Big Data Midterm Solutions

Sai Nishanth Mettu

sm11326

Question1) Hadoop Map Reduce – Random Sampling a big dataset.

50 points

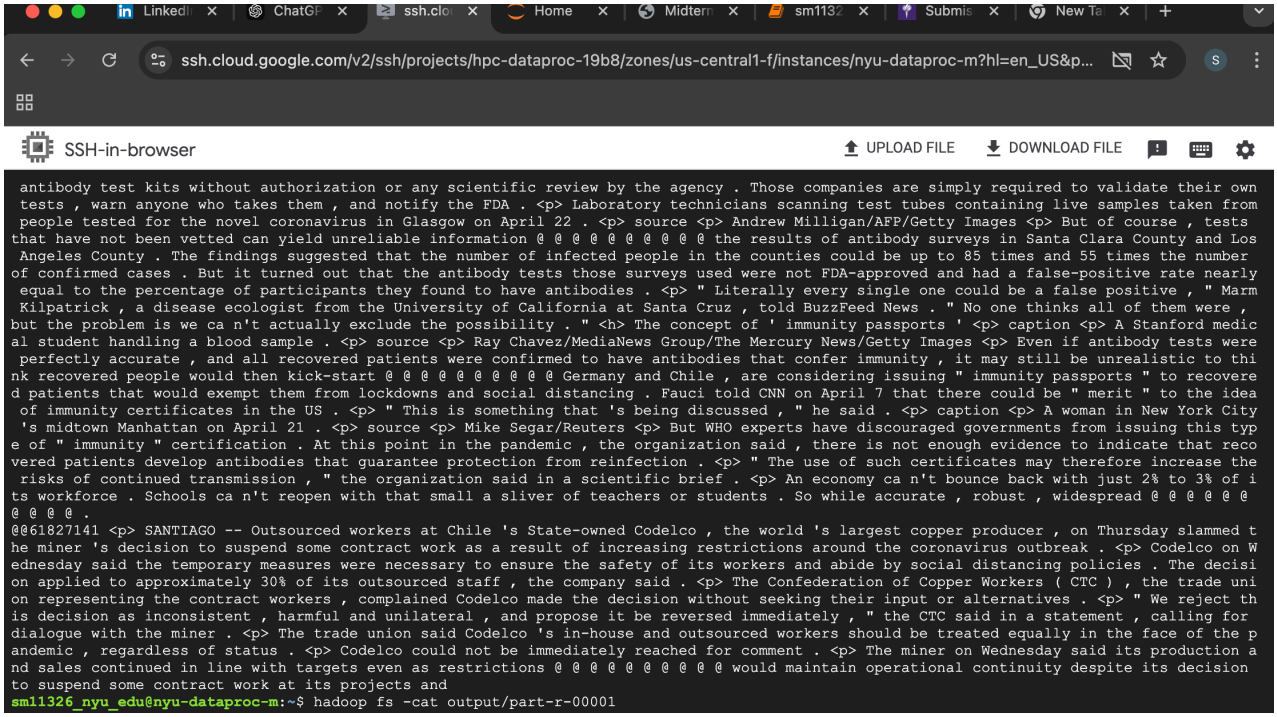
Commands to Run (depending on where you store the files, please modify the input location)

