

We load data into dynamoDB (AWS) which is NOSQL database for querying and processing.

We have user id and business id connected by the reviews given by the user for that business as from the above diagram.

We get the number of reviews given by the user for that particular business Ids and get the top 3 count of the restuarants that the user had attended (Assuming user reviews each restaurant they visit.)

We capture the user location each time he logs in and update the most recent place he visited in the users table. This will help the user to find the other users and the restaurants around.

In business table we will update the popularity by the number of check ins made by the people using the check in table.

Attributes are the type of restaurant example Chinese, indian etc.

Star rating of the user is given to a restaurant based on the reviews given by the user using sentimental analysis.

When a user logs in we get the location of the user where he logged in from. Based on that we will find all the users who are near to him (say 25 miles or 50 miles) and match each user using the review vector and show the suggestions.

Review vector is something which tells the top three business ids he likes based on the reviews given and most frequently visited business types (example Chinese, Italian etc). So we can match the users based on their review vector and the location. Review vector also contains if he likes alcohol, the food items he likes etc.

Review Vector in the user table contains the details of his likeness and the food user prefer.

Based on the number of reviews given by the user for particular business id, we check the ambience and the category of the business id. From the ambience we get the atmosphere of the restaurant user prefers and the category (Chinese, Italian) gives the type of food user likes to eat. If we count the number of business ids for each user id in the review table we can get the top 5 most business ids he reviewed or attended (Here the assumption is that user reviews almost, if not all, all the visits he goes to a restaurant). Price range also can be used to find the user preferences in a restaurant.

From the check in table we can predict the popularity of the restaurant or which time is best to check in for the user. We get the number of check in done at the given hour for a business id, which we update in the popularity field.

Process the reviews or the comments given by the user using sentimental analysis and then fill the star ratings in the reviews table.

Work done so far:

To process large data we are trying to use AWS EMR services, which is cloud service. EMR has in built capability to process large sets of data using hadoop. This will help in parallel processing and running several EC2 instances together will process the data fast.

We are also trying to learn EBS (Elastic beanstalk) which is used to host the web pages. So when we deploy our web coding on to the EBS it will create EC2 instance, and based on the performance parameters like CPU usage, load, bandwidth it will automatically create or destroy the EC2 instances on demand. This helps me more performance.

We are initially storing the data files on to S3 that will be picked by the AWS DynamoDB via pipelining from S3 to DynamoDB for few tables like reviews and check in. For few fields we need to process data using AWS EMR service ( which runs sentimental analysis and category finder) and fill the necessary fields in the tables.

We have created a web page where user enters his log in id and password and logs in. When user logs in it captures his current location and stores in the DynamoDB table in the user record.

We have created python file called category predictor, which we took reference from yelp data examples. It classifies the given food item in categories. For example, it classifies Pasta as a food, restaurant and Italian categories. So this will help to create the review vector for the user.

We have tried sample code given by AWS and getting to know about AWS sentimental analysis to classify the reviews given by the user.

We have uploaded check in and reviews table on to Dynamo DB tables using python code.

Final idea:

User logins to the web page, location is updated in the user table.

Program filters all the users near to the logged in user say around 25 miles using latitude and longitude locations. (Google APIs help in these calculations)

Then we compare the review vector of the user with the result from the above step. Review vector helps in finding out the likes and dislikes of the user. Thus we see if there are any common likeness between the users and print out the top 3 or 4 best match on the web pages. User data also contain friends list for that particular user. We are planning to user that friends data too to populate in the web page.

Thus user when logged in can find someone near to him to go to a restaurant.