Analyze and Compare Tech Hubs in Canada

Shivani Sheth

4 June 2020

1. Introduction

1.1 Background

Canada is one of the fastest-growing technical hubs in the world, with an increasing number of resources and scholarships provided by the government to attract bright minds all over the world. This is especially in the case of STEM students looking for study opportunities in the country. One of the most difficult questions that one faces while deciding upon a university/ college is its location. I myself being a prospective student in Canada, feel that a suitable neighborhood is an important factor as it not only enhances a student's college experience but also helps decide a preferable location for internships or exchange programs. Hence, here we explore and analyze the province of British Columbia, which is one of the top technical hubs of the country, in terms of science and engineering, and compare it with the previously analyzed data for Toronto, which is also famous for its prime opportunities in the field.

1.2 Proposal

In this project, we take leverage of the Foursquare API to find the most popular places in each of the neighborhoods consisting of British Columbia. A place is marked as "happening" by the Foursquare API according to the number of people present at a given place and hence the place is updated in real-time; it might change every few minutes. We then cluster the neighborhoods based upon their preferred places in the surrounding area. This will give a clear picture of the aura or the vibe of the place, which can help an individual know what to expect in the neighborhood, and hence decide upon a suitable location according to their preference. Finally, we also compare it with Toronto, another widely preferred city among prospective students, and brief upon the similarities and dissimilarities based upon its neighborhood.

1.3 Interest

This analysis can be useful for many people in different domains. First, it could be

used by business owners who might want to target students in the STEM domain, such as manufactures in the IT industry. Second, the analysis could be useful to the entrepreneurs in the engineering field, who are likely to open up their startups in the technical hubs of a country, to attract bright young minds freshly out of college. Third, it can also be used by prospective STEM-based students like me, to decide upon a preferable location for their universities, colleges, or exchange programs. Last but not the least, it can be used to compare the environment between the top two technical hubs of Canada and the type of crowd in the different neighborhoods, which could interest companies planning to shift their headquarters or open up a new branch.

2. Data

2.1 Acquisition

The data acquired for the purpose of this project can be obtained in many ways. The primary datasets that we need are the postal codes of the province of British Columbia, along with their latitude and longitude coordinates. After many tries with various datasets, I came upon the complete dataset used in this project on 'Postal Codes British Columbia, Canada - GeoNames'. We scrape the table from the given HTML page using the read_html function of the Pandas library. This will give us the primary table needed for the project, as shown in the figure below.

	Unnamed: 0	Place	Code	Country	Admin1	Admin2	Admin3
0	1.0	Port Moody	V3H	Canada	British Columbia	NaN	NaN
1	NaN	49.323/-122.863	49.323/-122.863	49.323/-122.863	49.323/-122.863	49.323/-122.863	49.323/-122.863
2	2.0	Pitt Meadows	V3Y	Canada	British Columbia	NaN	NaN
3	NaN	49.221/-122.69	49.221/-122.69	49.221/-122.69	49.221/-122.69	49.221/-122.69	49.221/-122.69
4	3.0	White Rock	V4B	Canada	British Columbia	NaN	NaN

2.2 Cleaning and Preprocessing

As we see, the scrapped table contains the address information in every alternate record, and following that is a row that contains the coordinates of the location. The coordinates are given in the form of 'lat/long' values, and the entire row consists of the same values. Our aim here is to fetch the alternate rows for the address values and just one cell from its next row for the coordinates.

We begin our data cleaning and preprocessing by extracting alternate address rows of this table in a separate data frame, say 'temp_df'. For this, we loop over

the number of records in a multiple of two and fetch each row. This will give us a total of 192 rows containing the neighborhood address of British Columbia, which looks like this:

	index	Place	Code	Country	Admin1
374	188.0	Vancouver (Strathcona / Chinatown / Downtown E...	V6A	Canada	British Columbia
376	189.0	Vancouver (NE Downtown / Harbour Centre / Gast...	V6B	Canada	British Columbia
378	190.0	Richmond South	V7A	Canada	British Columbia
380	191.0	Duncan	V9L	Canada	British Columbia
382	192.0	Parksville	V9P	Canada	British Columbia

Next, we delete the rows that have a float value in the index column of the original data frame, say 'original_df' and hence we are left with only the coordinate values in the table. Now, we keep any one of the columns and remove all the other ones as all the cells of a row contain the same data, and hence it is redundant. Hence, we obtain the coordinates for each address in one column. As the columns in both data frames, the 'original_df' and the 'temp_df', have data ordered according to their sequences in the table, we can merge both of them back into the original_df. We thus have all our data, a total of 192 records, in a single data frame as shown below:

