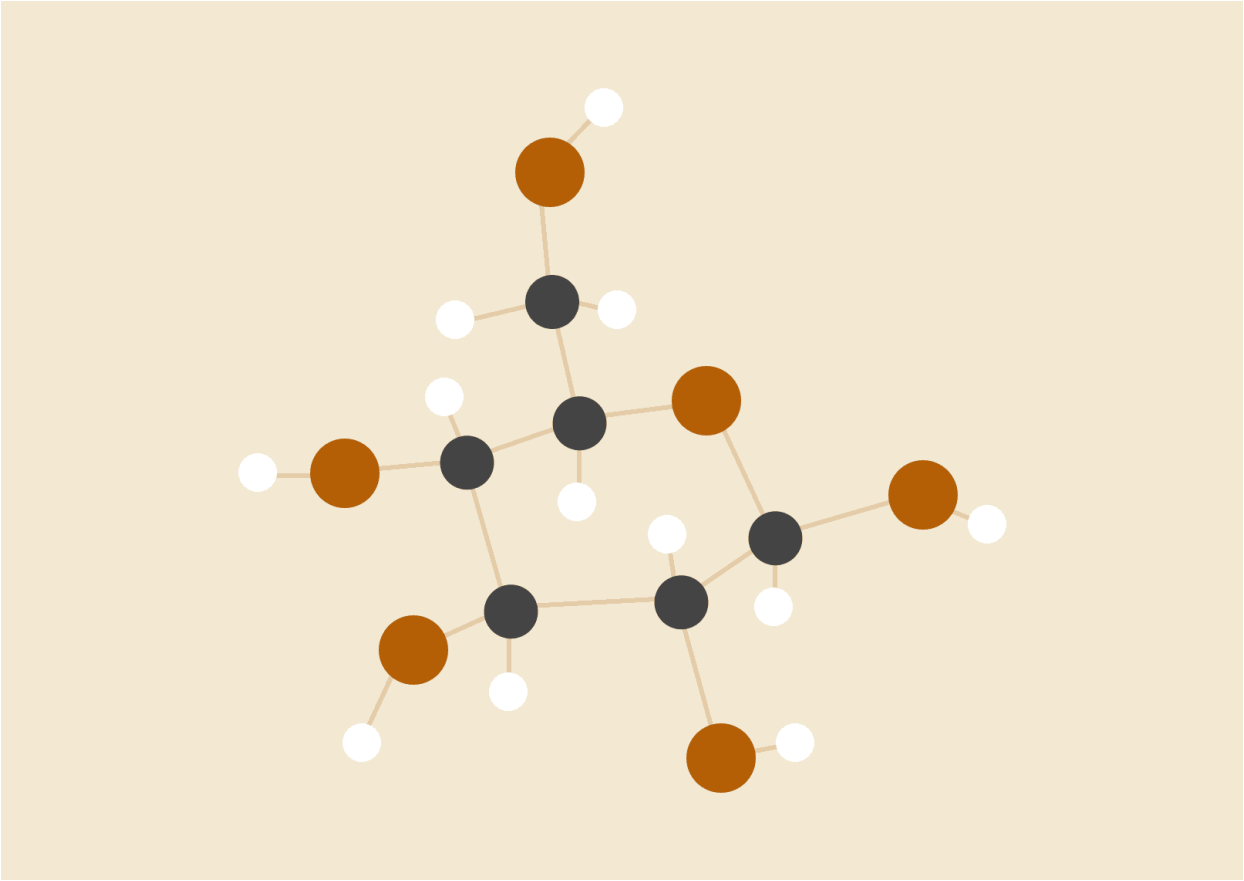
CMPT353 Final Project
OSM, Photos, and Tours



Team members:

Siwei Wang(Ryan) 301335885
Zhuo Liu(Able) 301385532
Teliang Yu(Leon) 301387502

1. Introduction

1.1 problem

Travelling is an excellent way of relaxing one's mind while enjoying views of various sceneries and landscapes of nature. It can be difficult for a person to determine and plan out the best route when travelling to a new city. In fact, it is difficult for one to sort out information on tourist attractions based on the Internet, for there is too much information to process, and the process can be complicated. As a result, it takes a lot of effort to sort out tourist routes according to individual situations. The purpose of this project aims to respond to the following problems:

1. If I were to plan a tour of the city (by walking/biking/driving), where should I go? What are some paths that can walk me past interesting things and various landmarks?
2. Some parts of the city have more chain restaurants (e.g. McDonald's or White Spot franchises, not independently-owned places): is that true? Is there a way to find these chained restaurants automatically and to visualize their density relative to non-chain restaurants?
3. If I was going to choose a hotel (or Airbnb), where should it be? What places have good amenities nearby?

