Collaborative Dining: A Social Recommender System for Restaurants

Hamilton Nguyen[hamiltonnguyen@berkeley.edu] and Sylvia Lin[sylviarlin@berkeley.edu]

Overview

Project Description

When people want to eat out, one of the difficult choices a customer has to consider is which restaurant to go to. It can be a matter of finding something suitable for a large group of people, or even just indifference to the locations at hand. Because there are so many choices out there, people spend a good amount of time deciding where to eat. Our project hopes to make this process more efficient by directly recommending restaurants to people. We are working on providing a recommendation application based off Yelp to provide users with a single choice of dining that is best matched with their needs and tastes.

Currently, Yelp is a local search engine that can help users find, learn about, and rate restaurants in a given location. It includes a social aspect where users can rate and review locations, creating a reputation system. However, it requires the user to individually look through each option, making the decision process demanding. Instead of having the user read through the choices, we are creating a recommendation algorithm that compares the user to the data accessed with the use of Yelp’s API. Given a person, we can profile him/her by their preferred style of restaurant, type of food, and price range. This is done through ratings this user may have given on restaurants through Yelp. We also take restaurants that the user has checked in at through Facebook and Foursquare, and then match these restaurants with new locations. Frequent check-ins to locations implies a preference for a given restaurant, which we can use to further profile the user.

Using this information, we then compile a list of options that could be compatible with the user’s restaurant preferences. There are several ways
restaurants can be added to this list. By comparing a Yelp user to other users with similar profiles, we can pull restaurants that other users have attended and liked that the current user hasn’t been to before. Additionally, we can find restaurants that match the user’s profile. These options are then given a certain weight based on their compatibility with the user’s profile. Check-ins made by their friends also affect weights, since users are more probable to go to a restaurant that someone they know has tried and liked. Given this collection, we use a specialized algorithm to output a single restaurant choice for the user to try, where options with higher weights are more likely to be recommended.

**Research Question**

We have two goals for this research project. The first, of course, is to build a working restaurant recommender system. To accomplish this, we will be using ‘conventional’ recommender techniques to build a User-User, memory-based active filtering system. This will in large part be similar to the existing prevailing implementations of recommender systems, such as *Entree* and *Zagat*. Our second goal is to differentiate ourselves from these existing implementations by expanding on the conventional algorithm with a social-based approach.

Although social-based recommender systems have been discussed in the literature, they are not seen in the popular restaurant recommender systems. This is the issue we hope to explore. What we want is to first use our implementation of a conventional User-User recommender system to establish a baseline. On top of this, we will then integrate social-based techniques and reevaluate our success rate. Ultimately, the research question we seek to answer is whether, within the domain of restaurants, a social-based recommender system will be more successful than a conventional recommender system.

**Why Social Based?**

There are two major motivations as to why we want to explore a social-based approach. The first is largely pragmatic - Yelp, and other restaurant rating sites already have social networking implementations. Almost all can be linked to one’s Facebook profile, and Yelp has its own system of friend networks. Since the information is already there, we may as well incorporate it into our algorithm, since it takes minimal additional effort to extract it.
The second motivation is that the domain of restaurants seems to naturally lend itself to a social-based approach. Restaurant dining, is itself a social activity. One generally goes to restaurants to eat out with friends. Consider this scenario: two users have nearly identical ratings for the same set of restaurants. However, the two users also have no social connections in common. Is it still likely that they will seek out the same restaurants to dine at? In a conventional recommender approach, these two users would be indistinguishable. But the reality is that they will probably not choose to dine at the same restaurants if they are part of distinct social circles. Our recommender system will account for this additional dimension.
Progress

Planning
There are currently three main stages in our schedule. First we want to create a recommendation algorithm that profiles and compares users to some extent. Our second stage consists of fine-tuning this algorithm. We can fix the base algorithm to be more accurate, as well as adding in additional data gathered from outside sources. Aside from Yelp’s API, we can also create better user profiles by pulling information from Facebook and Foursquare. By incorporating check-ins as part of the algorithm, we can tweak a person’s preferences, as well as add in the effects of social media such that users are more prone to go to places their friends have gone to. Our final stage is to create a User Interface to allow users to interact with the algorithm. We also hope to add functionality to this algorithm and expand on its uses as a continuation of this project.

This is the current work-flow chart for the algorithm we have created so far.
Our work-flow follows straightforward path. By working with the Yelp API, we can access Yelp’s database and mine data into our own local database. With this information, we create a profile table to organize the information as user and restaurant profiles. Given a user input, we can then draw structured data from our table and run it through our k-nearest neighbor based algorithm to output a restaurant recommendation. At this time, we have established a baseline algorithm using data pulled from Yelp. We simplified our task down to a very basic recommendation of new restaurants that will be compatible with a user’s tastes. While we hope to improve accuracy of our algorithm later on, we are working on fine-tuning the algorithm we have right now by focusing on Yelp users.

