Point of Interest Recommendations based on User-Defined Search

By Ethan Piette

1 Abstract

This project explores the creation of a flexible query system for recommendations on points of interest (POIs) through the use of Yelp's academic business. The goal of this project is to enable users to query businesses based on their preferences by inputting, the type of business, a list of keywords, and other specific filters, and receiving a ranked list of the top-K POIs that match their search criteria. To achieve this goal, a preprocessing pipeline was created to clean the JSON data, and transform it into a structured SQLite database. This required restructuring nested attributes, and ensuring data consistency, to create a schema that can be effectively queried. The algorithm for providing the recommendations dynamically constructs SQL queries based on the users' inputs, and applies optimizations to handle the large dataset efficiently. Through testing, we were able to determine that this solution works well, as it delivers accurate and customized recommendations. However, future work would likely include integrating user behavior, to enhance the recommendations that the query provides to match the users previous preferences.

2 Introduction

The Yelp business dataset provides an extensive collection of business information that contains business categories, reviews, ratings, and additional attributes related to businesses, in the form of a JSON. In order to query this data effectively it required pre processing the data, through cleaning, and formatting it such that it can be queried. While platforms like yelp provide basic filters for searching for POIs, they often lack the flexibility to take more detailed user-defined searches.

The objective of this project is to create a dynamic query system that allows users to search for POIs tailored to their specific needs. This would allow users to search for POIs via the use of specific filters, such as searching for “Mexican restaurants that offer delivery, with at least 100 reviews, and ratings of 4 stars or more”. The system will be able to efficiently handle these types of requests, and ensure accuracy and relevance in the recommendations that it provides, and will be able to provide these recommendations quickly, even with a large dataset such as the yelp business data.

This project contained significant challenges along the way. This includes transforming the raw JSON dataset into a structured format that can be queried, optimizing query performance for a large dataset, and handling nested fields within the data. This solution provides flexibility and control over the search criteria that hasn’t been seen in previous approaches in POI recommendation systems that have been created.

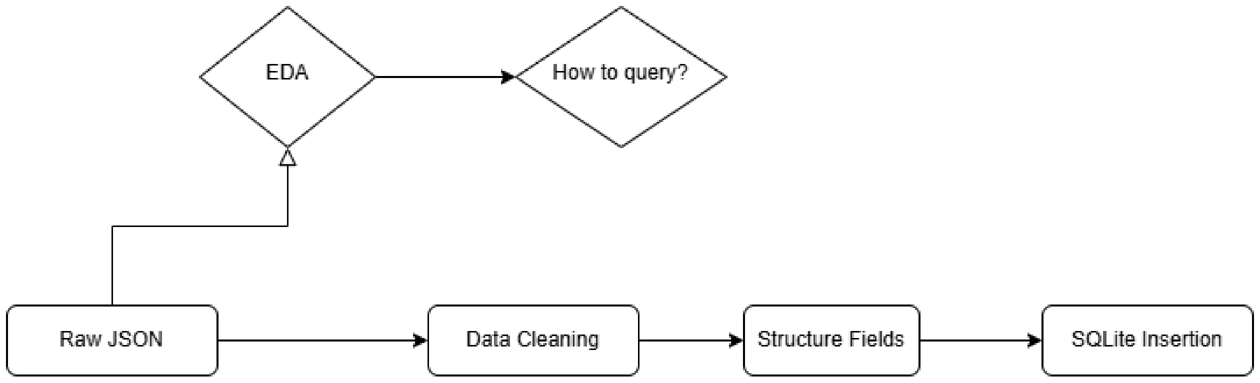
3 Related Works

The research on POI recommendation systems, looks into the importance of leveraging structured relationships, and robust preprocessing. An article in the journal Information Technology and Tourism from 2021 discusses that being able to utilize connections between businesses, users and attributes is necessary to enhance recommendation accuracy and relevance. This project aims to utilize these aspects to create a new system for querying the data on businesses to provide recommendations that are not just accurate, but also relevant to the user by allowing them to input specific information about what they are looking for in a POI, and providing recommendations from that information.