2. Methodology

2.1 Data gathering

In this program, we used 5 data sets:

- *amenities-vancouver.json.gz:* data provided in the project
- *additional_amenities.csv*: Location information of important amenities, collected from map data
- *listings.csv:* Hotel information downloaded additionally from Airbnb(http://insideairbnb.com/get-the-data.html)
- *all_restaurant.csv*: Location information of all restaurants in Vancouver after cleansing amenities-vancouver.json.gz
- *chain_restaurant.csv*: Location information of all chain restaurants in Vancouver after cleansing amenities-vancouver.json.gz

2.2 Data cleansing and processing

For *amenities-vancouver.json.gz* and *additional_amenities.csv*, we first identified the different types and categories of amenities, such as hotels and restaurants. To make sure our data are as consistent and accurate as possible, we used functions to identify words that may be spelled incorrectly. We then fixed data that may be partially incomplete or written in abbreviated forms into their corresponding amenity. For example, the word "caf" will be fixed into "cafe". Next, we eliminated amenity names that are empty to make sure we drop all the empty data. We then dropped data such as "timestamp" and "tags" because they are irrelevant data and will not be used. We also want to eliminate types of amenities that are not tourist attractions, such as toilets and postboxes. Finally, we want to concatenate the filtered amenities (those after data cleansing) along with our additional amenities gathered. This creates the finalized list of amenity data that we will be using for our project.

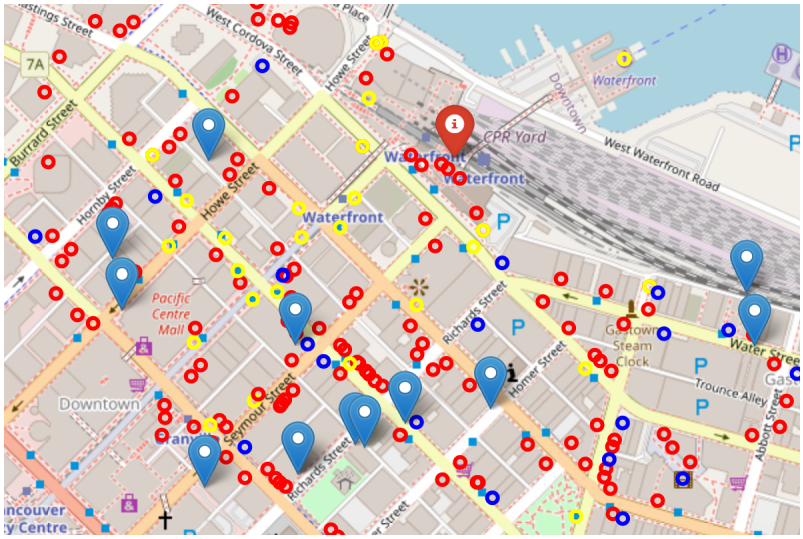
Similar procedures are used to cleanse data for *listings.csv* obtained from Airbnb. We first eliminate repeated data for the hotels. We used filtering to keep the hotels with more than one review per month. Lastly, we dropped the unwanted attributes in the hotel data.

The next step for us is to distinguish between chained restaurants and unchained restaurants. Firstly, we selected the amenity type "restaurant" and created an *all_restaurant.csv*, and dropped all empty data as well as duplicated data. Next, we categorize all those restaurants with repeated names at different locations as chained restaurants, such as McDonald's and Tim Hortons, separated and listed them in *chain_restaurant.csv.*

