Project Phase 2

Due Apr 24 by 11:59pm **Points** 0 **Submitting** a file upload
File Types zip, tar, tar.gz, gz, tgz, and jar
Available Apr 10 at 12am - Apr 24 at 11:59pm

This assignment was locked Apr 24 at 11:59pm.

This project requires two types of spatial data analysis: 1) Hot zone analysis and 2) hot cell/hotspot analysis.

The project's overall instructions are provided below.

Step 1: Set up your Hadoop and Apache Spark test environment. The following setup instructions are specific to Ubuntu operating system.

spark_ubuntu_install

Step 2: Complete the Hotspot Analysis programming assignment, including submitting your own hotspot source code as a .jar file.

Step 3: Write and submit your project report detailing your work on this work.

Instructions:

Project 2: Hot Spot Analysis

Requirement

In this project, you are required to do spatial hot spot analysis. In particular, you need to complete two

different hot spot analysis tasks.

1. Hot zone analysis

This task will need to perform a range join operation on a rectangle datasets and a point dataset. For each rectangle, the number of points located within the rectangle will be obtained. The hotter rectangle means that it includes more points. So this task is to calculate the hotness of all the rectangles.

Download the required templates here.

2. Hot cell analysis

Description

This task will focus on applying spatial statistics to spatio-temporal big data in order to identify statistically significant spatial hot spots using Apache Spark. The topic of this task is from ACM SIGSPATIAL GISCUP 2016.

The Problem Definition page is here: **http://sigspatial2016.sigspatial.org/giscup2016/problem (http://sigspatial2016.sigspatial.org/giscup2016/problem)**

The Submit Format page is here: **http://sigspatial2016.sigspatial.org/giscup2016/submit (http://sigspatial2016.sigspatial.org/giscup2016/submit)**

Special requirement (different from GIS CUP)

As stated in the Problem Definition page, in this task, you are asked to implement a Spark program to calculate the Getis-Ord statistic of NYC Taxi Trip datasets. We call it "**Hot cell analysis**"

To reduce the computation power need, we made the following changes:

1. The input will be a monthly taxi trip dataset from 2009 - 2012. For example, "yellow_tripdata_2009-01_point.csv", "yellow_tripdata_2010-02_point.csv".
2. Each cell unit size is 0.01 * 0.01 in terms of latitude and longitude degrees.
3. You should use 1 day as the Time Step size. The first day of a month is step 1. Every month has 31 days.
4. You only need to consider Pick-up Location.
5. We don't use Jaccard similarity to check your answer. However, you don't need to worry about how to decide the cell coordinates because the code template generated cell coordinates. You just need to write the rest of the task.

Coding template specification

Input parameters

1. Output path (Mandatory)
2. Task name: "hotzoneanalysis" or "hotcellanalysis"
3. Task parameters: (1) Hot zone (2 parameters): nyc taxi data path, zone path(2) Hot cell (1 parameter): nyc taxi data path

Example:

test/output hotzoneanalysis src/resources/point-hotzone.csv src/resources/zone-hotzone.csv hotcellanalysis src/resources/yellow_tripdata_2009-01_point.csv

Note:

1. The number/order of tasks do not matter.
2. But, the first 7 of our final test cases will be hot zone analysis, the last 8 will be hot cell analysis.

Input data format

The main function/entrace is "cse512.Entrance" scala file.

1. Point data: the input point dataset is the pickup point of New York Taxi trip datasets. The data format of this phase is the original format of NYC taxi trip which is different from Phase 2. But the coding template already parsed it for you. Find the data in the .zip file below.

yellow_tripdata_2009-01_point.csv

2. Zone data (only for hot zone analysis): at "src/resources/zone-hotzone" of the template

Hot zone analysis

The input point data can be any small subset of NYC taxi dataset.

Hot cell analysis

The input point data is a monthly NYC taxi trip dataset (2009-2012) like "yellow_tripdata_2009-01_point.csv"

Output data format

Hot zone analysis

All zones with their count, sorted by "rectangle" string in an ascending order.

Hot cell analysis

The coordinates of top 50 hotest cells sorted by their G score in a descending order. Note, DO NOT OUTPUT G score.

-7399,4075,15
-7399,4075,29
-7399,4075,22

Example answers

An example input and answer are put in "testcase" folder of the coding template

Where you need to change

DO NOT DELETE any existing code in the coding template unless you see this "YOU NEED TO CHANGE THIS PART"

Hot zone analysis

In the code template,

1. You need to change "**HotzoneAnalysis.scala** and **HotzoneUtils.scala**".
2. The coding template has loaded the data and wrote the first step, range join query, for you. Please finish the rest of the task.
3. The output DataFrame should be sorted by you according to "rectangle" string.

Hot cell analysis

In the code template,

1. You need to change "**HotcellAnalysis.scala** and **HotcellUtils.scala**".
2. The coding template has loaded the data and decided the cell coordinate, x, y, z and their min and max. Please finish the rest of the task.
3. The output DataFrame should be sorted by you according to G-score. The coding template will take the first 50 to output. DO NOT OUTPUT G-score.

Submission

Submission files

1. Submit your project jar package in My Submission Tab.
2. Note: You need to make sure your code can compile and package by entering sbt clean assembly. We will run the compiled package on our cluster directly using "spark-submit" with parameters. If your code cannot compile and package, you will not receive any points.

Tips (Optional)

This section is same as that in Phase 2.

How to debug your code in IDE

If you are using the Scala template,

1. Use IntelliJ Idea with Scala plug-in or any other Scala IDE.
2. Replace the logic of User Defined Functions ST_Contains and ST_Within in SpatialQuery.scala.
3. Append .master("local[*]") after .config("spark.some.config.option", "some-value") to tell IDE the master IP is localhost.
4. In some cases, you may need to go to "build.sbt" file and change % "provided" to % "compile" in order to debug your code in IDE
5. Run your code in IDE
6. **You must revert Step 3 and 4 above and recompile your code before use of spark-submit!!!**

How to submit your code to Spark

If you are using the Scala template

1. Go to project root folder,
2. Run sbt clean assembly. You may need to install sbt in order to run this command.
3. Find the packaged jar in "./target/scala-2.11/CSE511-Project-Hotspot-Analysis-Template-assembly-0.1.0.jar"
4. Submit the jar to Spark using Spark command "./bin/spark-submit". A pseudo code example: ./bin/spark-submit ~/GitHub/CSE511-Project-Hotspot-Analysis-Template/target/scala-2.11/CSE511-Project-Hotspot-Analysis-Template-assembly-0.1.0.jar test/output hotzoneanalysis src/resources/point-hotzone.csv src/resources/zone-hotzone.csv hotcellanalysis src/resources

/yellow_tripdata_2009-01_point.csv