Recommendation System Based on Yelp Dataset

University of New Mexico
Electrical & Computer Engineering Department
ECE 551

Shiqian Shen
5/5/2016

Index

PROJECT DESCRIPTION

Object

In this project, I built a Recommendation System based on Yelp Dataset, including 2 million reviews by 552 thousands users for 77 thousands businesses. The dataset is located on DynamoDB of Amazon Web Service (AWS) online. And I will use the AWS-SDK for JavaScript in Node.js with Express Frame to write programs to query and update data in the database, and build a Recommendation System WebApp.

There are two parts for the recommendation system.

- User based recommendation system
  Input is the requirements from a logged user, including location and categories of business. Then the recommended businesses will be shown. Some features taken into consideration for ranking the businesses includes stars of business and preference of the user. The results should be shown in several seconds.
- Business based recommendation system
  Input is the information of the logged business, including location and categories of business. Then potential customers will be shown. Some features taken into consideration for ranking the users includes categories of the business, preference of users and some characteristics of users, including average stars the user give, friends number, elite years of the user. The results should be shown in several seconds.
  Then use a self-built push module to push the business information to these potential customers. Next time, these users log in the system, they will see the push notices in their home page.

Dataset

The dataset used in this program is downloaded from the official website of Yelp. The link is https://www.yelp.com/dataset_challenge .

Business, User and Review are the three tables we used for the project.

Business Table:	65.9MB, including 77,445 businesses.
User Table:	225MB, including 552,339 users.
Review Table:	1.80GB, including 2,025,379 reviews.

The details and relationship among them are shown in Fig.1 below,

In the recommendation system, we take some attributes in the dataset into consideration to make the recommendation decision. They are ‘business_id’ , ‘neighborhoods’, ‘city’, ‘stars’, ‘categories’ from Business; ‘user_id’ , ‘average_stars’, ‘friends’, ‘elite’ ‘fans’ from User; ‘business_id’, ‘user_id’, stars’ from Review. And sometime we need to modify the content in ‘push’ from User to achieve the push module of the recommendation system.

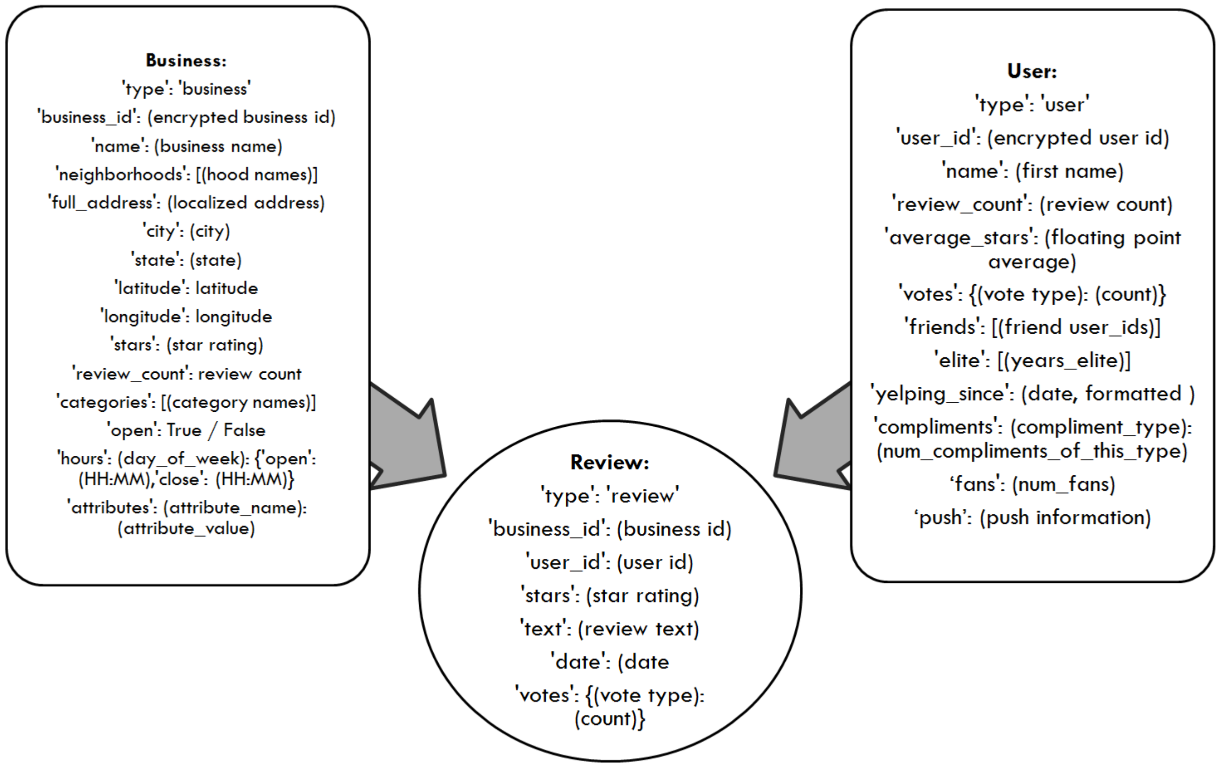

Business:
'type': 'business'
'business_id': (encrypted business id)
'name': (business name)
'neighborhoods': [(hood names)]
'full_address': (localized address)
'city': (city)
'state': (state)
'latitude': latitude
'longitude': longitude
'stars': (star rating)
'review_count': review count
'categories': [(category names)]
'open': True / False
'hours': (day_of_week): {'open': (HH:MM),'close': (HH:MM)}
'attributes': (attribute_name): (attribute_value)
Review:
'type': 'review'
'business_id': (business id)
'user_id': (user id)
'stars': (star rating)
'text': (review text)
'date': (date
'votes': {(vote type): (count)}
User:
'type': 'user'
'user_id': (encrypted user id)
'name': (first name)
'review_count': (review count)
'average_stars': (floating point average)
'votes': {(vote type): (count)}
'friends': [(friend user_ids)]
'elite': [(years_elite)]
'yelping_since': (date, formatted)
'compliments': (compliment_type): (num_compliments_of_this_type)
'fans': (num_fans)
'push': (push information)

Fig.1 Diagram of the Dataset

Database System

The dataset used for the project is located online in DynamoDB of Amazon Web Servise (AWS). Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability. NoSQL database, also called Not Only SQL, is an approach to data management and database design that's useful for very large sets of distributed data. DynamoDB automatically spreads the data and traffic for tables over a sufficient number of servers to handle the throughput and storage requirements, while maintaining consistent and fast performance. All of the data is stored on solid state disks (SSDs) and automatically replicated across multiple Availability Zones in an AWS region, providing built-in high availability and data durability.

