Yelp SQL Database Analysis and Permission Control

Tianchang Hou, t6hou, 20713409
Qicheng Hu, q45hu, 20702348
Mengchen Shi, m38shi, 20578193

University of Waterloo
WInter 2018

Abstract—This project presents the method and results of analyzing and data mining of the Yelp Dataset as well as a permission control application of database access. As for part 1, and in aid to the analysis and mining process of the Yelp database an Entity Relationship (ER) diagram was first created to better visualize the interrelated relations; then the Yelp Database was cleaned and indexed so the data mining can be performed on non-garbage data efficiently to reveal some underlying information and knowledge that associated with the Yelp Database. For part 2, we developed a program to control the levels of access permissions for various users.

Keywords—Yelp Database; Data Cleaning; Data Indexing; Data Mining; Permission Control; SQL

Introduction

Yelp develops, hosts and markets Yelp.com and the Yelp mobile app, which publish crowd-sourced reviews about local businesses, as well as the online reservation service Yelp Reservations [1]. The public datasets provided by Yelp contains a large quantity of data stored in relational databases, and there is a huge amount of information that can be implied by its dataset. For example, if the person has booked many such flights to the same general location, but it is not particularly a business location, it may imply a sick relative. Other data may be available allowing confirmation or refutation of such a hypothesis (*e.g.*, there may be data about a parking lot payment near a hospital, which would support the hypothesis). By analyzing the data in a database, we may infer information and knowledge beyond the raw data.

In this project, we have cleaned and analyzed the Yelp dataset to reveal some underlying information by following the steps of constructing an E-R diagram, data cleaning to improve the quality of the data and data indexing to improve the speed of aggregation. Then we created an application to better handle the permission control of types of users to avoid data corruption and information leak.

1.0 Entity Relationship Diagram

The Entity Relationship diagram was constructed by MySQL workbench which is in Crow's foot notation. As shown in figure 1 below, the ER diagram represents the conceptual design of the Yelp database and there are eleven entities in the Yelp Dataset. The primary key for each entity is marked by a golden key sign in each entity, and cardinality was marked on the graph using Crow's foot notation.