javac -classpath $(hadoop classpath) ReservoirSampling.java

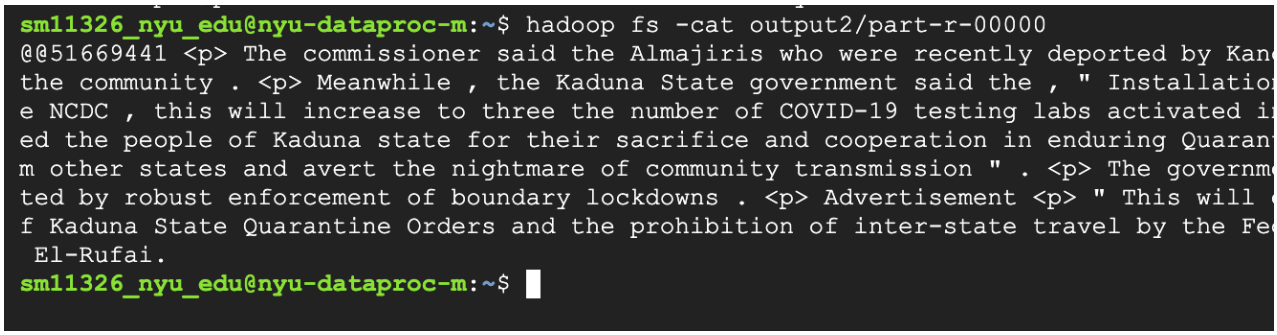
jar cf ReservoirSampling.jar ReservoirSampling*.class

hadoop jar ReservoirSampling.jar ReservoirSampling data/ output/ 900

Sampling Size 900



Sampling Size 1



Explanation – ReservoirSampling.java

In this Hadoop MapReduce program for reservoir sampling, the **Mapper** reads each input line and simply passes it as a key-value pair to the **Reducer**, where all lines are grouped under a single key ("sample"). The **Reducer** implements the reservoir sampling algorithm, which is designed to randomly select a subset of k lines (default 900) from the input data. It iterates through all the input lines, filling the reservoir until it reaches k elements, and then replacing elements in the reservoir with a certain probability as more lines are encountered. Finally, the **Driver** class configures and runs the MapReduce job, setting paths for the input and output directories, specifying the Mapper and Reducer classes, and ensuring that only one Reducer task is used (job.setNumReduceTasks(1)) for simplicity. The job then processes the input data and outputs a random sample of lines to the specified output directory.

Note : I Have the driver class, reducer class and mapper class chained within the

same ReservoirSampling.java class for ease of use.

Question 2) Spark – Language Models - in Spark - 50 points

To find the probability of the third word occurring in a trigram given that two words have already occurred,

- I am using the formula = P(w3 | w1 w2) = count(trigrams(w1 w2 w3)) / count (bigrams(w1 w2))
- I have three versions –
- Version 1 : without applying any regex or filtering and extracting meaningless trigram probabilities.

```
spark.stop()
```

```
+-----------------+-------------+----------------------+
|trigram          |trigram_count|conditional_probability|
+-----------------+-------------+----------------------+
|[@, @, @]        |113349       |0.8886563021850084     |
|[., <p>, "]      |7959         |0.16716022934912733    |
|[., <p>, The]    |6287         |0.13204376955873395    |
|[., ", <p>]      |4810         |0.4664468580294802     |
|[said, ., <p>]   |2415         |0.5625436757512229     |
|[,, ", he]       |1487         |0.16955530216647663    |
|[<p>, ", We]     |1376         |0.15076147693656186    |
|[., <p>, In]     |1321         |0.02774452355449142    |
|[the, spread, of]|1180         |0.9161490683229814     |
|[,, ", said]     |1142         |0.13021664766248575    |
+-----------------+-------------+----------------------+
```

- Version 2 : Filtering out everything apart from [A-z a-z 0-9]

```
+---------------------------+-------------+----------------------+
|trigram                    |trigram_count|conditional_probability|
+---------------------------+-------------+----------------------+
|[the, spread, of]          |1180         |0.9161490683229814    |
|[of, the, coronavirus]     |854          |0.04978430686720298   |
|[as, well, as]             |824          |0.7285587975243147    |
|[the, number, of]          |819          |0.9457274826789839    |
|[one, of, the]             |791          |0.6233254531126872    |
|[spread, of, the]          |772          |0.5482954545454546    |
|[due, to, the]             |719          |0.42244418331374856   |
|[the, coronavirus, pandemic]|711          |0.20887191539365452   |
|[of, the, virus]           |691          |0.04028214993587501   |
|[the, end, of]             |615          |0.9057437407952872    |
+---------------------------+-------------+----------------------+
```

