Midterm, Fall 2024
150 Points

Due Sunday, Nov. 10, 11:59PM. via Brightspace
4 Questions, answer any 3, or all 4 for 50 extra points.

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

Imagine you're working with a terabyte-scale dataset, and you have an application you want to test with that dataset. Running your application against the dataset may not be possible due to memory constraints, or take hours or days, and constantly iterating with code refinements and re-running against it isn't an optimal workflow.

To solve this problem, you look to random sampling, which is a statistical methodology for extracting a relevant subset of a population that hopefully faithfully represents the same data distribution(s) of the original dataset.

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 HW1 and HW2 hw1text/*.txt.
- The solution should accept a parameter via command line with the number of samples that should be extracted from the input. For this problem, sampling size = 900
- 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

Discussion: If you have ***bigrams*** and ***trigrams***, you can implement a simple Language Model that predicts the third word in a three-word sequence.

For example, to compute the probability P(times/new york), that is '**times'** following the word '**new york'**, we use Maximum Likelihood Estime , MLE (ignoring out of vocabulary words, i.e. no smoothing)

Use MLE as follows:

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

The input here is the same text input as homeworks 1 and 2: hw1text/*.

TO-DO: Compute the top 10 trigrams (count) for the input text, and show the conditional probability of the third word of each trigram.

3. Ranking over Partitions – in Spark. - 50 points

Compute the **top 3 items sold per *daypart*** for the problem in Homework 2, Question 1. (Bakery.csv)

Daypart =

morning if time >= 6AM, < 11AM

noon if time >= 11AM, < 2PM

afternoon if time >= 2PM, < 5PM

evening if time >= 5PM, <6AM

The format **MUST** be in the format (example): Show ALL rows:

morning coffe, bagel, pastry

noon soda, ham, potatoes

etc.

4. Duplicate Detection with Minhash – 50 points

Minhash/LSH is an algorithm based on cryptographic hashes that computes similarity between a pair of ***entities***. It is heavily used in Web Search and product similarity applications.

Details of the algorithm can be found in Chapter 3 of *Mining Massive* Datasets.
http://infolab.stanford.edu/~ullman/mmds/ch3.pdf

For example, assume I teach a course at NYU and I trust the students to learn the material by doing original work 😉. However, I think I see too many cases of plagiarism and copying. I could use this algorithm to easily detect near-duplicates and copying so you I can fail those students.

Problem Statement

For this problem, let's use the similarity algorithm to find the most similar items.
Assume I you are reading news items in the HuffPost website (https://www.huffpost.com/) and last clicked on a new article about gun control in the USA. The next day, I return to the website and it decides to prioritize new items for me based on the last news item I read.

The Data:

In Jupyterhub: 'shared/data/huffpost.json'

Base news item short description
"Kitten Born With Twisted Arms And Legs Finds A Mom Who Knows She\u2019s Perfect"

TODO (in Spark): 75 points

Use the Minhash/LSH algorithm to find URL link, headline, category, and short description of **the 5 most similar** items to the Item above (based on the "short_description" field).
The algorithm relies on the concept of distances to define similarity. For this exercise, **use Jaccard** similarity.

NOTE:

You can use any a third party Python library like 'datasketch'.
(http://ekzhu.com/datasketch/minhash.html). Datasketch is installed in the JupyterHub environment for this class. Note, SparkML also has a Minhash/LSH library.