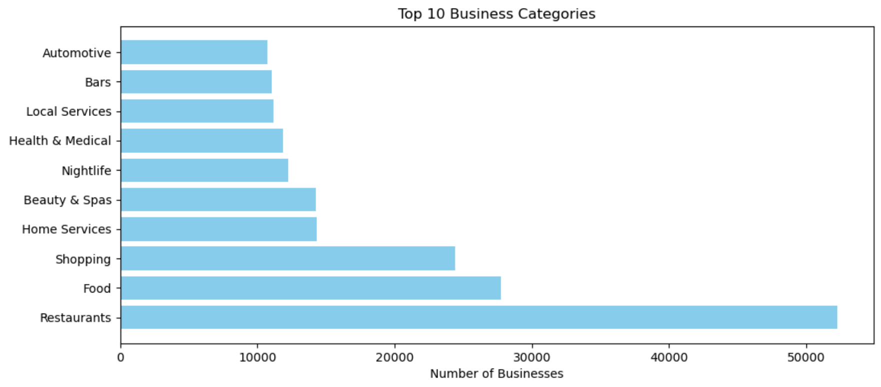
4 Solution

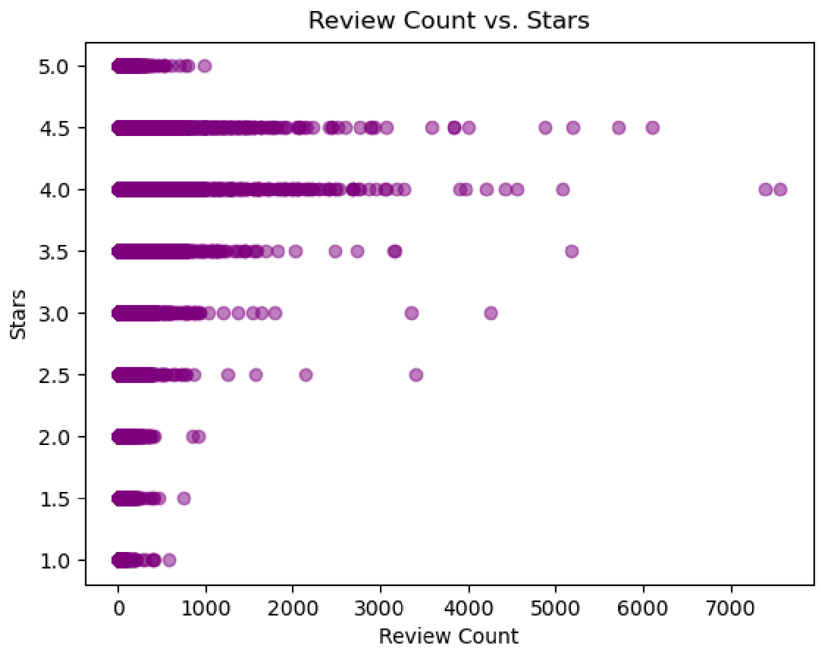
The solution for this project can be broken down into two main parts. Firstly, preprocessing, which involves cleaning the data, restructuring the data so that all of the data can be queried properly, and inserting the data into a SQLite database so that the data can be properly queried using SQL. Secondly, the construction of the query algorithm itself, which involves a dynamic query construction, parsing the data, and properly ordering the results of the recommendation.

The process taken for preprocessing the data is outlined in the below flow chart. For each of the steps python was used to take the raw data and send it through the preprocessing steps.



The exploratory data analysis involved looking at the data to determine what preprocessing needs to be done, and insights into the data to help drive the creation of the query algorithm. This involved the creation of insightful graphics(seen below).





In trying to work with the data it was discovered that I would need to clean the data so that it could be properly queried, and also structure specific problematic nested attributes so that they could be properly queried, and lastly properly loading the data into a SQLite database so that SQL queries can be performed on the data. Below is pseudocode for the python script that was used to perform these actions.

