INF 553 – Fall 2018
Assignment 3: Frequent Itemsets – Frequent Words in Yelp Reviews

Assignment Overview

The goal of the assignment is to identify frequent words in Yelp reviews. This assignment contains one main algorithm. You will implement the SON algorithm using the Apache Spark Framework. You will use three different datasets, ranging from small to large. This will help you to test, develop and optimize your algorithm given the number of records at hand. More details on the structure of the datasets and instructions on how to use the input files will be explained in details in the following sections. The goal of this assignment is to make you understand how you can apply the frequent itemset algorithms you have learned in class on a large number of data and more importantly how you can make your implementation more performant and efficient in a distributed environment.

Environment Requirements

Python: 2.7 Scala: 2.11 Spark: 2.3.1

Write your own code!

For this assignment to be an effective learning experience, you must write your own code!
Do not share code with other students in the class!!
Here's why:

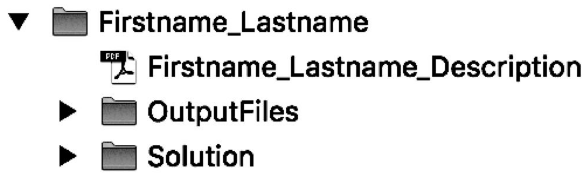
- The most obvious reason is that it will be a huge temptation to cheat: if you include code written by anyone else in your solution to the assignment, you will be cheating. As mentioned in the syllabus, this is a very serious offense, and may lead to you failing the class.
- However, even if you do not directly include any code you look at in your solution, it surely will influence your coding. Put another way, it will short-circuit the process of you figuring out how to solve the problem, and will thus decrease how much you learn.

So, just don't look on the web for any code relevant to these problems. Don't do it.

Submission Details

For this assignment you will need to turn in a Python or Scala program depending on your language of preference. We will test your code using the same datasets but with different support thresholds values. This assignment will surely need some time to be implemented so please plan accordingly and start early!

Your submission must be a .zip file with name: **<Firstname>_<Lastname>_hw3.zip**. The structure of your submission should be identical as shown below. The Firstname_Lastname_Description.pdf file contains helpful instructions on how to run your code along with other necessary information as described in the following sections. The *OutputFiles* directory contains the deliverable output files for each problem and the *Solution* directory contains your source code.



▼ Firstname_Lastname
  Firstname_Lastname_Description
  ▶ OutputFiles
  ▶ Solution

SON Algorithm

In this assignment we implement the SON Algorithm to solve every problem (Problems 1 to 3) on top of Apache Spark Framework. We will rely on the fact that SON can process chunks of data in order to identify the frequent itemsets. You will need to find **all the possible combinations of the frequent itemsets** in the Yelp Round 12 Dataset. In order to accomplish this task, you need to read Chapter 6 from the Mining of Massive Datasets book and concentrate on section 6.4 – Limited-Pass Algorithms. Inside the Firstname_Lastname_Description.pdf file we need you to describe the approach you used for your program. Specifically, in order to process each chunk which algorithm did you use, A-Priori, MultiHash, PCY, etc…

At the end of the assignment, Appendix A provides some more guidelines that will help you with the implementation and Appendix B specifies how to organize your Description pdf file.

For assignment 1 you used the Spark framework and most probably at this point you have a better understanding of the MapReduce operations. You can write your program in Python or Scala. For this assignment you will need to find the collection of frequent itemsets of words in reviews in the Yelp dataset with which you are already familiar with. You will need to compute the frequent itemsets using SON algorithm, initially for **yelp_reviews_test**(Problem 1), **yelp_reviews_small** (Problem 2) dataset, and **yelp_reviews_large** (Problem 3) for large dataset.

The dataset was prepared by extracting the text from reviews in Yelp Challenge Dataset Round 12. All special characters were removed from the reviews by running `sub('\W+', '',word)` and then converting the words to lowercase. Duplicate entries and stop words were removed from a review and words were written to a file. Every word is written on an individual line with the format "review_index, word". The small dataset (yelp_reviews_small) contains 10000 reviews and approximately 36000 unique words. The large dataset (yelp_reviews_large) contains 1 million reviews with approximately 620K words.

