Design Document

Yelp Reviews

Group Members: Tommy Chen, Emily Yamauchi and Sindhu Madhadi

Project Introduction

This project focuses on restaurant data retrieved from Yelp's Fusion API. Our primary motivation is the fact that Yelp's search function does not support filtering by business rating. As our project evolved, our main questions were:

→ Can we build a visualization app that allows for easy filtering (rating, price, category)?
→ Can we extract keywords of the restaurant based on user review text?
→ Can restaurant rating be predicted from the review text?
→ What else can we extract from the text data? How accurate is the restaurant categorization?

Language processing and machine learning tasks proved to be beyond the scope of this project, and are shelved as natural next steps; however we have completed significant text preprocessing

The Python packages involved in this project are **Requests, BeautifulSoup, MapBox,** and **NLTK**.

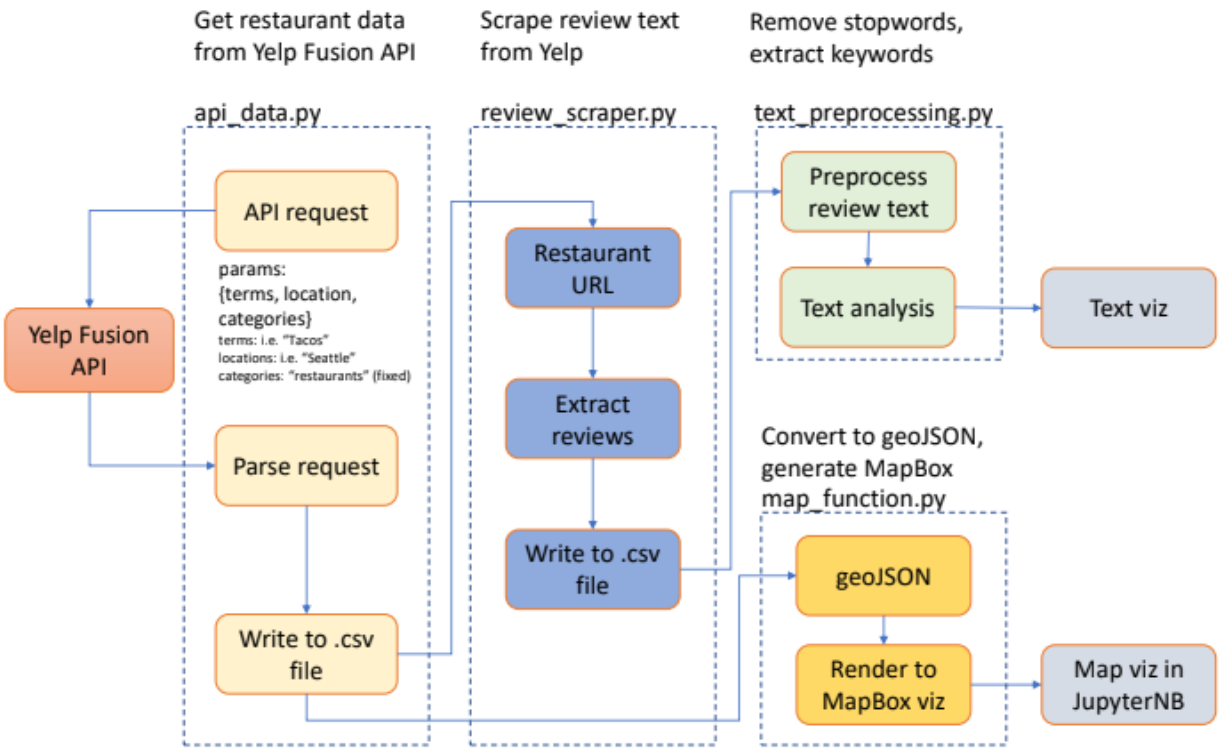
Data

We obtain the restaurant data from Yelp Fusion API, and individual restaurant reviews through an implemented web scraper which takes the URL retrieved from the API call.

- Yelp Fusion API allows extraction of business data on Yelp.com. While free, it requires a private key to authenticate all endpoints (discussed further below regarding limitation). Queries are constructed based on parameters chosen by users, and the responses are in JSON formats. For more details, please see Yelp's Fusion API page.

