CMPT 353: Computational Data Science

Simon Fraser University

Summer 2020

Project Report:

OpenStreetMap, Photos, and Tours

Project Team:

Qianhui (Maggie) Xu	301260853
Vaanyi Igiri	301296662
Aleksandar Vranjes	301290015

Date:

August 13, 2020

Table of Contents

Project Overview

We decided to use the OpenStreetMap data as well as additional external data (which is provided in the project repo) to implement a few different things the user can do with it. There are three options for running the program. The user can supply a collection of photos to the program and the program does one of the user's choice from the following:

- Take the raw paths between all photos and choose from a selection of attractions/amenities they would like to see the most of near the photo locations, i.e. restaurants, parks, recreational facilities, etc.
- Find the shortest path from the start and end of their tour (Printing the new shortest path coordinates into a gpx file titled smoothed.gpx)
- Recommend to the user the best Airbnb with the most restaurants and bars nearby in a given Vancouver neighbourhood and generate a route with the most interesting variety of amenities

Approaches and Methods

Data Collection

We used the given OSM data (amenities-vancouver.json.gz) and imported it into a Spark Dataframe in closeAmenities.py. The dataset contains a total of 17,718 unique amenities with 6 different columns which included amenity characteristics, geographic information, and some tags associated with WikiData.

In order to make recommendations on Airbnbs, we acquired the data (*listings.csv.gz*) from Inside Airbnb (http://insideairbnb.com/get-the-data.html), which publishes Airbnb listings data for several cities. In our case, we have used the detailed listings which consist of 106 features for 5,618 Airbnb listings as of June 2020. This, in conjunction with geographical data outlining Vancouver's local area boundaries, enabled a feature through which users can search for Airbnbs throughout several districts in Vancouver. The names of the accounted-for districts can be found in the *neighbourhoods.csv* file.

After referencing several points of latitude and longitude to get a feel for the scope of the dataset, to test the feature of finding geographic information in photos, we gathered two sets of data to use. The first was a set of three ferret photos in the scope of about one small city block around Production Way in Burnaby and the second was a set of seven photos in a wide tour of the Greater Vancouver area: starting from Rocky Point Park in Port Moody and ending on a busy street corner in West Vancouver, with locations such as a small mall near Commercial Drive and East 1st Street and the New West pier in between.

Data Understanding and Preprocessing

By analyzing the OSM data in this huge dataset, we initially decided to omit some amenities that were deemed unnecessary or otherwise uninteresting to a user who had larger attractions in mind. This included benches, parking spaces, toilets, and various other smaller, insignificant locations. But upon manually analyzing the OSM data, we realized there were many different kinds of amenities and decided to offer the user sixteen specific kinds of amenities, as well as an option for seeing every kind in chooseAmenities.py. In addition, we noticed that there are some tags that are associated with WikiData. We found them useful for determining which amenities are attractions, so we filtered those with the Wiki tag 'tourism', and included them when planning a tour past those attractions and other amenities in tourPlanning.py.

For the Airbnb dataset, we dropped columns that were not relevant to the problem like URL, host picture, etc. Besides, we noticed that the price column had some outliers, however, upon closer examination, we observed that for some the price given was for minimum nights and not per night. Thus, to calculate price per night, we filtered prices over $1,000 and divided them by minimum nights.