Finally, in the section Problem 1, we will describe explicitly how you should run your program, and what should be the format of your expected output. Everything that is described in section Problem 1 must be applied to the subsequent sections as well (i.e., Problem 2)

Problem 1 (20 Points)
Implementing SON using Spark with the Testing Dataset

We have a prepared a test set with few 1000 reviews. The *Data/yelp_reviews_test.txt* dataset can be used to verify the correctness of your implementation. We will also require you to submit an output as described in the following Deliverables section.

Execution Requirements

Input Arguments:

1. **Input.txt:** This is the path to the input reviews file. Every line contains a word and the index of the review the word belongs to, with the format "index_review, word". For Problem 1 you can use the *Data/yelp_reviews_test.txt* file to test the correctness of your algorithm.

2. **Support:** Integer that defines the minimum count to qualify as a frequent itemset.

Output:

A file in the format shown in the snapshot of the Execution Example section below. In particular, for each line you should output the frequent itemsets you found for the current combination followed by an empty line after each combination. The printed itemsets must be sorted in ascending order. A higher level description of this format is:

(frequent_singleton1), (frequent_singleton2), …, (frequent_singletonK)

(frequent_pair1), (frequent_pair2), …, (frequent_pairM)

(frequent_triple1), (frequent_triple2), …, (frequent_tripleN)
…

Execution Example

The first input is the path to the reviews input file and the second is the support threshold value. Following we present examples of how you can run your program with spark submit both when your application is a Java/Scala program or a Python script.

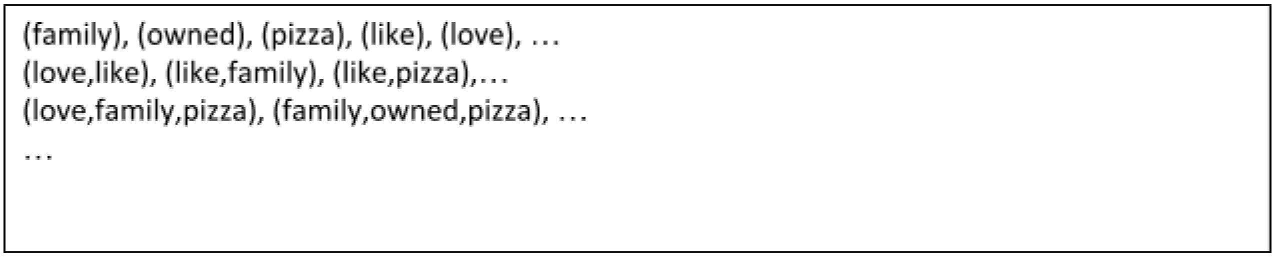
A. **Example of running a Java/Scala application with spark-submit:**
Notice that the argument class of the spark-submit specifies the main class of your application and it is followed by the jar file of the application.

```
$SPARK_HOME/bin/spark-submit --class Firstname_Lastname_SON
Firstname_Lastname_SON.jar <input_file> <support_threshold> <output_file>
```

Example of running a Python application with spark-submit:

```
$SPARK_HOME/bin/spark-submit Firstname_Lastname_SON.py <input_file>
<support_threshold> <output_file>
```

Example output



```
(family), (owned), (pizza), (like), (love), …
(love,like), (like,family), (like,pizza),…
(love,family,pizza), (family,owned,pizza), …
…
```

Deliverables for Problem 1

1. **Script or Jar File and Source Code**
   Please name your Python script as: **<firstname>_<lastname>_SON.py**
   Or if you use scala, **use class name as <firstname>_<lastname>_SON (For example, Anirudh_Kashi_SON) and jar file as: <firstname>_<lastname>_SON.jar (For example, Anirudh_Kashi_SON.jar)**

   The python script or scala script (with .jar file) of your implementation should be inside the *Solution* directory of your submission. You must also include a directory, any directory name is fine, with your source code inside *Solutions*.

2. **Output Files**
   We need two output files for Problem 1.
   Run your program against *Data/ yelp_reviews_test.txt* dataset with support 30.
   Run your program against *Data/ yelp_reviews_test.txt* dataset with support 40. **Support threshold values are the arguments.**

   The format of the output should be exactly the same as the above snapshot for both cases.
   The names of the output files should be as:
   <firstname>_<lastname>_SON_ *yelp_reviews_test*_30.txt
   <firstname>_<lastname>_SON_ *yelp_reviews_test*_40.txt

   The above output files should be placed inside the *OutputFiles* directory of your submission.