Programing Language

There are two major language I used for the project.
Python is used for creating table and uploading the dataset to the database system.
Javascript is used for query and update the data from DynamoDB online. Also it is used for build a WebApp based on Node.js with Express Frame.

Project Frame

System Frame

The system frame is shown as Fig.2 below,

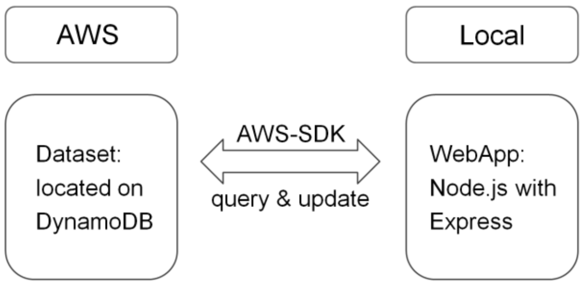


Fig.2 system frame

As shown in Fig.2, we have a WebApp of the recommendation system in local machine, and dataset is located in DynamoDB of AWS. The local machine access to the database via AWS-SDK, an API provided by AWS.
When the system is working, the WebApp query specific data which satisfies some submitted parameters from DynamoDB on AWS, and then use the downloaded data for calculation. Finally, we will get the recommendation. After that, in "business based recommendation system", the WebApp will send update query to DynamoDB to modify some data. In detail, add the information of the business to the 'push' attribute of the recommended users. Next time, when these users log in the system, they will see the push information.

Processing Frame

Then let's talk about the processing frames of "user based recommendation system" and "business based recommendation system".
For the "user based recommendation system", the processing frame is shown in Fig.3 as below,

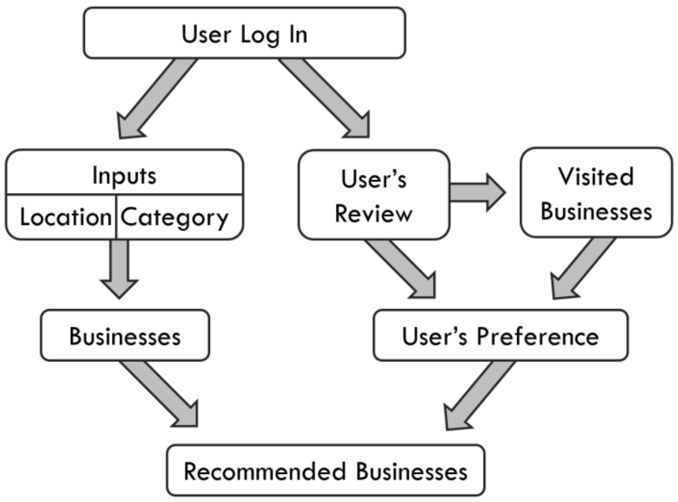


Fig.3 processing frame for "user based recommendation system"

In the case of "user based recommendation system", after a user log in the system, he/she will be asked for some inputs about the location and category of the required business. Then the system will query for businesses based on these requirement. At the same time, the system will statistic the user's preference based on his/her reviews and visited businesses in the reviews. The main features taken into consideration is categories of visited businesses, how many time the user went to these kind of businesses, and the difference between stars the user give to the visited business and the business' average stars. Combining the result of query and user's preference, we can rerank these businesses based on the average stars and get the recommendation result and then show the recommendation to the user.
For the "user based recommendation system", the processing frame is shown in Fig.4 as below,

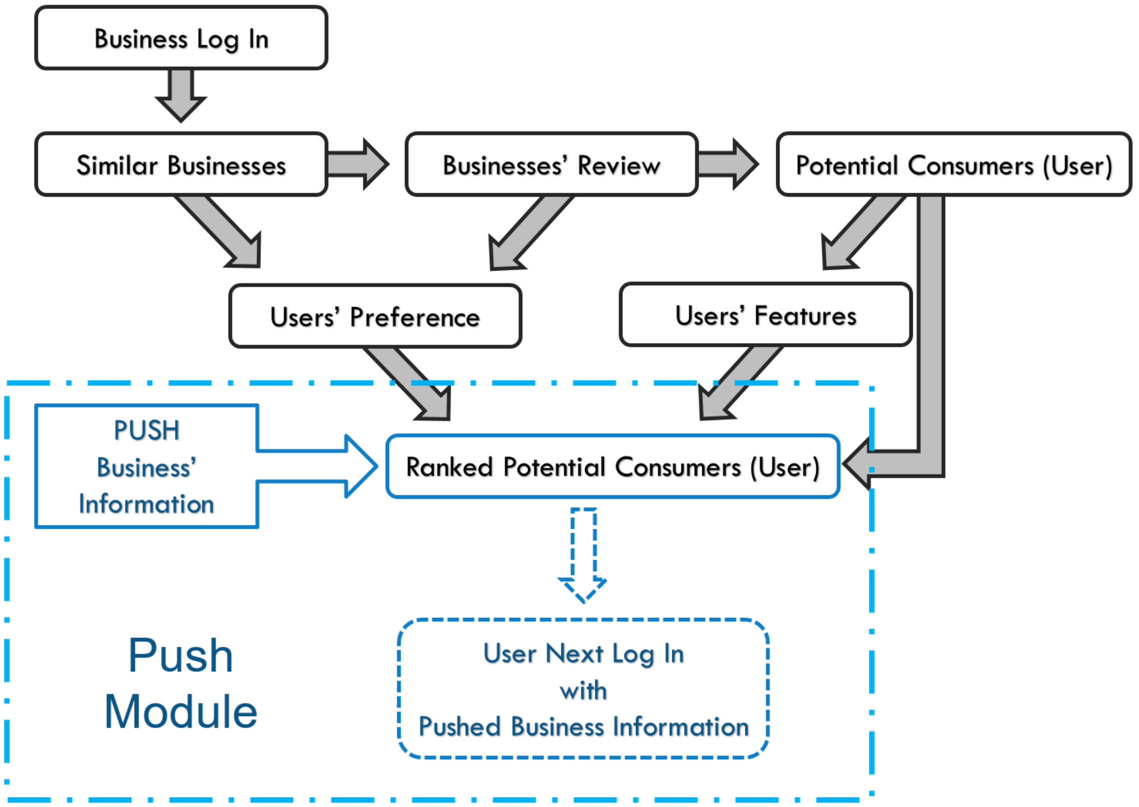

Business Log In
Similar Businesses
Businesses' Review
Potential Consumers (User)
Users' Preference
Users' Features
PUSH Business' Information
Ranked Potential Consumers (User)
Push Module
User Next Log In with Pushed Business Information

Fig.4 processing frame for "business based recommendation system"

In the case of "user based recommendation system", after business log into the system and start the recommendation processing, the system will search for the similar businesses with the same location and categories. And then via the reviews of these similar businesses, we can find the potential consumers. Based on the users' preference and some features, including friends number, elite years and average stars the user give, we have a reranked potential consumers list.
In this push module, the WebApp will send update query to DynamoDB to modify user data. In detail, add the information of the business to the 'push' attribute of the recommended users. Therefore, next time, when these users log in the system, they will see the push information.

