2020 ECE656 project

Database System

He Bing(20848700)/Li Junyu(20837856)

Instructor: Paul Ward

ECE
University of Waterloo
March 2020

1 Introduction

The purpose of the project is to design a simple social network. And the basic functions of the social network is to render people build relationship with other people. In social network, people can follow other people's post, and see what's going on recently. What is more, they can also follow the topics that they are interested in.

Here are the sample functions of the project:
(a) A person should be able to initial a post on a topic
(b) A person should be able to follow/join a group with another person
(c) A person should be able to follow a topic
(d) A person should be able to determine what posts have been added by people and/or topics that they are following since they last read from those people/topics
(e) A person should be able to respond to a post with thumbs up/down and/or a response post.

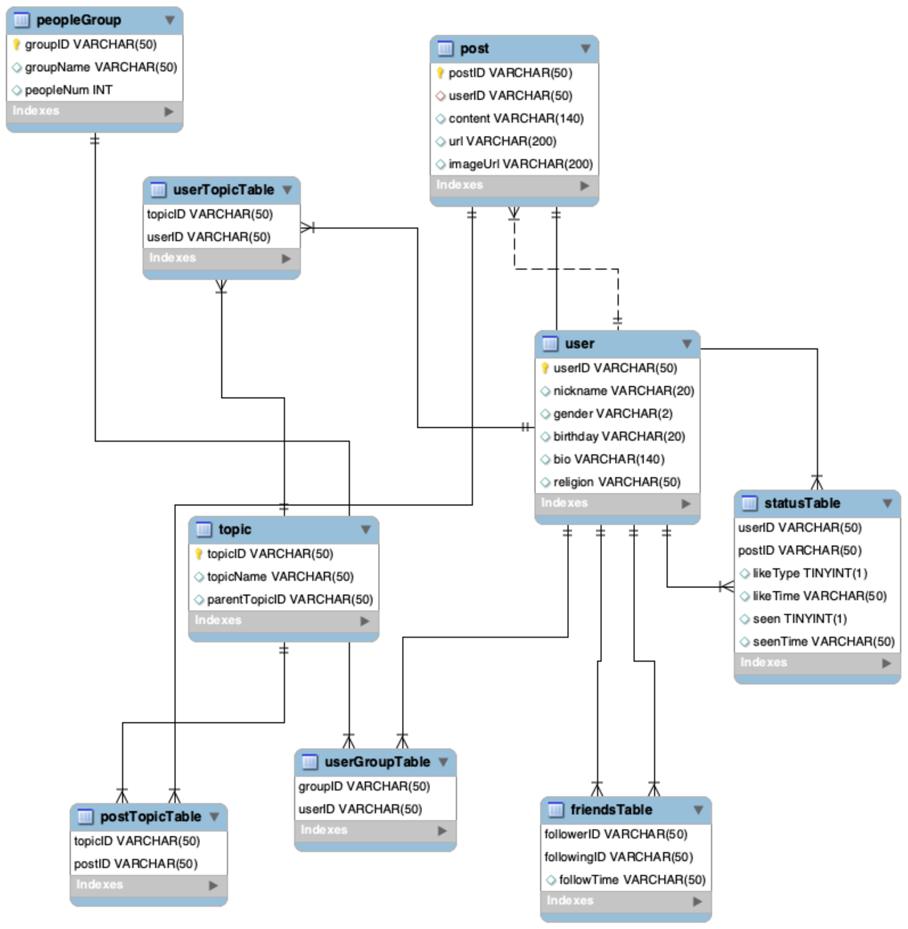
The data is from yelp in kaggle. For yelp which is a website that published crowd source reviews to help users and business owners, it is a local business directory service and review site with social networking features. That's why we choose it to support our simple social network. It allows users to give reasonable ratings and review businesses. The review is usually short text consisting of few lines with about hundred words.

We should design a database that can satisfy all the demands of the social network. Then make a server and also a client with CLI that can fetch the data from server.

2 Database design

2.1 Graph

Here is the E-R graph of our database schema.



