INFSCI 2480 Adaptive Information Systems Final Project Report

Personalized Restaurant Recommendation System – Yelp

Ruochen Liu, Shashank BR, Xinyi Li, Yimeng Liu

1. Introduction

The final project we designed and implemented for the IS2480 Adaptive Information Systems course is a restaurant recommendation system built on top of Yelp. It is an application which can recommend different restaurants to the users based on the users' rating history and the ratings given by other users across the platform because user given ratings are one of the most reliable sources of information. Our application has several functions such as user registration/login, favorites selection, adaptive searching, rating system, and recommendation system. The original idea was from IMDB. IMDB started out as an archive for ratings and reviews for movies but they now use their existing data to recommend other movies to users who have rated movies in their platform.

We decided to adopt this idea for restaurants using Yelp. Yelp is a platform containing ratings and reviews about various businesses, primarily focusing on restaurants across the country. In this website, people can view the reviews and ratings about any of the business posted by other registered users. Registered users can post reviews and ratings. The posts or reviews themselves can be further rated by other users. We noticed that when it comes to restaurants, there was not a lot of personalization or even adaptive recommendations available. Most of the time it was basically few filters which such as price, cuisine, hours and location. So we decided to use what Yelp already had and build a recommendation system on top of that. Every few months for the past 3 to 4 years, Yelp has been conducting what's popularly known as Yelp Challenge aimed at students where they make some of their data publicly available which can be used to conduct research or analysis. We decided to build our application using

this data, so we can start with real data containing real reviews. This can save us the hassle of creating fake data for testing our application.

2. System Interface

The interface of this project was built with React.js. The front-end communicates with back-end API with Axios. There are three main page tags of the user interface, Register, Recommendation, and Search. There are several functions have been implemented in the register page, which include sign-up verification and questionnaire survey.

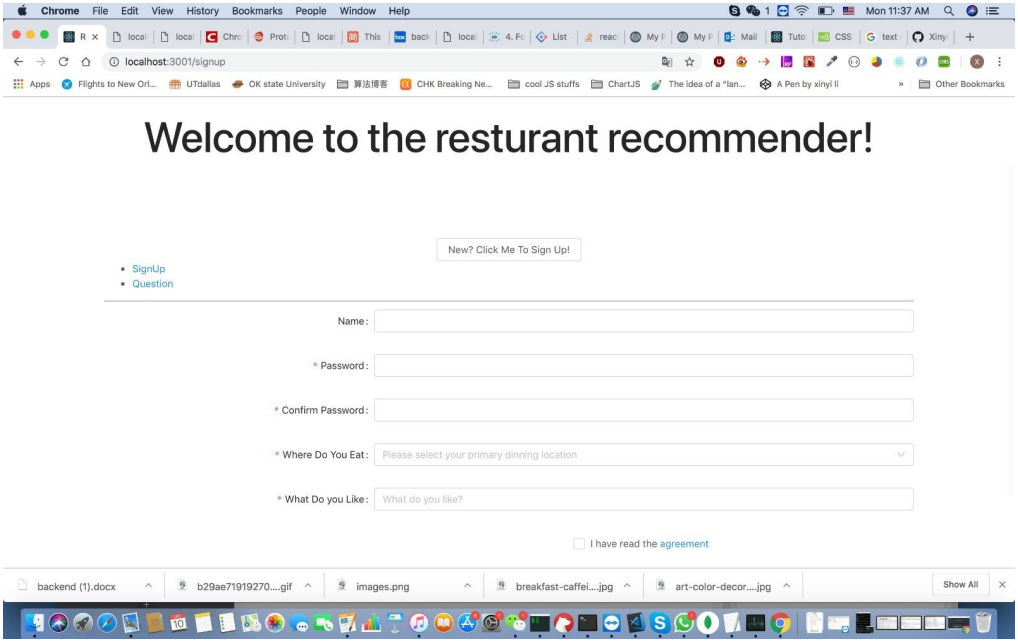


Fig. 1. Sign up page

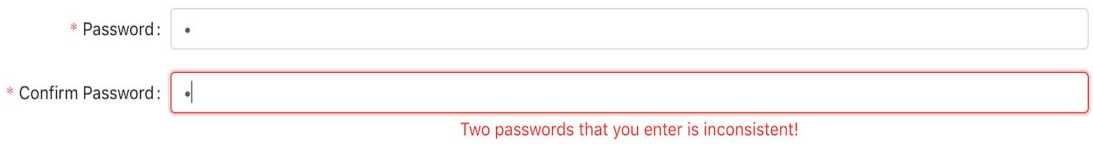


Fig.2 Confirm password when sign up

For the questionnaire survey, in order to engage user finish the survey, we decided to make it a fun process while keep it simple. We provide user top 20 popular restaurants in his area. If the user has been to some restaurants of those, he can rate the restaurants with 0 to 5 score by clicking stars.