DATABASE PREPARATION

After downloaded the data from the official website of Yelp, we have to modify the format in the data files to make them fit for the required format of json. Here, the attached file “python.py” is used. Especially, in order to achieve the push module in the recommendation system, an extra attribute ‘push’ is added into the dataset of User table. So “python2.py” is used to modify the format of data file for User.

Table Creation

Python language is used to create table in DynamoDB. We have three tables in the project. Here is the sample code (User table creation) for this step.

```
from __future__ import print_function # Python 2/3 compatibility
import boto3

dynamodb = boto3.resource('dynamodb', region_name='us-west-2',
endpoint_url="https://dynamodb.us-west-2.amazonaws.com")

table = dynamodb.create_table(
    TableName='UserInfo',
    KeySchema=[
        {
            'AttributeName': 'user_id',
            'KeyType': 'HASH'  #Partition key
        }
    ],
    AttributeDefinitions=[
        {
            'AttributeName': 'user_id',
            'AttributeType': 'S'
        }
    ],
    ProvisionedThroughput={
        'ReadCapacityUnits': 5,
        'WriteCapacityUnits': 200
    }
)
print("Table status:", table.table_status)
```

Here are the key words information of our three tables:

• Business Table:	Partition key: ‘type’, Sort key:’business_id’.
• User Table:	Partition key: ‘user_id’.
• Review Table:	Partition key: ‘business_id’, Sort key:’time’.

Actually, it is a mistake that the partition key for Business table is set as ‘type’. It leads to efficiency losing. I will fix the problem using secondary index creation later.

The read/write capacity of the table is the major limitation of the speed you access to the dataset from local machine. It is supposed to be [number] items per second. For example, if the capacity is set to 10, the speed to access the dataset is about 10 items per second. While if the number is larger than about 40, the speed cannot reach it anymore. Maybe the speed is influenced by the

internet. But even if the volume of each item is 1KB, to support capacity of 100, just 100KB/s is needed.
So I am not so sure about what the capacity refers to. But generally speaking, the more the faster. But it doesn’t matter now. It can be revised later on the AWS website.

Data Upload

Python language is used to upload the data to DynamoDB. We have three tables in the project. Here is the sample code (User data upload) for this step.

```
from __future__ import print_function # Python 2/3 compatibility
import boto3
import json
import decimal

dynamodb = boto3.resource('dynamodb', region_name='us-west-2',
endpoint_url="https://dynamodb.us-west-2.amazonaws.com")

table = dynamodb.Table('UserInfo')
with open("user.json") as json_file:
    users = json.load(json_file, parse_float = decimal.Decimal)
    for user in users:
      typename = user['type']
      push = user['push']
        user_id = user['user_id']
        name = user['name']
        review_count = user['review_count']
        average_stars = user['average_stars']
        votes = user['votes']
        friends = user['friends']
        elite = user['elite']
        yelping_since = user['yelping_since']
        compliments = user['compliments']
        fans = user['fans']

        print("Adding user:", user_id)

      if typename=='':
            typename = 'user'
      if compliments=='':
            compliments = 'unknown'

        table.put_item(
           Item={
             'type': typename,
             'push': push,
               'user_id': user_id,
               'name': name,
               'review_count': review_count,
             'average_stars': average_stars,
             'votes': votes,
             'friends': friends,
             'elite': elite,
             'yelping_since': yelping_since,
             'compliments': compliments,
             'fans': fans,
            }
        )
```
In this step, we don’t care about the speed. In total, I spent about 120 hours for uploading.

Secondary Index Creation

For efficient access to data in a table, Amazon DynamoDB creates and maintains indexes for the primary key attributes. This allows applications to quickly retrieve data by specifying primary key values. However, many applications might benefit from having one or more secondary (or alternate) keys available, to allow efficient access to data with attributes other than the primary key.

A secondary index is a data structure that contains a subset of attributes from a table, along with an alternate key to support Query operations. With a secondary index, data can be retrieved using the alternate key defined by the secondary index to increase efficiency.

To increase the efficiency of the WebApp and fix the former mistake of partition key setting in Business table. I created three secondary index.

- Business Table: business_id-index, city-stars-index
- Review Table: user_id-stars-index

They greatly improve the performance of the recommendation system WebApp.

WEB APPLICATION PROGRAM

For the WebApp part, I use javascript on Node.js with Express.
The major challenge here is to deal with asynchronous I/O event-driven architecture capable of Node.js. For example, if we run the following code in Node.js,

```
var a = 0;
a = function();
return a;
```

The return value will always be 0. Because the return command won't wait for the accomplishment of the former function and return the current value of variable a.
In order to solve the problem, I use 'express-promise' and 'promise-resolver' module in the project. Also, I user 'client-session' module to save temperature global variables.

User Based Recommendation System

I have app.js, index.js, DataQuery.js, ScoreTable.js, UserPush.js, login.ejs, user_home_before.ejs, user_home_after.ejs for this part of recommendation system. Here is the WebApp processing shown in Fig.5,