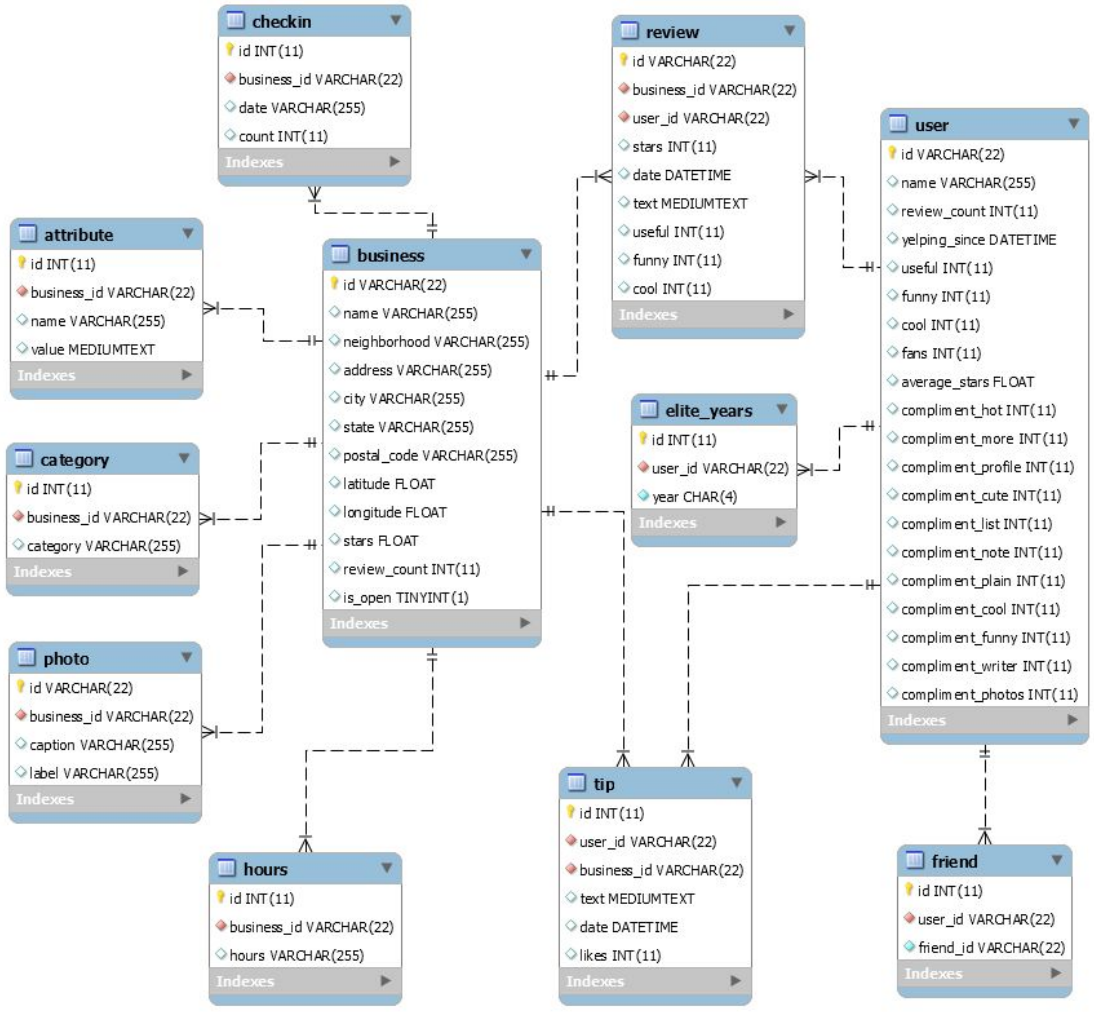

Fig. 1. Entity Relationship Diagram of Yelp Dataset

2.0 Data Cleaning and Sanity Check

Data cleansing or data cleaning is the process of detecting and correcting (or removing) corrupt or inaccurate records from a record set, table, or database and refers to identifying incomplete, incorrect, inaccurate or irrelevant parts of the data and then replacing, modifying, or deleting the dirty or coarse data [2]. Data cleaning is a vital step to improve the data quality before we could extract any useful information that infers from a dataset, especially for the case of Yelp dataset where the database is large. There are four components of data quality – accuracy, precision, consistency, and completeness [3], which were used as the guideline to clean the data for each relation of the Yelp Database.

2.1 Attribute

(1) *id* is the primary key of attribute table which means id can uniquely identify a record in the attribute table. Any duplication with the same *id* should be removed from the attribute table.

(2) The *business_id* of attribute table should preserve consistency to the *id* of the business table so that the business_id should be a foreign key that references the *id* of the business table.

2.2 Business

(1) *id* can uniquely identify each record of the business table and thus can be the primary key of the business table. Any duplication with the same *id* should be removed from the business table.

(2) The number of reviews received by a store as recorded in the business table cannot be smaller than the number of reviews in the Yelp dataset for that particular store (review_count >= num). Figure 2 shows a snippet of inconsistencies from the business table before cleaning, and figure 3 shows the result of a sanity check after cleaning.

id	review_count	num
cYwJA2A6I12KNkm2rtXd5g	5447	5448
DkYS3arLOhA8si5uUEmHOw	4869	4870
hihud--QRriCYZw1zZvW4g	3262	3264
Wxxvi3LZbHNIDwJ-ZimtnA	3101	3103
g8OnV26ywJlZpezdBnOWUQ	3050	3051
u_vPjx925UPEG9DFOAAvFQ	2938	2941

Fig. 2. Inconsistency of review_count before cleaning

id	review_count	num

Fig. 3. Sanity check after cleaning

2.3 Category

(1) *id* is the primary key of the category table which means *id* can uniquely identify a record in the category table. Any duplication with the same *id* should be removed from the category table.