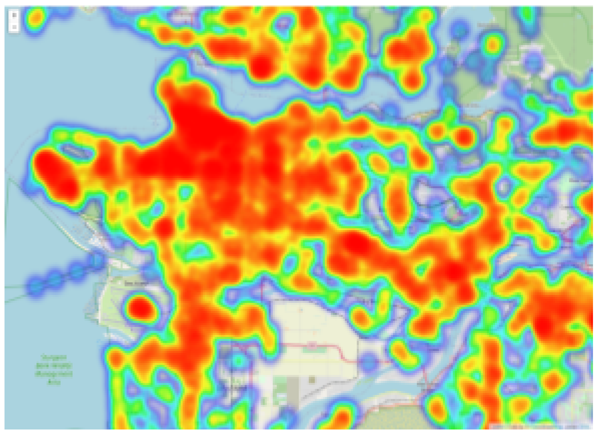


Figure 1: OSM dataset heatmap

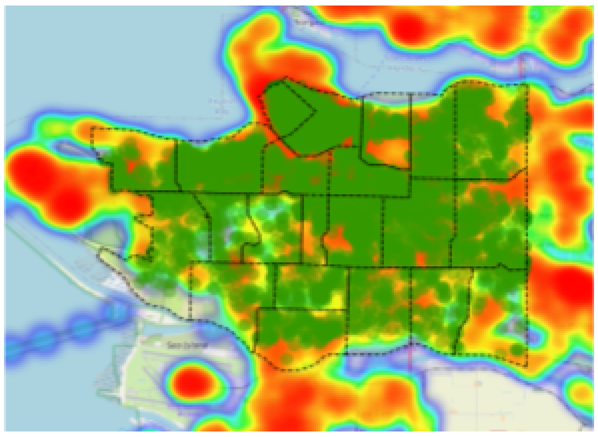


Figure 2: OSM heatmap with Airbnb dots and Vancouver local boundaries

By generating the OSM dataset heatmap, we noticed that the Vancouver area has the most density of amenities, and by plotting a dot for each Airbnb in the map, we found that it only contains Vancouver's Airbnb information. With these photos, we pulled out the EXIF information then extracted and transformed the latitude and longitude information in our getRoute.py so that we can use them to calculate our user's path.

Data Analysis Techniques

Get Route

Using the code from Exercise 3 as a base, we wrote the program getRoute.py to extract the EXIF info from a folder of photos, and use their latitude and longitude to calculate the distance between all of them. The unfiltered distance between the three ferret photos was approximately 77 metres (The filtered and unfiltered distances are shown and the path coordinates are printed to gpx files when you choose the second option in main.py). Once we had the program working as intended and tested on the first set, we went out and gathered a larger set of data. The total unfiltered distance of the second route was approximately 48,000 metres (48km) as the crow flies. We then used a Kalman filter to smooth the distance between the first and last photo locations in the set, creating the shortest path with new coordinates in between.

Close Amenity

For the first feature, we wanted to accomplish a few things. Using the EXIF data from a given user's photos, we aimed to calculate the shortest path traveling through each of the coordinates (lat/lon) given by the photos. We then wanted to calculate a path along with the photo coordinates that wasn't necessarily the shortest but yielded the most amenities. Finally, we hoped to make a combination of both – the shortest path with as many amenities as possible.

In our approach, we first needed to calculate the distance from each of the seven photos given in van-tour-photos to each amenity given by the OSM data. We converted the OSM data from a Spark to Pandas Dataframe ensuring a matching format across all pieces. Next, we assigned a common key to both which allowed us to perform a full outer join and create distinct combinations of all amenities and each photo coordinate as given by the EXIF data. We were then able to apply a derivative of our distance function to determine the closest amenities – defined by a threshold of 100 metres – and manipulate them as required.

Choose Amenity

As part of our presentation for this feature, we derived an interactive prompt allowing users to select what type of amenity they would like. This included a wide array of sixteen options (and a seventeenth for ‘all’) ranging from public washrooms to entertainment centres and healthcare facilities. Subject to the user’s choice input, we would then offer them a list of locations to close-by amenities, sorted by their distance in metres.

Tour Planning

We developed this elegantly simple feature with tourists in mind. It allows its users to find Airbnb’s that most suit their specifications in terms of location, method of transportation, cost, and the density of surrounding amenities. When our user is planning a tour of the city and going to choose an Airbnb, the feature allows the user to choose from several districts in Vancouver, select the room type, and input a maximum price to narrow down the range of Airbnb listings. Further, we used the OSM data to choose the Airbnb with the most restaurants and bars nearby.