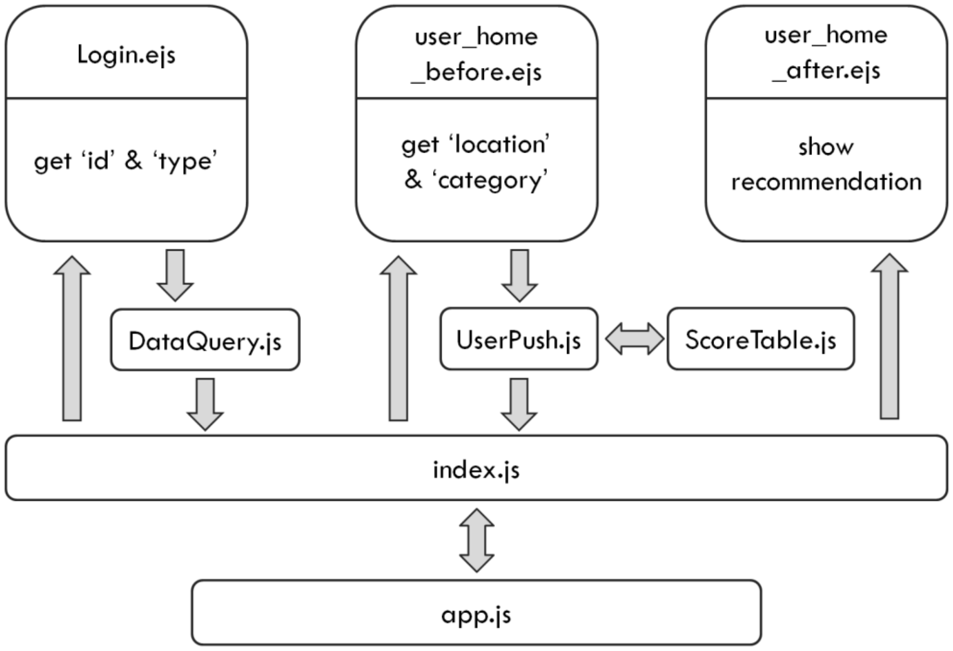


Fig.5 WebApp processing for user based recommendation system

Functions of these files are listed below,

- app.js: define the routers and handles used in the app.
- index.js: use the routers and handles to control the processing of the app
- DataQuery.js: function - query the data from DynamoDB according to the given parameters

- ScoreTable,js: function - calculate the preference table of given user based on his/her reviews and visited business.
- UserPush.js: function - get the recommendation result given user id, location and category.
- login.ejs: view - login page, get the inputs (id and user/business)
- user_home_before.ejs: view - home page before start the recommendation process, get the requirement from user like location, category (optional).
- user_home_after.ejs: view - home page after get the recommendation result, show the recommended business to user.

Here I have to explain how to get the user preference based on the dataset. In this part, it is the score table which is scores of different categories. The scoring process is shown in Fig.6,

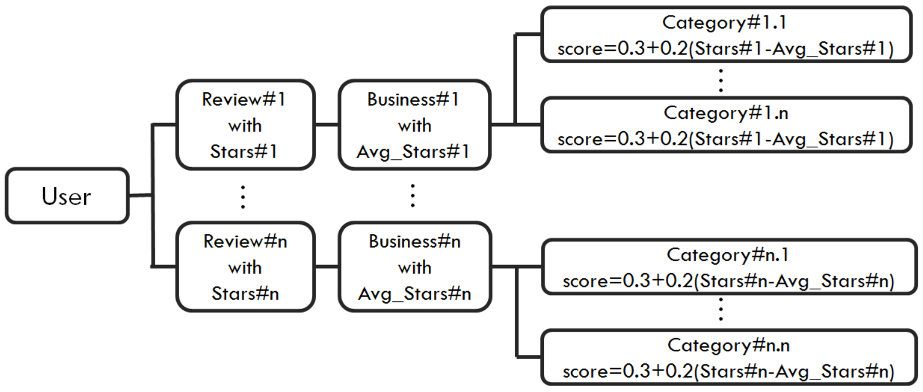

User
Review#1 with Stars#1
Review#n with Stars#n
Business#1 with Avg_Stars#1
Business#n with Avg_Stars#n
Category#1.1 score=0.3+0.2(Stars#1-Avg_Stars#1)
Category#1.n score=0.3+0.2(Stars#1-Avg_Stars#1)
Category#n.1 score=0.3+0.2(Stars#n-Avg_Stars#n)
Category#n.n score=0.3+0.2(Stars#n-Avg_Stars#n)

Fig.6 Categories scoring process

We suppose that 1) if the user ever visited this kind of business, he/she should prefer to it; 2) If a user give more stars to the business, we can conclude that he/she prefer to it. So as shown in Fig.6 , a score of a category consist of two parts. 0.3 is the basic score, and the other part is the difference between stars the user rated the business and the business' average stars. If Business#1 and Business#2 have a same category, the score of the category is the sum. Finally, the score of a recommended business is the sum of average stars and scores of all the attributes.

Business Based Recommendation System

I have app.js, index.js, DataQuery.js, ConsumerList.js, BusinessPush.js, Push.js, PushView.js, login.ejs, business_home_before.ejs, business_home_after.ejs, push.ejs, user_home_before.ejs, push_view.ejs to handle this part to recommendation system. Their functions are listed below,

- app.js: define the routers and handles used in the app.
- index.js: use the routers and handles to control the processing of the app
- DataQuery.js: function - query the data from DynamoDB according to the given parameters

- ConsumerList,js: function - get the potential consumers list for given business. They are users ever consumed in businesses similar to given business.
- BusinessPush.js: function - get the recommendation result given business id.
- Push.js: function - update the data in DynamoDB. Add the business id to the top 200 users in reranked potential consumers list.
- PushView.js: function - read the ‘push’ attribute of given user and show the return the information of pushed business.
- login.ejs: view - login page, get the inputs (id and user/business)
- business_home_before.ejs: view - home page before start the recommendation process.
- business_home_after.ejs: view - home page after get the recommendation result, show the top 200 users in reranked potential consumers list and include a ‘push’ button.
- push.ejs: view - show this page when push process is success.
- user_home_before.ejs: view - home page before start the recommendation process, include a notice of push number.
- push_view.ejs: view – push view page to show all the pushed businesses to user.