(2) The *business_id* of the category table should preserve consistency to the *id* of the business relation which means that the *business_id* should be a foreign key which references the *id* of business table.

2.4 Checkin

(1) *id* is the primary key of checkin table which means *id* can uniquely identify a record in the checkin table. Any duplication with the same *id* should be removed from the checkin table.

(2) The *business_id* of the checkin table should preserve consistency to the *id* of business which means that the *business_id* should be a foreign key which references the *id* of business table.

2.5 Elite_years

(1) *id* is the primary key of elite_years table which means *id* can uniquely identify a record in the elite_years table. Any duplication with the same *id* should be removed from the elite_years table.

(2) The *user_id* of elite_years table should preserve consistency to the *id* of a user which means that the *user_id* should be a foreign key which references the *id* of the user table.

(3) The identified Elite years must be both consistent with the time the user started reviewing for Yelp (e.g., the user cannot be Elite in 2005 if he or she only started reviewing in 2007) and with our own knowledge of current data (e.g., the user cannot be Elite in 2001 since Yelp did not start until 2004; Years Elite cannot include 2019, since that is the future).

2.6 Friend

(1) *id* is the primary key of the friend table which means id can uniquely identify a record in the friend table. Any duplication with the same *id* should be removed from the friend table.

(2) The *user_id* of the friend table should preserve consistency to the *id* of a user which means that the *user_id* should be a foreign key which references the id of the user table.

(3) The *friend_id* of the friend table should preserve consistency to the *id* of a user which means that the *friend_id* should be a foreign key which references the id of the user table.

2.7 Hours

(1) *id* is the primary key of hours table which means *id* can uniquely identify a record in hours. Any duplication with the same *id* should therefore be removed from the hours table.

(2) The business_id of the hours table should preserve consistency to the *id* of the business which means that the business_id should be a foreign key that references the id of the business table.

2.8 Photo

(1) *id* is the primary key of photo table which means id can uniquely identify a record in the photo table. Any duplication with the same *id* should be removed from the photo table.

(2) The *business_id* of photo table should preserve consistency to the *id* of the business table which means that the *business_id* should be a foreign key which references the id of the business table.

2.9 Review

(1) *id* is the primary key of review table which means id can uniquely identify a record in the review table. Any duplication with the same *id* should be removed from the review table.

(2) The *business_id* of the review table should preserve consistency to the *id* of the business table which means that the *business_id* should be a foreign key which references the id of the business table.

(3) The *user_id* of review table should preserve consistency to the *id* of the user table which means that the *user_id* should be a foreign key which references the id of the user table.

(4) For accuracy and consistency check, Yelp was founded since 2004 and a review should not have a date in the future time. If a record's date is out of range of 2004-2018, we will remove it from the database.

(5) If the user's review date is earlier than the date he or she registered in yelp, we will remove it from the database.

2.10 Tip

(1) *id* is the primary key of tip table which means *id* can uniquely identify a record in the tip table. Any duplication with the same *id* should be removed from the tip table.

(2) The *business_id* of the tip table should preserve consistency to the *id* of the business table which means that the *business_id* should be a foreign key which references the id of the business table.

(3) The *user_id* of tip table should preserve consistency to the *id* of the user table which means that the *user_id* should be a foreign key references the *id* of the user table.

(4) For the same reason as in 9.d, if a record's date is out of range of 2004-2018, we will remove it from the tip table.

