Fundamental of Big Data Analytics (DS2004)

Assignment #3

Deadline: 21-04-2024

Course Team		
Dr. Muhammad Ishtiaq	Course Coordinator	m.ishtiaq@nu.edu.pk
Ms. Kainat Iqbal	Course Instructor	kainat.iqbal@nu.edu.pk
Ibrahim Bin Umair	Teaching Assisant	i200567@nu.edu.pk
Muhammad Huzaifa Khan	Teaching Assisant	i212689@nu.edu.pk
Hashim Muhammad Nadeem	Teaching Assisant	i211675@nu.edu.pk

Assignment Guidelines:

- This assignment is meant to be completed as a team.
- Only one team member should submit the assignment. Multiple or duplicate
- submissions will lead to a deduction of marks.
- To submit the assignment, please create a GitHub repository and upload your source codes along with your result file. Ensure the repository is public. Share only the repository link on Google Classroom for submission. Do not include any additional materials on Google Classroom, as marks will be deducted for doing so.
- Additionally, you must provide a comprehensive report detailing your work and findings. This report should be written within the README.md file located in your GitHub repository.
- To enhance the readability of your submission, please use Markdown elements (such as headings) to organize the code and explanations/findings.
- Any portion of the code with an error, even minor ones, will not be accepted.
- You may refer to online sources such as websites and/or ChatGPT, but make you can explain the code you have written.
- To earn bonus marks, please use comments and adhere to correct PEP 8 coding conventions.

Plagiarism Policy:

Plagiarism is a grave academic offense that can severely undermine your academic integrity and reputation. Any instance of a student found to have plagiarised their assignment, whether from a peer or external source, will be subject to strict consequences. This may result in a zero score for the current or all assignments, or severe cases, even a failure in the course. Furthermore, all instances of plagiarism will be promptly reported to the Disciplinary Committee for further action.

Streaming Data Insights

Frequent Itemset Analysis on Amazon Metadata

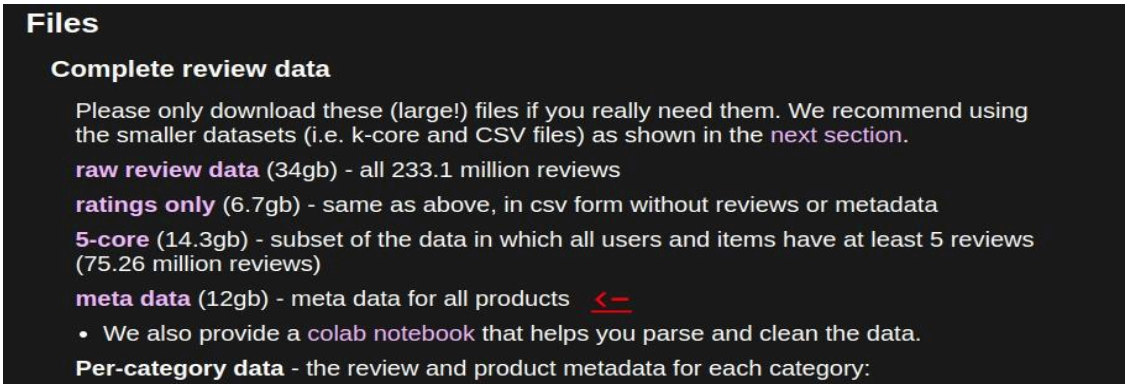
Dataset Description

In this assignment, you will use the amazon_metadata dataset. The dataset is of JSON (Javascript Object Notation).