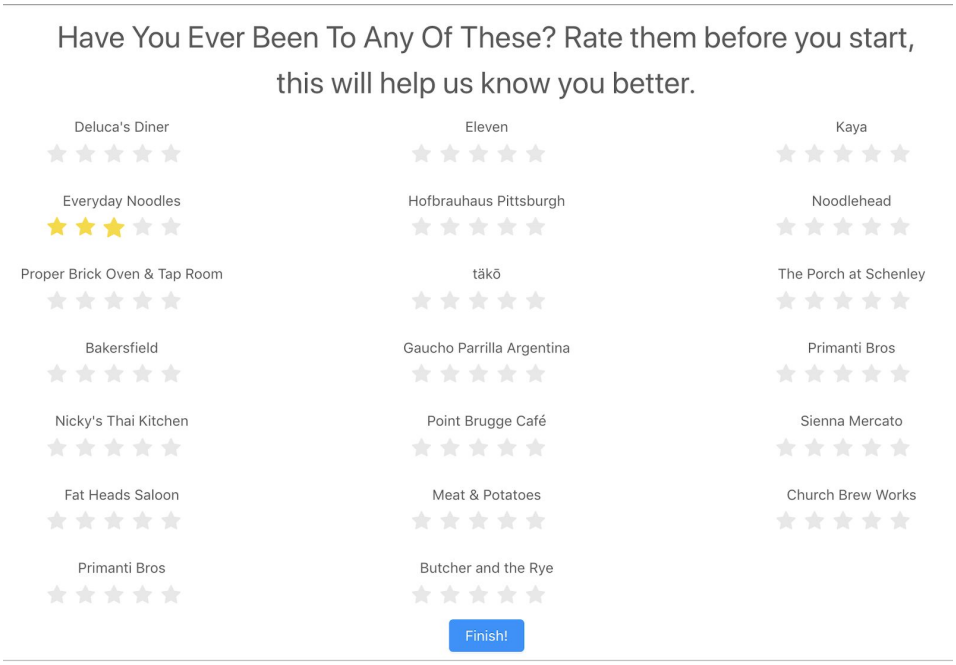


Fig. 3 Rate three new restaurants while sign up

The Recommendation page recommends various restaurants through collaborative filtering. The recommendation results are listed from top to bottom and those recommended restaurants are also linked to correlation photos in the photo database and vividly displayed in this page.

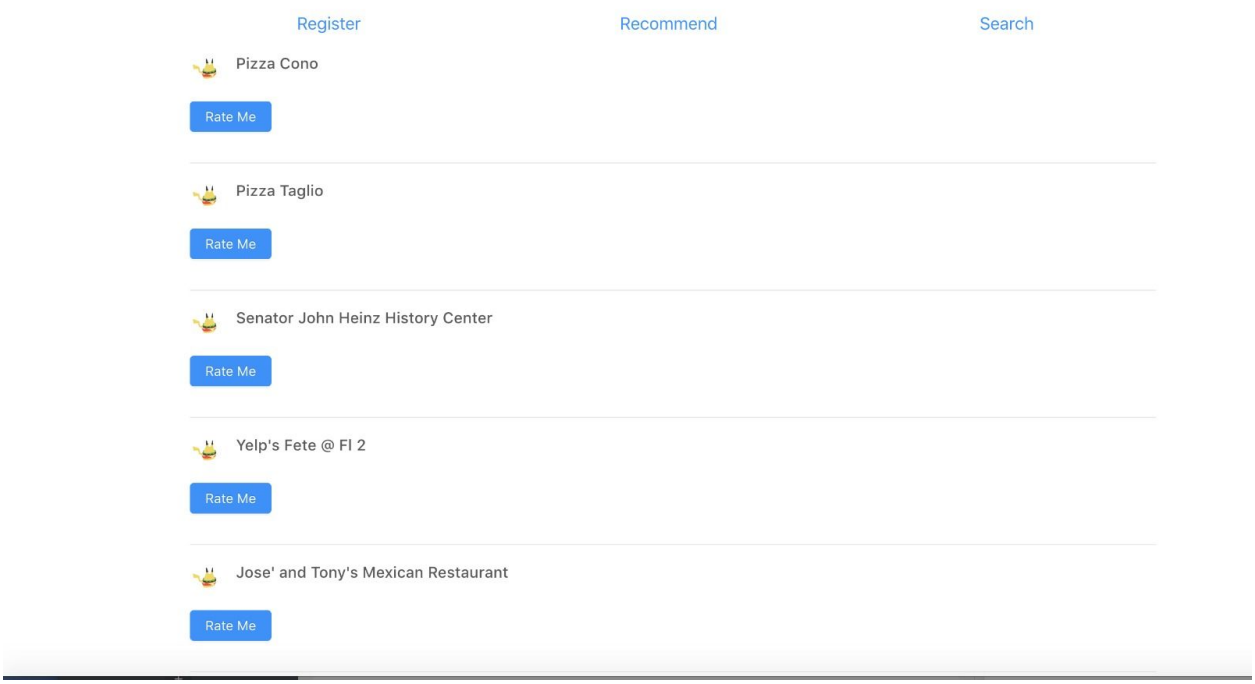


Fig. 4 Top 5 restaurants recommendation result

When users want to do some searching, click the search tag and it will jump to the search page. We imitated the layout of classical searching engines and designed ours. The search box lays in the middle. Once typed the search query and clicked “search bottom”. Searching results will be listed with the sequence of relevance. If user is satisfied with the search result, he can click the ‘thumbs up’ button and this will help the system to gain a better understand of his preference.(Figure. 5)