(5) The date in tip table must later than the date a user started using Yelp (date >= yelp_since). Figure 4 shows a snippet of inconsistencies from the tip table before cleaning, and figure 5 shows the result of a sanity check after cleaning.

id	date	yelping_since
VF49DRllHsERdrlcdZdibA	2014-06-11 00:00:00	2014-06-12 00:00:00
YAqFBgNu2bVk1VDUaOhYnA	2014-07-10 00:00:00	2014-07-11 00:00:00
2VymtqG7F4As-tc5OWwcvg	2016-08-17 00:00:00	2016-08-18 00:00:00
TPgT7r8YzxUVH_vt4DSx2Q	2016-05-27 00:00:00	2016-05-28 00:00:00
z4TmkMv7y1pLcGzULQzgxA	2014-09-29 00:00:00	2014-09-30 00:00:00
X9BlWb6A3EPOMc_ANdJlhA	2012-05-06 00:00:00	2012-05-07 00:00:00

Fig. 4. Inconsistency of tip table before cleaning

id	date	yelping_since

Fig. 5. Sanity check of tip table after cleaning

2.11 User

(1) *id* is the primary key of user table which means *id* can uniquely identify a record in the user table.

(2) Any duplication with the same *id* should be removed from the user table.

(3) For the same reason as in 9.d and 10.d, if the attribute *yelping_since* in the user table is not in the range of 2004 - 2018, we will remove it from the database.

(4) For the consistency check, the number of reviews written by a user as recorded in the user table cannot be smaller than the number of reviews of that particular user in the sample Yelp dataset. Figure 6 shows a snippet of inconsistencies of the user table before cleaning.

id	review_count	num
-61V4ZkRsKUChYFZtdZDvQ	0	4
-9TyYbKtEz-pxeZyLICOgA	0	1
-arJ-0bq2eyclNnHrm0LFA	0	2
-CgbL7Rygee52C3ylCV3iQ	2	3
-d8nnc-pp6qj_6qnp4IN-g	0	1
-Dhxu5B36bkm65ciME0vxg	0	22
-G0sArycZyO_-AHlCPI4IQ	0	11

Fig. 6. Inconsistencies of the user table before cleaning

3.0 Data Indexing

Index is a data structure that speeds up the lookup of records by specific search keys. They improve the performance of query from two different aspects, the first is organizing data in ways that benefit specific types of queries and the other aspect is the smaller scale of the index file.

Adding index can greatly improve the performance of a database system, for instance, accelerating the search of some unique search-keys. An index can get the result in much lesser time as compared with linear search algorithm and it can speed up the joining operation between tables.

However, there are few drawbacks with indexing as well. First, the process of indexing takes time, and the time will increase with the growth of the data. Secondly, it decreases the performance on inserts, updates, and deletes by taking additional time to deleting or updating some columns with indices corresponding to the specific operations. Thirdly, the index file structure takes plenty of physical storage space.

In this project, we demonstrate the use of indices to speed up several parts of cleaning and querying processes. The strategy and process of indexing various tables in the Yelp database are documented in this section and we will introduce them respectively.

3.1 Cleaning reviews whose date is illegal

```
delete from review
where date<'2004-01-01 00:00:00' or date>='2019-01-01 00:00:00' ;
```

Fig. 7. Cleaning Statement

Before indexing, the running time for cleaning reviews is 7.998 seconds. By adding a B+ Tree index to the 'date' attribute, the running time is reduced to 0.0012 seconds for the same cleaning process.

3.2 Cleaning businesses with illegal review numbers

```
DELETE business FROM business
    INNER JOIN (
        SELECT business_id, COUNT(DISTINCT id)AS rev_num FROM review GROUP BY business_id
    )AS X
    ON business.id=X.business_id
    WHERE review_count<X.rev_num ;
```

Fig. 8. Cleaning Statement

Without indexing, the cleaning operation took approximately 10 minutes. By adding a foreign key to attribute business_id of the review relation references business:id, and a B+ Tree index to the business_id, the execution time was reduced to 19sec for the same deletion operation.