We hoped to explore more of the usefulness of the OSM data, so we implemented a feature that planned the tour for our user. First, we ask the user his/her preferred way of traveling (by walking/biking/driving). If the user chose walking, we will only consider the amenities within 1.5 km with the selected Airbnb as the center, if by biking, the range will be 3.5 km, and driving will be 15 km. Then, from these amenities, we filtered the attractions, leisure, parks, etc. which the tourists might be interested in and chose the nearest one for each type of amenity. With recording all the locations’ geographic data, we could then generate a route by different traveling methods that pass through an interesting variety of things.

Such versatility would enable those with limited means of transportation to receive recommendations to listings that afford them the most interesting and diverse experience possible, regardless of their limited mobility. Users with little to no mobility issues are presented with an even more comprehensive variety of choices.

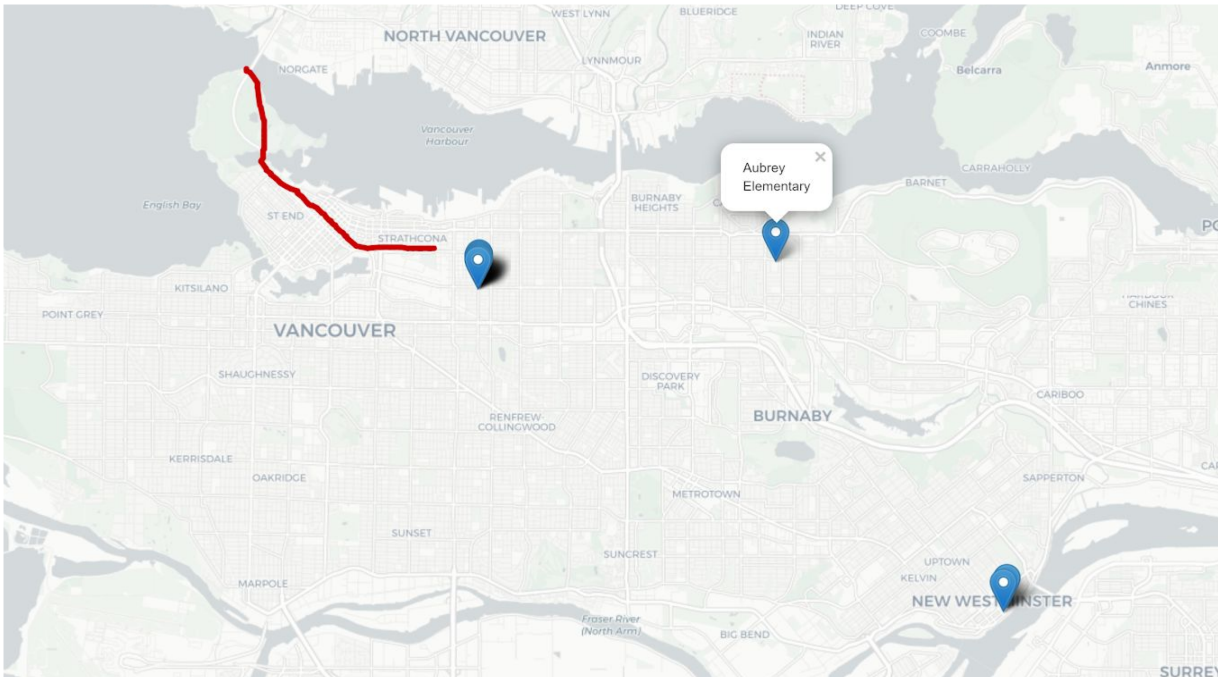
Results and Conclusions

During our group project, we tried out many methods of data cleaning, processing, and map visualization to show how the OSM data (and other data) can be used for many different tour-planning tasks. With the implemented approaches, we offer a proactive and retroactive method of exploring Greater Vancouver, whether you are a resident of the city or a curious tourist. The retroactive approach lets the user supply photos of their trip and locates all the nearby amenities while generating the total distance of their path travelled. The proactive approach is implemented by recommending the best Airbnb in a chosen Vancouver neighbourhood (with room types and budget in mind) and planning the most interesting path for our user, whether by foot, bike, or car.

