

Data Checkpoint

Name: Kedong He

Uniqname: kedongh

Github repo: https://github.com/kedongh/507_final_proj

Data Source

My data comes from two sources. One is Web API to search some business related to the condition, the other one is Yelp business websites to scrape detailed information.

Firstly, I use Web API to fetch basic information of business according to the given term and the location and serialize the response to JSON format. The endpoint is <https://api.yelp.com/v3/businesses/search>, and there are two parameters for the user to input. The first parameter is search term. It can be a general description, like “food” and “restaurant”, or even a detailed business name, like “Starbucks”. The second parameter is location. It indicates the geographic area to be used when searching for businesses. Each response would contain at most 50 business results. In the response body, some parameters are of interest. I select rating, price, business name, address, Yelp url, review count, phone number, city and Yelp id as the basic information to present to the user and save to the database. To be time efficient, I use caching to store the retrieved results after the first time search. The following is the snapshot of the code that implements caching.

```
138 def make_request(baseurl, params):
139     '''Make a request to the Web API using the baseurl and params
140
141     Args:
142         baseurl (str): The URL for the API endpoint.
143         params (dict): A dictionary of param:value pairs.
144
145     Returns:
146         results (dict): The JSON response from the request.
147     '''
148     response = requests.get(baseurl, params=params, headers=headers)
149     results = response.json()
150     return results
151
152
153 def make_request_with_cache(baseurl, term='', location='', count=50):
154     ''' Check the cache for a saved result for this baseurl+params:values
155         combo. If the result is found, return it. Otherwise send a new
156         request, save it, then return it.
157
158     Args:
159         baseurl (str): The URL for the API endpoint
160         term (str): The search term passes to the API.
161         location (str): The search location passes to the API.
162         count (int): The number of business results to return.
163
164     Return:
165         results (dict): The JSON response from the request.
166     '''
167     params = {
168         'term': term.lower().replace(" ", "+"),
169         'location': location.lower().replace(" ", "+"),
170         'limit': count
171     }
172     request_key = construct_unique_key(baseurl=baseurl, params=params)
173
174     if request_key in CACHE_DICT:
175         # The data has been fetched before and stored in the cache
176         return CACHE_DICT[request_key]
177     else:
178         results = make_request(baseurl=baseurl, params=params)
179         CACHE_DICT[request_key] = results
180         save_cache(cache_dict=CACHE_DICT)
181         return results
```

Figure 1: Snapshot of code implementing caching.

The second source is Yelp business website. I plan to scrape some pictures and recent reviews loaded by customers with Beautiful Soup. To scrape a page, I will first get the HTML page contents in text format and then “parse” the contents, looking for particular tags with particular attributes and then extracting the

contents of those tags. In each Yelp business website, I scrape at most 3 pictures and at most 3 top and recent reviews. Since the pictures and reviews are likely to be updated day-by-day, although it is possible to use caching, the information stored in the cache might be out-of-date. So I decide not to use caching for this data source.

Database

I create two databases. One, named as “business_info”, is to store all of infor of business and the other one, named as “category_info”, is to store some categories of business. The left picture is the database schema of “business_info” and the right picture is the schema of “category_info”.

```
CREATE TABLE "business_info" (
  "yelp_id" TEXT UNIQUE,
  "business_name" TEXT NOT NULL,
  "city" TEXT NOT NULL,
  "phone_number" TEXT NOT NULL,
  "review_count" INTEGER NOT NULL,
  "rating" REAL NOT NULL,
  "price" INTEGER NOT NULL,
  "address" TEXT NOT NULL,
  "url" TEXT NOT NULL,
  PRIMARY KEY("yelp_id")
);
```

```
CREATE TABLE "category_info" (
  "yelp_id" TEXT UNIQUE,
  "category_1" TEXT NOT NULL,
  "category_2" TEXT NOT NULL,
  "category_3" TEXT NOT NULL,
  PRIMARY KEY("yelp_id")
);
```

Figure 2: Database schema of “business_info” and “category_info”.

It is worth noting that “yelp_id” is the primary key in two tables and the foreign key in “business_info” pointing to the business in “category_info”. And the following two figures are snapshots of two tables.

	yelp_id	business_name	city	phone_number	review_count	rating	price	address
1	O4AKSCQvrvOs...	Cliff House Terr...	San Francisco	+14153863330	449	4.0	3	1090 Point L...
2	dN_v0Lu6xhv0y...	Hakkasan Temp...	San Francisco	+14158298148	1772	4.0	3	1 Kearny St
3	nXYmPSI2Gy_ge...	Osmanthus Dim ...	San Francisco		36	4.5	2	504 Broadwa...
4	6bFXiikXLUHnr...	Larkin Restaurant	San Francisco		1	4.0	0	607 Larkin St
5	oOVILL4bz_wAe...	Monsoon Seattle	Seattle	+12063252111	614	3.5	2	615 19th Ave
6	4FD0TAxe70nb...	Uluh	New York	+19172615963	269	4.0	3	152 2nd Ave
7	vRs8JtL_MKDRt...	Food Zone Chin...	Dexter	+17344263410	36	3.5	1	7023 Dexter
8	WA7XyZ8FhZSL...	Akira Japanese R...	San Francisco	+14158008498	352	4.5	2	1634 Bush St
9	ZNj4xwodiEfo...	The Shota	San Francisco	+16282242074	158	4.5	4	115 Sansom
10	pNe8ZCKGT4c...	Ise Restaurant	New York	+1212284152	456	4.5	2	63 Cooper St
11	J8VuX3wGrY5O...	Katsu-Hama	New York	+12127585909	1319	4.0	2	11 E 47th St
12	6qzNc9Z9DHYY...	Takashi	New York	+12124142929	978	4.0	3	456 Hudson
13	ssi9ivccrDBT_R...	Greens Restaurant	San Francisco	+14157716222	2042	4.0	3	2 Marina Blv
14	Fin7_nhVUfkh...	Taste of the Hlm...	San Francisco	+14153311335	317	4.0	2	2633 Bridg...

Figure 3: Snapshot of “business_info” table.

DB Browser for SQLite - /Users/kedongh/Desktop/SI-507/507_final_proj/yelp.sqlite

Database Structure Browse Data Edit Pragmas Execute SQL

Table: category_info

yelp_id	category_1	category_2	category_3
1	Breakfast & Brunch	NULL	NULL
2	American (New)	Cantonese	Dim Sum
3	Cocktail Bars	Dim Sum	NULL
4	Vietnamese	Chinese	NULL
5	Vietnamese	Asian Fusion	Bars
6	Tea Rooms	Dim Sum	NULL
7	Chinese	NULL	NULL
8	Japanese	Sushi Bars	NULL
9	Japanese	Sushi Bars	NULL
10	Japanese	Sushi Bars	NULL
11	Japanese	NULL	NULL
12	Japanese	Korean	Barbeque
13	Vegetarian	Vegan	Gluten-Free
14	Himalayan	Indian	NULL

1 - 14 of 9523

Go to: 1

Mode: Text Import Export Set as NULL

1222

Type of data currently in cell: Text / Numeric
4 char(s)

Identity

Name Commit Last modified Size

SQL Log Plot DB Schema Remote

UTF-8

Figure 4: Snapshot of “category_info” table.

Interaction and Presentation Plans

I plan to use Flask platform to realize interaction and presentation. As shown in figure 5 (just a demo), there are two options. The first one “search” is to allow the user to get some business information after typing term and location. The second one is “summary” which is designed to give a rating/price/review_count summary for different business types or different cities. For better presentation, I decide to use Plotly to draw barplot or other formats (not decided yet).



Figure 5: Homepage.