SENG 550/ENSF 612
Assignment 1
Marks: 25
This assignment is worth 5% of your course grades
Due 11:59 PM Friday October 11, 2019

1) **Producing a random sub-sample of a big dataset using Hadoop MapReduce (Marks: 3)**

Text analytics projects often start by sampling a small fraction of data from a bigger dataset to conduct some preliminary analysis, e.g., estimating the expected skew of words in the dataset. For this problem, you will use Python and Hadoop streaming to develop a job that **randomly** selects approximately **10%** of the lines in a large input dataset and writes that result to the distributed file system for further analysis. Test your program using the "Shakespeare.txt" file used in the Hadoop tutorial.

2) **Building n-grams using Hadoop MapReduce. (Marks: 5)**

An n-gram is a contiguous sequence of n items in a sequence of text or document. Computing n-grams has many applications including natural language processing, modeling languages, and speech recognition. For example, calculating the counts for various n-grams can help an AI program predict the next word given a speech pattern.

For this problem, you will use Python and Hadoop streaming to build counts for a word-level 2-gram or a di-gram. Your program should be able to scan a set of text documents and then record all unique 2 word sequences in the document. As a final output, it should list all the unique di-grams along with their counts. Similar to the Hadoop tutorial, make sure you remove punctuations in the text document scanned and convert words to lowercase as part of your analysis.

Your program should be scalable to large datasets and hence should use MapReduce. You need to test your program using the "shakespeare.txt" file (the file containing in text the complete works of William Shakespeare) used in the Hadoop tutorial.

3) **Building an inverted index of a text corpus using Hadoop MapReduce. (Marks: 7)**

An inverted index is often used in information retrieval and search applications. An inverted index maps content, e.g., words, to locations where the content is found, e.g., file names of documents. Such an index allows search engines to quickly locate content relevant to search queries.

For this problem, you will use Python and Hadoop streaming to build an inverted index for contents in an input directory. Your program should scan all the documents in the input directory. For each file, it should map every unique word in the file to the name of that file. The final output of your program will consist of several records. The key of each record is a unique word occurring in the documents contained in the input directory. The value of each record is a list of file names

of documents that contain that word. Similar to the Hadoop tutorial, make sure you remove punctuations in the text documents scanned and convert words to lowercase as part of your analysis.

Your program should be scalable to large datasets and hence should use MapReduce.

Hint: You might need a bit of Googling to figure out the name of the file from which a mapper is getting its records.

4) Sorting using Hadoop MapReduce. (Marks: 10)

For this problem, you are going to sort words found in a large text document. Specifically, your program should scan the document and then output the words in the document in ascending order. Similar to the Hadoop tutorial, make sure you remove punctuations in the text document and convert to lower case letters as part of your analysis.

Your program should be scalable to large datasets and hence should use MapReduce and **multiple reducers**. Single reducer solutions will only get partial credit since they won't be scalable. Part of the challenge involved is to deduce the correct number of reducers that will result in a total order sort, i.e., results appearing sorted across all reducers.

Hint: You might need to read up the ***KeyFieldPartitioner*** option and examples given in the Hadoop streaming tutorial.

Deliverables: Need to individually demonstrate working code to TA (during lab hours). Need to submit code via D2L.