Vancouver Tour Planner
CMPT 353-D100 Final Project

Daniel Bogusz
#301237992

2022 Fall Semester

1. Report

1.1 Introduction

For this project I ran with the idea of a Vancouver trip planner. My intention was to create an app which allows users to input an initial point and a destination point (specified by latitude and longitude) and outputs a map which indicates a route (by street) between the initial point and the destination which passes potential tourist attractions. I expanded this concept by adding customizability: users may specify the number of points, the maximum distance they are willing to go out of their way, and how many unique routes to generate; these routes can be cycled through.

1.2 Data

The data used in this project was obtained from OpenStreetMap. The amenities data was provided and stored locally, while the graph data (streets) was obtained using osmnx.graph_from_place(); but both sets of data are stored locally before generating routes for efficiency.

1.3 Techniques

The amenities data is cleaned by removing unnecessary columns, and filtering for wanted rows; this is done by isolating tags and looking for wiki entries, a tourism tag, or other tags likely to be be of interest to tourists. This task was largely trivial, through given the tags are stored within the column as a map which required unique filtering methods (checking for specific tags, counting the number of tags, comparing tags, etc., while using Pandas efficient operations). Once the separate categories of desired amenities have been found, the data is concatenated, and then checked to ensure no necessary columns have missing values.

I realized during this process that it is scaling this app to other cities would not be easy, as the use of tags on OpenStreetMap is not consistant. This would mean that inclusive filtering would cause many amenities to be left out, while exclusive filtering would cause far too many amenities to be left out.

After the data has been prepared, the user may begin generating routes using **route.py**. After a source and destination are specified, a linear regression is used to limit the scope of the data to an area between and around the source and destination. This is done in part to make the rest of the algorithm more efficient, ignoring amenities which should not be chosen; but also to allow users to specify the maximum distance they are willing to go out of their way during their travel. This is specified by the **max_distance_multiplier**, I use a multiplier rather than a static value because it makes more sense to make this value relative to the distance between the source and destination.

Next, the best amenities for the route are chosen. This was the most time consuming part of the project, and I went through several iterations before choosing the best sorting metric. Amenities are sorted based on being as evenly spread as possible. Users specify the **num_amenties** they wish to see along their route.

Locations are then converted to nodes on the graph using a built in **nearest_node** function; this works like a **Nearest Neighbor** function, selecting the nearest node in the graph with respect to the amenities latitude and longitude.

Once all nodes have been generated, the route is constructed using a **weighted graph traversal**. The weight is determined purely by distance between the nodes. This is appropriate, as the app is intended for tourists walking on foot. (If the app were accounting for drivers, other consideration would be needed.)