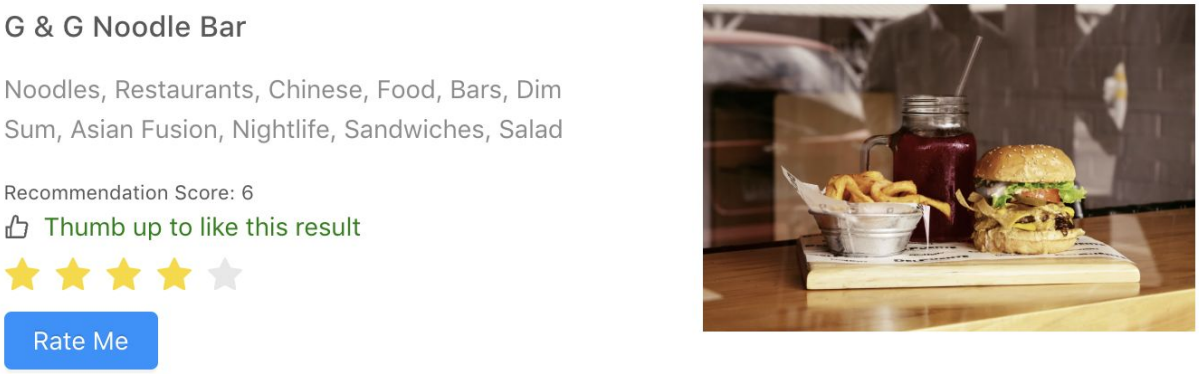


Fig. 5 User can give feedback for the search result by clicking the ‘thumbs up’ button

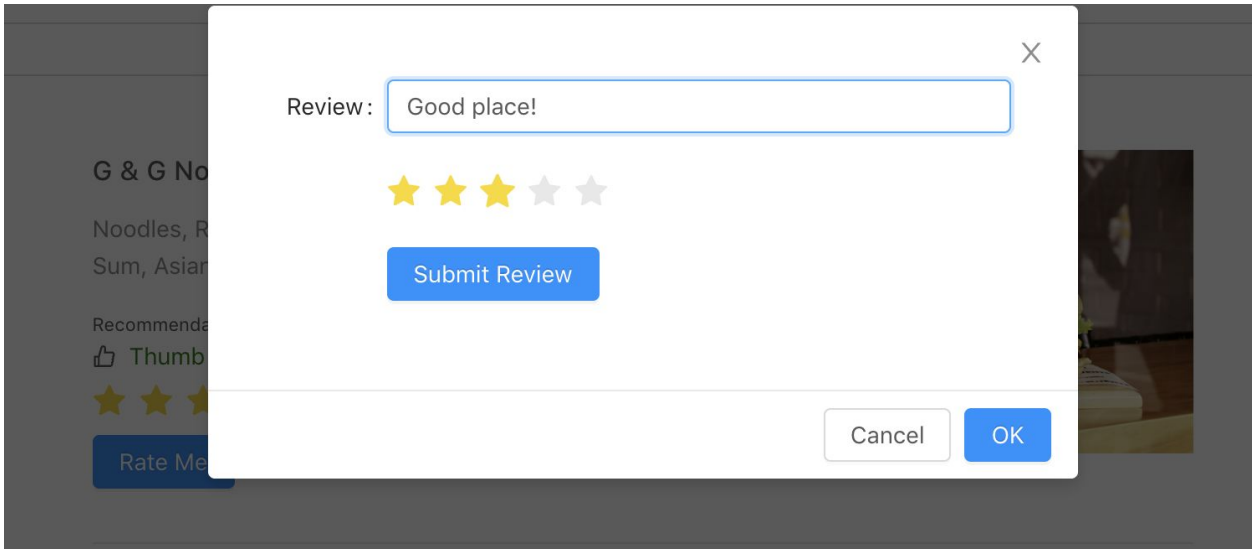


Fig. 6 User can rate the place and leave review

	Home Page	Business Page	User Page	Search Result
Function	Recommendation	Write Review, Stars	Username	Search an input word
	Click in to see restaurants	Like it or not		
Display	Similar Recommendation	Star Level		Restaurants sorted by relevance
		The restaurants' details		
		Review area		

Table. 1 User Interface

3. Database Design

Dataset provided by Yelp was in the form of JSON files. Following files were provided by Yelp :

A. Business: Contains business data including location data, attributes, and categories.

B. Review: Contains full review text data including the user_id that wrote the review and the business_id the review is written for.

C. User: User data including the user's friend mapping and all the metadata associated with the user.

D. Check-in: Check-in on a business.

E. Tip: Tips written by a user on a business. Tips are shorter than reviews and tend to convey quick suggestions.

F. Photo : Contains photo data including the caption and classification (one of "food", "drink", "menu", "inside" or "outside").

