Where2Move

Capstone Project Presentation

Overview

- Motivation
- Data and implementation
- Example of usage
- Future directions

Motivation

- Answering the question “**Which area of A is most similar to areas around B?**” would normally involve much manual work, compiling data from various sources and then comparing the available data to identify the best match
- Example:
  - John Doe is living in Toronto, Canada, but he got an exciting job offer in Cambridge, UK
  - He likes the area of Toronto he currently lives in, and would like to have a similar setting in terms of the venues available to him within a short distance
  - The software presented here, **Where2Move** can help John to find the area in Cambridge which is most similar to his current location in Toronto

Motivation (II.)

- **Where2Move** is a lightweight software that aims to address the following 2 main use cases:
    - **A.)** A user is considering to move to a new location and would like to get assistance in terms of finding an area that is most similar to the user's current location
    - **B.)** A user thinks that it would be great to live at a certain location, they can get an idea of which area around their current location is most similar to the dream location in terms of available venues

Data and Implementation

- **Where2Move** is implemented in Python 3.6+
  - Can be used as a Jupyter Notebook, or exported as a Python script
- Package dependencies
  - requests
    - Used for sending GET requests
  - pandas
    - Used for data parsing and transformations
  - scipy
    - Used for hierarchical clustering and dendrogram plotting
  - folium
    - Used for rendering interactive maps

Data and Implementation (II.)

- Data used by **Where2Move**
  - Input coordinates in decimal degrees format
    - Start locations coordinates, e.g. the user's current locations
    - Destination location coordinates, e.g. a city the user plans to move to
  - Venues/Locations data
    - Used the Places API of Foursquare
      - Documentation available at https://tinyurl.com/y4e9qmx5
      - Limited number of API requests per day (using free plan)

Data and Implementation (II.)

- Workflow of **Where2Move**
  - The user provides 2 pairs of latitude and longitude coordinates, “location” and “destination”
  - A grid of 25 areas, each sized 2x2 km, centred around the “destination” coordinations is generated
  - Venues data is retrieved using Foursquare’s Places API for the 25+1 coordinates, with a radius of 1 km
  - The retrieved data contains venues
    - The data for all coordinates are merged
    - The venue categories are then encoded (0/1) in a data frame
    - The venue categories are summed up per locations
  - The combined, encoded and summed up data frame is then clustered using hierarchical clustering
    - The “destination” grid which is most similar to the input “location” (i.e. the closest neighbour) is retrieved
    - The best hit area is then rendered on an interactive map

Example of usage

Step 1. - Initialise the program with input coordinates

Step 2. - Run the calculations

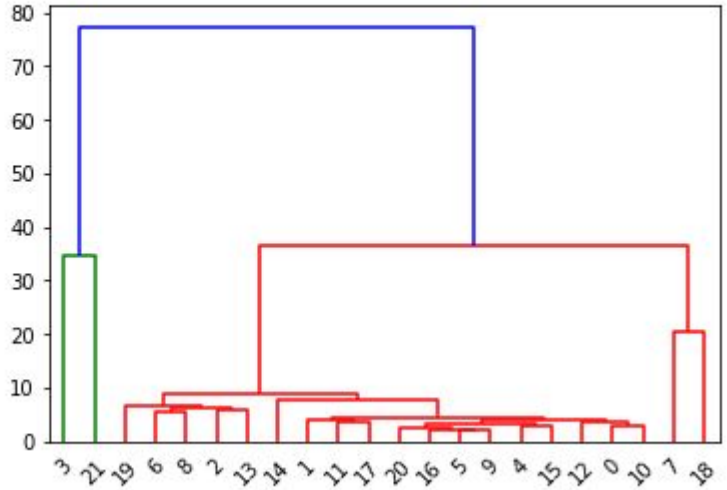
The example below will initialise the program with coordinates for central Toronto, Canada as the current location (“location”) and central Cambridge, UK as the destination location (“destination”):

```
sl = SimilarLocations(location_latitude=43.704689, location_longitude=-79.403590, destination_latitude=52.1988895, destination_longitude=0.0849678)
sl.run()
```