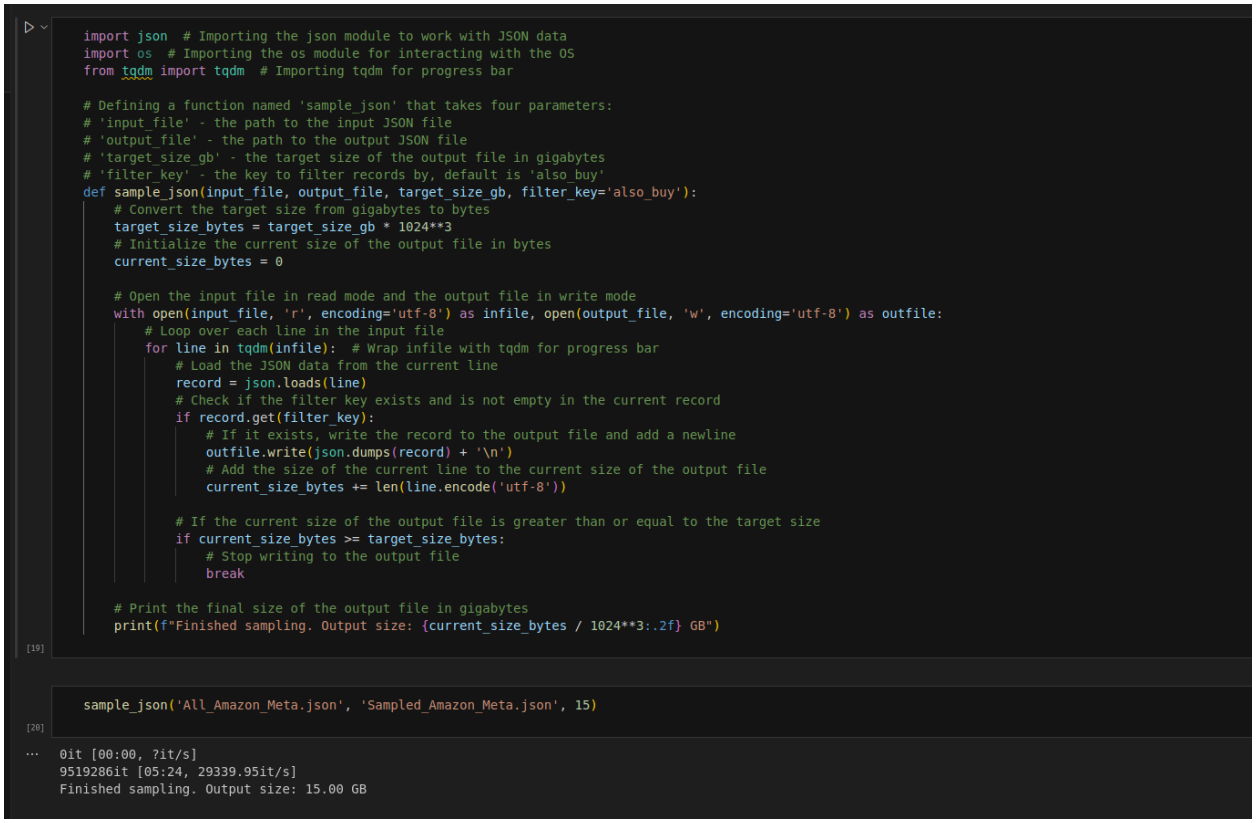
- asin - ID of the product, e.g. 0000031852
- title - name of the product
- feature - bullet-point format features of the product
- description - description of the product
- price - price in US dollars (at time of crawl)
- imageURL - URL of the product image
- imageURL - URL of the high-resolution product image
- related - related products (also bought, also viewed, bought together, buy after viewing)
- salesRank - sales rank information
- brand - brand name
- categories - list of categories the product belongs to
- tech1 - the first technical detail table of the product
- tech2 - the second technical detail table of the product
- similar - similar product table

Downloading and Sampling the Dataset [5]

You can download the Amazon Metadata dataset through this link.