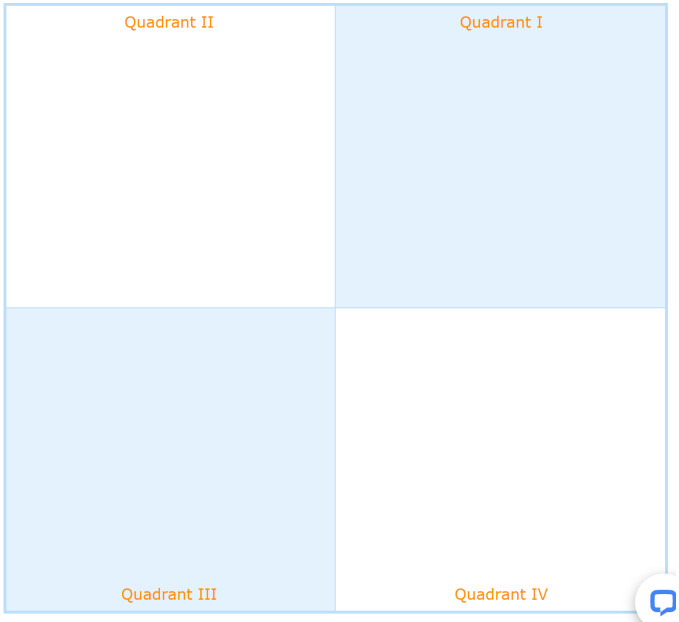
2.3 Techniques

To demonstrate our project in a scenario, imagine a user coming to Vancouver and facing the problem of choosing a hotel. They can take photos of their surrounding buildings and upload them to our program. The program first identifies the location of the image based on latitude and longitude. Next, the program will search for nearby hotels for the user based on the location of the uploaded photo. Finally, the program will recommend nearby hotels to the user, by showing the user (a) the location of the hotel on the map, and (b) the facilities near the hotel such as bars and coffee shops. As shown in the figure below, the red location icon on the map represents the current location of the user, and the blue location icon represents nearby hotels. The other circles represent different amenities. For example, the blue circle represents the bar, the red circle represents the

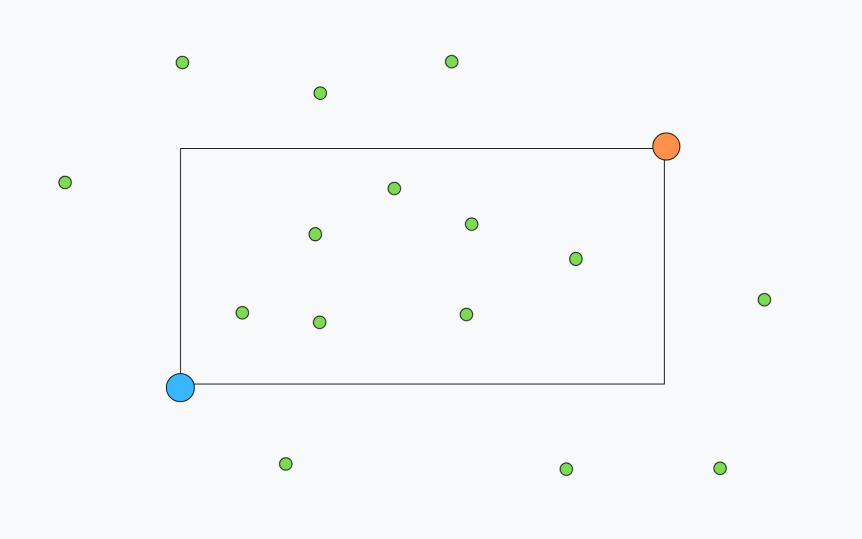
restaurant, and the yellow circle represents the bus station. Users can choose a hotel that suits them according to their preferences of nearby amenities.



