APARTMENT FINDR

Megan Baker, Najla Elmachtoub, Michael Mazzola

Abstract

Choosing a place to live is one of the most important and most difficult choices that people make on a regular basis, but current web applications do not allow for a truly personalized search based on the features of a neighborhood that may be important to an individual user. This is especially difficult when moving to a new city. Location is an increasingly important factor in apartment selection as people become busier and have less time to commute. The ideal location varies, since every person has different places that he or she values. We have built a web application that allows the user to input how much he values different business categories and returns an ideal listing for an apartment currently on the market. This application uses Craigslist apartment listings in New York City and Yelp business reviews.

Project

Motivation

We are three college students who have searched for apartments in different areas of the country over the past few years. In Ithaca, it is an almost trivial task for us to figure out where the most valuable place to live is: Collegetown. This value comes from an underlying interest to be close to the Cornell campus, restaurants, friends, the bookstore, and other amenities. The older population of students may prefer the Commons, while families might prefer to live farther away from the rambunctious college crowd. Searching for apartments in other areas has not been so easy, especially in areas where there are more unknowns and a wider array of choices. Our Apartment Findr seemed like it would be a useful tool to sort apartments by desirability, using location as a metric rather than price and size.

Application

Our application is in the form of a simple web page where the user is asked, “What’s important to you?” The user will use a slider ranging from 0-100 to indicate his interest in these particular categories of business: food, shopping, nightlife, active life, education, restaurants, arts and entertainment, and beauty and spas. These categories were selected from the general list of Yelp business categories, as they reflect common and easily associated types of business. We combed through the general list to pick out what might be most important to a user.

Once the user has made his selections, he can click the “Find an apartment” button, which will redirect him to a page with the highest scoring apartments on Craigslist. Each apartment is given a score based on how well the surrounding businesses satisfy the user’s interests using a weighted average encompassing distance, ratings, and the number of businesses. A business is considered for the score if it is within 500 meters of the apartment. The top ten apartments are shown on the page.

Datasets

As mentioned above, we used Craigslist and Yelp data for this project. To access the Craigslist data, we used the 3Taps[1] API. The query we used to obtain all listings for apartments for rent in New York City is expressed below:

```
http://3taps.net/search?
      category=RHFR&
      regionCode=USA-NYM-MAN&
      annotations={source_map_google:*%20AND%20source_loc:newyork}
```

The raw listing data is transformed from JSON into a dictionary representation, storing the title, number of bedrooms, price, source URL, description, and image link. Before storing the location, we use the Google Maps API[2] to retrieve the latitude and longitude. This is because the locations are user-posted and not uniform, so we need some way to standardize them. This is how we get those values:

```
raw_location=raw_apartment['annotations']['source_map_google'].split('?q
      =loc%3a')[1]
google_response=urllib2.urlopen(google_geolocator_url + raw_location)
google_response_dict=json.loads(google_response.read())
refined_location=google_response_dict['results'][0]['geometry']['locatio
      n']
refined_apartment.update({'location':refined_location})
```

[1] 3taps.com

[2] developers.google.com/maps

For each apartment, we use the Yelp API[3] to find all businesses of each category within one thousand meters. These businesses are sorted by rating. An example query is as follows:

```
http://api.yelp.com/v2/search?
        term=food&
        ll=37.78,-122.39&
        radius_filter=500&
        sort=2
```

For each apartment listing, we take the Bayesian posterior mean for each category of business. The formula used is $\frac{3+ny}{1+n}$, where *n* = how many businesses in the category are within 500 meters of the apartment listing and *y* = the average rating for these businesses. We assume that the average rating for a business on Yelp is 3 with a standard deviation of 1. Now each apartment listing has eight Bayesian posterior means associated with it, one for each category.

The newly created dictionary of apartments is stored into a MongoDB, along with its means and all of the associated businesses.

Technical Architecture

The front end of the application is written in HTML/CSS with Skeleton[4] because it is a simple and elegant way of displaying our user interface. The results page is actually an HTML template with python commands to output the data. We have used schema.org for structured data markup.

The scripts used to extract the data are written in python, which we use to make the MongoDB. We chose MongoDB because our data is not very relational and we were just interested in learning how it works. The underlying web application is written in python with the Tornado framework, which was both familiar and useful given the tasks we are completing.

The values for the user's preferences are grabbed using JavaScript when the button on the initial page is clicked. This builds a request that is used to generate output. Eight rankings--one for each category--are used to create a weighted average of the Bayesian means for each listing. The apartments with the highest averages are the ones suggested to the user.

[3] yelp.com/developers

[4] getskeleton.com

We have used all of these technologies to create a RESTful API for the underlying structured dataset. We have not implemented POST, PUT, or DELETE, as it does not make sense to in this case, and because we do not have the permissions to modify Craigslist or Yelp. However, we can submit GET requests for specific apartment listings and businesses. We use the following commands:

```
curl localhost:8888/apartment/0
curl localhost:888/business/0
```

To maintain connectedness, each apartment will have links to the businesses that are within 500 meters of it, and each business will have a list of the apartments it is close to.

Relation to Course

In this project, we are using many of the technologies learned in this course. At a high level, we are taking two seemingly unrelated data sets to expose something new. From this course we were able to see the abundance of available public data and ways of manipulating it. We are also using technologies and concepts that we used in class: HTML templates, python, Tornado, RESTful APIs, etc.

Work Distribution

Each of our group members participated in the ideation process and the mapping of our technical architecture. GitHub[5] was used for version control. This work was completed prior to the progress check done for class. Afterwards, we met once a week in three-hour blocks and all worked side by side. This increased communication and production. It also enabled each of us to be involved in all parts of the project, though we split the actual work up as follows:

- Najla-Wrote all HTML and CSS and most of this paper.
- Megan-Extracted Yelp data and contributed to the python web application.
- Michael-Extracted Craigslist data and setup MongoDB. Contributed to python web application and wrote JavaScript.

Future Work

There are many possible extensions for this project in addition to making it a more robust RESTful web application.

[5] github.com/mazzola/apartment_findr

- Extending to other cities. We chose New York City because there exist many more data points out there than there are for other cities. However, this would be useful in a number of locations where this same problem exists, such as Boston, Chicago, or Seattle.
- Using other apartment listings. Craigslist apartment listings were one of the few options when we looked for data sets, and even this data is not very clean or uniform. Craigslist postings may lack uniformity, and using this data may also result in not having enough listings to give good suggestions. It would be ideal to aggregate apartment listings from several different sources, if available.
- Refining the categories. Some users may prefer to have advanced settings, where they can specify their preferences for even more specific categories of businesses. This could be extended using the subcategories available on Yelp, but careful consideration would have to be made to ensure that the level of refinement still produces good results.
- Adding specific locations. Users may value specific locations that they would like to be close to, like family addresses, their offices, or their usual delis. We could use the Google Maps API to factor this into the score creation.
- Including typical search criteria. It would likely be of interest to users to be able to add restrictions to their search results based on price, the number of bedrooms, the number of bathrooms, etc.
- Creating a visual aid. Google Maps could be used to create a visual aid of where the apartments of interest are located. This could help the user spot patterns to understand which neighborhood of the particular city would be a best fit for him. This could also be a way to unobtrusively view which businesses are nearby each location on the map.
- Defining closeness. We currently take into consideration businesses that are “close” to each apartment based on some distance cutoff that we choose. We could instead allow the user to define what “close” means, as this does vary by city and available modes of transportation.
- Different ways of scoring. It could be interesting to try different methods for creating the scores.