FoodiePal Project Report

- CS411 Course project

Our website FoodiePal is a “local yelp” website that provides information about Restaurants located at Urbana-Champaign. Users can perform search by restaurant name or category name, review restaurants and get personalized recommendations based on past reviews. We implemented fuzzy search and machine learning as our two advanced functions.

Spring 2016

Authors: Jiayu Chen, Chenying He, Jingjing Huang, Xinyao Huang

University of Illinois at Urbana Champaign

Spring 2016

1. **Brief description**

Our website FoodiePal is a “local yelp” website that provides information about Restaurants located at Urbana-Champaign. Users can perform search by restaurant name or category name, review restaurants and get personalized recommendations based on past reviews. We implemented fuzzy search and machine learning as our two advanced functions.

1. **Usefulness**

FoodiePal not only provides quick and easy restaurant information look-up (such as location on map, short description, average rating), but also recommends eatery choices based on past reviews through trained machine learning model.

1. **Discussion of our data**

We got initial data about businesses and reviews from Yelp’s API. We then added a table for users to allow visitors to register on our website for us to record their preferences. We then added necessary tables that enable mapping of keys between tables. For more information, see our ER diagram below.

1. **ER Diagram and Schema**

ER diagram and schema has changed significantly since the Stage 2.

* Initial Design:

- Schema:

Users(UserID,name,gender,age) Restaurant(RestaurantID,name,phone,Address,rating,businessHours,reserveationURL,category,reviewCount,price,imageURL)

like(UserID,RestaurantID,rank)

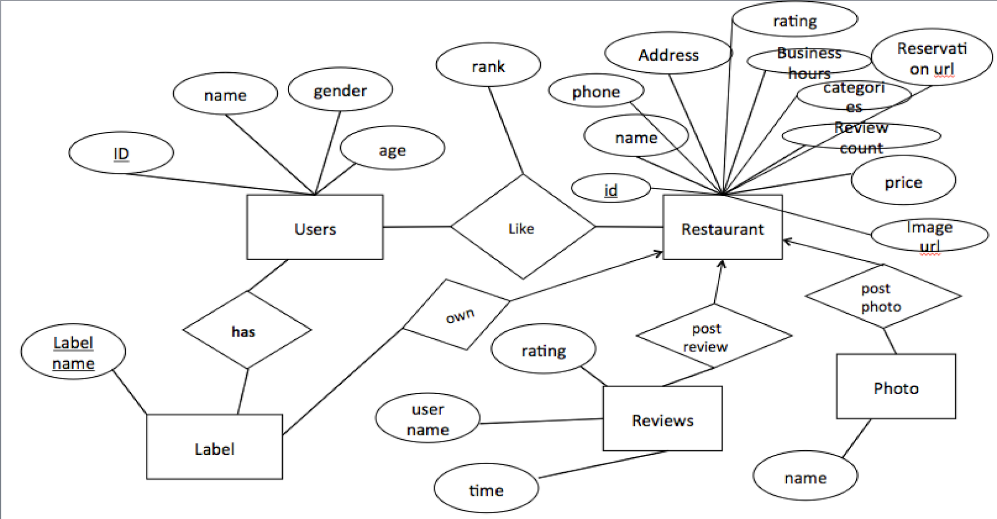
label(labelName)

has(UserID,LabelName)

own(LabelName,RestaurantID)

postReview(RestaurantID,rating,userName,time)

postPhoto(RestaurantID,Photoname)

* ER diagram: 

**Final Design:**

* Schema:

Categories(id,category,subcategory,class\_tag,class\_tag2)

Ratings(id,user\_id,bus\_id,rating,rating\_date)

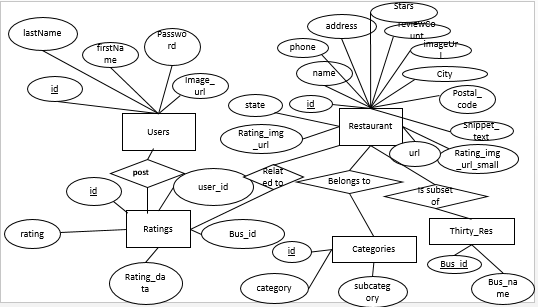
Restaurant(id,name,phone,address,stars,reviewCount,imageUrl,city,postal\_code,snippet\_text,state,rating\_img\_url,rating\_img\_url\_small,url)