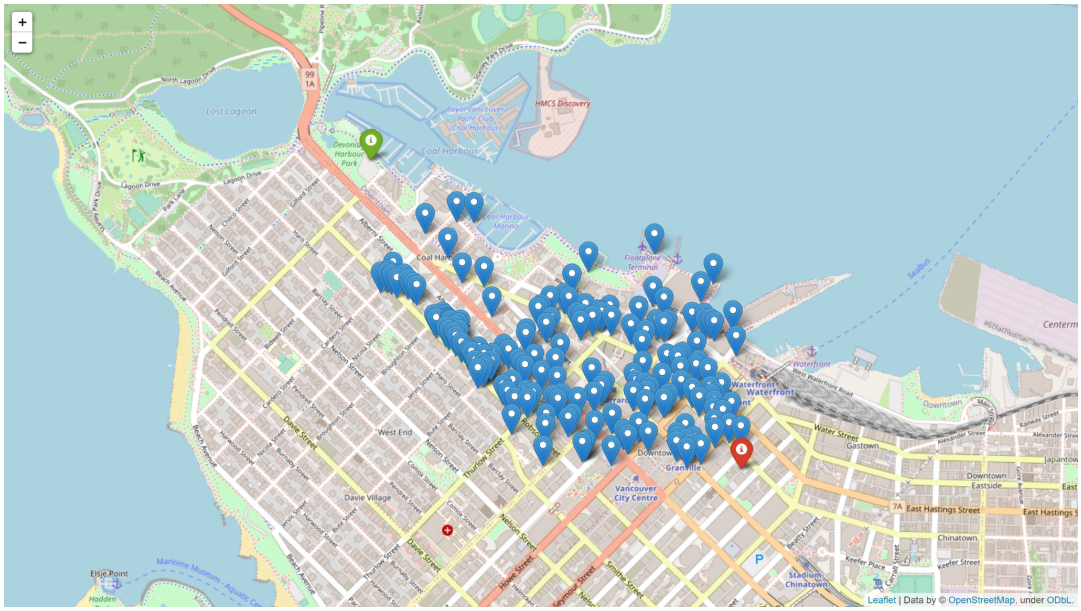
After the user chooses the hotel where he wants to stay, he can then start planning a route to the tourist attraction he wants to go to. Our program will then set the hotel address as the starting point. Users can either take photos and upload them themselves or search for the destination tourist attraction photo from the web and enter the program to set them as the final destination. First, the program will read the location of the start and endpoints, and generate a coordinate system to determine which quadrant the endpoint is in.



Then, a rectangular boundary is generated according to the latitude and longitude of the start point and the endpoint and only retains the amenities in this boundary.

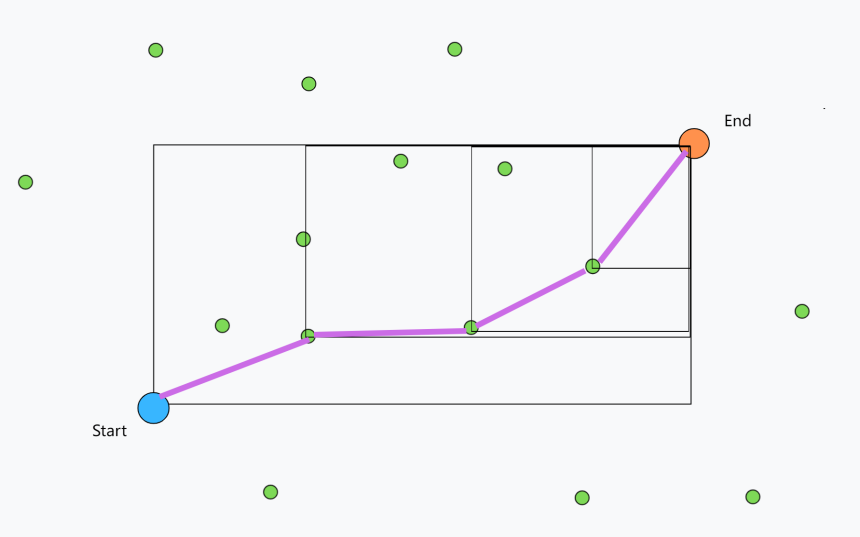


The program will then generate a map based on the boundary, containing all the amenities within this boundary. This provides users with an overview of all the scenic views and amenity spots, where the user may choose freely. The program will also recommend different travel methods for the users based on the distance of travel. If the distance is less than 2 kilometres, the recommended way of travel is walking. If the distance is between 2 kilometres and 5 kilometres, then the recommended way of travel is through a bike. Driving is recommended for distances greater than 5 kilometres.