2.2 explaining

The goal of this project is to build a small social network, so it is not necessary to transform all the data from yelp.json to the database.
So we decided to use business as topic and if a user has a review to the business, then we assume he or she has followed this topic.
For all the many-to-many relations, we should create a middle table which can present the relation from one entity to other entity.
In addition, we should make a table to record which post has been liked or seen by a specific user. So we need a status table of all the post and user.

What's more, based on the BCNF principle and the relationship in real world, we have the E-R graph as above.

3 Server and client design

3.1 server

Use the library flask in python 3.
base url=127.0.0.1:(port)
All the api will be listed below.

3.1.1

url: /
type(get or post):N/A
function description:give some description of this project.
error:N/A
sql:N/A

3.1.2

url: /user?user_id=%
type(get or post):get
function description: get the information of a user
error: Error: No id field provided. Please specify an id.
sql:

```
select * from user where userID=%
```

3.1.3

url:/friends_post?user_id=%
type(get or post):get
function description:get the unseen posts from all the friends of the user
error: Error: Cannot find the user id.
sql:

```
select * from (select * from post where userID in (select followingID
from friendsTable where followerID = ? and postID not in (select postID
from statusTable where userID= ? and seen=true)) as A inner join user
on A.userID=user.userID;
```

3.1.4

url:/topic_post?user_id=%
type(get or post):get
function description:get all the unseen following topic's posts
error: Error: Cannot find the user id.
sql:

```
select * from post where postID in (select postID from postTopicTable
where topicID in (select topicID from userTopicTable where userID =
?)) and postID not in (select postID from statusTable where userID=?
and seen=true);
```

3.1.5

url:/group_post?user_id=%
type(get or post):get
function description:get the the post of groups that the user follows
error: Error: Cannot find the user id.
sql:

```
select * from post where userID in (select userID from userGroupTable
where groupID in (select groupID from userGroupTable where userID =
?)) and postID not in (select postID from statusTable where userID=?
and seen=true);
```

3.1.6

url:/all_topic?topic_id=%
type(get or post):get
function description:get all the posts in this topic
error: Error: Cannot find the topic id.
sql:

```
select nickname,post_time,content from (select * from post where postID
in (select postID from postTopicTable where topicID=?)) as A inner
join user on A.userID=user.userID;
```

3.1.7

url:/all_person?user_id=%
type(get or post):get
function description:get all the posts of an user
error: Error: Cannot find the user id.
sql:

```
select * from post where userID=?
```

3.1.8

url:/friend_list?user_id=%
type(get or post):get
function description:get the friend list of an user
error: Error: Cannot find the user id.
sql:

```
select userID,nickname from user where userID in (select followingID
from friendsTable where followerID=?
```

3.1.9

url:/all_group
type(get or post):get
function description:get all the group's information
error:N/A
sql:

```
select * from peopleGroup;
```

3.1.10

url:/all_like?user_id=%
type(get or post):
function description:get all the post that the user liked
error: Error: Cannot find the user id.
sql:

```
select * from post where postID in (select postID from statusTable
where userID=? and likeType=True)
```

3.1.11

url:/all_unlike?user_id=%
type(get or post):
function description:get all the post that the user disliked
error: Error: Cannot find the user id.
sql:

```
select * from post where postID in (select postID from statusTable
where userID=? and likeType=False)
```

3.1.12

url:/send_post
type(get or post):post
function description:send one post
error:N/A
sql:N/A

3.1.13

url:/follow_person
type(get or post):post
function description:follow a person
error: Error: you have followed this person already.
Error: user id does not exist
sql:N/A

3.1.14

url:/follow_topic
type(get or post):post
function description:follow a topic
error: Error: you have followed this topic already
Error: topic id does not exist.
sql:N/A

3.1.15

url:/create_group
type(get or post):post
function description:create a group
error:N/A
sql:N/A

3.1.16

url:/follow_group
type(get or post):post
function description:follow a group
error: Error: you have followed this group already.
Error: group id does not exist.
sql:N/A

3.1.17

url:/create_user
type(get or post):post
function description:create a new user
error:N/A
sql:N/A

3.1.18

url:/like
type(get or post):post
function description:like or dislike a post
error:N/A
sql:N/A

3.1.19

url:/create_topic
type(get or post):post
function description:create a new topic
error:N/A
sql:N/A