These files contained almost 6 million reviews given to 188 thousand businesses in 10 metropolitan areas. For our application, we have used Business, Review and User files. We further filtered out the data to have reviews pertaining just to the Pittsburgh region. Our final data contains over 200 thousand reviews given to 6800 businesses.

We use MongoDB to store and management our data. The data from Yelp are in the form of JSON file. MongoDB is undoubtedly the best choice since it uses JSON as the default input.

The initial business, photos, and reviews tables are open source datasets provided by Yelp downloaded from its website. Among the countless restaurants in the dataset, we only selected businesses based in Pittsburgh. The size of the business table is reduced to 6000 records. The user profile database is created to store any new sign up users. Once they joined the system, they have to enter their names, select and rate several restaurants we provided. These data are also inserted into the reviews table, and taken into the collaborative filtering immediately.

The photo database stores photos download from yelp, it is downsized to an advisable size with only contains local photos. The photo dataset has a business_id field which can be referenced to the same field in the business dataset.

There are 2 aggregative fields in the business collection. Stars are calculated by averaging all the stars in the review of a particular restaurant. Review counts are calculated by summing up the number of reviews toward a particular restaurant.

The categories dataset aggregates keyword such as “Japan”, “pizza”, “spicy” in the category field of the business field. It counts how many times each keyword occurs for all the restaurants.

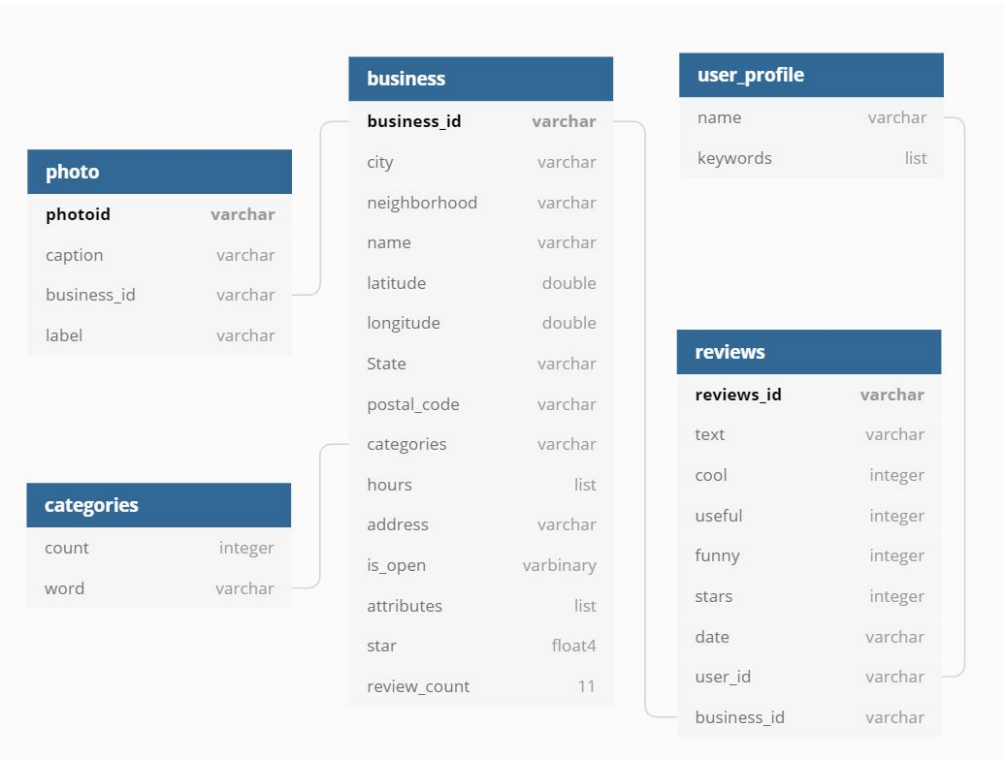


Fig. 7: Database structure diagram

4. Algorithm

Adaptive Search

New users are signed up with 5 initial keywords that they pick up at the registering questionnaires(Figure 3). Users are asked to select at least 5 categories that they are interested at. Words at questionnaires are generated with the most popular ones. The most popular ones are used to describe most frequently on business records. They are adaptive, that means, when new business comes in, this table should be re-calculated.