Here is the processing shown in Fig.7,

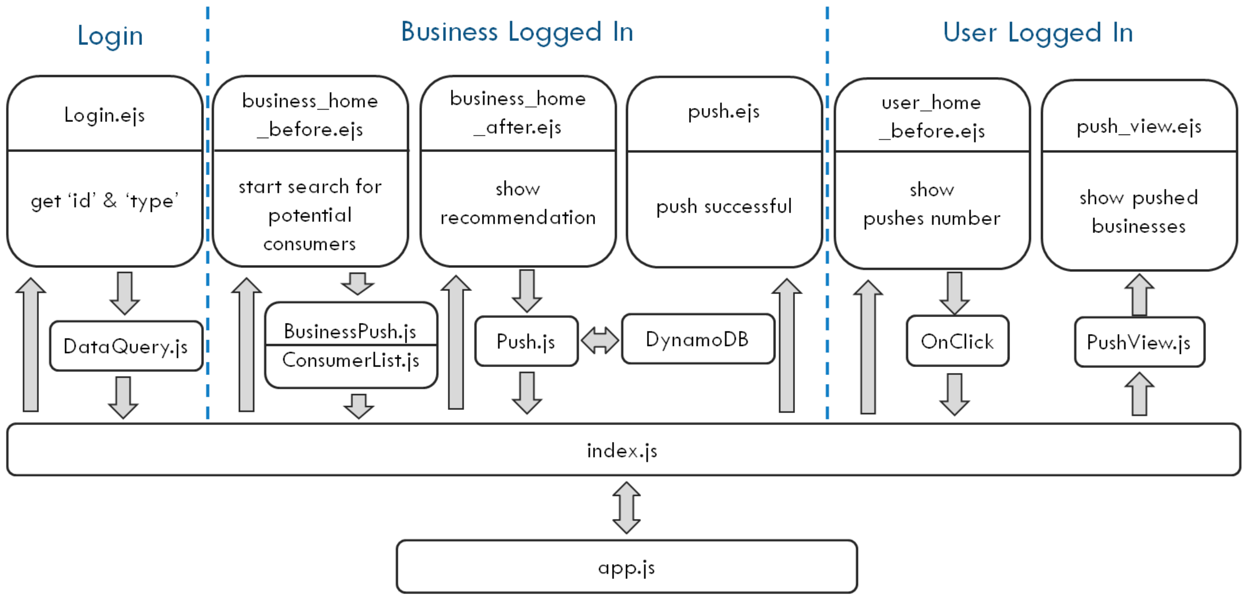


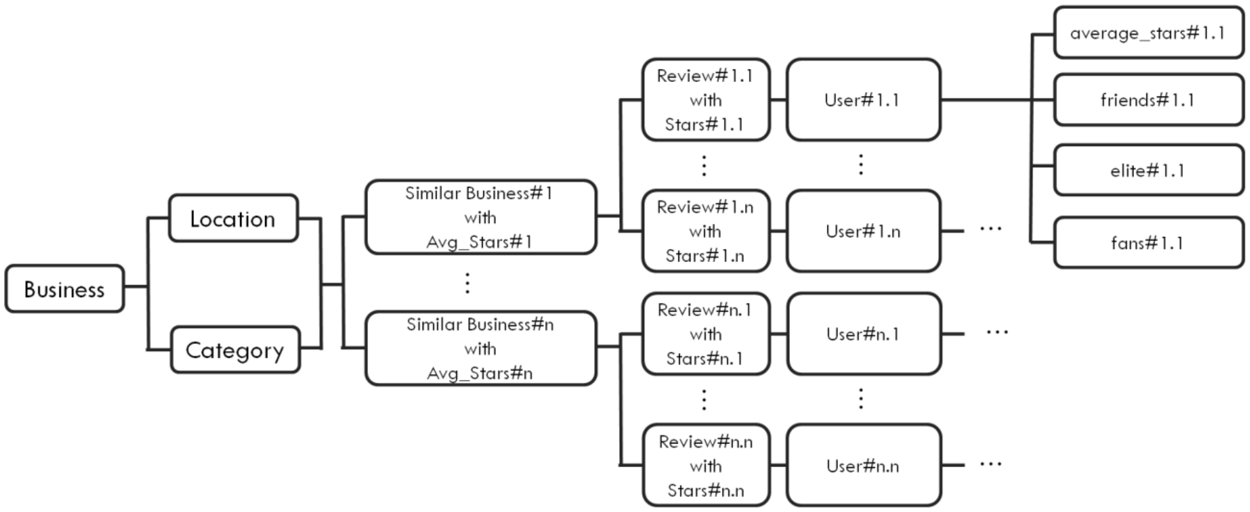
Fig.7 WebApp processing for business based recommendation system

In this part, BusinessPush.js and ConsumerList.js is to get the ranked potential consumers. The processing is shown in Fig.8.

We suppose that 1) if the user ever visited this kind of business, he/she should prefer to it; 2) if a user give more stars to the business, we can conclude that he/she prefer to it; 3) if a user with higher average stars, he/she prefer to give a higher stars to business; 4) a user who has more friends is a better potential consumer; 5) a user who has more elite years is a better potential consumer; 6) a user who has more fans is a better potential consumer. So as shown in Fig.8, the User#1.1 … User#n.n is the potential consumers, and the score is based on the situations we talked about before and can be separated into two parts: preference part and features part. For example,

Score of User#1.1
= (0.3+stars#1.1-Avg_Stars#1)
+ (average_stars#1.1+0.02 friends[number]+0.5*elite[number]+0.3*fans)*

If User#1.1 and User#2.1 is the same user, the score of the user is sum[preference part] + features part.



Business
Location
Category
Similar Business#1 with Avg_Stars#1
Similar Business#n with Avg_Stars#n
Review#1.1 with Stars#1.1
Review#1.n with Stars#1.n
Review#n.1 with Stars#n.1
Review#n.n with Stars#n.n
User#1.1
User#1.n
User#n.1
User#n.n
average_stars#1.1
friends#1.1
elite#1.1
fans#1.1

Fig.8 Rank the potential consumers

Sample Code

BusinessPush.js

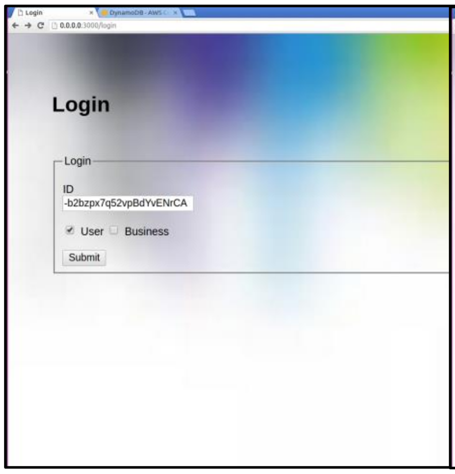
```
var express = require('express');
var p = require('promise-resolver');
var Q = require('./DataQuery.js');
var C = require('./ConsumerList.js');

var AWS = require("aws-sdk");
AWS.config.update({
    region: "us-west-2",
    endpoint: "https://dynamodb.us-west-2.amazonaws.com"
});

var sortBy = function (filed, rev, primer) {
    rev = (rev) ? -1 : 1;
    return function (a, b) {
        a = a[filed];
        b = b[filed];
        if (typeof (primer) != 'undefined') {
            a = primer(a);
            b = primer(b);
        }
        if (a < b) { return rev * -1; }
        if (a > b) { return rev * 1; }
        return 1;
    }
};

exports.BusinessPush = function(id){
    var def = p.defer();
    var ConsumerList = C.ConsumerList(id);
    ConsumerList.then(function(value){
        value = value.sort();
        var number = 0;
        for (var i = 1; i < value.length; i++){
            if (value[i][0]!=value[number][0]){
                number+=1;
                value[number]=value[i];
            }else{
                value[number][1]+=value[i][1];
            }
        }
        number+=1;
        var PushList = new Array();
        var count = 0;
        for (var i = 0; i < number; i++){
            var params ={
                TableName : "UserInfo",
                KeyConditionExpression: "#id = :iiii",
                ExpressionAttributeNames:{
                    "#id": "user_id"
                },
                ExpressionAttributeValues: {
                    ":iiii": value[i][0]
                }
            };
            var docClient = new AWS.DynamoDB.DocumentClient();
            docClient.query(params, function(err, data) {
                if (err) {
                    console.error("Unable to query. Error:",
JSON.stringify(err, null, 2));
                } else {
                    data.Items.forEach(function (item) {
                        var score = value[i][1];
                        score += 0.02*item.friends.length +
0.5*item.elite.length + 0.3*item.fans + item.average_stars;
                        item.score = score;
                        PushList[PushList.length] = item;
                        count+=1;
                    });
                }
                if (count == number) {
                    PushList.sort(sortBy('score', true, parseFloat));
                    def.resolve(PushList);
                }
            });
        }
    });
    return def.promise;
}
```