Unset
```
1. Define function `create_business_table(connection)`:
   - Execute SQL command to create the `business` table with fields:
     business_id, name, address, city, state, postal_code, latitude, longitude,
stars,
     review_count, is_open, attributes, categories, hours (if table doesn't
exist).
   - Commit the changes.

2. Define function `clean_json_field(data)`:
   - If data is a string: strip unwanted characters.
   - If data is a dictionary: recursively clean key-value pairs.
   - If data is a list: recursively clean each element.
   - Return cleaned data.

3. Define function `prepare_for_insert(data)`:
   - Clean `attributes` and `hours` fields using `clean_json_field`.
   - Ensure `categories` is a list and join it into a comma-separated string.
   - Return a dictionary with cleaned and JSON-formatted data.

4. Define function `load_json_to_sqlite(file_path, connection)`:
   - Call `create_business_table(connection)`.
   - Open JSON file and parse each line into a dictionary.
   - Prepare data for insertion using `prepare_for_insert`.
   - Insert data into the `business` table using SQL `INSERT OR IGNORE`.
   - Commit the changes.

5. Connect to SQLite database (or create it if it doesn't exist).

6. Call `load_json_to_sqlite` with the path to the Yelp JSON dataset and the
database connection.

7. Print "Data loaded into SQLite."
```

Once the data was cleaned a query algorithm was created, so that recommendations could be provided based on a user-defined search. This required creating a dynamic query, which allows filters to be added conditionally keeping the queries efficient and flexible. SQLite's json_extract function was used to enable advanced filtering based on more complex data attributes. Lastly, the recommendations are ordered by businesses with the highest store rating, and then by the highest number of reviews. Pseudocode for this algorithm can be found below along with the pseudocode for an example query.

Unset
```
1. Define function `query_pois(poi_type, keywords, filters, k)`:
   - Connect to SQLite database.
   - Initialize base SQL query to select POIs by `poi_type` in `categories`.

2. Add keyword filters:
   - For each keyword, add a condition to match `categories` using `LIKE`.
   - Combine all keyword filters with `OR`.
   - Append keyword parameters to the query.

3. Add additional filters:
   - For `attributes`: Use `json_extract` to filter nested JSON fields.
   - For other filters:
     - If value is a string: Add a `LIKE` condition.
     - If value includes an operator (e.g., `>=`, `=`): Add the operator and
value as conditions.

4. Order results:
   - Sort by `stars` (descending) and `review_count` (descending).
   - Limit the results to top-k.

5. Execute the query:
   - Bind all parameters to the SQL query.
   - Fetch and return results.

6. Example Usage:
   - Input:
     - `poi_type` (e.g., "Restaurant").
     - `keywords` (e.g., ["Chinese"]).
     - Filters (e.g., stars ≥ 2, reviews > 20, delivery available, state =
"ID").
     - Top-k value (e.g., 5 results).
   - Output: Ranked POIs matching the criteria.

7. Close the database connection.
```

