Midterm, Fall 2022
150 Points

Due Tuesday October 25, 11:55PM, via Brightspace
4 Questions, answer any 3, or all 4 for extra points.

1. Hadoop Map Reduce – Sampling a dataset. 50 points

Imagine you're working with a terabyte-scale dataset and you have a MapReduce application you want to test with that dataset. Running your MapReduce application against the dataset may take hours, and constantly iterating with code refinements and re-running against it isn't an optimal workflow. To solve this problem you look to sampling, which is a statistical methodology for extracting a relevant subset of a population.

In the context of MapReduce, sampling provides an opportunity to work with datasets without the overhead of having to wait for the entire dataset to be read and processed.

TO-DO – In Hadoop MapReduce code:

- The input test data for this problem is lines of text given in cookbook_text/*.txt.
- The solution should accept a parameter via command line with the number of samples that should be extracted from the input. e.g. 500
- The sampling method ***must be*** the reservoir sampling algorithm: http://en.wikipedia.org/wiki/Reservoir_sampling.
  The reservoir algorithm can be summarized as:
  - given a reservoir that can hold k samples:
    - fill the reservoir with samples until k lines are added.
    - for line k+1, pick probability p where p is a random integer number from 1 to k+1. If p is <= k, replace line at p with the new line
    - repeat for all j, where j = k+1 to the total input.

2. Spark – Language Models (in Spark)
50 points

Compute the probability of some word following any given word.

Discussion: In this problem, you will re-use the work in Homework 2 for wordcount.
If you have unigrams and bigrams, you can implement a simple Language Model that predicts the next word in a sequence of two words.

For example, to compute the probability, P, of 'york' following the word 'new', using Maximum Likelihood Estime (MLE) and ignoring out of vocabulary words (no smoothing), we can use

$$P(york|new) = \frac{count(bigrams('new\ york'))}{count(unigrams(new))}$$

As a starter, you can use the wordcount logic from Homework 2 to compute the unigrams. You will also need to compute bigrams, and the solve for the probabilities.

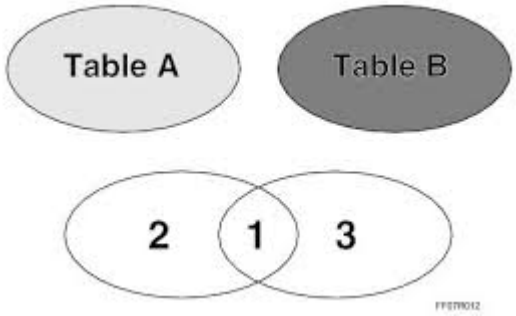
The input here is the same input as homework 2 question 4. In Jupyterhub 'shared/hw2/romeo-juliet-pg1777.txt '

The output should be the probability of each pair of words, e.g.

new|york
cat|the
cat|bad

3. Spark Bloom Filters and Broadcast Joins
50 points

Suppose you are interested in records on one dataset, Table A, based on values of another dataset, Table B. Generally, an inner join is used as a form of filtering.



Consider, for example, if Table A has 100's of millions of rows, while Table B has only a few thousands.

In cases like this, you might want to ***avoid the shuffle*** that the join operation introduces, especially if the dataset you want to use for filtering is ***significantly smaller*** than the main dataset on which you will perform your further computation.

Definition: Broadcast Join/Filter

In a broadcast join or filter, you **send the *entire*** smaller dataset, Table B, to **each** node/worker in our cluster. A join or filter in this manner is called a broadcast join/filter.

Spark Broadcast Variables:
https://spark.apache.org/docs/2.2.0/rdd-programming-guide.html#broadcast-variables

"Broadcast variables allow the programmer to keep a read-only variable cached on each machine rather than shipping a copy of it with tasks. They can be used, for example, to give every node a copy of a large input dataset in an efficient manner."

Definition: Bloom Filter
https://en.wikipedia.org/wiki/Bloom_filter

Bloom filters are efficient probabilistic data structures constructed of a set of values to be used for membership tests. It can tell you if an arbitrary element being tested **might** be in the set, or **definitely not** in the set, that is false positives are allowed, but false negatives are not.

The data structure is a bit array, onto which elements are mapped using a hash function. The mapping basically sets some bits to 1 leaving the rest as 0's. The size of the bit array is determined by how much false positives you are willing to tolerate, so most implementations accept an FPR param in the constructor of the data structure (typical value is 1%).

TO-DO – Implement a broadcast filter using Bloom filters. **in Pyspark**

Using a broadcast join and a bloom filter, filter all rows in Table A for only those containing 'model' in Table B. The common key is the '**model'** column.

You must filter Table A for the values in Table B's ***model*** column, using the Bloom Filter as the filter. (Your result may contain false positives).

You can use the Pyhton 'bloom filter' package (already installed in our environment). https://pypi.org/project/bloom-filter/

Data Source: data_Q1.2019.zip (**do not copy to JupyterHub. I will put it in the shared data folder')**

Instructions:

Table A: all files from 2019-01-01.csv through 2019-03-30.csv

Table B: 2019-03-31.csv

Reference: https://www.duedil.com/engineering/efficient-broadcast-joins-using-bloom-filters

4. Ranking over Partitions – In Spark
50 points

Compute the **top 2 and bottom 2 items sold per hour** for the problem in Homework 2, Question 1. (Breadbasket_DMS.csv)