Data

The data we are using is user-centered. More specifically, we query a user via Yelp’s API and then receive a reply that contains data detailing that user’s activity across all of Yelp. We record that user’s location, restaurants reviewed, and friends, storing the information into a local database (the local database is necessary only to avoid redundant queries in our testing). Using this data, we can begin creating profiles for the users.

In general, these profiles are how users are viewed by our recommender algorithms. In the context of this project, these profiles encode the relevant information by which we decide to recommend them a restaurant. In our current baseline implementation, the profiles are simple - a user’s profile is simply a mapping between the restaurants they have reviewed, and the ratings given in those reviews. Sample profiles will look as follows:

Alice = {‘Subway’ : 4.0, ‘Top Dog’ : 3.5, ‘Bongo Burger’ : 5.0, ‘La Burrita’ : 1.5}

Bob = {‘Subway’ : 3.0, ‘Top Dog’ : 1.5, ‘La Burrita’ : 2.0, ‘Intermezzo’ : 4.5}

As seen above, it is not necessary for profiles to contain an entry for every restaurant, nor do we expect any two profiles to have entries for the exact same set of restaurants. Although it is not apparent right now, it is important to note that our process does distinguish between a ‘highly negative’ rating (i.e. a rating of 0) and the absence of a rating.
These profiles are then stored in a *rating table*, which has the following format:

<table>
<thead>
<tr>
<th></th>
<th>Rest. 1</th>
<th>Rest. 2</th>
<th>...</th>
<th>Rest. M</th>
</tr>
</thead>
<tbody>
<tr>
<td>User 1</td>
<td>1.5</td>
<td>2.0</td>
<td>3.5</td>
<td></td>
</tr>
<tr>
<td>User 2</td>
<td></td>
<td>2.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td>3.5</td>
<td>5.0</td>
<td></td>
</tr>
<tr>
<td>User N</td>
<td>1.0</td>
<td>3.0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Each row of the table corresponds to the profile of a single user. This organization of the table lends itself to a User-User comparison - profiles can be evaluated by examining rows of the table relative to each other.

**Algorithm**

A high-level summary of our recommendation process is this: similar users prefer similar restaurants. It is a fairly straightforward mode of thinking - to recommend a restaurant to a specific profile, we make N-1 user-user comparisons with all other profiles in our database. We find some small set of these that have the highest degree of similar with the profile we wish to recommend to. We then use this set of ‘best matches’ to then decide upon a restaurant to recommend to the user.
The algorithm we are employing is the *K-Nearest Neighbors* process. Very simply, it finds the k most similar profiles to a given user, using *distance* as the metric of similarity. In our process, we view the user profiles as vectors, and distance is defined simply as Euclidean distance:

```python
def distance(self, user_ratings_1, user_ratings_2):
    sum_of_squares = 0
    for restaurant in user_ratings_1:
        if restaurant in user_ratings_2:
            rating1 = user_ratings_1[restaurant]
            rating2 = user_ratings_2[restaurant]
            sum_of_squares += square((rating1 - rating2))
    return sqrt(sum_of_squares)
```

This gives us a straightforward metric to determine the likeness of two users. One thing to be noted is that, as mentioned previously, if a rating appears in one profile but not the other, it does not contribute to or affect the distance in any way. This is an algorithm design decision that may be revisited later.

Once the concept of distance has been defined, it is natural to define a way to find nearest neighbors.

```python
def nearest_neighbor(user_to_compare):
    for user in rating_table:
        dist = distance(rating_table[user], rating_table[user_to_compare])
        if dist < min_so_far:
            min_so_far = dist
            nearest_neighbor = user
    return nearest_neighbor
```

The last step in the process is to move from a set of k most similar profiles to an actual restaurant to recommend. Our current algorithm for this step is also simple: for each restaurant, we sum together its ratings as given across the profiles in the k nearest neighbors. The restaurant to recommend is the restaurant with maximal sum that the user has not already visited.
Problems Faced

One of the biggest problems we faced when trying to create a baseline algorithm was deciding on a user group to focus on. We originally wanted to generalize this application for public use, but we ran into the trouble of how we would profile users. Our original hope was to use Facebook as a user base, taking check-ins for restaurants and matching those names with restaurant data through Yelp to create user profiles. However, Yelp also had its own user base, and the users were not necessarily linked to Facebook so intersections were unavoidable. Thus we decided that we should keep our user group simple for now, and focus on members of Yelp. Our final decision for our current algorithm profiles users according to their reviews, however, so profile accuracy is not consistent, especially if the user is not active in reviewing restaurants. The fewer reviews a user makes, the less data we have to profile a user with.