Example of usage

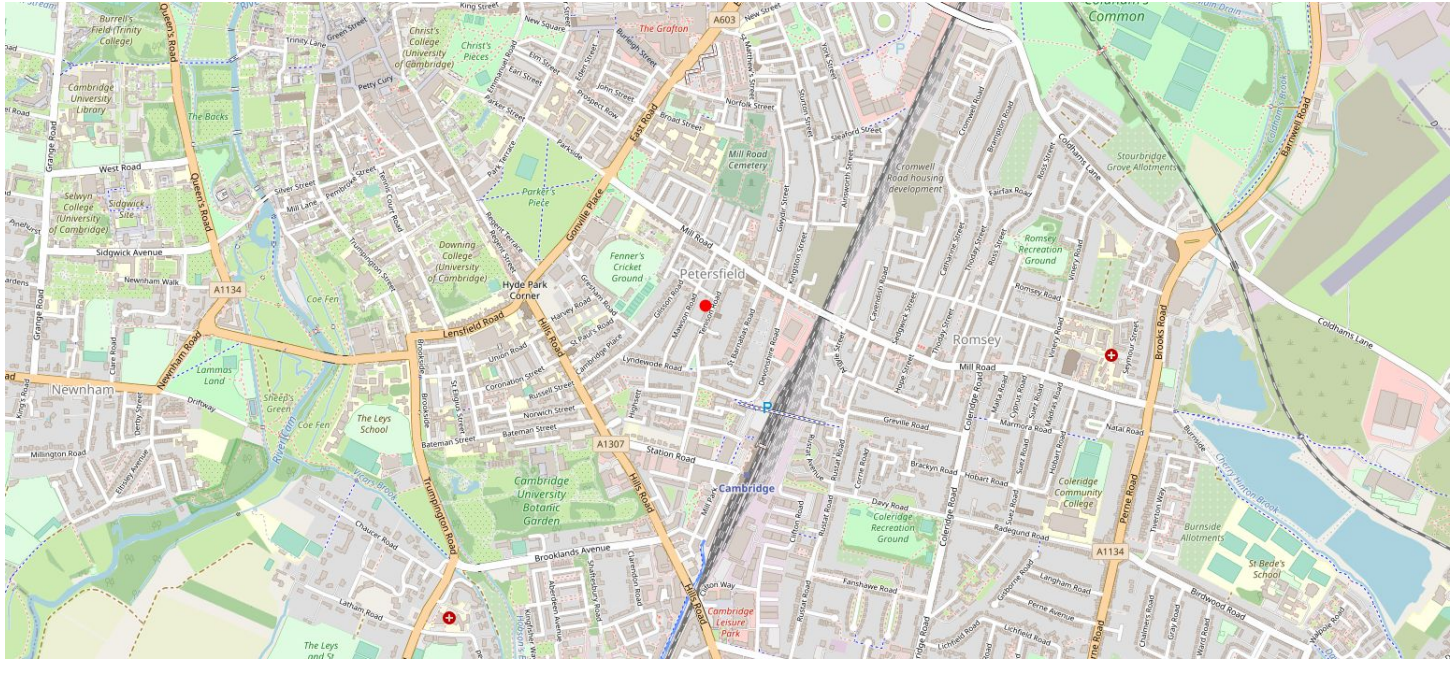
- The program will then retrieve data for central Toronto, and 25 locations centred around Cambridge.
- The data is then processed, transformed and clustered
- The hierarchical dendrogram can be displayed by calling the “render_dendrogram()” method
- In this example, the green fork has the location in Toronto (21) and the location in Cambridge which is most similar to it (3)

```
# Display the dendrogram
sl.render_dendrogram()
```



Example of usage

- The map of the destination with the best hit area highlighted with a red circle is rendered:



Example of usage

- Optionally, the venues can be displayed for comparison between the “location” and the best matching “destination” area
    - This allows for a more detailed look at what venues are available

```
In [50]: sl.get_location_and_best_hit_venues()
```

Out[50]:

	location	destination_area_12
African Restaurant	0	2
Airport	0	0
Airport Lounge	0	0
Airport Terminal	0	0
Arts & Crafts Store	1	0
Asian Restaurant	0	1
Athletics & Sports	0	0
Auto Garage	0	0
BBQ Joint	0	1
Bakery	3	2
Bank	1	0
Bar	2	6
Bed & Breakfast	0	0
Beer Bar	0	0
Beer Garden	0	0
Bookstore	2	1
Botanical Garden	0	1
Breakfast Spot	1	1
Brewery	0	1
Bubble Tea Shop	1	0
Buffet	1	1
Burger Joint	1	2
Bus Line	0	0

Future directions

- There are a number of potential improvements to consider:
  - Support using city & country as input instead of decimal degree coordinates
  - Support changing the range of the venues the API returns
  - Support changing the limit of the API
  - Provide not only the best hit but also a sorted list of all the locations from the destination grid
  - Support larger grids (not only 5x5)
  - Allow the user to filter what venue categories are irrelevant and/or emphasise venue categories that should weight more
- These would fall outside the scope of a 2-weeks long project though