Spatial Data Science & Engineering

Project Phase 1

Maximum points Possible – 8

Required Tasks

Set Up Working Environment:

Setting up working environment is similar to the setup for Assignment-3 or Part-1 of Assignment-1. If you successfully did the setup for Assignment-1/Assignment-3, you are done with the setup. If you didn't, setup Java, Hadoop, and Spark following the instructions provided with Assignment-1. In order to work with an IDEA such as Intellij IDEA, you may need to add the scala plugin. Install 'sbt' on your computer if you didn't for Assignment-1. Assignment-1 also provides instructions on how to run a project and test with input datasets.

Tasks:

1. A separate PDF file Trajectory-Data.pdf is available along with the assignment materials. Try to understand how trajectory data and trajectory queries work. Also check the existing databases and tools that support trajectory data. Some of those databases and tools are available at the end of Trajectory-Data.pdf file.
2. Think about how you can store trajectory data in an Apache Sedona DataFrame. Which attributes will be required? You should store the trajectory data in such a way that you can efficiently answer any type of trajectory queries available in the Trajectory-Data.pdf file.
3. A sample trajectory dataset is given along with the project template. Also, src/main/scala/ ManageTrajectory.scala file in the project template contains 4 empty functions. You need to complete those 4 functions and submit the project.
4. Function loadTrajectoryData() takes the path to the trajectory data and store it in an Apache Sedona DataFrame. Once you load the given json file as a DataFrame, restructure it to efficiently answer trajectory quries. Return the restructured DataFrame from the function. Structure of this DataFrame may vary from group to group. There is no ground truth structure.

5. Three other functions – getSpatialRange(), getSpatioTemporalRange(), and getKNNTrajectory() return trajectories or trajectory segments according to the definition of these queries. Definitions are available in the Trajectory-Data.pdf file. For finding the K-Nearest Neighbor Trajectories, assume that the distance between two trajectories A and B is the minimum of all pairwise distances between points in trajectories A and B. Only consider those points of a trajectory which consist of a location and a timestamp. Don't consider the intermediate locations between two consecutive timestamps. If you find the same minimum distance for multiple trajectories, pick the nearest trajectories based on ascending order of trajectory id.

Dataset:

A sample trajectory dataset data/simulated_trajectories.json is given. It contains a list of trajectories in JSON format. For each trajectory, it contains a trajectory id, moving object/vehicle id, and trajectory points. A point in a trajectory consists of a latitude, a longitude, and a timestamp. Note that same object/vehicle may have multiple trajectories. So, there might be duplicate object/vehicles ids in the dataset, but they will always have a unique trajectory id.

Sample Output:

Sample dataset simulated_trajectories.json should be placed under project root/data folder when generating the jar file SDSE-Phase-1-assembly-0.1.jar by running the command – 'sbt clean assembly'. You should run *sbt clean assembly* command under the project root folder where build.sbt file is located. Sample outputs for three query functions - getSpatialRange(), getSpatioTemporalRange(), and getKNNTrajectory() are available under sample-output folder. Those files represent the output for the given sample dataset on the following query. The following query should be run after generating the jar.

$SPARK_HOME/bin/spark-submit Path_To_SDSE-Phase-1-assembly-0.1.jar Path_to_output_folder get-spatial-range 33.41415667570768 -111.92518396810091 33.414291502635706 -111.92254858414022 get-spatiotemporal-range 1664511371 1664512676 33.41415667570768 -111.92518396810091 33.414291502635706 -111.92254858414022 get-knn 0 5

Submission Instructions:

- You should submit a .zip file containing the full Spark Scala project. In the build.sbt file, if you changed the dependencyScope to compile for debugging with IDE, change it back to provided before submission. You may delete the following folders before before generating the zip file for submission: SDSE-Phase-1/sample-output folder, SDSE-Phase-1/target folder, SDSE-Phase-1/project/target folder, SDSE-Phase-1/project/project folder.
- You need to make sure your project code can be compiled by entering *sbt clean assembly* command under the root folder where build.sbt file is located.