Capstone Project Report

IBM Data Science Specialisation

Introduction

The proposed software will have the following main functionality. Provided an input location and a destination, it will investigate various characteristics of the input location, and suggest the most similar locations within a predetermined area of the destination.

This functionality is expected to address the following two use cases:

A.) Suppose a user would like to move from his/her current location to a new city, perhaps due to professional reasons, climate change or religious beliefs. It is difficult to find an area that has similar characteristics to the location the user is moving away from, and this software could help

B.) Suppose a user has a “dream” location in mind and would like to see if something similar is available closer to his/her current location.

The Minimal Viable Product (MVP) of this software will do the following:

- Accept 2 input coordinates
- Retrieve information on various characteristics of around the 2 input coordinates
- Perform hierarchical clustering
- Display the top 10 areas within a predefined range of input coordinate 2 most similar to input coordinate 1

Optional features:

- Allow users to set the range parameter for the 2 input coordinates
    - E.g. Find the top 10 most similar locations within a 10 miles range of input coordinate 2
- Allow users to set the limit for displaying the top *n* most similar locations
- Allow users to specify characteristics which should be considered
    - E.g. There needs to be a school in the are and a Mexican restaurant
- Allow users to provide city and country names as input instead of coordinates

Data

This software will use publicly accessible location data from Foursquare - As prescribed by the capstone project assignment.

For the MVP, no additional data source is required. The user will provide 2 sets of coordinates and the software will retrieve location characteristics data for both locations using the Foursquare API. The data will be parsed, cleaned, trimmed and combined into a data frame. Hierarchical clustering algorithms will be deployed on the data frame, centred around the input coordinate. The output will be a map, using the folium package, that will highlight the top 10 locations around the second input coordinate which are highly similar to the first input coordinate.

A high-level overview of the workflow:

1. The user provides input parameters (***location***, ***destination***, ***range_limit***, ***top_hits***)
2. The software calls the Foursquare API and gets data within a 1-mile range of ***location*** and within ***range_limit*** miles of ***destination***
3. Both JSON data will be cleaned, selecting useful features and discarding others
4. The 2 data frames will be combined
    a. ***n*** columns (selected features)
    b. ***m*** rows (1 row for ***location*** and m-1 rows for ***destination***)
5. A hierarchical clustering algorithm will be called, centred around the row of ***location***
6. The results will be sorted by similarity to the row of ***location***
7. The ***top_hits*** number of similar rows will be listed to the user
8. The ***top_hits*** number of similar rows will be plotted on a map

Examples for the data used:

```
get_similar_locations(location=(43.753259, -79.329656),
                destination=(43.746143, -79.324630),
                range_limit=10,
                top_hits=10)
```

A Foursquare API endpoint is then called to get venue information for both ***location*** and ***destination*** using the following URL:

```
venues_url +
'/explore?client_id={}&client_secret={}&v={}&ll={},{}&radius={}&limit
={}'.format(
    CLIENT_ID,
    CLIENT_SECRET,
    VERSION,
    lat,
    lng,
    radius,
    range_limit)
```

Example (fragment) of the returned data:

```
{'reasons': {'count': 0,
       'items': [{'summary': 'This spot is popular',
         'type': 'general',
         'reasonName': 'globalInteractionReason'}]},
      'venue': {'id': '51819802498ee3c6834b8e0b',
       'name': 'De Mello Palheta Coffee Roasters',
       'location': {'address': '2489 Yonge St',
        'crossStreet': 'Erksine Ave',
        'lat': 43.71179137179029,
        'lng': -79.39940300399607,
        'labeledLatLngs': [{'label': 'display',
          'lat': 43.71179137179029,
          'lng': -79.39940300399607}],
        'distance': 679,
        'postalCode': 'M4P 2H6',
        'cc': 'CA',
        'city': 'Toronto',
        'state': 'ON',
        'country': 'Canada',
        'formattedAddress': ['2489 Yonge St (Erksine Ave)',
         'Toronto ON M4P 2H6',
         'Canada']},
       'categories': [{'id': '4bf58dd8d48988d1e0931735',
         'name': 'Coffee Shop',
         'pluralName': 'Coffee Shops',
         'shortName': 'Coffee Shop',
         'icon': {'prefix':
'https://ss3.4sqi.net/img/categories_v2/food/coffeeshop_',
          'suffix': '.png'},
         'primary': True}],
       'photos': {'count': 0, 'groups': []}},
      'referralId': 'e-0-51819802498ee3c6834b8e0b-1'}
```

The resulting low-level categories (e.g. Cafe, Caffee Shop, etc) will then be mapped to higher-level categories, provided by another FourSquare API endpoint at URL:

```
venues_url +
"/categories?&client_id={}&client_secret={}&v={}".format(
    CLIENT_ID,
    CLIENT_SECRET,
    VERSION
```

)

Example (fragment) of the returned data:

```
{'id': '4bf58dd8d48988d182941735',
    'name': 'Theme Park',
    'pluralName': 'Theme Parks',
    'shortName': 'Theme Park',
    'icon': {'prefix':
'https://ss3.4sqi.net/img/categories_v2/arts_entertainment/themepark_
',
     'suffix': '.png'},
    'categories': [{'id': '5109983191d435c0d71c2bb1',
      'name': 'Theme Park Ride / Attraction',
      'pluralName': 'Theme Park Rides/Attractions',
      'shortName': 'Theme Park',
      'icon': {'prefix':
'https://ss3.4sqi.net/img/categories_v2/arts_entertainment/themepark_
',
       'suffix': '.png'},
      'categories': []}]},
```

All the low-level categories will be merged based on the higher-level category (i.e. parent category) they belong to and grouped by the area they belong to (1-by-1 mile grid around the coordinates). These merged and grouped categories will be the features used for the hierarchical clustering algorithm. The features would look something like this:

Arts & Entertainment	College & University	Food	Nightlife Spot	Outdoors & Recreation	Professional & Other Places	Residence	Shop & Service	Travel & Transport
1	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0
0	0	0	0	1	0	0	0	0