Praveen Iyer (5549510111)
Sankalp Sharma (7672471024)

ZIP-Find

(Location based Restaurant Search and Recommendation System)

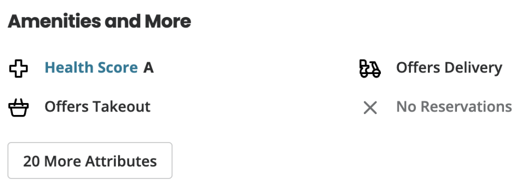
Project Goals/Motivation/Research Questions:

ZIP-Find is a location based Restaurant Search and Recommendation System. The motivation for building this project comes from the fact that there currently exists no platform that clearly allows users to search for restaurants that are similar to a restaurant they like, specifically in the proximity of a tourist attraction that they plan on visiting. We are looking at two main dimensions: Proximity to an attraction and Restaurant Similarity.
For instance, we consider a user who really likes Cava, a Mediterranean restaurant and searches for it on Yelp's website. The user will be recommended a few restaurants similar to Cava in their neighborhood but these restaurants are not necessarily located in the proximity of an attraction that the user wants to visit. Hence, the dimension of proximity to the desired location is not accounted for in Yelp's case. Additionally, if the user searches Tripadvisor's website for an attraction like Griffith Observatory, then they will be recommended a set of restaurants located near the attraction but the notion of similarity or user's preference is not taken into account. In both cases, one of the two aforementioned dimensions is getting compromised.
Hence we decided to make a Knowledge Graph connecting restaurants and attractions together using zip code (Location) as it acts as a common link between the two entities. Our KG has four nodes (Restaurant, Attraction, Zip code and City) connected to each other using properties (HasRestaurant, HasAttraction, Nearby, InCity). Having information about nearby zip codes for a specific zip code helped us to account for the dimension of Proximity to an attraction. For Restaurant-based Similarity Recommendation, we used Jaccard Similarity between the restaurants obtained from querying our KG. Presence of 'City' nodes allowed the user to perform city-based attraction search (sorted by expert opinion, ratings and number of reviews) and city-based restaurant search based on filters like (Parking, Dog Status, Vegetarian Options, Reservations etc.)

1st Technical Challenge:

Problem: While trying to scrape restaurant-related data from Yelp's website, we came across the section of "Amenities and More" which in most cases had a button (e.g.: '20 More Attributes') which upon clicking loads the entire list of amenities . Scrapy does not function in the case of data that is dynamically loaded. Since this section was crucial to calculate Restaurant's similarity, we had to find a way to extract all amenities hidden by the button.
Solution: To scrape this information dynamically, we used Selenium which allows us to locate and click the button and scrape the entire list of amenities. This was particularly challenging because we were not familiar with Selenium and having to use it with Scrapy made it even more challenging. The code took 16 hours to run and we scraped 7446 restaurants.
Evaluation: We manually compared 25 random restaurants obtained in the output file and checked whether the entire list of amenities was successfully scraped or not. Out of 25 results, 24 of them were scraped entirely, except one in which the entire list wasn't obtained due to stability issues with Selenium.
Examples: Images below show the button that needs to be clicked to see the full amenities:

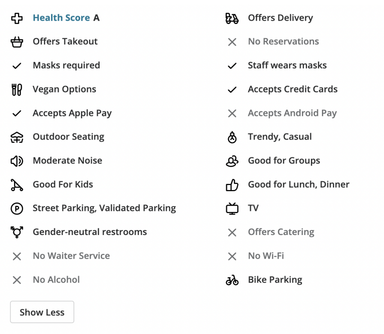
Praveen Iyer (5549510111)
Sankalp Sharma (7672471024)



Amenities and More

Health Score A
Offers Takeout
Offers Delivery
✕ No Reservations

20 More Attributes

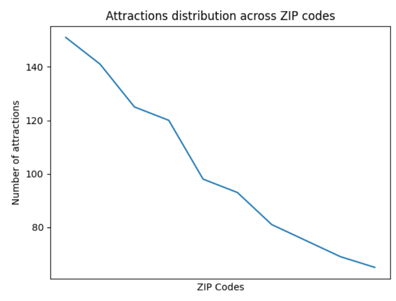


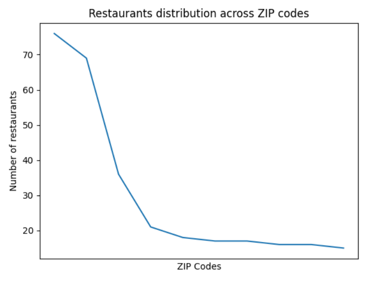
Health Score A
Offers Takeout
✓ Masks required
Vegan Options
✓ Accepts Apple Pay
Outdoor Seating
Moderate Noise
Good For Kids
Street Parking, Validated Parking
Gender-neutral restrooms
✕ No Waiter Service
✕ No Alcohol
Offers Delivery
✕ No Reservations
✓ Staff wears masks
✓ Accepts Credit Cards
✕ Accepts Android Pay
Trendy, Casual
Good for Groups
Good for Lunch, Dinner
TV
✕ Offers Catering
✕ No Wi-Fi
Bike Parking

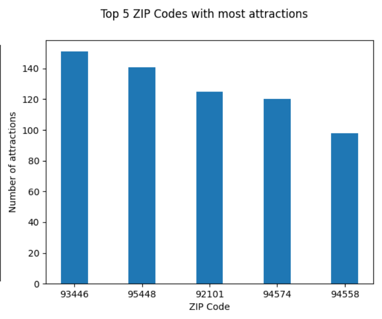
Show Less

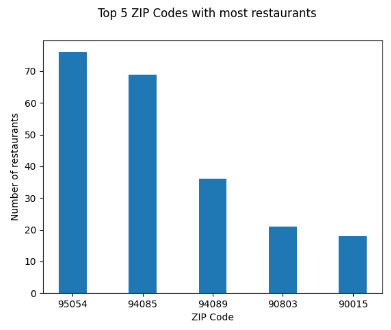
2nd Technical Challenge:

Problem: Evaluating and assessing the quality of our Knowledge Graph was also a challenge.
Solution: Quality is something that cannot be quantified hence we used measures like completeness, objectivity, accuracy, timeliness, reputation etc. to assess quality of KG and Data.
Evaluation with Examples:
1) Completeness: Restaurants and attractions data had to be dropped due to missing critical information such as Zip Code, Reviews, Rating and so on. From 1596 standard ZIP codes, 43 were removed due to absent nearby ZIP codes data.
2) Timeliness: Data is up to date.
3) Reputation: Data comes from reputed/trusted websites like Tripadvisor, Yelp and official US website for zip codes.
4) Accuracy: Sometimes Selenium failed to extract complete amenities lists that resulted in some data loss. ZIP codes from nearby states (AZ , NV, OR) that appeared in our dataset were dropped to mitigate errors.
5) Objectivity: Some zip codes appeared more frequently than others based on the popular cities which introduced some bias in our data. Hence our platform will have more data for popular cities (like San Francisco, Los Angeles, San Diego etc.) in California.Graphs below demonstrate the distribution of zip codes:









Lessons Learnt:

- Combining structured (list of attractions and restaurants, list of zip codes) and unstructured data (address, ratings in the form of stars, price in the form of $$$) became much easier with a KG.
- Querying and retrieving information (nearby restaurants or attractions using zip code) is faster using KG than other conventional data structures in python and traditional relational databases.
- Use of KG facilitated smooth linking of entities between two different data sources for tourist attractions.