3.2 client

Here is the function list:
a.send a post
b.follow a person
c.like/dislike a post
d.follow a topic
e.follow a group
f.create a topic
g.create a group
h.see new posts of people
i.see new posts of topics
j.see new posts of groups
k.see my friend list
l.see all the posts of one person
m.see all the posts of one topic
n.see all the groups' information
o.see all my likes
p.see all my unlikes

4 Data analyse

For business:

```
business_data = [json.loads(line) for line in open('../656/business/business_a',
'r')]

scores={}
for d in business_data:
    if d['categories']:
        for c in d['categories'].split(','):
            c=c.replace(' ','')
            if c in scores.keys():
                scores[c]=scores[c]+d['review_count']
            else:
                scores[c]=d['review_count']
a = sorted(scores.items(), key=lambda x: x[1])[::-1]
a=pd.DataFrame(a)
a=a[:10]
plt.bar(range(10),a[1],color='pink')
plt.xticks(range(10),a[0],rotation=30)
plt.title("reviews top 10 categories")
fig = plt.gcf()
fig.set_size_inches(18.5, 5)
plt.show()
```

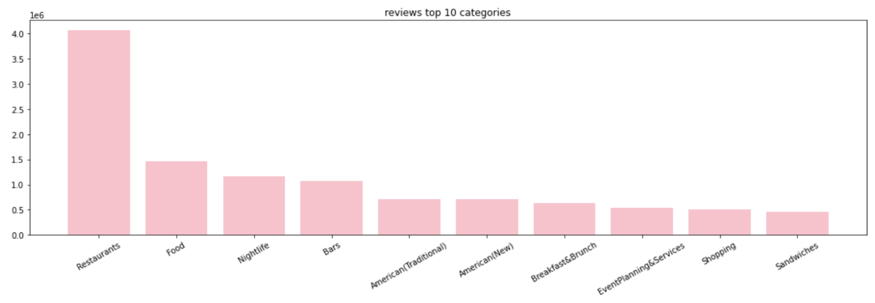


Figure 1: Bar chart of reviews top 10 categories

The bar chart shows top 10 most popular categories of store.
According to this chart, the percentage of business for Restaurants is the highest, which is the most popular one. By contrast, the percentage of business for Eventplannning&Services, Shopping and Sandwiches are much lower than the percentage of business for Restaurants.
Overall, people will pay more attention to Restaurants, Food, Nightlife and Bars, that's why these categories are popular.

```
scores={}
reviews={}
for d in business_data:
    if d['state'] in scores.keys():
        scores[d['state']]=(scores[d['state']]*reviews[d['state']]
        +d['stars']*d['review_count'])/(reviews[d['state']]+d['review_count'])
        reviews[d['state']]=reviews[d['state']]+d['review_count']
    else:
        scores[d['state']]=d['stars']
        reviews[d['state']]=d['review_count']
for k,v in reviews.items():
    if v<100:
        del scores[k]
a = sorted(scores.items(), key=lambda x: x[1])[::-1]
a=pd.DataFrame(a)
plt.bar(range(len(a)),a[1],color='tomato')
plt.xticks(range(len(a)),a[0],rotation=30)
plt.title("average stars in states")
fig = plt.gcf()
fig.set_size_inches(18.5, 7)
plt.show()
```

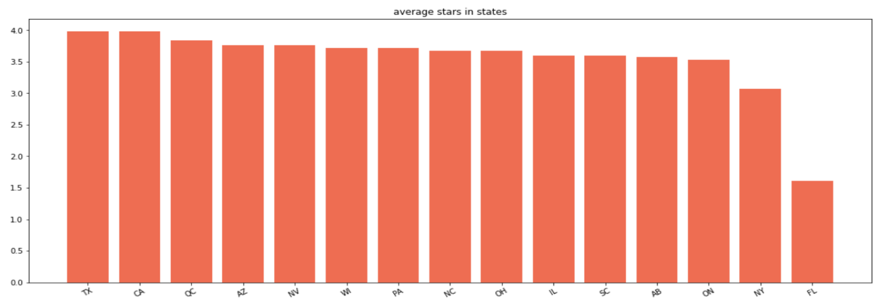


Figure 2: Average stars in states

This bar chart compares average stars in 15 different states.
According to the bar chart, state TX and CA have most high average stars,(about 4.0).In comparison, only state FL has the lowest average stars. Apart from these 3 states, the average stars of the rest of 12 states are almost the same(between 3.0 to 4.0 points).

