Frequent item set mining in E-commerce transaction logs

Background: Frequent item set mining is a fundamental task in data mining. It involves identifying sets of items (or elements) that frequently occur together in a given dataset. This technique is widely used in various applications, including market basket analysis, recommendation systems, bioinformatics, and network analysis.

The item set is a collection of one or more items. For example, in a market basket analysis, an item set could represent a list of products that a customer purchases in a single transaction. In order to find out the frequent item sets, we need to first compute the "support" of each item set, which is the proportion of transactions in which the item set appears.

To be specific, the support of an item set is the number of transactions in which the item set appears, divided by the total number of transactions. For example, suppose we have a dataset of 1000 transactions, and the item set {milk, bread} appears in 100 of those transactions. The support of the item set {milk, bread} would be calculated as follows:

```
Support({milk, bread}) = Number of transactions containing
                          {milk, bread} / Total number of transactions
                         = 100 / 1000
                         = 10%
```

So the support of the item set {milk, bread} is 10%. This means that in 10% of the transactions, the items milk and bread were both purchased.

Problem Definition: You are given an E-Commerce dataset of customer purchase transaction logs collected over time. Each record in the dataset has the following five fields (see the example dataset):

```
InvoiceNo: the unique ID to record one purchase transaction
Description: the name of the item in a transaction
Quantity: the amount of the items purchased
InvoiceDate: the time of the transaction
UnitPrice: the price of a single item
```

Your task is to utilize Spark to detect the top-k frequent item sets from the log for each month. To make the problem simple, you are only required to find frequent item sets containing three items. The support of an item set X in a month M is computed as:

Support(X) = (Number of transactions containing X in M) / (Total number of transactions in M)

Output Format: The output format is "**MONTH/YEAR,(Item1|Item2|Item3), support value"**, where the three items are ordered alphabetically. You need to sort the results first by time in ascending order, then by the support value in descending order, and finally by the item set in alphabetical order. If one month has less than k item sets, just output all item sets in order for that month.

For example, given the sample dataset and k=2, your result should be like this:

```
1/2010,(A|B|C),0.6666666666666666
1/2010,(A|C|D),0.6666666666666666
2/2010,(A|B|C),1.0
```

Code Format: The code template has been provided. You need to submit two solutions, one using only RDD APIs and the other one using only DataFrame APIs. Your code should take three parameters: the input file, the output folder, and the value of k. Assuming k=2, you need to use the command below to run your code:

```
$ spark-submit project2_rdd.py "file:///home/sample.csv"
"file:///home/output" 2
```

Submission

Deadline: Tuesday 31st October 11:59:59 PM

If you need an extension, please apply for a special consideration via “myUNSW” first. You can submit multiple times before the due date and we will only mark your final submission. To prove successful submission, please take a screenshot as the assignment submission instructions show and keep

Late submission penalty

5% reduction of your marks for up to 5 days, submissions delayed for over 5 days will be rejected.

Some notes

1. You can read the files from either HDFS or the local file system. Using the local files is more convenient, but you need to use the prefix "file:///...". Spark uses HDFS by default if the path does not have a prefix.
2. You are not allowed to use numpy or pandas, since we aim to assess your understanding of the RDD/DataFrame APIs.
3. You can use coalesce(1) to merge the data into a single partition and then save the data to disk.
4. In the DataFrame solution, it is not allowed to use the spark.sql() function to pass the SQL statement to Spark directly.
5. It does not matter if you have a new line at the end of the output file or not. It will not affect the correctness of your solution.

Marking Criteria (Each solution has 8 marks)

- You must complete this assignment using Spark RDD/DataFrame APIs. Submissions only contain regular Python techniques will be marked as 0.
- Submission can be compiled and run on Spark => +3
- Submission can obtain correct results (including correct item sets, correct support values, correct format and order) => +3

- The efficiency of top-k computation => +1
- Submissions correctly using Spark APIs (RDD/DataFrame solution only RDD/DataFrame APIs allowed) => +0.5
- Submissions with excellent code format and structure, readability, and documentation => 0.5