	Coordinates	Place	Code	Country	Admin1
0	49.323/-122.863	Port Moody	V3H	Canada	British Columbia
1	49.221/-122.69	Pitt Meadows	V3Y	Canada	British Columbia
2	49.026/-122.806	White Rock	V4B	Canada	British Columbia
3	49.481/-119.586	Penticton	V2A	Canada	British Columbia
4	49.866/-119.739	Westbank	V4T	Canada	British Columbia

Now, for processing convenience, we convert the coordinate values into different columns consisting of the latitude and longitude values separately. We do this by storing the coordinate values into a list and by iterating through each value. We split the string by the "/" character, and hence obtain two parts of the same string. The former would be the latitude values and the latter would be the longitude values. We initially append both lat and long values in a list and after the iteration process completes, we assign it to the data frame column. Finally, we rename our columns appropriately, which completes our data cleaning and preprocessing. Our final data frame looks like this:

	Coordinates	Place	Code	Country	Province	Latitude	Longitude
0	49.323/-122.863	Port Moody	V3H	Canada	British Columbia	49.323	-122.863
1	49.221/-122.69	Pitt Meadows	V3Y	Canada	British Columbia	49.221	-122.69
2	49.026/-122.806	White Rock	V4B	Canada	British Columbia	49.026	-122.806
3	49.481/-119.586	Penticton	V2A	Canada	British Columbia	49.481	-119.586
4	49.866/-119.739	Westbank	V4T	Canada	British Columbia	49.866	-119.739

2.3 Alternatives

The same data can also be obtained from various other sources such as the Wikipedia pages that contain the postal codes for all the provinces in Canada, ordered by their regions. The page that can be used to obtain the postal codes for British Columbia, along with their neighborhoods is given here. Although this has the complete information needed for the scope of the project, the table is not ordered in a scraping-friendly manner. Hence, upon scraping, we obtain the information in a manner shown below.

	V1AKimberley	V2APenticton	V3ALangley Township(Langley City)	V4ASurreySouthwest
0	V1BVernonEast	V2BKamloopsNorthwest	V3BPort CoquitlamCentral	V4BWhite Rock
1	V1CCranbrook	V2CKamloopsCentral and Southeast	V3CPort CoquitlamSouth	V4CDeltaNortheast
2	V1ESalmon Arm	V2EKamloopsSouth and West	V3ECoquitlamNorth	V4EDeltaEast

This happens because the data stored for a complete row is in the same box, and hence for each cell in the table, we have the complete address of the region. To iterate through this table and make this data readable, we first store the table values in a list format using the df.values.tolist() function. This gives us a nested list structure that contains the rows and columns of the table which looks like this:

```
[['V1BVernonEast',
  'V2BKamloopsNorthwest',
  'V3BPort CoquitlamCentral',
  'V4BWhite Rock',
  'V5BBurnaby(Parkcrest-Aubrey / Ardingley-Sprott)',
  'V6BVancouver(NE Downtown / Gastown / Harbour Centre / International Village
  'V7BRichmond(Sea Island / YVR)',
  'V8BSquamish',
  'V9BVictoria(West Highlands / North Langford / View Royal)'],
 ['V1CCranbrook',
  'V2CKamloopsCentral and Southeast',
  'V3CPort CoquitlamSouth',
  'V4CDeltaNortheast',
  'V5CBurnaby(Burnaby Heights / Willingdon Heights / West Central Valley)',
  'V6CVancouver(Waterfront / Coal Harbour / Canada Place)',
  'V7CRichmondNorthwest',
  'V8CKitimat',
  'V9CVictoria(Colwood / South Langford / Metchosin)'],
```

Now, we iterate through this list, fetch the first three alphabets of each string as the postal code, and the remaining characters as the neighborhood value. We store this information in two separate lists known as the 'postal_code' and 'neighborhood' and add them to a data frame. Now, our data frame consists of postal codes and neighborhoods in an orderly and readable manner, which looks something like this:

	Postal Codes	Neighborhood
0	V1B	VernonEast
1	V2B	KamloopsNorthwest
2	V3B	Port CoquitlamCentral
3	V4B	White Rock
4	V5B	Burnaby

Once we have obtained the addresses and the postal codes, we can use geocoder from the geopy library to find the latitude and longitude values for each pair of postal code and address and merge that in this data frame. Hence, we will have the same data as the previous method through different datasets.