For users:

```
user_data=[]
for l in letters:
    try:
        user_data.extend([json.loads(line) for line in open('../656/user/user_'+l,
'r')])
    except:
        break

friend_and_star={}
num={}
for d in user_data:
    friend_num=len(d['friends'].split(','))
    if friend_num in friend_and_star.keys():
        friend_and_star[friend_num]=(friend_and_star[friend_num]*num[friend_num]
        +d['average_stars'])/float(num[friend_num]+1)
        num[friend_num]=num[friend_num]+1
    else:
        friend_and_star[friend_num]=d['average_stars']
        num[friend_num]=1
plt.scatter(list(friend_and_star.keys()),list(friend_and_star.values()),color='indigo')
fig = plt.gcf()
fig.set_size_inches(18.5, 7)
plt.xlabel('number of friends')
plt.ylabel('stars')
plt.show()
```

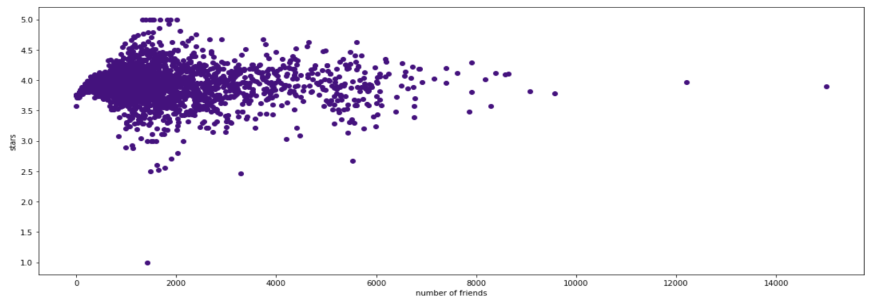


Figure 3: Scatter plot of friends and stars

From this scatter plot, we try to find the connection between the number of friends and stars, then we find that there is no relationship between them.

```
review_data=[json.loads(line) for line in open('../656/review/review_aa',
'r')]

star={}
num={}
for d in review_data:
    text_len = len(d['text'])
    count = d['stars']+d['useful']+d['funny']+d['cool']
    if text_len in star.keys():
        star[text_len]=(star[text_len]*num[text_len]+count)/float(num[text_len]+1)
        num[text_len]=num[text_len]+1
    else:
        star[text_len]=count
        num[text_len]=1
a = sorted(star.items(), key=lambda x: x[0])[::-1]
a=pd.DataFrame(a)
a[1]=a[1].rolling(200, win_type='triang').sum()
plt.plot(a[0],a[1],color='violet')
fig = plt.gcf()
fig.set_size_inches(18.5, 7)
plt.xlabel('length of text')
plt.ylabel('total average number of stars etc.')
plt.title('"whether useful" vs. length of text')
plt.show()
```

