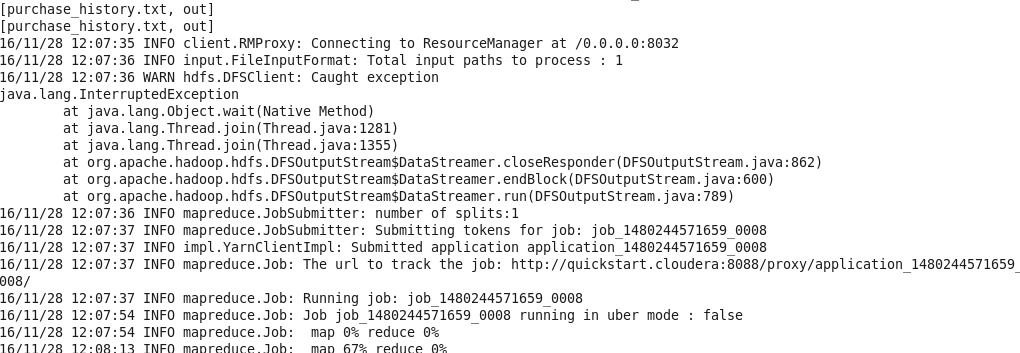
1. **Reducer**

The reduce function creates a hashmap for each key. The hasmap converts all values of the key (a user id) to a hashmap key and for the first occurrence, it sets the value to 1. For all other occurences, it will increment the value by 1 for that user id. Once the hashmap is complete, it creates a string of all the key-value pairs. So it’s in the form: key(value),key (value). The string then gets transformed into a Texy object and written to the output using context.write(key, new Text(res));

1. **Images**
   1. File structure in eclipse

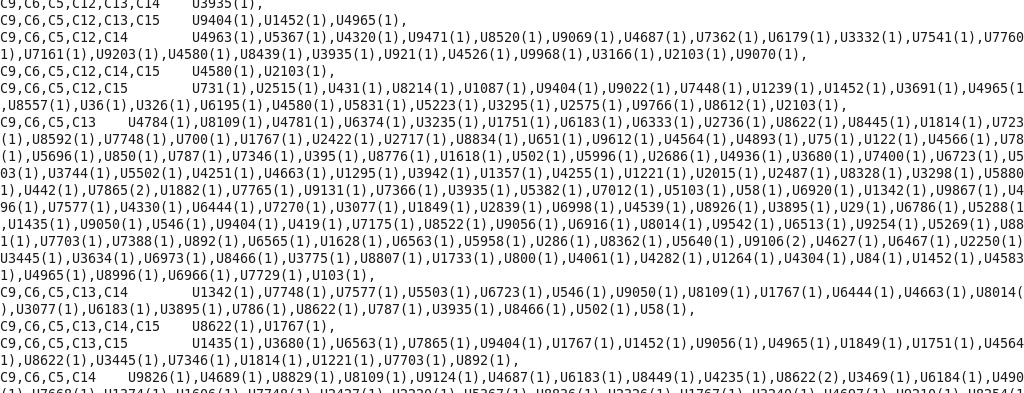


* 1. Running on Hadoop terminal



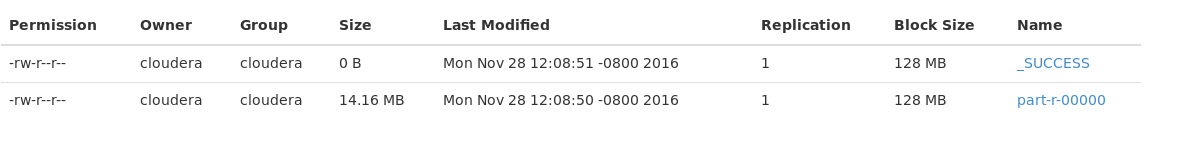
* 1. Output





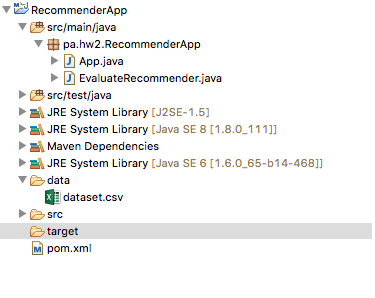
* 1. Localhost:8088



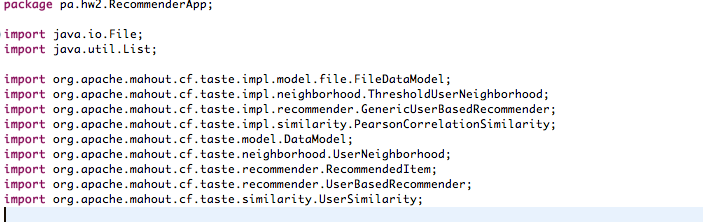
* 1. <http://localhost:50070/explorer.html#/user/cloudera/out>

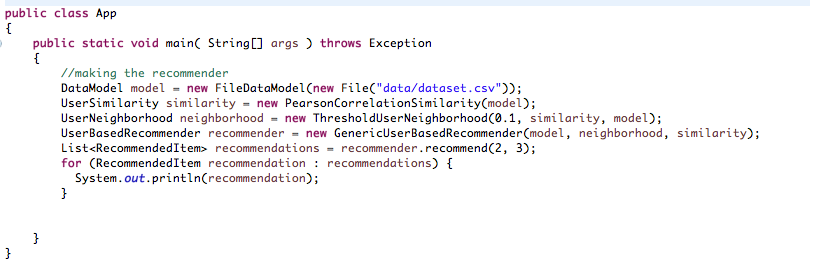
**2. Recommender System using Apache Mahout**

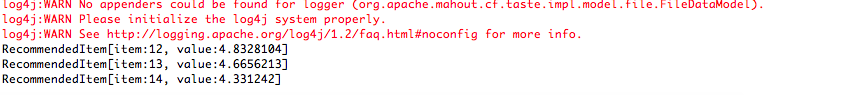
a) Eclipse Project Structure



b) App.java







b) EvaluateRecommender.java