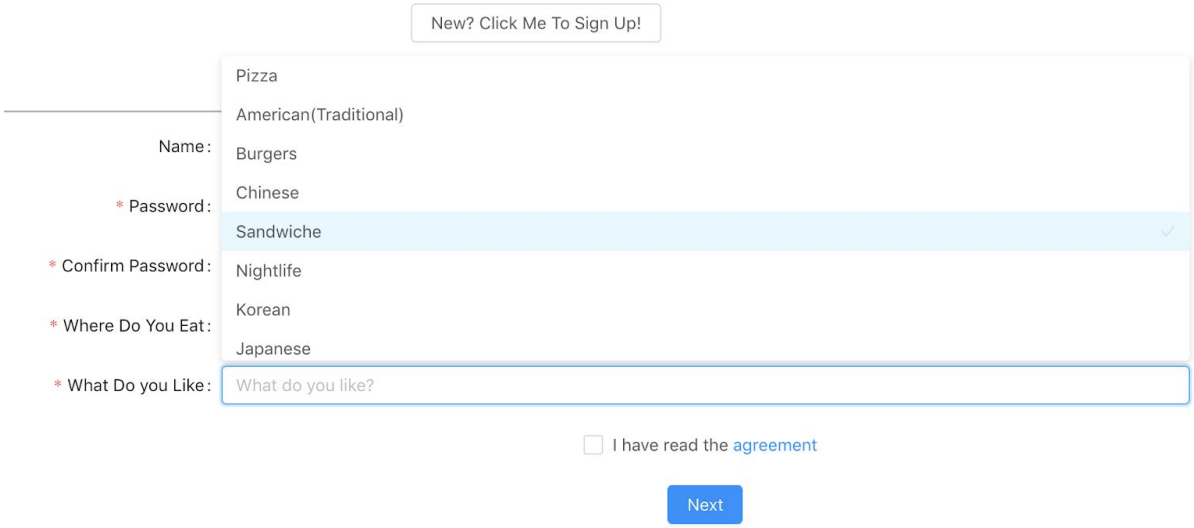


Fig. 8 Select at least five keywords while sign up

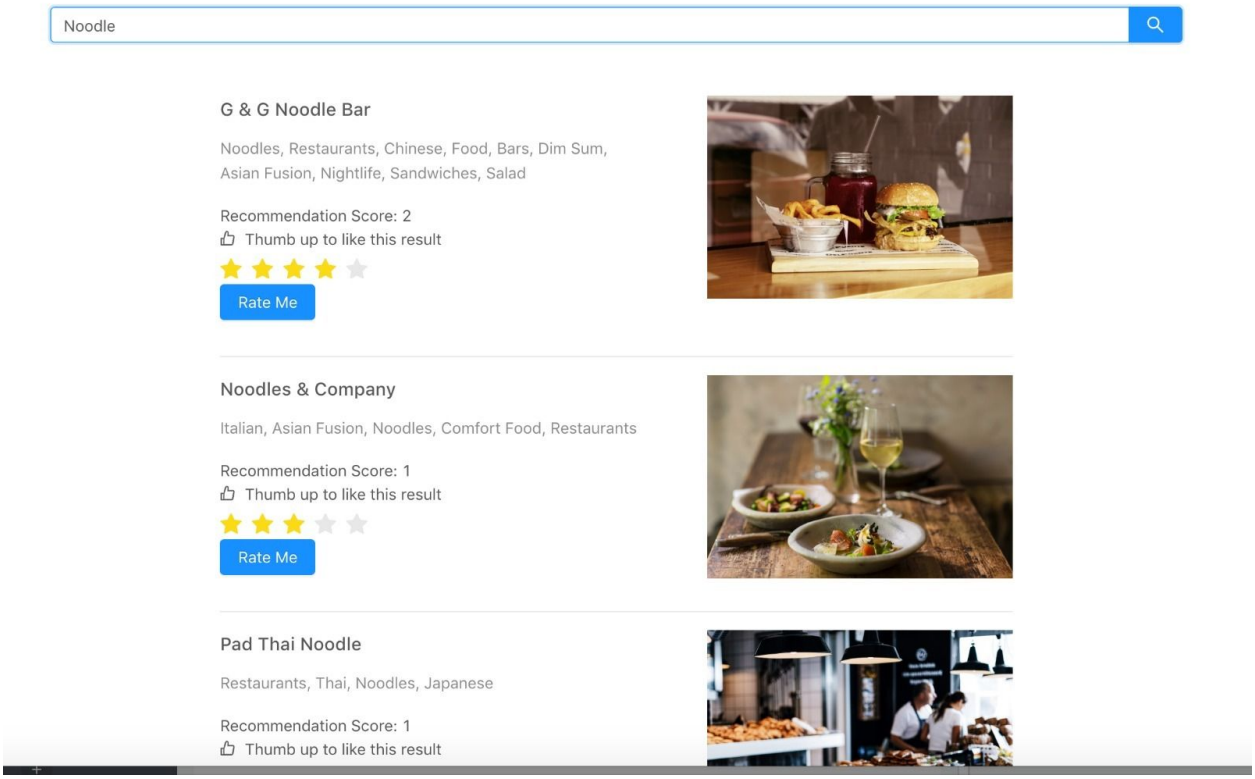
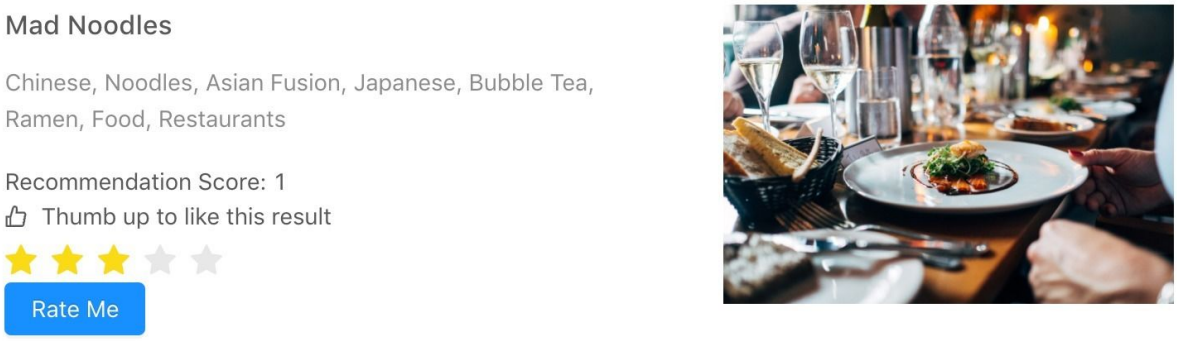