We also had to consider which parameters were useful for profiling a user. Using the Yelp API, we could access a large amount of data about a given user, and we had to decide which parameters could be considered random variables. We felt that we would start with the most relevant variables first, such as restaurant reviews, and focus on organizing that data into a profile table. Some other things that could possibly influence taste were age and location, since user preferences are heavily influenced by people around them.

Next Steps

At this point, we have a functional recommender system that given some dataset as input, can generate restaurant recommendations. We also have a means to extract data from Yelp. Our baseline implementation is running, so our next steps are to fine tune our current implementation, establish an accuracy metric to evaluate the performance of our system, and to begin the implementation of an expanded social-based approach.

Algorithm Design

There are several design parameters that can be reexamined, and the choices for each of them deserve some consideration:
Data Collection - Although there is a myriad of user data available, the only information actually taken into consideration is user ratings of restaurants. Other factors are surely at play, including price, location, hours of availability, etc. Exploring these other factors may lead to changing the definition of profiles and thus expanding the algorithm type - we currently employ User-User comparison, but it may be fruitful to consider User-Item comparisons, in which we consider how well a restaurant’s characterizations match a user’s ideal restaurant, or Item-Item comparisons in which we compare known favorably rated restaurants with other restaurants.

Notion of Distance and Similarity - Our current working definition is the Euclidean distance between the entries of each profile. This may not necessarily be the best metric of similarity, in which case the question becomes how do we define similarity, and how do we represent user profiles to function in the new system?

Rating Normalization – Users may vary greatly in the nature of ratings they give to restaurant. Of course, users are operating on different criterion and different scales. For example a 2 from one user may connote the same opinion as a 4 from another user. One way we may address this issue in the future is rescaling all of a user’s ratings relative to its average rating.

Absent Ratings - A more specific issue to consider regarding our current definition of user profiles is absent ratings. Currently, we ignore them - that is, when comparing two users, we only examine the restaurants that are rated by both users. The consequence of this is that if we look at all the pairwise comparisons, different sets of restaurants will be considered in each comparison. The effect of this on our accuracy is unknown. The problem becomes whether we continue to ignore absent ratings, or treat them as implicit ratings. Does a lack of a rating imply a lack of interest, or simply a lack of knowledge?

Value of k - The value of k in our kNN algorithm is a small part of the overall picture, but an important point of consideration nonetheless. Depending on how we move from the set of k nearest neighbors to our final result (discussed in the next section), the value of k can affect the restaurant we decide on. A small value of k may not accurately capture the subtleties of the entire picture, while a large value of k may dilute the information too much.

Deciding on a recommendation - The kNN algorithm can only output the set of the k most similar profiles to the user for whom we wish to recommend. The final step is to bridge the gap from having this set of profiles to generating a
single restaurant recommendation. Our current approach is simple - we find the restaurant that is best rated across the entire set of k profiles (and that has not been visited by the user before) and recommend that restaurant. But there could be alternative methods that have greater yields. For example, we could use this set of k profiles to initially filter a majority of the restaurants, and then employ a different approach, such as User-Item comparison to come up with a final answer.

**Accuracy Metric**

Because our algorithm and output are based on calculations created with opinions, it’s very difficult to gauge how accurate our results are. One way we can measure our results is to see how satisfied users are with the restaurants we recommend. Our goal is to give users new options that they will enjoy. If the user gives the recommended restaurant a high rating on Yelp, then we will know he or she was satisfied. We can also ask for user input through a survey, to see how compatible the choice was with the user. We will need to gauge the user satisfaction through survey questions using the Likert scale so we can quantify how well our algorithm worked. After every stage of improvements, we will need to test our new algorithm on a set of users to test its accuracy.

**Social Context Data**

As we begin to implement a social-based approach on top of our baseline recommender, we must first consider what social data to consider, and how to retrieve that information.

The first piece of information we need to determine is the social network graph - what users are connected to each other. Yelp itself has its own social networking mechanism in which users can ‘friend’ each other over Yelp. On top of this, many users link their Yelp accounts to their Facebook profiles that can be extensively mined for more social network information.

On top of the social network, we can use ‘check-ins’ of various social media sites such as Facebook and Foursquare. Although much sparser and less well-reported, check-ins are of arguably greater weight than a simple social network connection, since if a user checks in with a group of other users, this establishes a history of dining with these other users.
A separate task is how to interpret this information. There are two prevailing kinds of interpretation of a social connection in the context of a recommender system. The first is that a social connection denotes common opinion. If two users are friends, it is plausible they have the same likes and dislikes, or that one user follows or “trusts” the ratings of the other user.

The second is that a social connection denotes common activity. If two users are friends and dine together option, it is likely that they will visit a restaurant that at least one of them has interest in.

Our end goal is to be able to consolidate the information discussed here into a workable social-based recommender system for restaurants whose effectiveness can be measured against our baseline current implementation.

References

1. Kalseth, Fredrik: “Developing a Restaurant Recommender System” UC Santa Cruz, May 2005