PERFORMANCE

Run the WebApp, and the server is listening on http://0.0.0.0:3000.

User Based Recommendation System

Now let’s check the performance of the user based recommendation system. The capture of the running system is shown below in Fig.9.



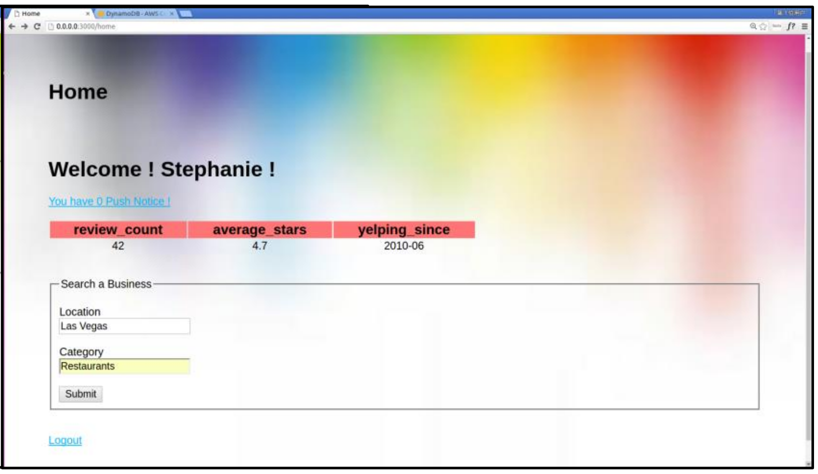
Login

Login

ID
-b2bzpx7q52vpBdYvENrCA

☑ User ☐ Business

Submit



Home

Welcome ! Stephanie !

You have 0 Push Notice !

review_count	average_stars	yelping_since
42	4.7	2010-06

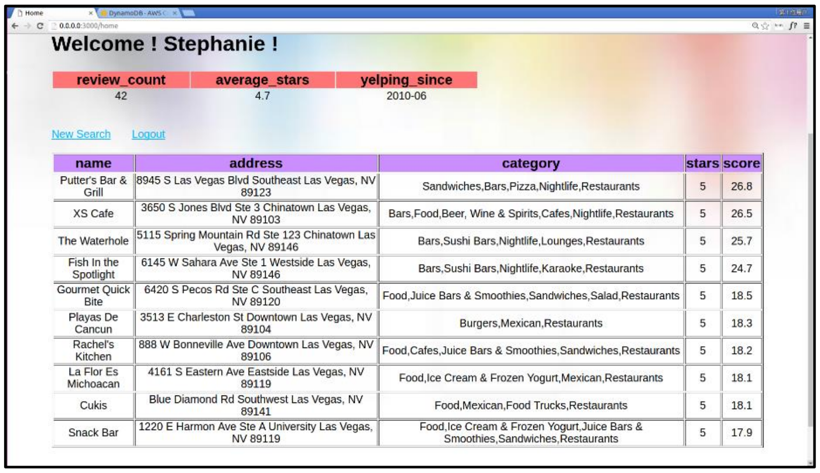
Search a Business

Location
Las Vegas

Category
Restaurants

Submit

Logout



Welcome ! Stephanie !