- The JSON dataset obtained from the API includes the following columns which we selected to convert to a Pandas Dataframe: Name, Category, Coordinates (longitude and latitude), Price, Rating, Url, and Review Count.
- However, this API has its limitations: it allows only 20 results for a single request, and up to 1000 returns per business in a single search query. Further, it caps the request limit per user on a free plan. Most importantly, the API only supports up to 3 reviews per business, which would be insufficient for our project.
- To resolve this issue, our group implemented a web scraper using the **Requests** package to retrieve HTML from URL, and the **Beautiful Soup** package to parse the HTML. The URL is obtained through the API call, and the resulting scraper would find all the JSON-formatted objects within the HTML and extract all 20 reviews on the first page of the business website.

Architecture



Components

- Data collection and parsing tool using Yelp Fusion API
    - Inputs:
        - Requires Yelp API key
        - Parameters specified by user (search terms, location, etc.)
    - Outputs:
        - JSON file of restaurants meeting search criteria
        - Also converts JSON file to a dataframe, keeping only select columns for further visualization (as specified in “Data” section above)
    - Interactions:
        - Initial data collection method which returns the dataframe used for populating the map visualization in the component below
        - Also returns URL of the restaurants, which is used in the web scraping component below which in turn returns the user review text
- Web scraping tool for user review text
    - Inputs:
        - Dataframe containing URL of each restaurant web page
    - Outputs:
        - HTML of the restaurant page from the given URL
        - Parses the HTML page and extracts the review text, writes to a .csv file
    - Interactions:
        - Requires the data collection module to return the URL for the restaurants meeting the search criteria
        - The output .csv file is read by the text preprocessing module
- Text preprocessing tool
    - Inputs:
        - .csv file containing restaurant review text obtained from web scraping module
        - Optional: User selected custom stop words
    - Outputs:

        - Cleaned text file as a list of words that has been lemmatized, with stop words and rare words removed
- Visualization app
    - Inputs:
        - Dataframe containing restaurant data from the API call
        - Requires MapBox token
        - Optional: filter parameters from user
        - Review text data from text preprocessing module
    - Outputs:
        - Optional: filters input dataframe per parameters
        - Converts input dataframe to geoJSON format
        - Reads geoJSON and renders map
        - Renders word cloud and distribution chart from text data
    - Interactions:
        - Reads dataframe from data collection module
        - Requires review text from .csv file from the web scraper tool, which in turn is cleaned up by the text preprocessing module

Use Cases

Supported use cases:

Case 1: Search for and plot restaurant data based on user-selected parameters without filtering (same functionality as existing Yelp searches)

- Terms: “Pizza”, “Tacos”, “Boba”, etc.
- Location: “Seattle”, or specific neighborhoods such as “University District, Seattle”

Case 2: Search for and plot restaurant data based on user-selected parameters, but include filtering

- Same parameters as above Case 1
- Returned plots can be filtered by ratings (Yelp “stars”), or categorical variables such as restaurant category/cuisine, price, or transactions supported (delivery, pickup, etc.)

Proposed, but “next step” use cases:

Case 3: View latest reviews based on searches from either Case 1 or Case 2

- Using the search function from either Case 1 or Case 2, populate the map with restaurants meeting criteria. Upon selection, the user could view the latest 20 reviews for the selected restaurant.
- Visualization of the review text. Currently considering word cloud, but if time allows, attempt to extract keywords from the review list.

Next Steps

- More flexible and intuitive filtering logic in map visualization
    - Currently, the map visualization only allows for a single filter, and does not allow complex if-statements such as “rating > 4” AND “cuisine == Mexican”.
    - Further, the existing filtering logic should be amended so that it would be more intuitive to the user. Currently, price category in the dataset is filtered based on a “is” logic, but should be adjusted to allow for a range- i.e., return restaurants less than “$$$”, rather than only “$$$” restaurants.
- Include more variables in the map visualization
    - The JSON file returned from the API call includes variables that were not selected in our data preprocessing step, such as hours of operation
- Interactivity and dashboard creation
    - Our goal was to create an interactive dashboard which allows users to filter returned map visualization through widgets rather than directly in the code cells, but we scaled back our scope to focus on completion of existing specification.
    - As a natural next step, we would like to also link the text visualization to the filtered dataset, and display the two visualizations in the dashboard next to the map.
- Text processing and machine learning
    - Our original objective was to create a model to predict the restaurant rating based on the user reviews, but this proved to be far beyond the project scope. As we have completed some preliminary preprocessing steps, we would like to also continue to work on the model creation as a next step.

Testing

We have included a unit test framework to test our modules, included in the “tests” folder. Each test file corresponds to a base module. Methods are tested using either the test cases included in the “data” subfolder, or if the methods are tested using the actual API call, parameters other specifications are provided in the “data” subfolder.