Final Project Document for SI 507

Jiachen Jiang
jiachenj@umich.edu

1 Project Code

Link to github repo for final project code: SI507_Final_Project. You can find README.md containing instructions for running the code, including all required python packages for the project to work, how to supply API keys as well as a brief description of how to interact with my program.

2 Data Source

GeoDB Cities API is a Web API I haven't used before that requires API key or HTTP Basic authorization. Its challenging score is 4. Twitter API v2.0 is a Web API I haven't used before that requires OAuth. Its challenging score is 6. Since I only used Twitter API v1.1 before, it's different from Twitter API v2.0 which uses OAuth2.0.

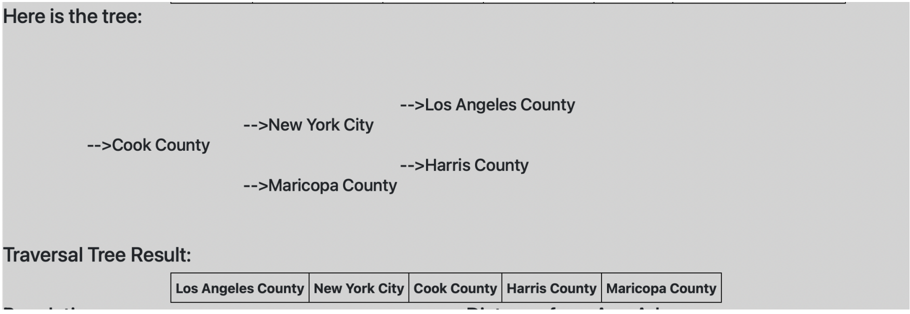
- **GeoDB Cities API** **GeoDB Cities API** provides basic information about cities, counties, regions, and countries throughout the world. It allows us to constrain and sort the data in various ways, retrieving just what's relevant to our use-case. We get cities' information and distance from Ann Arbor through it. The data is in JSON and I accessed by GET method with caching. I use cacheCountry.json to cache results of city list in that country and cacheCity.json to cache the distance from Ann Arbor of each city. I only retrieve the top 5 records. We use the API in two ways. Firstly, we get cities' information by setting Optional Parameters to the country's comma-delimited country codes. It returns several fields, including "id", "wikiDataId", "city", "country", "region", "latitude", "longitude" and "population". Secondly, we get the cities' distance from Ann Arbor using "City Distance" API by setting the Optional Parameters to Ann Arbor's wikiDataId: Q485172. It returns fields of "wikiDataId" and "distance" in km. Since it's a free API, it constrains the rate to one request per second. So it would be a little bit slow when searching for new countries not in the cache.

- **Twitter API v2.0** **The Tweets Look up API** is one of the primary resources on Twitter. In its simplest form, a Tweet can contain up to 280 characters and can be posted either publicly or privately, depending on an account's settings. We look up tweets related to the city on it. The data is in JSON and I accessed by GET method without caching. Since the recent tweets in API are updated and changing all the time, we want to see the recent tweets every time we enter the system. Therefore, caching is removed. I retrieve about 10 records for each city. The important field is "id" and "tweets".

3 Data Structure

In README.md, I explains how data is organized into data structure. Basically, I would sort the cities according to the population and distance from Ann Arbor. I would build a balanced Binary Search Tree(BST) of the cities based on their population or distance from Ann Arbor according the user's input. I would show the tree's structure in the result page. Also, I would show the traversal result of tree in three different ways, including pre-order, mid-order and post-order traversal, according to the user's input.

Figure 1: Example of a BST structure and traversal by Mid-Order



Here is the tree:

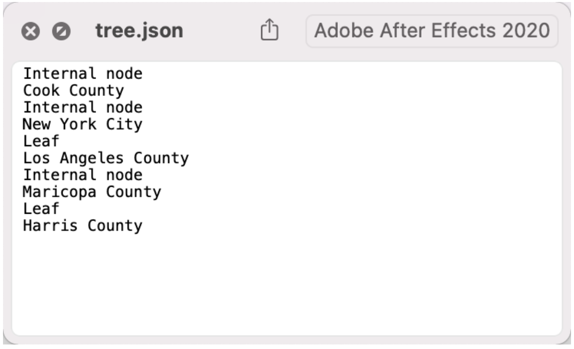
```
                              -->Los Angeles County
                  -->New York City
      -->Cook County
                              -->Harris County
                  -->Maricopa County
```

Traversal Tree Result:

Los Angeles County	New York City	Cook County	Harris County	Maricopa County

Finally, I would save the tree structure to a JSON file. You can load the tree using "loadTree.py". The "tree.py" and "tree.json" demonstrate organization of data into data structures. Screenshots of data structures are as follows.

Figure 2: Example of a tree organized in JSON format



```
Internal node
Cook County
Internal node
New York City
Leaf
Los Angeles County
Internal node
Maricopa County
Leaf
Harris County
```

4 Interaction and Presentation

I would create a **Flask** App that uses HTML links/form elements to prompt for the user to choose country and sort options. Display selected data using HTML tables. I would also use **Plotly** to show the populations and distance. The flask would get the user input from forms and pass it to data API module, then the module would request the GeoDB Citieis API and Twitter API v2.0 with cache to get data and store it to database. Then flask would retrieve related information from database and show it to user.

After supplying API keys, please run "myflask.py" file. You'll see the server is up; click on the localhost this link to view the index page. There are four different 4 different options you can choose to interact with my program.

- You can select a country you interested and find the top 5 cities in population in this country.
- You can choose sort the cities by their population or by the distance from Ann Arbor.
- A tree structure of the cities would be constructed and you can choose to traversal the tree by pre-order, mid-order and post-order.
- You can click the city name to get the recent tweets related to the city.

After selecting all options, you hit search and you'll see all information in a table. Note you can easily return back by clicking on top left "Michigan" icon.

5 Demo Video

Here is the demo video link: Final Project Demo Video