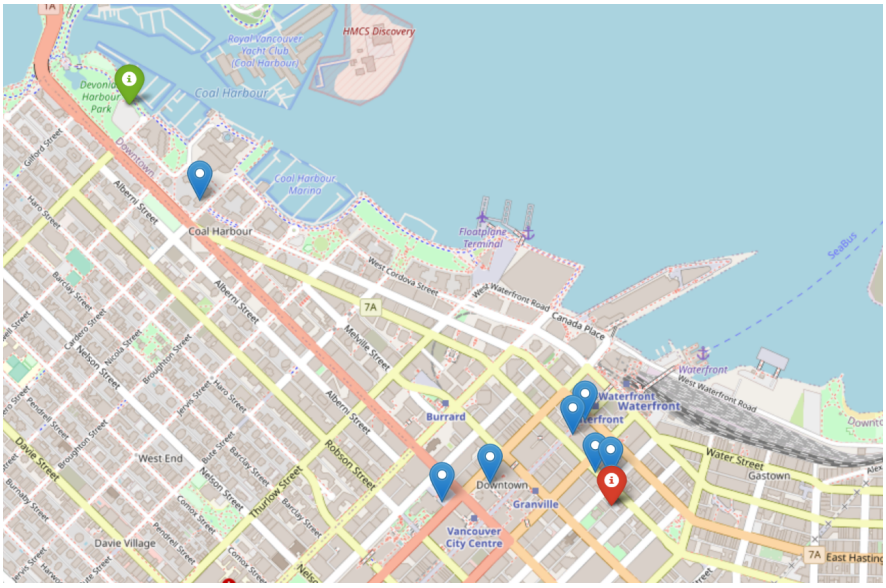


After that, the algorithm will recommend a route through different types of amenities. The specific steps of the algorithm are as follows:

1. Taking the user's chosen hotel as the starting point, find the amenities closest to the starting point in the boundary, and then set this as the new starting point.
2. Set the boundary with a new starting point, and then look for the nearest amenity, and so on. The same type of amenity is only considered once.
3. The algorithm stops when there are no amenity spots in the range or all amenity spots of the same type, the algorithm stops.

