Assignment 7

Tasks:

- Study Hadoop and MapReduce.
- Write the record-reader routine, to read a csv file and extract information (as mentioned in the following sections).
- Write the Mapper, Combiner and Reducer routines (those necessary) to implement the queries mentioned in the following sections.
- Write the output-format routine to print results of queries to a text file. Write the code for these queries in python.

Dataset:

The dataset is a csv file which contain information about songs. Each line contains information about one song. The following fields are available:

● List of artists who sang the song (artists separated by semicolon (;)).

●Tags: list of tags for the song. Example: “New York”, “classic”, “rock” etc. The tags are based on both genre and lyrics of the song (tags separated by semicolon (;)).

● Title: title of the song.

● Song-id: alphanumeric format.

To download the dataset, use

https://drive.google.com/open?id=1OYigN9Dt6aBCdZxB-7NxPw7tCkUcrZa8

Information to extract:

From the Input file, extract names of the artists, title and tags of the song. Group all this information under the song-id. This work is to be done in the Record-reader phase.

Queries: The queries are to be implemented in Mapper, Combiner and Reducer phases. Some of them may be empty based on the query.
Here are some simple queries to get you started.

- Print titles and ids of the songs which have more than NUM_TAGs tags.
- Print names of the artists, along with song names and song ids, who have sung of more Than NUM_SONGS songs. If a song is sung by multiple artists together, consider it separately for each artist

You need to implement these following queries in the assignment.

- **Query 1:** Report all the artists who have sung songs with the artist having maximum number of songs.
- **Query 2:** Report the 10 most frequently used tags along with their frequencies.
- **Query 3:** Report the artist pairs who have sung more than NUM_SONGS together having at least NUM_TAGS tags.
- **Query 4:** In this query we develop an index for artist names. I.e. given an artist name you shall be able to retrieve names of all the songs sung by that artist. Example: on input "Arjit Singh", print all the songs sung by Arjit Singh.

  Important, you shall not search the whole dataset to find the songs sung by Arjit Singh

  Write code to construct an index for artist names. Corresponding to an artist, store all the songs sung by him/her. Since it is an index on artist names, it shall be sorted on artist names. On a query search in this index and print the results.

  Also print the whole index to a separate file once. An index like this is called inverted index.

Deliverables:

Python code for above functionality compressed as .tar.gz, named A7_<ROLL_NO>.tar.gz. All

your code must be inside a folder called A7_<ROLL_NO>.

Strictly adhere to the naming convention above. Submissions not following the above will attract a late submission penalty!!

****

Tutorial to solve a problem using map reduce

Map Reduce is just a different way of solving problems. Let us start with a simple word count problem. Say, we have a text file and we want to count the frequency of occurrence of each word. The tutorial below explains how to solve this problem using a map reduce algorithm.

http://www.michael-noll.com/tutorials/writing-an-hadoop-mapreduce-program-in-python/

We have also attached a series of tutorials for a more in-depth explanation of map reduce.

****

How to run and test your code

Since we do not have access to a hadoop cluster, we will be testing our codes on a linux system as follows:

```
cat input.txt | python mapper.py | sort | python reducer.py
```

Explanation of the above commands:

"cat" is a linux command to print the contents of a file on console.

You must be familiar with the pipe operator (|). It directs the output of previous command to next command.

"sort" is a linux command to sort the input lexicographically.

****

LOGIC:

Hadoop has following phases:

1. record reader. // Reading and parsing the input to records.
2. map. // Execute an operation on each record.
3. combiner. // We do not need to code this.
4. partitioner. // We do not need to code this.
5. reduce. // Combine the results from the combiner.
6. Output format.

The assignment covers all the user editable phases along with three important uses of Hadoop i.e.

- Filtering (finding artists with at least 3 vowels in their names)
- Numerical summarizations (Counting number of songs of an artist)
- Indexing