- Version 3 : Applying Broadcasting and making it even faster as Professor suggested in the class.

```
+---------------------------+-------------+----------------------+
|trigram                    |trigram_count|conditional_probability|
+---------------------------+-------------+----------------------+
|[the, spread, of]          |1180         |0.9161490683229814    |
|[of, the, coronavirus]     |854          |0.04978430686720298   |
|[as, well, as]             |824          |0.7285587975243147    |
|[the, number, of]          |819          |0.9457274826789839    |
|[one, of, the]             |791          |0.6233254531126872    |
|[spread, of, the]          |772          |0.5482954545454546    |
|[due, to, the]             |719          |0.42244418331374856   |
|[the, coronavirus, pandemic]|711          |0.20887191539365452   |
|[of, the, virus]           |691          |0.04028214993587501   |
|[the, end, of]             |615          |0.9057437407952872    |
+---------------------------+-------------+----------------------+
```

Code Explanation :

In this code, I am building a trigram language model using PySpark to calculate the conditional probability of a third word, given the first two words in a sequence. The input text is read from text files and split into words. I then filter out any sequences that contain non-alphanumeric words, ensuring that only valid words remain. Using explode and transform, I generate bigrams (pairs of consecutive words) and trigrams (triplets of consecutive words) from the word sequences.
I calculate the counts of each bigram and trigram, and then create prefixes for the trigrams to match them with their corresponding bigrams. After joining the trigram counts with the bigram counts, I compute the conditional probability of a trigram by dividing its count by the count of its bigram prefix. Finally, I order the trigrams by their count in descending order and display the top 10 trigrams along with their respective conditional probabilities. This gives insights into which word triplets are most likely to occur together in the text.

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

In this code, I am analyzing bakery sales data to find the top 3 items purchased during different parts of the day (morning, noon, afternoon, and evening). First, I read the data and convert the "Time" column into a timestamp, then create a new column "Daypart" to categorize each entry into one of the four time periods based on the hour of the day. After grouping the data by "Daypart" and "Item", I count the number of sales for each item within each daypart. Using a window function, I rank the items within each daypart by their sales count. I then filter to get the top 3 items per daypart and aggregate them into a list, which is displayed as the final result.

This process leverages Spark's window functions and groupings to efficiently compute the top 3 items per daypart in large datasets. The window function partitions the data by "Daypart" and orders it by the item sales count, while the rank() function helps identify the top 3 items. Finally, collect_list() is used to gather these top items for each daypart into a list, making it easy to view the most popular items for each period of the day.

```
+---------+----------------------+
|Daypart  |TopItems              |
+---------+----------------------+
|afternoon|[Coffee, Bread, Tea]  |
|evening  |[Coffee, Bread, Tea]  |
|morning  |[Coffee, Bread, Pastry]|
|noon     |[Coffee, Bread, Tea]  |
+---------+----------------------+
```

Manual Verification :

```
Top 3 Items per Daypart (Ranked):
+---------+----------------------+
|Daypart  |TopItems              |
+---------+----------------------+
|afternoon|[Coffee, Bread, Tea]  |
|evening  |[Coffee, Bread, Tea]  |
|morning  |[Coffee, Bread, Pastry]|
|noon     |[Coffee, Bread, Tea]  |
+---------+----------------------+


Verification: Raw counts for the top 3 items per daypart:
+---------+------+-----+
|Daypart  |Item  |count|
+---------+------+-----+
|afternoon|Coffee|1476 |
|afternoon|Bread |847  |
|afternoon|Tea   |566  |
|evening  |Coffee|87   |
|evening  |Bread |55   |
|evening  |Tea   |49   |
|morning  |Coffee|1615 |
|morning  |Bread |1081 |
|morning  |Pastry|453  |
|noon     |Coffee|2293 |
|noon     |Bread |1342 |
|noon     |Tea   |540  |
+---------+------+-----+
```