Fig. 9 search result for "noodle" after sign up, one of the keyword for current user is 'nightlife'

User's keywords have a limitation of 21, and when length limitation is reached, oldest words can be removed until it meets the limitations.User's keyword vectors changed by their behavior by clicking "like" button on restaurants page. Each like behavior will attach 3 keywords, selection from[0,200), [200,700] and (700, 2305], which is divided evenly by distribution. Each word is selected randomly from each interval, and if there is not a word in that interval, the system will skip this interval.



Search results are sorted according to users keywords vectors. When he types in a word to search a particular restaurant, for example, “Noodle”, restaurants with “Noodle” in their name are returned. They are then calculated to have their relevance to this user’s interest. The logic behind relevance calculation is simple. We add 1 to the restaurant relevance if it meets one word of the user’s keyword vector, and sort them according to their relevance.

For example, User “Yimeng” has [“Japanese”, “Vegetarian”, “Coffee & Tea”], and after she like the business “Hampton Inn Pittsburgh University/Medical Center”, her keywords vector becomes[“Japanese”, “Vegetarian”, “Coffee & Tea”, ”Screen Printing”, “Local Service”, ”Hotels”], based on the category words according to the business itself.

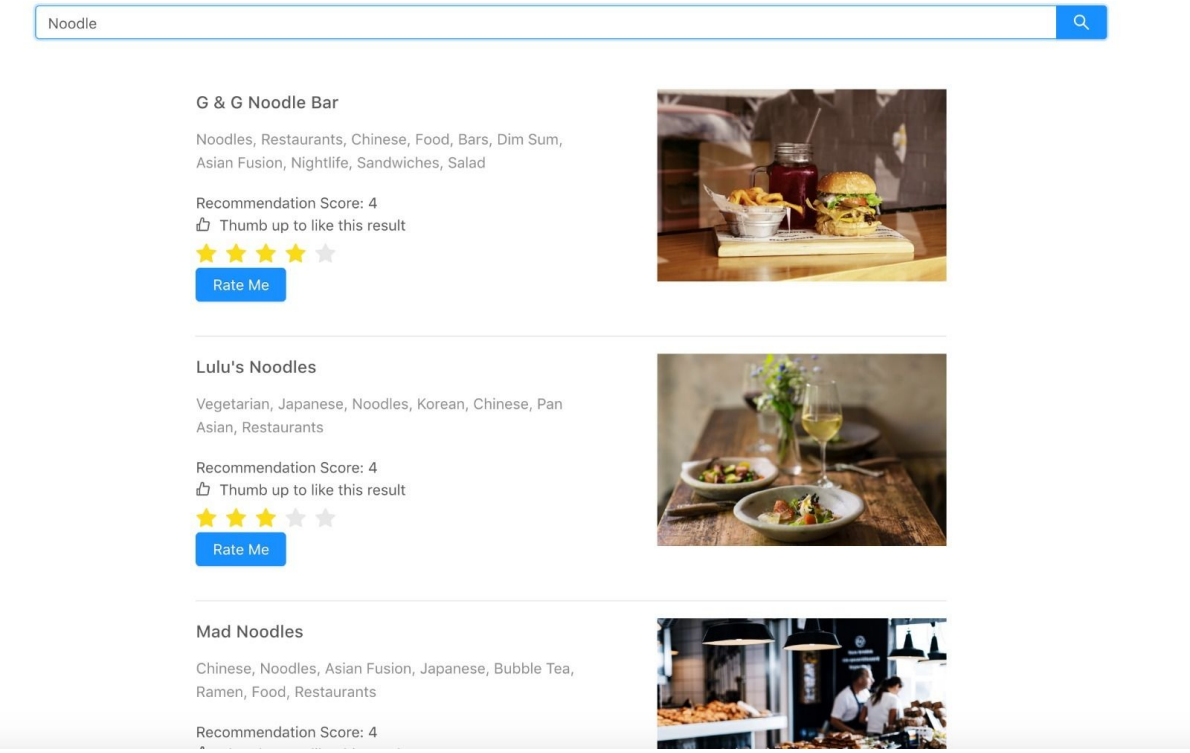


Fig. 10 New search result for the current user, there are more keywords for the user after he click like for 'mad noodle', and the search result changed with new keywords

```
set([u'Screen Printing', u'Coffee & Tea', u'Vegetarian', u'Japanese', u'Local Services', u'Hotels'])
<pymongo.results.UpdateResult object at 0x10b024248>
```

