Twitter Hashtag Filtering Using Hadoop

1. Introduction

The program is used to find the recent popular hashtag by implementing decaying window algorithm in Hadoop by map-reduce. It uses the data "**2016 United States Presidential Election Tweet Ids**", provided by The President & Fellows of Harvard College. I have used 30 thousand data as a sample but it can be downloaded from https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/PDI7IN.

2. Environment Setup

I have used Hadoop 2.9.1 and Eclipse together for this project. The other dependencies are the Java Development Kit (JDK) and Java Runtime Environment (JRE) as well as JAR files of Hadoop for Eclipse.

3. About the code

The program uses three jobs to accomplish the task in Hadoop. The role of each jobs is:

a) Job 1: It is the job that is responsible for finding a set of unique dates. It extracts all the dates from the input file and put it in a set. It can further be understood by the role of mapper and reducer:
Role of mapper: Grabs all the dates from the input file and turn it into <key, value> pair. Key represents the date and value represents "1" for each time the date occurs.
Role of reducer: It takes the <key, value> pair from the mapper and groups the same key (i.e. same date). So, the output of this reducer will be a set of unique dates.

b) Job 2: This job is responsible for finding all the dates that a hashtag occurs. For example: if #book occurs in date 7/12/2018 and 7/14/2018 then the output will be ***#book*** as key and ***7/12/2018, 7/14/2018*** as value. Let's look as its mapper and reducer.
Role of mapper: Grabs hashtag as key and date of the hashtag as value from the input file.
Role of reducer: It concatenates all the dates of a hashtag by comma. Then, the hashtag becomes the key and the concatenated dates become the value for this reducer.

c) Job 3: This is the main job of the program which is responsible for providing the score of the hashtag by decaying window algorithm. We can dive into its details by looking at its mapper and reducer.
Role of Mapper: It takes input from the path where the **Job 2** stores its output. So, it has hashtag as the key and concatenated date as the value. Along with this input it creates a set to store all the unique dates collected by **Job 1**, this is done in the setup step of the mapper. In the main function of the mapper, we will run an outer loop which runs over all the set of dates and an inner loop which runs for all the dates of the hashtag. Inside the inner loop, we will provide score to the hashtag by checking if the hashtag has occurred in the date provided by the set of all dates. If the hashtag does not occur in the date, which we are currently looking at, then its score is decayed by a factor of (1-c) but if it occurs, then the score is incremented with its frequency along with decaying the score by (1-c). 'c' is the decaying factor which is a very small number such as $1/10^6$. Mathematically,

If the hashtag occurs in the date that we are currently looking in:

$$\mathbf{score(x)} = \mathbf{score(x)} \cdot (\mathbf{1} - \boldsymbol{c}) + \boldsymbol{frequency(x) at\ current\ time}$$

else

$$\mathbf{score(x)} = \mathbf{score(x)} \cdot (\mathbf{1} - \boldsymbol{c})$$

where 'c' is a small positive number such as $1/10^6$. 'c' is known as the decaying factor.
Its output will be score of a hashtag as key and the hashtag as value.
Role of Reducer: The Hadoop internally runs shuffle and sort between the mapper and reducer phase. So, all the hashtags are sorted based on their score before being received at reducer. So, the only task left is to output the hashtags and their scores. This is done by this reducer.

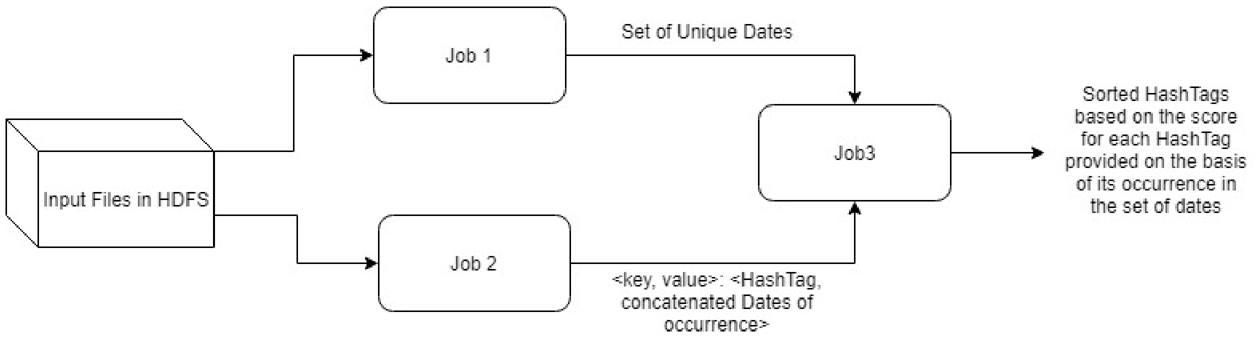
Job 1
Set of Unique Dates
Input Files in HDFS
Job3
Sorted HashTags based on the score for each HashTag provided on the basis of its occurrence in the set of dates
Job 2
<key, value>: <HashTag, concatenated Dates of occurrence>

Fig: Explanation of how all the three jobs runs in coordination

4. **Steps to run the code**

After the Hadoop has been downloaded and configured properly in pseudo distributed environment as well as JDK and JRE has been installed and all the environment files has been setup, we can use following step to execute the code:

a) For making input directory in hdfs, run the command:
```
hadoop fs -mkdir /input_dir
```
b) For deleting all the previous output and intermediate output files:
```
hadoop fs -rm -r /output*
```
c) Set the classpath for creating jar file:
```
set HADOOP_CLASSPATH=%JAVA_HOME%\lib\tools.jar
```
d) For copying input files to the hdfs, run the command:
```
hadoop fs -put democratic-candidate-timelines.csv /input_dir
```
e) For creating the jar file of the java code:
```
hadoop com.sun.tools.javac.Main HashTagFiltering.java
jar cf htFilter.jar HashTagFiltering*.class
```
f) For executing the file:
```
hadoop jar htFilter.jar HashTagFiltering /input_dir /output_dir
```
g) Finally, for reading the output:
```
hadoop dfs -cat /output_dir /part-r-00000
```

5. **Important Links**

a) Hadoop Download Link:
http://www.apache.org/dyn/closer.cgi/hadoop/common/hadoop-2.9.1/hadoop-2.9.1.tar.gz

b) JRE and JDK download link: https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

c) To download Hadoop jar(s): https://mvnrepository.com/artifact/org.apache.hadoop