Question 4) : Duplicate Detection with Minhash – 50 points

Top 5 Urls - MINHASH LSH using Approx Nearest Neighbours algorithm which uses Jaccard Similarity

```
[Stage 22:=============================>                          (1 + 1) / 2]
+-----------------------------------------------------------------------------------------------------------------------------------
-------------+----------------------------------------------------------------------------------+-------------+----------------------
--------------------------------------------------------------------------------------------------+
|link
|headline                                                                          |category     |short_description
|
+-----------------------------------------------------------------------------------------------------------------------------------
-------------+----------------------------------------------------------------------------------+-------------+----------------------
--------------------------------------------------------------------------------------------------+
|https://www.huffingtonpost.com/entry/mom-hilariously-nails-what-getting-ready-looks-like-for-mothers_us_595666f0e4b0da2c73230b40
|Mom Hilariously Nails What Getting Ready Looks Like For Mothers                  |PARENTS      |"Maybe she's born with it ... Maybe
she's a tired mom who doesn't have time for this."                                     |
|https://www.huffingtonpost.com/entry/how-to-make-the-perfect-c_us_5b9dc24ee4b03a1dcc8c8503
|How to Make the Perfect Chocolate Chip Cookie                                     |FOOD & DRINK |A cookie with the perfect combinatio
n of fat, flavor, and comfort. Who needs detox?                                         |
|https://www.huffingtonpost.com/entry/andrew-garfield-lip-syncs-whitney-houston-in-epic-drag-show-act_us_592ff5d1e4b0540ffc84b79b
|Andrew Garfield Lip-Syncs Whitney Houston In Epic Drag Show Act                   |ENTERTAINMENT|With a back flip and everything.
|
|https://www.huffingtonpost.com/entry/linked-by-their-choice-to-become-single-mothers-two-women-share-their-stories_us_59b91b63e4b
0edff9717de69|Linked By Their Choice To Become Single Mothers, Two Women Share Their Stories|PARENTS      |The stories of an Idaho
mom who gave birth and a California mom who adopted.                                                |
|https://www.huffingtonpost.com/entry/carol-brady_us_5b9c5c92e4b03a1dcc7e15be
|What Carol Brady Is Really Saying (INFOGRAPHIC)                                   |PARENTING    |Here's the story of a lovely lady --
who somehow kept six kids, a husband, a housekeeper and a dog in perfect order with|
+-----------------------------------------------------------------------------------------------------------------------------------
-------------+----------------------------------------------------------------------------------+-------------+----------------------
--------------------------------------------------------------------------------------------------+
```