Fig .6 Shows the changed user keywords vector

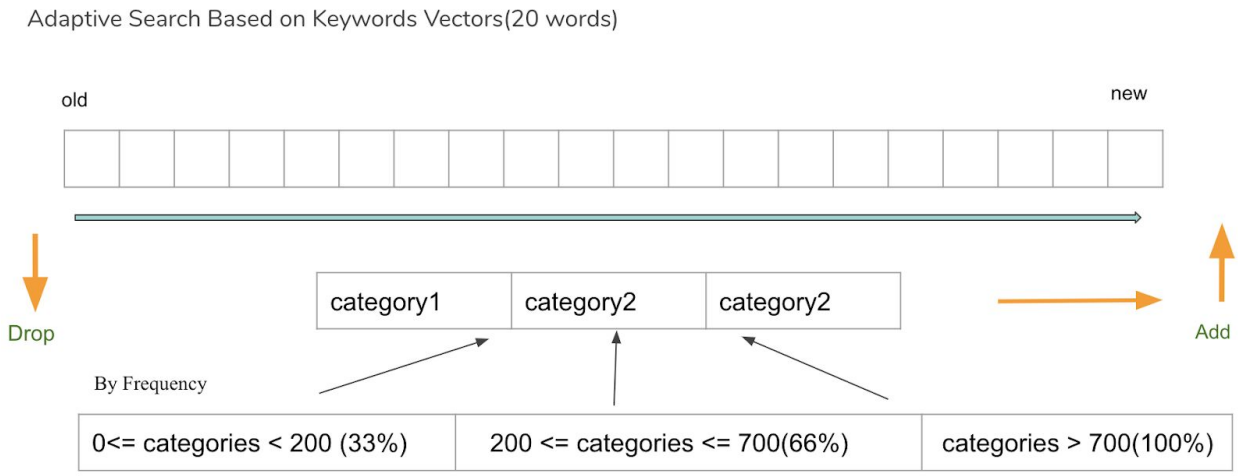


Fig. 11 The keywords vector's adaptive changing process

Personalized Recommendation

We started with just over 200 thousand reviews and computed what's known as a pivot table which is in the format of a matrix. Columns represent businesses and rows represent every user, so every cell corresponds to the rating given by a user to a business. This matrix was very sparse. As recommended in class, we decided to go with item-item collaborative filtering to get our recommendations.

We start with checking how active the user is. If he is a new user or he has less than 3 ratings, we ask him to rate at least 3 restaurants before we can recommend anything(Figure 8). If he has more than 3 ratings, we look at the distribution of his ratings to see his 75th percentile of ratings and how many restaurants has he given those ratings to. This is used to find top restaurants for that particular user. Following flowchart illustrates the process explained above.

