report 1

> - 2024.10.18 18:30 HJ-302 Main Campus

讲稿

下面由我来介绍一部分基础算法，包括轨迹重建、兴趣点匹配以及基于栅格的潮汐指数计算。

Next, I will introduce some basic algorithms, including trajectory reconstruction, POI matching, and tide index calculation based on grid.

我想要介绍的第一个基本算法是关于轨迹重建的。就我们得到的数据而言，每条轨迹都存在一定的乱序现象，因此我们需要对轨迹进行重建，以便后续的分析。

The first basic algorithm I want to introduce is about trajectory reconstruction. As far as the data we get is concerned, each trajectory has a certain disorder phenomenon, so we need to reconstruct the trajectory for subsequent analysis.

Algorithm 1: Trajectory Reconstruction

可以首先观察一下左边的图片，可以发现原始轨迹数据具有明显的“跳跃现象”，即相邻的轨迹点之间的距离可能会很远。

You can first observe the picture on the left, and you can find that the original trajectory data has obvious "jump phenomenon", that is, the distance between adjacent trajectory points may be very far.

这个问题可以用一个很简单的思路来解决，即认为最近的点大概率是下一个点。这样，在确定了轨迹的起点之后，我们就可以依次寻找下一个点，直到重建出整条轨迹。

This problem can be solved with a very simple idea, that is, the nearest point is probably the next point. In this way, after determining the starting point of the trajectory, we can find the next point one by one until the whole trajectory is reconstructed.

算法具体实现部分。这里我们使用集合这个数据结构来实现去重，但是会导致坐标精度损失，也就无法保证点的唯一性。因此，我们首先使用GeoHash算法对轨迹点进行编码，然后存入集合。对于我们的数据集，该算法大约执行了90分钟。

The specific implementation part of the algorithm. Here we use the set data structure to achieve deduplication, but it will lead to coordinate precision loss, which means that the uniqueness of the point cannot be guaranteed. Therefore, we first use the GeoHash algorithm to encode the trajectory points, and then store them in the set. For our data set, the algorithm took about 90 minutes to execute.

在重建完成轨迹之后，我们发现轨迹中存在锯齿状噪声及冗余点，就考虑使用道格拉斯-普克抽稀算法对轨迹进行简化。但是，经过小组内商讨，我们决定转而使用路网匹配算法来进一步优化轨迹数据。

After the trajectory is reconstructed, we find that there is jagged noise and redundant points in the trajectory, so we consider using the Douglas-Peucker simplification algorithm to simplify the trajectory. However, after discussion within the group, we decided to use the road network matching algorithm to further optimize the trajectory data.

Algorithm 2: POI Matching

在拥有了轨迹数据后，我们还希望进一步探索用户的骑行目的地，以分析动机。这里我们想到使用兴趣点匹配算法，即将轨迹终点与兴趣点进行匹配，以确定用户的骑行目的地类别。

After obtaining the trajectory data, we also hope to further explore the user's riding destination to analyze the motivation. Here we think of using the POI matching algorithm, that is, matching the trajectory end point with the POI to determine the user's riding destination category.

有两个关键思想使得我们的算法具有实际意义。首先，地理学第一定律：事物都是彼此相关的，但是距离越近，相关性越大。其次，共享单车很好地解决了“最后一公里”问题，即用户骑行共享单车会尽可能地靠近目的地，而不像公共交通那样只能到达固定的站点。

- There are two key ideas that make our algorithm practical:
    - First, the first law of geography: everything is related to everything else, but near things are more related than distant things.
    - Second, shared bicycles have solved the "last mile" problem very well, that is, users riding shared bicycles will get as close as possible to the destination, rather than only to fixed stations like public transportation.

现在让我们来讨论一下具体实现。

Now let's discuss the specific implementation.

考虑到该算法涉及到地数据量非常巨大。我们有大约12万个兴趣点，有10万条轨迹数据。想要处理这么多数据，而不考虑优化算法地性能是一件非常可怕地事情。

Considering that this algorithm involves a huge amount of data. We have about one hundred and twenty thousand POIs and one hundred thousand trajectory data. It is a very terrible thing to process so much data without considering the performance of the optimization algorithm.

GeoHash 编码有两个关键特性：1. 编码长度越长，精度越高；2. 编码前缀相的部分越长，表示地理位置越接近。我们可以利用这两个特性，将空间查询动作转化为字符串匹配问题。

- GeoHash encoding has two key features:
    - a. The longer the encoding length, the higher the accuracy;
    - b. The longer the common prefix of the encoding, the closer the geographical location.
- We can use these two features to convert spatial query actions into string matching problems.

我的个人网站上实现了一个附带智能提示的终端。这个智能提示功能就用到了一个高效的字符串匹配算法，即前缀索引树。

There is a terminal with intelligent prompts on my personal website. This intelligent prompt function uses an efficient string matching algorithm, the prefix index tree.

我发现该数据结构可以很好的适配当前的场景。我们可以依次计算所有兴趣点的GeoHash值并插入前缀索引树，然后查询轨迹终点的GeoHash值，返回对应的兴趣点类别编码。

I found that this data structure can be well adapted to the current scenario. We can calculate the GeoHash value of all interest points in turn and insert them into the prefix index tree, then query the GeoHash value of the trajectory end point and return the corresponding interest point category code.

这样大约两分钟即可完成所有轨迹数据的兴趣点匹配。这种高效算法能够给科研工作带来一定的便利，因此是有价值的。在某些情况下，这种优化算法甚至是决定性的。

In this way, it takes about two minutes to complete the POI matching of all trajectory data. This efficient algorithm can bring some convenience to scientific research work, so it is valuable. In some cases, this optimization algorithm is even decisive.

例如，我们可以使用该算法对轨迹中的每一个点进行兴趣点匹配，从而将用户的骑行轨迹从地理坐标转化为由系列兴趣点类别编码组成的序列（句子），并深入分析用户行为。若不进行优化，这种分析过程会被视为不切实际的。

For example, we can use this algorithm to match each point in the trajectory with the POI, so that the user's riding trajectory is converted from geographical coordinates to a sequence (sentence) composed of a series of interest point category codes(words), and analyze user behavior in depth. Without optimization, this analysis process would be considered impractical.

Algorithm 3: Tide Index Calculation

最后，我们希望研究共享单车借还车的潮汐现象，以便更好地规划共享单车的投放。这里我们想到使用基于栅格的潮汐指数计算算法。

Finally, we hope to study the tide phenomenon of borrowing and returning shared bicycles in order to better plan the deployment of shared bicycles. Here we think of using the tide index calculation algorithm based on grid.

为了寻找潮汐现象较为突出的区域，我们需要对轨迹数据进行空间分布统计。这里我们采用划分规则时空格网的方式，统计2016年8月1日早晚高峰（7:00-10:00，17:00-20:00）期间上海市1741条共享单车轨迹数据的空间分布，并生成借、还车潮汐点。

In order to find the areas where the tide phenomenon is more prominent, we need to statistically analyze the spatial distribution of the trajectory data. Here we use the method of dividing the regular space-time grid to statistically analyze the spatial distribution of 1741 shared bicycle trajectory data in Shanghai during the morning and evening peak hours of August 1, 2016 (7:00-10:00, 17:00-20:00), and generate borrowing and returning tide points.