Data Visualization

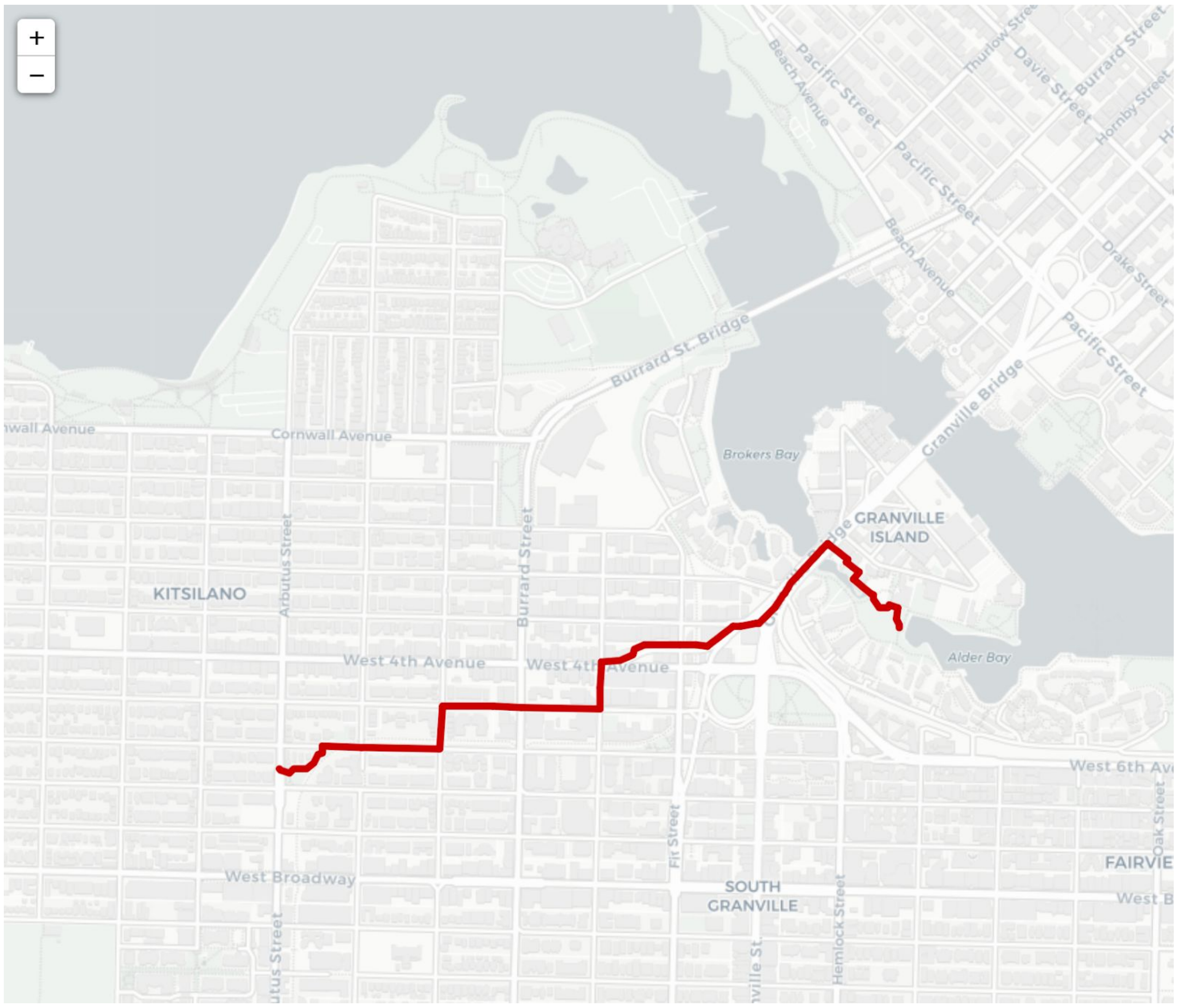
All the visualizations that have been created are in Python. For map visualization, we used both Folium and OSMnx library. Folium enables binding of data to a map for visualizations, and OSMnx can model walkable, drivable, or bikeable urban networks with a single line of Python code.