Restaurant\_Categories(rid,cid)

thirty\_Rest(bus\_id,bus\_name)

Users(id,firstName,lastName,password,image\_url)

* ER Diagram:

****

**Changes:**

There are several changes to the schema and ER diagram, not only because we are adding advanced functions, but also because we are changing some of the logic when we are making progress. Here are some noticeable changes:

* **Users:** We decided to drop name and gender, as they are not important to the development of recommendation system. Besides, we added password and image\_url for the sign in and register function.
* **Ratings:** this is the updated schema of the Reviews. It only contains the numeric data(0.0-5.0) now, as there’s already a snippet review in the Restaurant table
* **Categories**: this is the updated schema of the Label. We now saves all the categories that all of the restaurants show up in.
* **Restaurant\_Categories:** this represents the Restaurant Belongs to Categories relationship. We store rid(restaurant id) and cid(category id).
* **Restaurant:** Not much changes. We add a snippet test column.
* **Thirty\_Res**: This is a subset of all the restaurants. This table is mainly used for the choosing restaurants you may like page when the user is registering.

1. **Data source**

We accessed Yelp’s API (https://www.yelp.com/developers/) using code in python and filtered out informations of 200+ local restaurants and their reviews. For the purpose of machine learning, I found larger datasets released by Yelp. There are three json files for this part: businesses, reviews and users. We also get the info from the user register page. When the user registers for an account on our website, we will store the user’s personal information, such as name, gender and etc.

1. **Functionality of application (feature specs)**

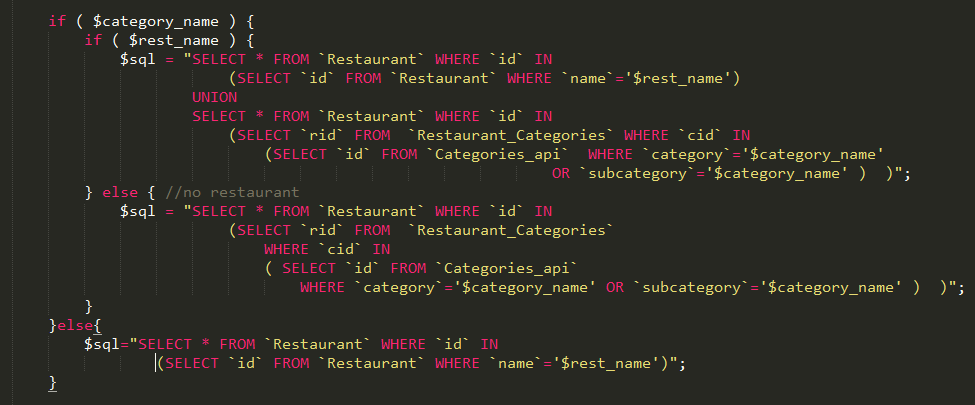
* User register
* User sign in
* Manage restaurants (insert, modify, delete)
* Fuzzy search (by name and category)
* Restaurant personalized recommendation

1. **Explanation of one basic function**

Users can sign up and keep an account at FoodiePal. They are directed to "/pages/register.html” to complete the registration. We record the profile pic as well as other general information about them by processing the data submitted along with the POST request sending to a PHP file, where we respond to the request by storing the user information in the user table. We notify user if the email address, which serves as user’s id, is already taken by another user.

1. **The actual SQL code snippet**

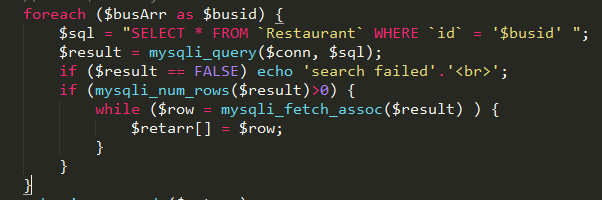
* After performing fuzzy search algorithm on search word, there are at most two estimated search terms sent to the back-end, one being category name and the other being restaurant name. PHP below matches restaurant by category or/and restaurant name based on the given search term(s).