Method 2 : Done by manually calculating Jaccard Similarity (Used as verification for the first!)

```
--------------------------------------------------------------------------------------------------+

Top 5 items using Manual Jaccard Similarity:

+-----------------------------------------------------------------------------------------------------------------------------------
-------------+----------------------------------------------------------------------------------+-------------+----------------------
--------------------------------------------------------------------------------------------------+-------------------+
|link
|headline                                                                          |category     |short_description
|jaccard_similarity |
+-----------------------------------------------------------------------------------------------------------------------------------
-------------+----------------------------------------------------------------------------------+-------------+----------------------
--------------------------------------------------------------------------------------------------+-------------------+
|https://www.huffingtonpost.com/entry/mom-hilariously-nails-what-getting-ready-looks-like-for-mothers_us_595666f0e4b0da2c73230b40
|Mom Hilariously Nails What Getting Ready Looks Like For Mothers                  |PARENTS      |"Maybe she's born with it ... Maybe
she's a tired mom who doesn't have time for this."                                     |0.25               |
|https://www.huffingtonpost.com/entry/how-to-make-the-perfect-c_us_5b9dc24ee4b03a1dcc8c8503
|How to Make the Perfect Chocolate Chip Cookie                                     |FOOD & DRINK |A cookie with the perfect combinatio
n of fat, flavor, and comfort. Who needs detox?                                         |0.21739130434782608|
|https://www.huffingtonpost.com/entry/andrew-garfield-lip-syncs-whitney-houston-in-epic-drag-show-act_us_592ff5d1e4b0540ffc84b79b
|Andrew Garfield Lip-Syncs Whitney Houston In Epic Drag Show Act                   |ENTERTAINMENT|With a back flip and everything.
|0.17647058823529413|
|https://www.huffingtonpost.com/entry/linked-by-their-choice-to-become-single-mothers-two-women-share-their-stories_us_59b91b63e4b
0edff9717de69|Linked By Their Choice To Become Single Mothers, Two Women Share Their Stories|PARENTS      |The stories of an Idaho
mom who gave birth and a California mom who adopted.                                                |0.17391304347826086|
|https://www.huffingtonpost.com/entry/carol-brady_us_5b9c5c92e4b03a1dcc7e15be
|What Carol Brady Is Really Saying (INFOGRAPHIC)                                   |PARENTING    |Here's the story of a lovely lady --
who somehow kept six kids, a husband, a housekeeper and a dog in perfect order with|0.16666666666666666|
+-----------------------------------------------------------------------------------------------------------------------------------
-------------+----------------------------------------------------------------------------------+-------------+----------------------
```

Step 1: Data Preparation and Tokenization

I started by loading a dataset of HuffPost articles and a base article description. I used Tokenizer to split each article's short_description into individual words,

creating a words column in the DataFrame. Tokenization was essential here, as it allowed me to compare articles based on shared words.

Step 2: Vectorization Using HashingTF

To enable the MinHashLSH model to work with this text data, I converted the tokenized words into feature vectors using HashingTF. This step generated a fixed-length feature vector for each article's words, allowing the model to perform efficient similarity calculations. HashingTF effectively reduced the dimensionality of the text data by hashing words into a numerical representation of a specific length (numFeatures=10000), capturing word frequency information while avoiding the computational expense of more complex embeddings.

Step 3: Finding Similar Items with MinHashLSH

With feature vectors in place, I applied MinHashLSH to find approximate nearest neighbors to the base article. MinHashLSH is particularly useful in high-dimensional data, as it groups similar items by hashing, allowing for fast and approximate similarity calculations. After fitting the MinHashLSH model to the data, I queried the model for the top 5 articles with the highest similarity to the base article. This gave me a set of candidates with similar content based on hash proximity, approximating Jaccard similarity efficiently.

Step 4: Validation Using Manual Jaccard Similarity

To validate the accuracy of the MinHashLSH results, I manually calculated the Jaccard similarity between the base article and each article in the dataset. I used a custom UDF (User Defined Function) to compute Jaccard similarity by finding the intersection and union of word sets for each article and the base description. I then selected the top 5 articles with the highest Jaccard similarity, creating a second list of most similar items based on exact similarity.

Results Comparison

Finally, I displayed the top 5 similar articles based on both MinHashLSH and manual Jaccard similarity. This allowed me to evaluate how closely the approximate similarity matched the exact Jaccard results, illustrating the balance between accuracy and computational efficiency offered by MinHashLSH in large datasets. This dual approach highlights the practical utility of MinHashLSH as a fast, scalable approximation for Jaccard similarity in text-based similarity tasks.

AI Acknowledgement

I have used ChatGPT & Perplexity to understand Jaccard Similarity, 1.MinHash LSH and HashingTF in Q4 because I didn't know this before. Also I used it to generate a validation code for my existing code in Q3, just to check if my answers are correct. I also used GPT to figure out Hadoop .jar compilation commands, and also to verify my Reducer, as it was erroneous in the beginning in Q1.

THANK YOU!

Sai Nishanth Mettu

sm11326