With our second set of photos, we generate a route that starts from Stanley Park to Grandview-Woodland district. The blue markers are the amenities that are close to the photo location, the name of the amenity will appear when you click on it.



Unfortunately, we generated the route by using OSMnx network_type = 'walk' instead of ‘drive’ and the route is incomplete. We think it might be due to the API being fragile and it cannot handle so many requests, so the bike and drive functions seem to have some errors.

As for the planning tour phase, we did a test case by choosing walking as the travel method and selected Kitsilano as the desired Airbnb location, then filtered with the entire home or apartment. We then input the maximum price per day as $500. The planned route is shown below.



Limitations/Issues/Future Plans

We encountered a few issues and limitations while designing the program. The first limitation was that while we could plot an extremely accurate map of where each photo was taken and connect them together, the amenities we filtered with closeAmenities.py only chose amenities within 100 metres of the photo locations, and not amenities that were along the path *between* the photos as well.

Another limitation we faced was during the process of implementing the third option with Airbnb information in the Vancouver area. While the OSM data contained information of locations in the Greater Vancouver area (i.e., Burnaby, Tri-Cities, Surrey), we could display the Airbnb data for the municipality of Vancouver alone. Using the map modules, we displayed the amenities and Airbnbs in Vancouver as shown in the generated file van_heatmap.html. Unfortunately, while the first option in the program can print the amenities near the photos taken, output the csv, and generally works as expected, when printing the your_route.html file you'll notice that the only part of the tour from van-tour-photos mapped in red is the part in the city of Vancouver. However, it does successfully show the locations of the amenities, even if they are not in Vancouver.

A bug in the code we could not solve in time was a couple of options for Vancouver districts in the third method showcasing the Airbnb information. When you choose districts seventeen or twenty-two (Shaughnessy and West End respectively), the program is not able to finish properly, as explained in the data visualization section as well.

A future feature we did not have the time to implement would be the combination of the shortest distance tour from a set of photos and filtered amenities close to the new coordinates. We used a Kalman filter to find the smoothed route from the first and last photo locations as explained previously, but we did not include functionality to find amenities close to the new coordinates given.

Accomplishment Statements:

Maggie Xu

- Formed a team of 3 people.
- Acquired datasets from Inside Airbnb and gathered 3 ferret photos for testing (test-photos).
- Gathered useful EXIF information from photos and exported them as a csv.
- Cleaned and manipulated data for better decision-making practices.
- Developed and completed the idea of the tour planning feature.
- Visualized data on an interactive map using OSMnx and Folium library.

Vaanyi Igiri

- Created distinct combinations of coordinates corresponding to the users' travelled path and all available amenities.
- Modified function to calculate distance between coordinates for each photo and each amenity.
- Sorted and filtered resulting dataset to include amenities within agreed upon threshold.
- Conducted isolation testing to help identify and polish bugged features.

Aleksandar Vranjes

- Gathered seven photos in the Greater Vancouver area to use for testing (van-tour-photos).
- Wrote chooseAmenities.py to give the user options for amenities.
- Wrote main.py to run the entire system and delete.py to delete files after execution.
- Edited all files to create uniform coding conventions, comments, and design.