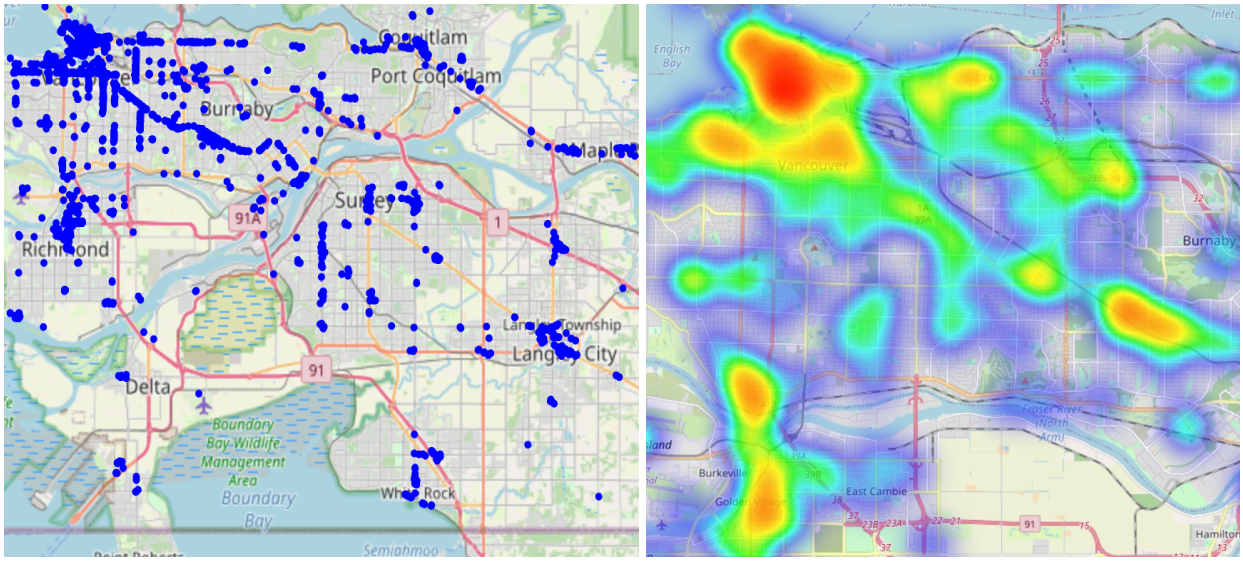


4. The program will generate a map based on the last remaining amenities so that the user can see them intuitively.



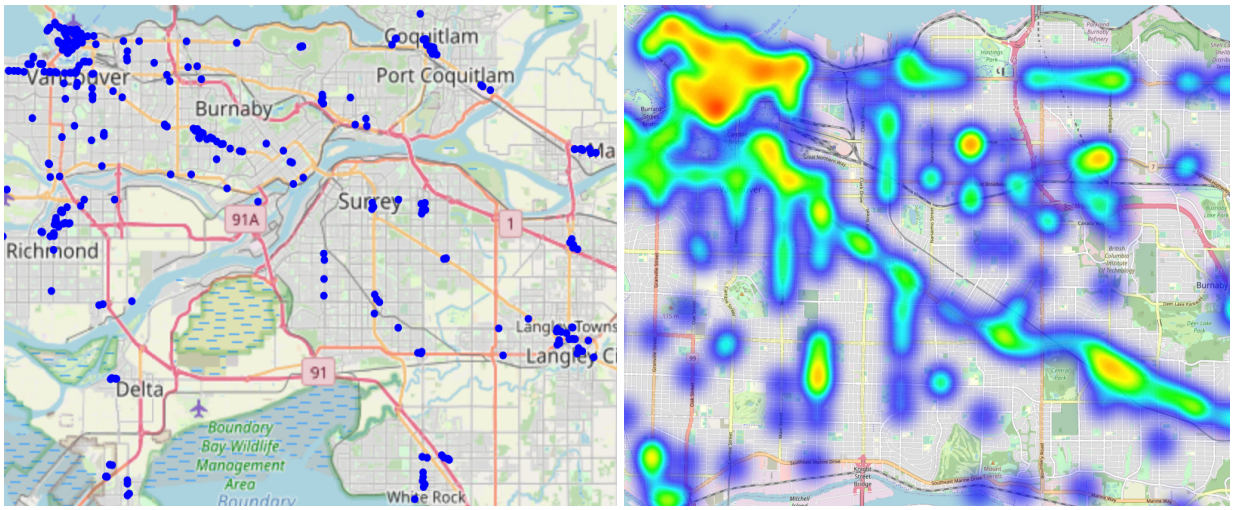
2.3 Location analysis of chained restaurants

The spatial distribution of restaurants is mainly based on geographic agglomeration. Restaurants that are significantly adjacent in geographic space can be classified as a cluster. The left figure demonstrates the distribution of all restaurants in Vancouver, and the heat map of the degree of aggregation is shown in the figure on the right.



It can be easily observed that the restaurants are mainly distributed near Downtown Vancouver as well as major highways. The distribution trend of the restaurants is taking Downtown Vancouver as the center point, with the main road as the web. The restaurants are mainly distributed on the center point and along with the web. In other non-chain areas, there are only a few distributions of restaurants, and it is difficult to form clusters.

The figures below show the distribution of all chained restaurants and the heat map of the degree of aggregation.



It can be observed that the distribution of chained restaurants is similar to the distribution of all restaurants, as they are concentrated near DownTown and some major highways. The density and distribution of chain restaurants are roughly the same as that of non-chain restaurants. The main reason may be that the density of the population determines the number of restaurants in the cluster.

3. Limitations

- The data for amenities is not complete enough. To accommodate for this, we have added a significant amount of famous tourist attractions added to the data provided to us and merged the two data sets. Although the new Amenities data set covers most places, it may still miss out on some unpopular tourist attractions.
- The scope of all data sets so far is only limited to the Vancouver area. Both amenities and hotel data are in the Vancouver area. Therefore, are larger data set that is more complete will be required if we hope to expand the service to more regions in the future, such as Burnaby and Lougheed.
- Our definition of a chained restaurant may not be detailed enough. We categorize a restaurant to be chained if the name of the same restaurant appears more than once in different locations. As a result, we may have categorized franchised restaurants as chained restaurants.

4. Conclusions

Our program is designed to help users choose a hotel in a suitable location with desired amenities. Our methodologies and techniques allow the program to effectively help users choose a nearby hotel, and provide the user with a list of nearby amenities located on the map. Additionally, the program can also benefit the user by recommending the most suitable travel route for the user, based on the user's starting and ending destination points. Our algorithm, therefore, saves both time and energy for the user in finding a travel route suitable to meet his or her individual needs. Lastly, the program also provides users with a distribution and heatmap of both chained restaurants and non-chained restaurants.

5. Project Experience Summary

Siwei's Accomplishments:

1. Extract the latitude and longitude data from the Exif data of the photo using GPSphoto in the Python library.
2. Experimented with different methods of calculating recommendation routes for users to get the best experience.

Zhuo's Accomplishments:

1. Implemented visualization of the heat map of the amenities around the hotel by using folium library in Python.
2. Cleaned up and organized the data set to make the subsequent analysis more accurate.

Teliang 's Accomplishments:

1. Coordinated with the team members to locate the specific problem to be solved and conduct a framework for the work content.
2. Implemented visualization of amenities on maps by using folium library in Python.