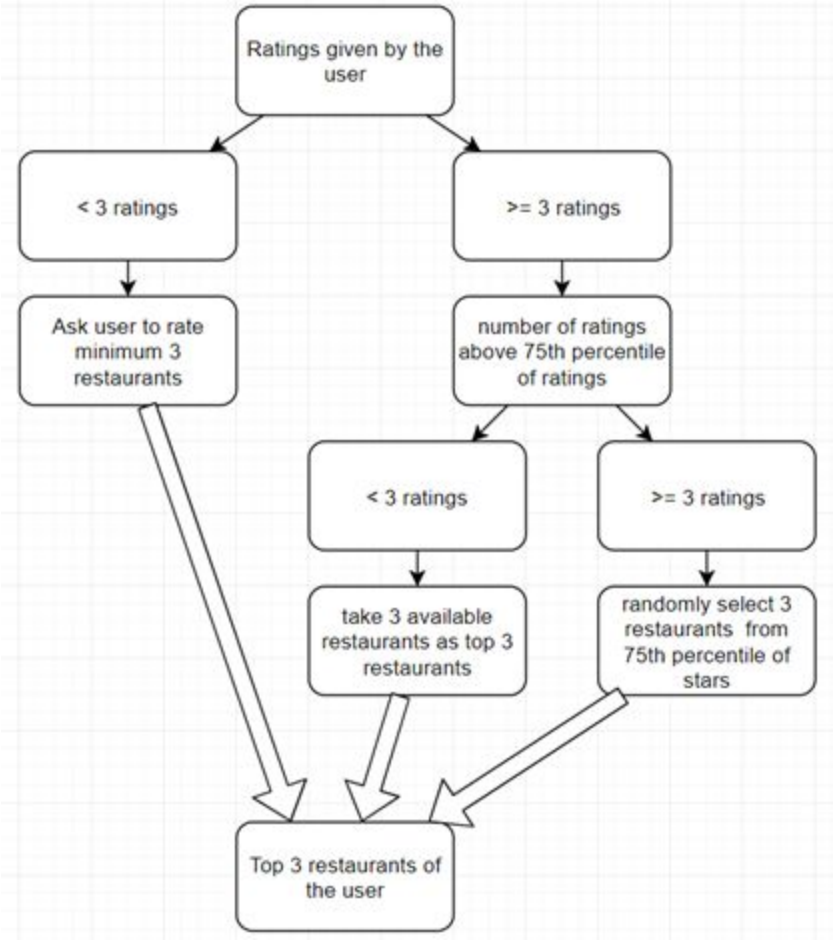


Fig 12. Computing top 3 restaurants of a user

Once we have the top 3 restaurants of a user, we use the pivot table generated earlier to find other restaurants that are similar to his top 3 restaurants. We find restaurant–restaurant similarity like item-item similarity and filter out the ones which are less than 50% similar. Among the ones that are at least 50% similar, we sort them by average ratings so that number of ratings are also considered. Then we choose 5 restaurants for each restaurant in top 3 of the user. Out of the final 15, we randomly select 5 and recommend just to keep things dynamic. Following flowchart illustrates the recommendation process.

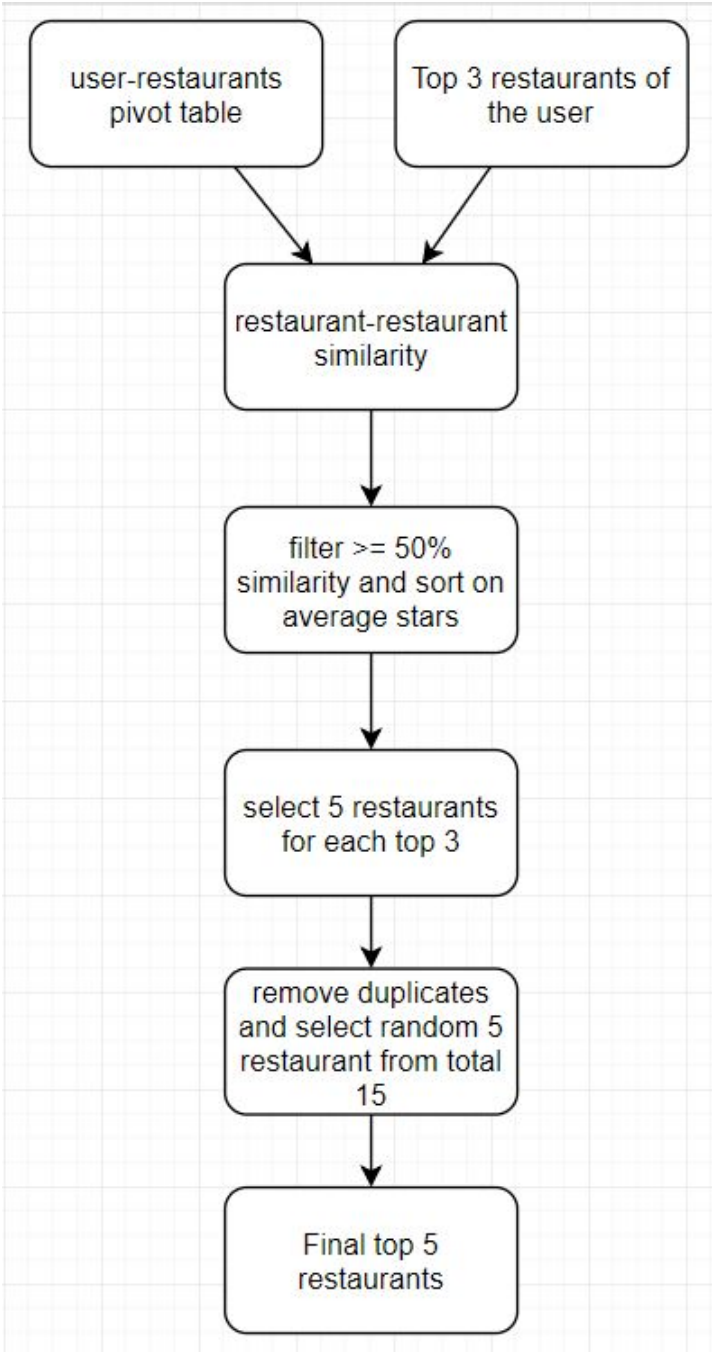


Fig 13. Item-Item Collaborative Filtering

5. Timeline

By October 27, we discussed the details of our system.

By Nov 7, we have a first version of how collaborative filter might be working.

By Nov 14, frontend and backend are started. The first table of all the business is set up.

By Nov 21, the backend can communicate with the MongoDB database.

By Nov 28, we build up other tables and filter the main business table to have only Pittsburgh business.

By December 3, the first version of the collaborative filter is working.

By December 5, the backend is almost done. Keywords and restaurants for registration are picked up.

By December 8, the adaptive search algorithm is done.

By December 9, the adaptive search is modified to version2.

6. Individual Contribution

Frontend : Xinyi Li

Backend: Yimeng Liu, Shashank, Ruochen Liu

We have 4 group members: Ruochen Liu, Shanshank, Xinyi Li, and Yimeng Liu. Xinyi Li did the client side of our system. Yimeng Liu and Ruochen Liu are in charge of database designing and backend coding. For adaptive personal part, Shashank is in charge of recommendation and Yimeng in charge of adaptive search.