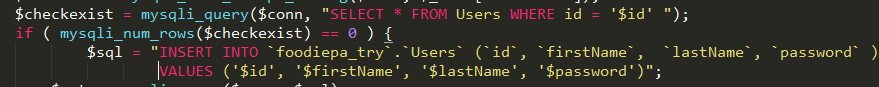
* When logged in user clicks “recommend” button, recommend.php receives user’s id and uses that as key to get all rating records by this user.



PHP then reformats it and send the string through TCP to our “Server”, which runs our Machine Learning R script and return business ids of top 20 recommendations. PHP then queries again to get restaurants list and send it back to the client to display。



* When user tries to login, we check if id is taken and insert new record if not



* When user tries to log in:



* When user rates restaurant:



1. **Dataflow**

* Log in
  1. When user log in, client side will send password and username for server to check the combination.
  2. If the combination is legal, user info will be send back and stored in local storage for later checking.
  3. When user click logout, the local storage will be cleared.
* Register
  1. When user registers and the email is not used, a new row will be created in the User table with all the details about the user.
  2. After registered, user will be lead to the page for selecting ten favorite restaurant out of 30 restaurants listed. When they submit the result, server will and 10 new rows in the Rating table about that user for later use of recommendation.
* SearchDetails page
  1. When user in the recommendation page or in searchResult page, he/she can click the title of the restaurant. Client will save the current restaurant information and display the details to the user
  2. At the same time, client will query the Google Map API with the current restaurant address to get the map generated and displayed
  3. If user logged in already, he/she can click the rating stars. Once he/she clicked, client will send restaurant id, user id, and the rating to backend. And backend will add a new row of user rating in the Rating restaurant with current dateTime.
* SearchResult page
  1. After login, user can click recommendation button on top of the nav bar to get a list of personalized recommended restaurant. Client side will send user id to backend. Backend will be charged of checking all the rated restaurants by that specific user and apply machine learning on it and send back client the list.
  2. register/login: see bullet points 1 & 2
* Index page
  1. When user type in restaurant name or category name in search box, front end will finish fuzzy search and ask server to query Restaurant table with category or/and name as keyword and return user a list of related restuarant
  2. register/login: see bullet points 1 & 2
* Recommendation page
  1. List of result of personalized recommendation.
  2. Every time click the ‘recommendation’ page, the result will be updated.

1. **Advanced Functions Explanation:**

* Recommendation system integrating Machine Learning
  1. **Summary**

A smart recommendation feature collects and makes use of our website user’s review history. Because our website only holds information on local restaurants and Yelp’s API only provides one review per business, we are unable to train our model on local restaurants. We found good datasets about a few cities released by Yelp. We wanted to identify a fair way of predicting user’s preference for a restaurant. Finally we decided to analyze and predict user’s preference for each categories since the Yelp’s categories are almost shared by the machine learning dataset and API’s dataset. With this information in mind and as well as what categories each restaurant has, we can calculate the preference users would have for all 200+ restaurants. Finally, we output the top 20 ones as our recommendation.

* 1. **Dataset**

- 11537 businesses with 500 + categories, reduced to 5400 restaurants with 109 categories related to restaurants, restaurants have star ratings

- 171296 reviews with ratings by 43876 users

* 1. **Classify categories using K-median Clustering**

Firstly, we want to study the distance between categories and cluster them in order to reduce dimensions of data we are dealing with. Our motivation of clustering categories is as below:

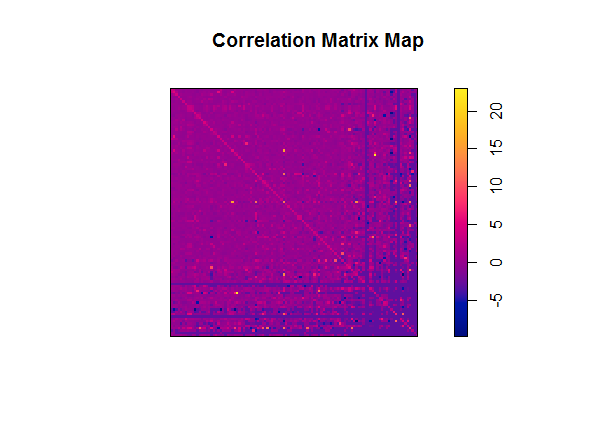
Basically as mentioned in the summary, we want a way to represent some User u’s preference on categories. We can easily figure out a way to obtain from past ratings a 109 dimension vector V for each User u, each dimension representing u’s preference on one category (recall there are 109 categories). For example, we can increment one count for category n, if User u rated a Restaurant B who has tag/category n. We can add weight to the count according to (this review’s rating – this B’s average rating). But with such big dimension along with few reviews, we will get very sparse vector which isn’t informing nor desirable.

We constructed a 109 \* 109 “covariance” matrix X, with entry X[i, j] = X[j, i] denoting the correlation between category i and j, i.e. more positive value means higher likelihood of users who prefers one of the two categories also prefers the other category.

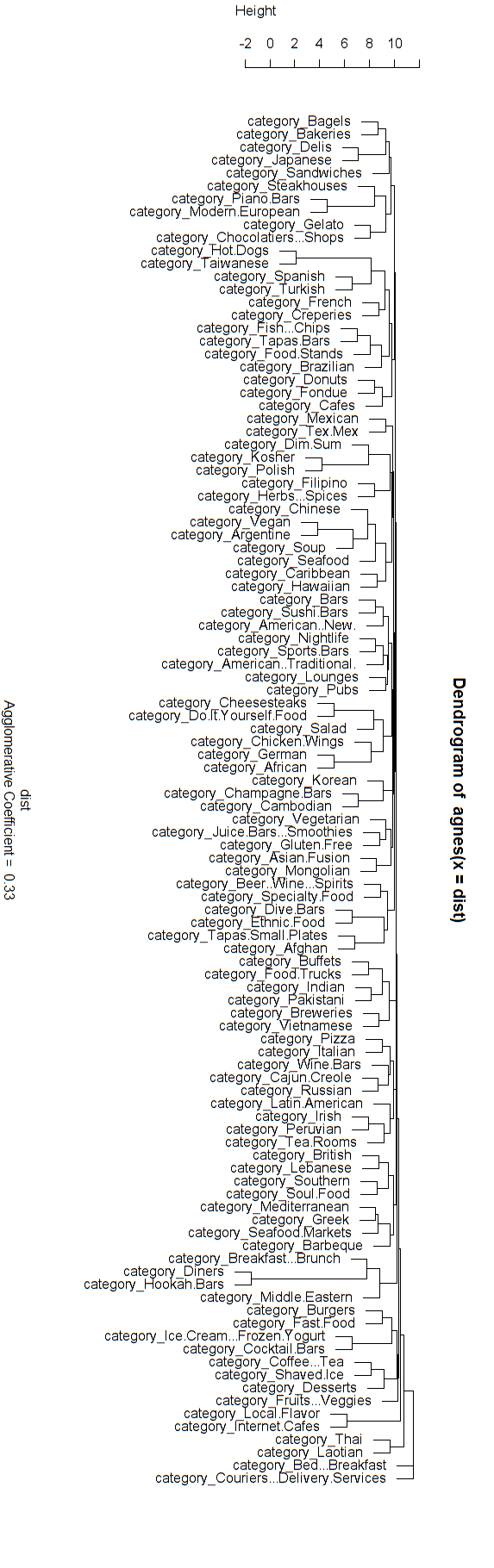
Divided by

\*we adopt the term covariance and correlation loosely here because we didn’t standardize it

Below is a visual representation of the distance matrix,

****

Convert this covariance matrix into distance matrix: each entry = (9 - each entry)/2. We then can perform k-median (more suitable than k-mean in this case) to cluster categories into 9 classes. Just for fun, we attached the hierarchy clustering result using the distance matrix. We used result from k-median because it’s hard to pick k for hierarchy clustering and it is ambiguous where the cluster broader is.