Once you download the dataset, you will extract it. The dataset size will go from 12 GB to 105 GB. To work with this data, you can first sample it. Here is a script you can use to do so:



```python
import json  # Importing the json module to work with JSON data
import os  # Importing the os module for interacting with the OS
from tqdm import tqdm  # Importing tqdm for progress bar

# Defining a function named 'sample_json' that takes four parameters:
# 'input_file' - the path to the input JSON file
# 'output_file' - the path to the output JSON file
# 'target_size_gb' - the target size of the output file in gigabytes
# 'filter_key' - the key to filter records by, default is 'also_buy'
def sample_json(input_file, output_file, target_size_gb, filter_key='also_buy'):
    # Convert the target size from gigabytes to bytes
    target_size_bytes = target_size_gb * 1024**3
    # Initialize the current size of the output file in bytes
    current_size_bytes = 0

    # Open the input file in read mode and the output file in write mode
    with open(input_file, 'r', encoding='utf-8') as infile, open(output_file, 'w', encoding='utf-8') as outfile:
        # Loop over each line in the input file
        for line in tqdm(infile):  # Wrap infile with tqdm for progress bar
            # Load the JSON data from the current line
            record = json.loads(line)
            # Check if the filter key exists and is not empty in the current record
            if record.get(filter_key):
                # If it exists, write the record to the output file and add a newline
                outfile.write(json.dumps(record) + '\n')
                # Add the size of the current line to the current size of the output file
                current_size_bytes += len(line.encode('utf-8'))

            # If the current size of the output file is greater than or equal to the target size
            if current_size_bytes >= target_size_bytes:
                # Stop writing to the output file
                break

    # Print the final size of the output file in gigabytes
    print(f"Finished sampling. Output size: {current_size_bytes / 1024**3:.2f} GB")
```

```
[19]
```

```python
sample_json('All_Amazon_Meta.json', 'Sampled_Amazon_Meta.json', 15)
```

```
[20]
```

```
0it [00:00, ?it/s]
9519286it [05:24, 29339.95it/s]
Finished sampling. Output size: 15.00 GB
```

Make sure your **sample size is at least 15 GB**. A sample size lower than this will result in a **deduction of marks**.

Pre-Processing [10]

1. Load the Sampled Amazon dataset.

2. Preprocess the data to clean and format it for analysis, ensuring it is suitable for the streaming and frequent itemset mining process.
3. Generate a new JSON file containing the preprocessed data.
4. **BONUS:** Perform batch processing to execute pre-processing in real time. [3]

Streaming Pipeline Setup [10]

1. Develop a producer application that streams the preprocessed data in real time.
2. Create three consumer applications that subscribe to the producer's data stream.

Frequent Itemset Mining [35]

1. Implement the Apriori algorithm in one consumer. There should be print statements showing real-time insights and associations.
2. Implement the PCY algorithm in one consumer. There should be print statements showing real-time insights and associations.
3. In the third consumer, you are free to do whatever you want. Be as innovative and creative as possible. We will **NOT** accept straightforward analyses like identifying duplicates or performing basic calculations. This consumer will hold the highest number of marks.

The challenge of applying Apriori or PCY algorithms in a streaming context is that these algorithms typically require access to the entire dataset to calculate the itemsets accurately. However, in a streaming data scenario, you only have access to a portion of the data at any given time. To address this challenge, you can employ techniques that adapt these algorithms to the streaming environment:

- Sliding Window Approach
- Approximation Techniques
- Incremental Processing
- Decaying Factor
- Online Algorithms

These are some of the methods you can use. You are free to use whichever one you want or search online for other approaches. **REMEMBER:** Tweaking things for optimization can snag you some sweet **bonus points**. [2]

Database Integration [15]

- You are free to choose any type of Database you want, however, I would strongly recommend using a non-relational or NoSQL one, such as MongoDB, as it fits perfectly for this project.
- Modify each consumer to connect to one of the databases and store the results.

ReadME [5]

The ReadME file should contain all necessary information about your file. It should tell the user about your approach, and why it's the one you chose.

Bonus: Enhancing Project Execution with a Bash Script [5]

- Set up a bash script that runs the producer, and consumers, and initalizes all Kafka components, like Kafka Connect, Zookeeper, etc.
- Again, I would strongly urge you to do this, as it makes things much easier.