SENG 501.5
Assignment 2
Marks: 20
This assignment is worth 5% of your course grades
Due in lab Wednesday November 1, 2017

1) Matrix transpose. (Marks: 3)

Using the MapReduce approach, write code that will perform the transpose of a large matrix. Assume that the input matrix is stored in text files with each line of the text file containing a record with the following format:

Row_index,Column_index,value

Both *Row_index* and *Column_index* should start with 0. The output from the MapReduce job should also have records with the above format. Test your code using Hadoop streaming and specifying more than one reducer. The matrix you need to transpose can be found in HDFS at **/user/instructor/matrix/matrix.txt**

2) Parallel Breadth First Search (Marks: 7)

Using the MapReduce approach, implement the parallel breadth first search algorithm that we discussed in class. For testing use, the following input graph:

1<tab>2,3|0|GRAY|source
2<tab>1,3,4,5|Integer.MAX_VALUE|WHITE|null
3<tab>1,4,2|Integer.MAX_VALUE|WHITE|null
4<tab>2,3|Integer.MAX_VALUE|WHITE|null
5<tab>2|Integer.MAX_VALUE|WHITE|null

The format of each line of the input is as follows:

source<tab>adjacency_list|distance_from_the_source|color|parentNode

Run a series of jobs using Hadoop streaming to figure out the distance of each node from the source and the shortest path from each node to the source. Use more than one reducer in all your jobs.

Hint: To fully automate this job you may need to play around a bit with the "counters" feature of Hadoop. In particular, lookup the section "how do I update counters in streaming applications?" of the Hadoop streaming documentation.

3) Normalized word co-occurrence matrix (Marks: 10)

Finding co-occurring terms in documents is a common task in text processing. In this problem we will consider an example related to a large retail store. The store wants to analyze point of sales records to figure out correlated product purchases, e.g., customers who buy *this* item are also likely to buy *that* item. This information can then be used for inventory management and stocking items in shelves.

Assume that the store has a vast collection of point of sale purchase records. Each record consists of a list of items checked out by a shopper. As a sample, consider the following three records:

cheese bread milk
candy milk coke
milk coffee eggs cheese

Given such records, one can characterize correlations using what is called as a normalized co-occurrence matrix.

$$f(w_j|w_i) = \frac{N(w_i, w_j)}{\sum_{w`} N(w_i, w`)}$$

In the above equation, $f(w_j|w_i)$ indicates the proportion of time item w_j occurs with item w_i. *N(.,.)* indicates the number of times a particular co-occurring pair occurs in the collection of records. The denominator represents the number of times *any* item occurs with item w_i. It is used to prevent pairs that get a high score simply because one of the items in that pair occurs very commonly, e.g., milk in a grocery store checkout. A high value of $f(w_j|w_i)$ indicates that there is a very high likelihood that a customer that purchases $w_j$will likely also purchase w_i.

For the example records listed above, the terms needed to calculate the entries of the normalized co-occurrence matrix would be as follows:

N(cheese, bread) = 1; N(cheese, milk) = 2; N(bread, cheese) = 1; N(bread, milk) = 1; N(milk, cheese) = 2; N(milk, bread) =1; N(candy, milk) = 1; N(candy, coke) = 1; N(milk, candy) = 1; N(milk, coke) = 1; N(coke, candy) = 1; N(coke, milk) = 1; N(milk, coffee) = 1; N(milk, eggs) =1; N(coffee, milk) = 1; N(coffee, eggs) = 1; N(coffee, cheese) = 1; N(eggs, milk)= 1; N(eggs, coffee) = 1; N(eggs, cheese) = 1; N(cheese, coffee) = 1; N(cheese, eggs)=1;

N(cheese,) = 5 ; N(bread,) = 2; N(milk,) = 7; N(candy,) = 2; N(coke,)=2; N(coffee,) = 3; N(eggs,) = 3

Using MapReduce, compute the elements of the normalized word co-occurrence matrix. Test your solution using Hadoop streaming with multiple reducers. You must make efforts **to avoid solutions that require potentially large in-memory data structures**, e.g., a large hash table, on mappers/reducers. You should also make an effort to **avoid solutions that require more than 1 MapReduce stage**. To get such a solution, you might need to explore the ***KeyFieldBasedPartitioner***.

The input dataset for this problem is available in HDFS at **/user/dkrishna/retail/retail.dat**. This contains anonymized contents of shopper shopping carts at an anonymous Belgian retail store. More information can be found at http://fimi.ua.ac.be/data/retail.pdf.