****

* 1. **Regression to get bias matrix of User u**

We mapped categories in our database to 109 categories in above dataset. Then given a new user’s review history, based on it we can initialize 109-D vector, each entry representing user’s rating bias comparing to all users’ reviews for this category. We perform regression referring to distance matrix X to get full vector of it (since the original one is sparse based on few reviews). We can then calculate the scores based on category ratings for each of 200 + restaurants and output the top 20 ones.

* Fuzzy search:

We consider fuzzy search as an advanced function because without it users would have to spell the name of the restaurant or the categories completely right in order to find the restaurants that they want. With fuzzy search function, user can still find the right restaurant even if they misspell some part of the search term.

We use dynamic programing and backtracking algorithm to find the closest search term in the list of restaurant name and the list of categories using fuzzy\_search\_helper function and compare the both return value and pick the closest one. More specifically, we keep track of the “distance” between input string and each examined keyword. Position mismatch and character difference both are considered as1 distance away. For example, if the input word is “word” then keyword “worr” or “oord” both have distance 1 from “word”. And“wrd” also has distance 1 from “word”. We built a 2d-array for each examined word to keep track of distance from input[0….i] to keyword[0...j],where input[0...i] represents the substring of input string from index 0 to index i and keyword[0...j] represents the substring of keyword string from index 0 to index j. We fill this 2D array up with bottom up manner and 2d-array [input.length-1] [keyword.length-1] is the output for this pair of input and keyword. We repeats this process for every keyword in the list of restaurant and list of categories and keep updating the closest results for both list. And finally do the comparison of distances of both returned results. If the return distances of list1 and list1 don’t vary larger than 2,we output result from both category and restaurant. Otherwise we pick the smaller one.

Also before applied the fuzzy search algorithm on the input string ,we will exam if this input string is the substring of the examined keyword. If it is we will directly return keyword as the closest word for the specific list and mark the distance to be 0. Then do the same comparison to the result of the other list.

For each pair of input and keyword, let n denotes the length of input string and m be the length of the examined keyword,the runtime will be O(nm). So if the category list have length x and restaurant list have length y, the total runtime for fuzzy search is O((x+y)(nm)). Since x,y,n and m are not very large, our fuzzy search will complete in a very short time.

Recommendation system via machine learning

1. **One technical challenge encountered**

One technical challenge is how to integrate our R code that trains machine learning code into our website. We processed relatively large datasets locally using R code and we’re able to save the R environment that contains the trained model so we don’t have to repeat training each time a new user asks for recommendations. However, we had trouble integrating it into PHP while using Cpanel, which doesn’t seem to support R.

We thought of using Microsoft Azure Machine Learning Api, where we can save our model and employ it as a web service. But Azure’s Api has many restrictions and specific formats that we need to follow when transporting data to and from it. Moreover, R is picky about its input data type also. We finally gave up on that when Azure seems to be having bugs within that is causing it to have extremely long responding time.

We ended up using a VM provided by school as a “server” and our recommand.php sends request to the server using TCP protocol. The general idea is that when server receives recommendation request from us using TCP, it parses the received buffer which contains the user’s past ratings and the corresponding business ids. It then runs the R code, and using the csv files exported from our database to get the necessary information about our restaurants. The server calculates the top 20 restaurant ids of restaurants most likely to be liked by the user and then send back the ids as a comma separated string back to the PHP.

However, this leads to a problem that our website can only performs this process when the TCP server python script written on the VM is run. Once we log out, we cannot access to the server. We are looking for free server service online, and hopefully we’ll transport our code there soon.

1. **State if everything went according to the initial development plan and proposed specifications, if not - why?!**

Yes we went with our initial plan.

1. **Labor division and teamwork**

* Front End: Xinyao Huang, Jingjing Huang
* Back End: Jiayu Chen, Chenying He
* Back End and Front End Connection: Jiayu Chen, Xinyao Huang
* Advanced functions:­­­­­­

- Machine Learning: Jiayu Chen, Chenying He

- Fuzzy Search: Xinyao Huang, Jingjing Huang

For the most part, the front-end and back-end groups work separately. But once a week all four of us meet to merge code and update each other on individual progress, discuss any challenges that we’ve encountered. We backup our work periodically.