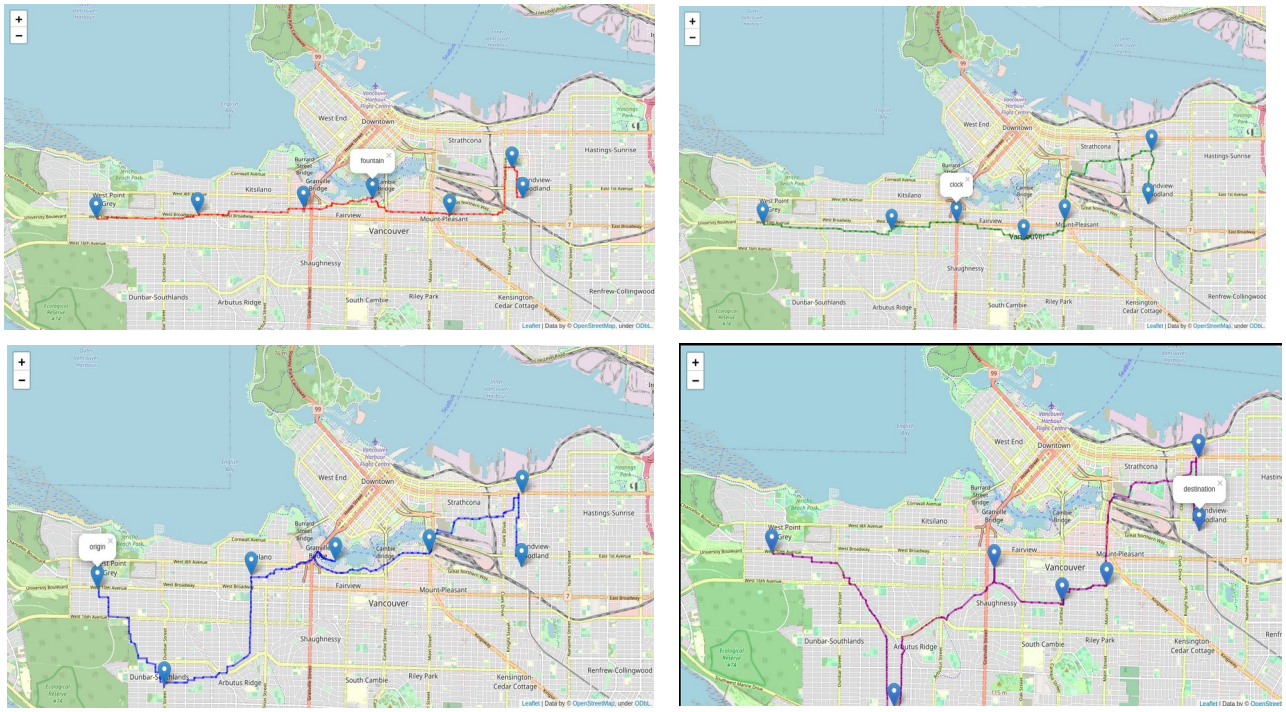
After the first route is generated, the process is repeated until the user specified **num_routes** have been generated. The algorithm is written in such a way as to ensure the same amenity is not chosen for more than 1 route, guaranteeing each route is wholly unique.

Loops are never used for altering dataframes, however, loops are used for other logic.

1.4 Results

Examples of the generated route are shown below.
Note: these images are multiple routes generated from the same starting point and destination.
These results can be replicated using:

python3 route.py landmarks_data.csv graph.osm 49.2635 -123.2067 49.26752 -123.07181 output 5 1.5 4



Several things to note:

- Colours are cycled through to clearly distinguish routes.
- Each amenity along the route has a marker, and each marker is given it's name: the type of amenity. (Origin and Destination are named as such.)
- Because potential amenities are removed after each route, and because each route is calculating the 'best' route, each subsequent route will be 'worse'.

1.5 Challenges

My project was developed using two mapping methods: folium, and ox plot. I was familiar enough with either to know which would be better. After implementing both and testing, I discovered folium not only looks much better, but is far faster as well. For this reason, ox plotting, and the code needed to display it a series of plots, has been completely removed.

1.6 Considerations

At present the app only works for the West Vancouver area. If I had more time I would extend this app to function globally. In order to do this I would need to wait to calculate the graph until after knowing the scope of the route, then fetch the data from OpenStreetMap. This would increase the runtime. At present, my code is (relatively) very efficient, the bottleneck is the size of the graph. Using timers, I have determined that the code I have written to construct logical routes (a list of amenities) can be done at least a dozen times before the external functions construct a single route. For this reason, I prefer to perform data fetching ahead of route generation, and limit the size of the graph to West Vancouver.

Notes:

- Currently, trying to use inputs/outputs from other regions will simply crash the program.
- I have chosen to include ‘touristy’ amenities that are not specifically labeled for tourists, as this improved route generation in a limited area.

2. Project Experience Summary

In order to complete this project I obtained real world geographic data and filtered tens of thousands of Open Street Map data points in order to create usable dataframes in Pandas.

I designed and tested a working application given a pre-determined specification. This process demanded I familiarize myself with a number of external tools and libraries, including folium, osmnx, and networkx.

I implemented several statistical techniques in order to isolate data in an efficient manner, including Linear Regression, data de-skewing, dataframe filtering and sorting, nearest neighbor finding, and weighted graph traversal.

I am proud of my final product. I created a functional and practical route planner with unique features. It works entirely as intended (within the scopes of the available data).

3. Testing

For testing this application, please use the provided **demo1.py** and **demo2.py,** and see the **README** for instructions on all operations.

Thank you.