review_count	average_stars	yelping_since
42	4.7	2010-06

New Search Logout

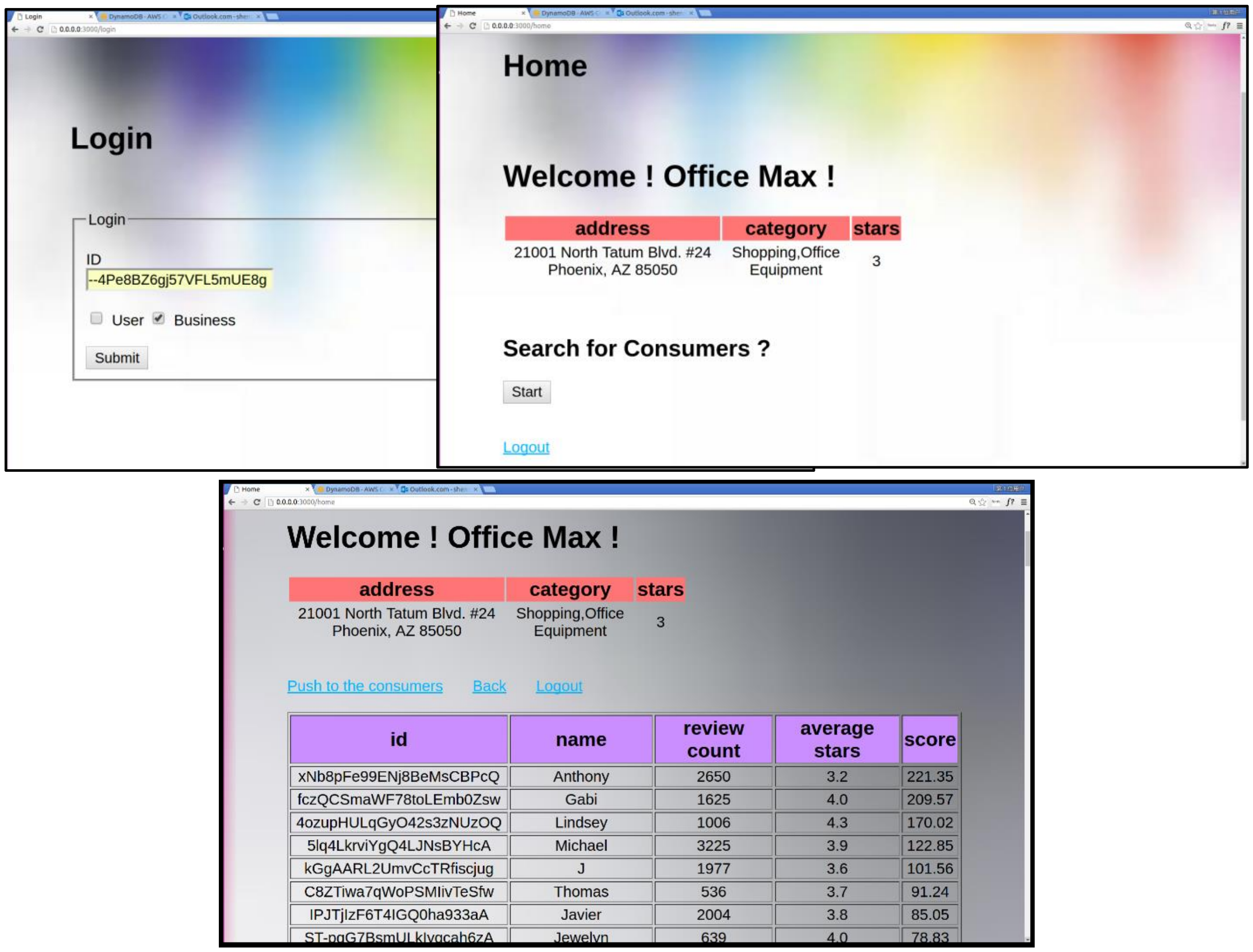
name	address	category	stars	score
Putter's Bar & Grill	8945 S Las Vegas Blvd Southeast Las Vegas, NV 89123	Sandwiches,Bars,Pizza,Nightlife,Restaurants	5	26.8
XS Cafe	3650 S Jones Blvd Ste 3 Chinatown Las Vegas, NV 89103	Bars,Food,Beer, Wine & Spirits,Cafes,Nightlife,Restaurants	5	26.5
The Waterhole	5115 Spring Mountain Rd Ste 123 Chinatown Las Vegas, NV 89146	Bars,Sushi Bars,Nightlife,Lounges,Restaurants	5	25.7
Fish In the Spotlight	6145 W Sahara Ave Ste 1 Westside Las Vegas, NV 89146	Bars,Sushi Bars,Nightlife,Karaoke,Restaurants	5	24.7
Gourmet Quick Bite	6420 S Pecos Rd Ste C Southeast Las Vegas, NV 89120	Food,Juice Bars & Smoothies,Sandwiches,Salad,Restaurants	5	18.5
Playas De Cancun	3513 E Charleston St Downtown Las Vegas, NV 89104	Burgers,Mexican,Restaurants	5	18.3
Rachel's Kitchen	888 W Bonneville Ave Downtown Las Vegas, NV 89106	Food,Cafes,Juice Bars & Smoothies,Sandwiches,Restaurants	5	18.2
La Flor Es Michoacan	4161 S Eastern Ave Eastside Las Vegas, NV 89119	Food,Ice Cream & Frozen Yogurt,Mexican,Restaurants	5	18.1
Cukis	Blue Diamond Rd Southwest Las Vegas, NV 89141	Food,Mexican,Food Trucks,Restaurants	5	18.1
Snack Bar	1220 E Harmon Ave Ste A University Las Vegas, NV 89119	Food,Ice Cream & Frozen Yogurt,Juice Bars & Smoothies,Sandwiches,Restaurants	5	17.9

Fig.9 Performance of user based recommendation system

In Fig.9, they are login page, home page, search result page of user “Stephanie”, and the business location is limited to Las Vegas. The input of category is optional. And then we can get the result of the recommendation. Stars is the average stars the business get, and the score of the business is modified based on the stars according to the preference of the user. Only the top 10 results will be shown on the website, but actually the length of the ranked businesses list ranges from several to thousands. It depends on the number of businesses in the location and satisfy the category.
The result came out within a second. I think the speed is good for a recommendation system.

Business Based Recommendation System

The performance of the business based recommendation system is shown below in Fig.10.



Login

Login

ID

--4Pe8BZ6gj57VFL5mUE8g

User [✓ Business]

Submit

Home

Welcome ! Office Max !

address	category	stars
21001 North Tatum Blvd. #24 Phoenix, AZ 85050	Shopping,Office Equipment	3

Search for Consumers ?

Start

Logout

Welcome ! Office Max !

address	category	stars
21001 North Tatum Blvd. #24 Phoenix, AZ 85050	Shopping,Office Equipment	3

Push to the consumers Back Logout

id	name	review count	average stars	score
xNb8pFe99ENj8BeMsCBPcQ	Anthony	2650	3.2	221.35
fczQCSmaWF78toLEmb0Zsw	Gabi	1625	4.0	209.57
4ozupHULqGyO42s3zNUzOQ	Lindsey	1006	4.3	170.02
5lq4LkrviYgQ4LJNsBYHcA	Michael	3225	3.9	122.85
kGgAARL2UmvCcTRfiscjug	J	1977	3.6	101.56
C8ZTiwa7qWoPSMIivTeSfw	Thomas	536	3.7	91.24
IPJTjIzF6T4IGQ0ha933aA	Javier	2004	3.8	85.05
ST-pqG7BsmULklygcah6zA	Jewelyn	639	4.0	78.83

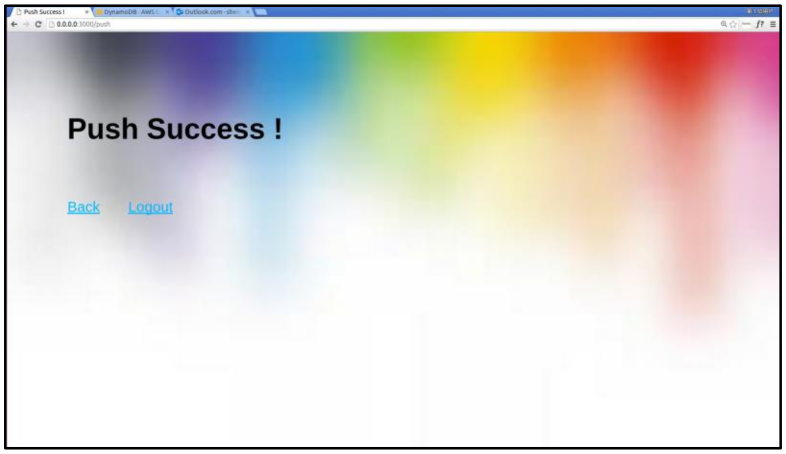
Fig.10 Performance of business based recommendation system

In Fig.10, they are login page, home page, potential consumer search result page of business “. Average Stars is the stars the user prefer to give to business, and the score of the users is modified based on the average stars according to the preference of the user and some features of users. Only the top 200 results will be shown on the website, but actually the length of the ranked potential user list is thousands as usual.

