Recommending Favourable Dishes in nearby Restaurants

Tarun Tater(IMT2012047)

May 19, 2016

Problem Statement

Quite a few online services take user reviews and use them to provide better services by taking in consideration the suggestions and comments in the user reviews. But there are many other things user reviews can do. The user reviews are also being used to personalise the user's experience. This project also aims to do something similar.

Given a user's location(in Bangalore), the aim is to suggest a known favourable dish to a user in a nearby restaurant given. The dishes suggested are one of the dishes the user has already reviewed before.

Data Processing

The data about the user reviews for dishes and restaurants were taken through Zomato's API. Zomato's API gives all user reviews for a particular restaurant in a city. The project only targeted Bangalore. So, a list of all restaurants in Bangalore was retrieved from the Zomato API. There were 7730 restaurants in total. For each restaurant, there were a list of attributes related to it. The attributes retrieved for each restaurant were:

name	hasDelivery	hasBar
longitude	avgCostForTwo	smokingArea
latitude	hasDineIn	hasAC
cuisines	hasTakeAway	acceptsCreditCards
timings	isPureVeg	hasWifi

Table 1: List of Attributes for Restaurants

The data related to timings had different formats and had to be extensively processed to bring it into a standardised format.

Then, for each restaurant, all the user reviews were extracted. The user reviews were filtered to cater to the need of the project. Only those user reviews

were used which had a dish name in them. A list of dishes was compiled from different sources which served as the database of dishes available.

The user reviews were further processed by removing unwanted punctuations. Now, the adjectives which are generally used to describe the dishes was also compiled.

Analysis of Dataset

Each user review was shortened by identifying and keeping only sentences which had opinions in them. Each opinion was classified using a trained sentiment analyser for food reviews. A Naive Bayes classifier was used which gives a positive and negative rating to a sentence. These opinions were thus given a positive or negative tag with respective probabilities.

The opinions with their respective probabilities for positive sentiment were chunked together for each user. There were more than 80,000 users who have written reviews. This rendered a json of each user having dish wise reviews and their probabilities for positive sentiments.

Algorithm

For each user, the list of dishes he/she had commented upon were taken as the dishes he/she would prefer to eat. This seemed a reasonable assumption as the dishes considered were only those which the user had eaten previously and was passionate enough to write a review about it. This may result in missing a few dishes but would never give a false positive.

The number of reviews about a particular dish were also given importance in deciding the favourable dish. The idea was that if a user had reviewed a particular dish multiple times, it was more likely that the user prefers this dish over another.

Algorithm 1 Retrieve Dishes Preferred By User

```
def getPreferences(userId):
#Getting user preferences from dishes reviews
get all reviews for this userId.
for dish in eachReview:
    increment the score for the dish. #Based on a metric
sort the dishes score in decreasing order
#the dish with the highest score is the most preferred dish.
return the top n dishes with their scores from the sorted list.
```

The metric used to increment the score or evaluate the order of preferences of dish can vary. The metric used for this project depends on the number of reviews a user has given about a particular dish and the positive sentiment related to each comment. Taking the number of reviews about a dish by a user as n

- $if\ n < 3,\ score = average\ positive\ sentiment\ about\ dish\ given\ by\ user$
- $if\ n \geq 3\ and\ n \leq 5,\ score = average\ positive\ sentiment + 0.01 * n$
- $if\ n > 5,\ score = average\ positive\ sentiment + 0.02 * n$

Algorithm 2 Recommend

```
def recommend(userId, distancePrefered, lat, lon, currentDay, currentTime):
    feasibleRestaurants = []
    recommendations = []
    for eachRestaurant in restaurantsList:
        check distance from current location
        if distance from current location < distancePrefered:
            if the restaurant is currently open:
                append to feasibleRestaurants
    dishesPrefered = getPreferences(userId)
    for eachRestaurant in feasibleRestaurants:
        if any dish preferred by the user is served by the restaurant:
            calculate the average positive sentiment of that dish in this restaurant
            append this dish,restaurant and avg. positive sentiment in recommendations
    sort the recommendations in decreasing order of their positive sentiment
    return the recommendations
```

For a user, his/her preferred dishes are evaluated depending on his/her previous reviews and based on the above mentioned metric. Then, in the vicinity of user's location i.e., in a user given preferred distance(radius), a search is made for all the restaurants using the Haversine distance formula. The restaurants in the vicinity are then checked if they are open at that particular time on present day. Now, all these open restaurants in the vicinity are checked if they serve any of the dishes preferred by the user. All such restaurants (where any of the preferred dishes are served) are then ranked based on their average positive sentiment corresponding to the preferred dish. The top 10 results are displayed to the user.

The user can put other constraints such as averagar cost, take away facility, AC, vegetarian restaurant etc. All the available options are mentioned in Table1.

Result

Although, there was no direct way to test the results. The hope is that such a system would keep learning from user's behaviour and improve over time. Since, the dishes suggested are one of the dishes that the user has already tried and commented upon, it can be reasonably assumed that the user likes the dish. And since the number of comments about a particular dish are also taken into

account, the suggested dish seems to be one of the favoured dishes to a particular customer, if not the most favoured dish.

The metric to evaluate preferred dishes can be changed depending on the user's feedback. Also, the restaurants can be filtered based on varying importance given by a user to different attributes like AC, Wifi etc.