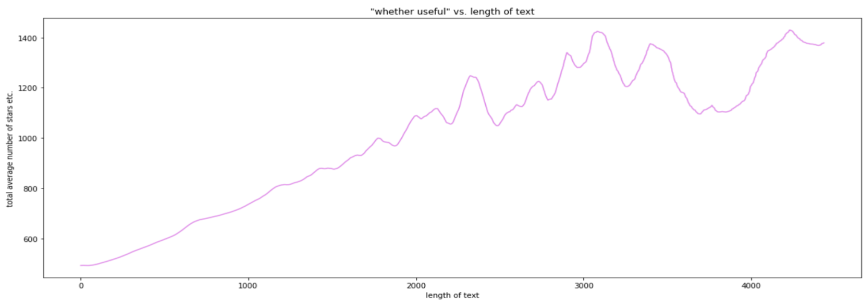


Figure 4: whether useful" vs. length of text graph

This graph shows a significant growth when the length of text from 0 to 2000. The total average number of stars fluctuates dramatically when the length of text rises from 2000 to 4000. From this trend, it is predicted that the total average number of stars will increase when the length of text has a rising trend.

```
review_data_pd=pd.DataFrame(review_data)
text_and_star=review_data_pd[['stars','text']]

from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report, accuracy_score
from sklearn.metrics import confusion_matrix
X_train, X_test, y_train, y_test = train_test_split(text_and_star['text'],
text_and_star['stars'])
vectorizer = TfidfVectorizer(stop_words="english", max_features=1000,
decode_error="ignore")
vectorizer.fit(X_train)
X_train_vectorized = vectorizer.transform(X_train)
cls = MultinomialNB()
cls.fit(vectorizer.transform(X_train), y_train)
y_pred = cls.predict(vectorizer.transform(X_test))

m=confusion_matrix(y_test, y_pred)
m1=[[] for i in range(5)]
for i in range(5):
    s=0
    for k in range(5):
        s=s+m[i][k]
    for j in range(5):
        m1[i].append(float(m[i][j]/float(s)))

import seaborn as sns; sns.set()
m1=pd.DataFrame(m1)
fig = plt.gcf()
fig.set_size_inches(10, 7)
ax = sns.heatmap(m1, annot=True)
plt.title('predict accuracy using MultinomialNB model')
```

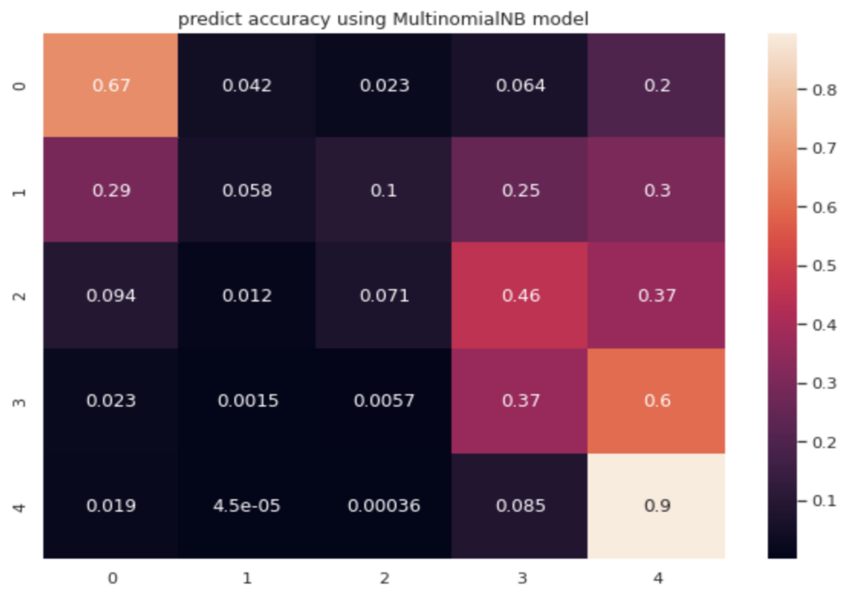


Figure 5: predict accuracy using MultinomialNB model

The model above shows that the accuracy prediction. This heatmap shows that values of accuracy in a data matrix. Smaller values were represented by small dark purple or black squares (pixels) and larger values by lighter squares. Number 0 to 4 represents star 1 to 5. From the different values of all squares in this model, we find that when star value is 1 and 5, the value of accuracy is high. It is hard to predict the star 2, 3 and 4, which means that people always like to choose the best or the worst review when they write the review.

5 Conclusion

This course is a database-design and implementation exercise. We create an appropriate database design for the problem space and implement a prototype of the design. At first, we create an Entity-Relationship Model of the problem space. Then we design the reasonable SQL to create the tables. What's more, we use the real-word yelp data set to support our simple social network.
Our social network has lots of functions, users can send posts, follow other people or a topic or group, like or dislike a post, create a topic or a group, see new posts or friend list, etc. Because of these functions, we didn't use all the data from yelp.json. We use business as topic, if users write review for a business, then we suppose that users follow this topic.
For the data analysis part, we listed the top 10 popular categories, and we compared the average stars in different states. We didn't find the relationship between number of friends and stars, but we found that when the length of text is longer, the stars are higher. It is understandable that people always choose extreme evaluation when they write a review.