The result came out within several second. I think the speed is good for a recommendation system. The business based recommendation system need much more calculation than the user based one. Because we take all the businesses located in the city into consideration. And usually, a business has more reviews than a user. Which make things obvious is the required capacity of the tables on DynamoDB. For user based recommendation system, about 20 capacity is enough. While in the business based system, we have to set the capacity to at least 500 to reach a satisfying speed.

And then if the business click on “Push to the consumers”, push page will be shown if the push command is successful, as shown in Fig.11.

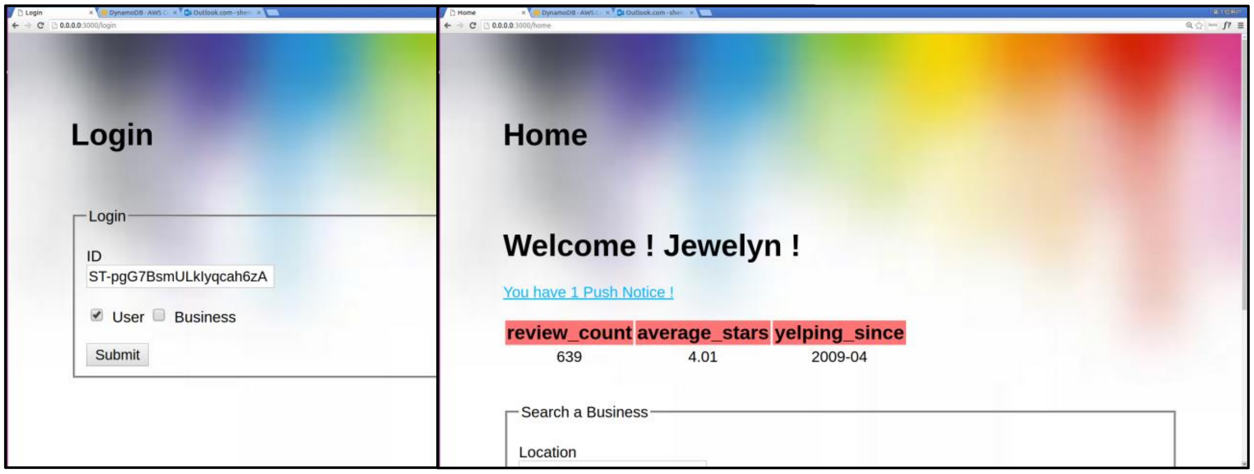
So next time if the top potential consumers log in the project, they will see a push notice on his/her home page. And the pushed business details will be shown on click on the notice. The process is shown in Fig.12. Jewelyn has a push notice right under the “welcome”. After click on the notice, the detail of the pushed business will be shown in push view page.



Push Success !

Back Logout

Fig.11 Push page



Login

Login

ID

ST-pgG7BsmULklyqcah6zA

User Business

Submit

Home

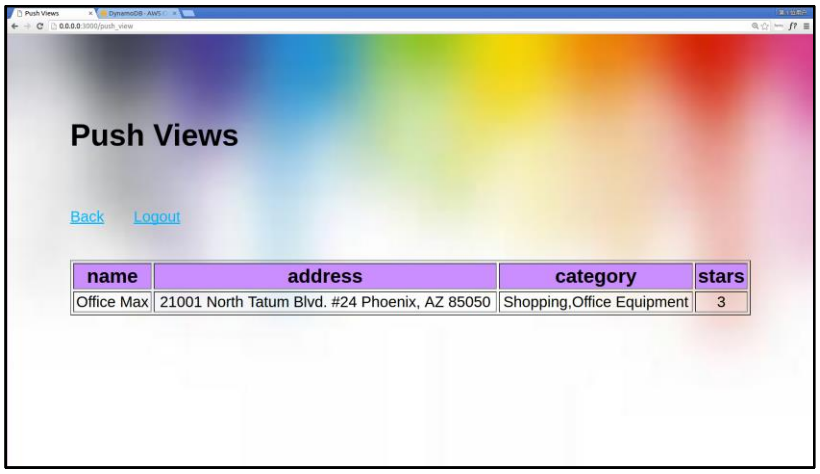
Welcome ! Jewelyn !

You have 1 Push Notice !

review_count	average_stars	yelping_since
639	4.01	2009-04

Search a Business

Location



Push Views

Back Logout

name	address	category	stars
Office Max	21001 North Tatum Blvd. #24 Phoenix, AZ 85050	Shopping,Office Equipment	3

Fig.12 Push notice on user’s page

CONCLUSION & FUTURE

In the project, we achieved a recommendation system based on the dataset from Yelp, and built a WebApp for it. From the very beginning, we created the table, loaded data, design the scoring algorithm for users and businesses, and finally show the results in a website.
The performance of the project is good, the recommendation result is modified based on the specific requirement and the user/business. And we just need at most several seconds to get the results of the recommendation. It is good for millions of calculation.
In the user based recommendation system, without any review, we cannot get the preference of the user according to the dataset, and can just recommend the businesses with the highest stars. Even worse, in the business based recommendation system, without any review, we cannot find the potential consumer for business.
But it make sense. If there is a new user without any details, it can hardly to make a recommendation fit for the specific situation of the user. That is why nowadays most of the website will ask for tags from you when you sign up at the first. And almost all the website will save the watch history and search history for you. It is not only to make you convenience to get you're history, but a record of your preference.
While in the dataset we used in the project, review is the only relationship between users and businesses. Of cause, the more relationships the better. As a conclusion, making use of the big data is not the only problem of big data. How to get the data is another key word.

FILE DIRECTORY

python.py – used to format the downloaded business and review dataset file
python2.py – used to format the downloaded user dataset file

BusinessCreateTable.py – create business table
BusinessLoadData.py – upload the business data
UserCreateTable.py – create user table
UserLoadData.py – upload the user data
ReviewCreateTable.py – create review table
ReviewLoadData.py – upload the review data

yelp-problem folder – WebApp