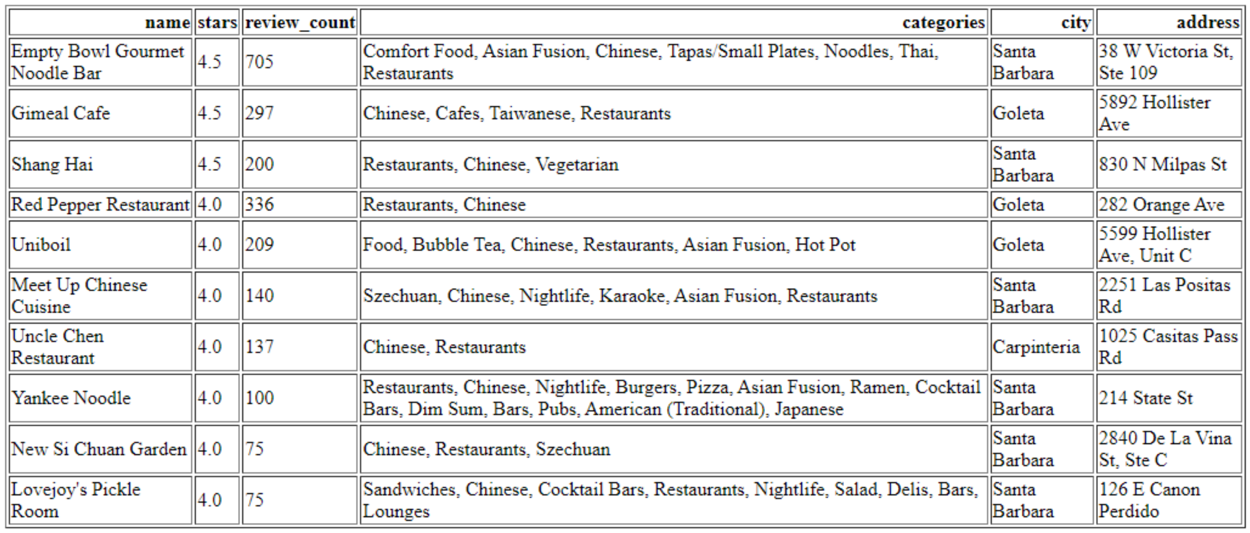
5 Evaluation

5.1 Setup

The dataset used was the Yelp Academic Dataset, which contains information about various different businesses in JSON format. In order to query the data it was necessary to transform the data into a structured SQLite database using the preprocessing methods described in the solution section. For evaluation of the algorithm we used python to run example queries, manage the database with SQLite, and create visualizations using matplotlib.

5.2 Results

To evaluate the performance of the algorithm both the efficiency and the accuracy of the algorithm were evaluated. Using example queries the algorithm was tested to make sure that it would provide accurate recommendations quickly. Thanks to the algorithm being able to run O(n) time it was able to provide recommendations for all example queries in under a second. In addition it provided accurate recommendations based on the filters provided in the query properly ordered by star rating and total number of ratings. It also proved to be flexible through the use of multiple complex queries for recommendations. The results for an example query of "Chinese restaurants that offer delivery in California with 4 or more stars and 70 or more reviews".



name	stars	review_count	categories	city	address
Empty Bowl Gourmet Noodle Bar	4.5	705	Comfort Food, Asian Fusion, Chinese, Tapas/Small Plates, Noodles, Thai, Restaurants	Santa Barbara	38 W Victoria St, Ste 109
Gimeal Cafe	4.5	297	Chinese, Cafes, Taiwanese, Restaurants	Goleta	5892 Hollister Ave
Shang Hai	4.5	200	Restaurants, Chinese, Vegetarian	Santa Barbara	830 N Milpas St
Red Pepper Restaurant	4.0	336	Restaurants, Chinese	Goleta	282 Orange Ave
Uniboil	4.0	209	Food, Bubble Tea, Chinese, Restaurants, Asian Fusion, Hot Pot	Goleta	5599 Hollister Ave, Unit C
Meet Up Chinese Cuisine	4.0	140	Szechuan, Chinese, Nightlife, Karaoke, Asian Fusion, Restaurants	Santa Barbara	2251 Las Positas Rd
Uncle Chen Restaurant	4.0	137	Chinese, Restaurants	Carpinteria	1025 Casitas Pass Rd
Yankee Noodle	4.0	100	Restaurants, Chinese, Nightlife, Burgers, Pizza, Asian Fusion, Ramen, Cocktail Bars, Dim Sum, Bars, Pubs, American (Traditional), Japanese	Santa Barbara	214 State St
New Si Chuan Garden	4.0	75	Chinese, Restaurants, Szechuan	Santa Barbara	2840 De La Vina St, Ste C
Lovejoy's Pickle Room	4.0	75	Sandwiches, Chinese, Cocktail Bars, Restaurants, Nightlife, Salad, Delis, Bars, Lounges	Santa Barbara	126 E Canon Perdido

6 Conclusion and Future Work

The algorithm created successfully leverages the Yelp dataset to provide personalized POI recommendations based on a user-defined search, that contains the type of business, a list of keywords, attribute filters, and the number of recommendations that they wish to receive. By preprocessing the data, and implementing a dynamic query algorithm, recommendations are returned quickly and accurately based on the user defined search criteria. However, there are some challenges that would be important for future work on POI recommendation. The primary of which is implementing user preference history to be incorporated into the decisions for the provided recommendations. It would be important to incorporate machine learning algorithms to adapt the query system to work with the user data to provide these personalized recommendations that go beyond the scope of the search. Additionally, this would help to implement using real time data being added to the dataset, and handling larger more complex datasets.