3. **Description**
   Inside the Firstname_LastName_Description pdf document please write the command line that you used with spark-submit in order to run your code. Specify also the Spark version that you use to write your code. **Also, see Appendix B.**

Problem 2 (50 Points)

Implementing SON using Spark with the Yelp Reviews Small Dataset

The requirements for Problem 2 are similar to Problem 1. However, here we would like to **check for the performance of our implementation for a larger dataset** (*Data/yelp_reviews_small.txt)*. We would like to find the frequent itemsets among a larger number of records. For this purpose, a good indicator of how well our algorithm works is the total execution time. In this execution time **we take into account also the time of reading the files from the disk.** Following, we provide a table of execution time for two threshold values for each case described in the first section. You can use this array as an evaluation metric of your implementation.

Support Threshold	Execution Time
500	<80secs
1000	<40secs

<u>**Deliverables for Problem 2**</u>

1. **Output Files**

The format of the output should be exactly the same as the one for Problem 1.
The output files should be named as:
<firstname>_<lastname>_SON_ *yelp_reviews_small*_500.txt
<firstname>_<lastname>_SON_ *yelp_reviews_small*_1000.txt

The above output files should be placed inside the *OutputFiles* directory of your submission.

Grade breakdown
a. Two correct output files (10pts each)
b. Your execution time needs to be smaller than the ones in the table (10pts each)
c. Your execution time needs to be less than 2/3 of the ones in the table (5pts each)

Problem 3 (30 Points)
Implementing SON using Spark with the Yelp Reviews Big Dataset

For the last part of this assignment we will run our application using the large Yelp Reviews dataset (*(Data/yelp_reviews_large.txt)*. Since the purpose of this assignment is to test the efficiency and see how you can optimize your implementation we also provide some execution times similarly as we did in Problem 2. In this execution time we take into account also the time of reading the files from the disk.

Support Threshold	Execution Time
100000	<600secs

120000	<500secs

Deliverables for Problem 3

1. **Output Files**

The format of the output should be exactly the same as the one for Problem 1.
The output files should be named as:
<firstname>_<lastname>_SON_*yelp_reviews_large*_100000.txt
<firstname>_<lastname>_SON_*yelp_reviews_large*_200000.txt

The above output files should be placed inside the *OutputFiles* directory of your submission.

Grade breakdown
a. Two correct output files (10 pts each)
b. Your execution time needs to be smaller than the ones in the table (5pts each)

Bonus (5pts): Describe why did we need to use such a large support threshold and where do you think there could be a bottleneck that could result in a slow execution for your implementation, if any.

General Instructions:

1. Make sure your code compiles before submitting
2. Make sure to follow the output format and the naming format.

Grading Criteria:

1. If your programs cannot be run with the commands you provide, your submission will be graded based on the result files you submit and 20% penalty for it.
2. If the files generated by your programs are not sorted based on the specifications, there will be 20% penalty.
3. If your program generates more than one file, there will be 20% penalty.
4. **If you don't provide the source code and just the .jar file in case of a Scala application there will be 60% penalty.**
5. **If your submission does not state inside the Description pdf file how to run your code, which Spark version you used and which approach you followed to implement your algorithm there will be a penalty of 30%.**
6. There will be 20% penalty for late submission.

APPENDIX A

- You need to take into account the Monotonicity of the Itemsets
- You need to leverage Spark capabilities of processing partitions/chunks of data and analyze the data within each partition.
- You need to reduce the support threshold according to the size of your partitions.
- You should emit appropriate (key, value) pairs in order to speed up the computation time.
- Try to avoid data shuffling during your execution.

Pay great attention on the thresholds number for each case. The lower the threshold the more the computation. Do not try arbitrary threshold values. Try testing values within the given ranges.

APPENDIX B

Please include the following information inside your description document.

- Succinctly describe your approach to implement the algorithm.
- Command line command to execute your program
- Problem 2 execution table
- Problem 3 execution table