3.3 Cleaning users who are not representative

```
delete `user` FROM `user` INNER JOIN (
       SELECT  user_id, COUNT(DISTINCT id
       )AS rev_num
       FROM review GROUP BY user_id)AS X
    ON `user`.id=X.user_id
    WHERE review_count<X.rev_num ;
```

Fig. 9. Cleaning Statement

Without indexing, the cleaning operation of removing users with inconsistent number of reviews in the database took approximately 5.141 minutes without indexing and foreign key. By adding a foreign key to attribute user_id of the review relation references user:id, and a B+ Tree index to the user_id, the execution time was reduced to 9.438 sec for the same deletion operation.

3.4 Querying average ratings of each business before and after 2016

```
select A.business_id, before_2016, cnt_before_2016, after_2016, cnt_after_2016
from
(select business_id, avg(stars) before_2016, count(id) cnt_before_2016 from review where date < '2016-01-01 00:00:0' group by business_id) A
inner join
(select business_id, avg(stars) after_2016, count(id) cnt_after_2016 from review where date >= '2016-01-01 00:00:0' group by business_id) B
using(business_id);
```

Fig. 10. Query Statement

Before indexing, the execution took more than 5 minutes. Then we added a B+ Tree Index to the column ‘date’ and ‘business_id’ which accelerated the running time to 42.007 seconds.

4.0 Data Analysis

4.1 Determination of improving or declining of ratings of businesses

We analyzed the ratings of each business to determine whether its ratings are improving or declining. For a business, although a long-term rating can reflect the customers’ long-term preference of it, the short-term ratings can be more reflective in the aspects of capturing some changes in the business and evaluating whether some of the improvements done by the business are helpful or not.

To capture the rating trend of a business, we applied two distinct techniques to reveal it. The first is to compare the average rating of a business between two specific date ranges, i.e., before and after 2016-1-1. The second is using regression to display the general tendency of ratings.

4.1.1 Comparison of average ratings before and after 2016-1-1

To compare the average ratings for businesses before and after 2016-1-1, we excluded the reviews of businesses which are opened later than 2016-1-1 as there is no review for these businesses before the date and thus the tendency is meaningless.

```
drop view table_trend_2016;
create view table_trend_2016 as
select A.business_id, before_2016, cnt_before_2016, after_2016, cnt_after_2016
from
(select business_id, avg(stars) before_2016, count(id) cnt_before_2016 from review where date < '2016-01-01 00:00:0'  group by business_id) A
inner join
(select business_id, avg(stars)  after_2016, count(id) cnt_after_2016 from review where date >= '2016-01-01 00:00:0'  group by business_id) B
using(business_id);
```

Fig. 11. Query Statement of Average Ratings

business_id	before_2016	cnt_before_2016	after_2016	cnt_after_2016
--9e1ONYQuAa-CB_Rrw7Tw	4.1491	369	4.2170	106
-050d_Xlor1NpCuWkbIVaQ	4.3448	87	3.0000	1
-0qht1roIqleKiQkBLDkbw	3.0682	88	2.5625	64
-1UMR00eXtwaeh59pEiDjA	4.0233	43	3.3645	107
-2ToCaDFpTNmmg3QFzxcWg	1.3596	178	1.3426	108
-3zffZUHoY8bQjGfPSoBKQ	3.9225	142	4.2973	37

Fig. 12. Comparison of Average Stars

A snippet of average rating comparison between before and after of 2016 for each business is shown in Fig.12, before_2016 and cnt_before_2016 represents the average rating and count of rating before 2016 respectively. Similarly, after_2016 and cnt_after_2016 represents the average rating and count of rating after 2016.

cnt_improving	cnt_unchanging	cnt_declining
21810	16373	24242

Fig. 13. Number of Businesses which are improving or declining

The results are shown in Fig. 13, 21810 businesses got improved average rating after year 2016, while 24242 businesses got their averaged rating decreased after year 2016. The middle column shows 16373 businesses kept their ratings approximately unchanged as compared to that of before 2016.

4.1.2 Regression of ratings of the specific business

We also used the method of regression analysis to analyze the general trend of ratings of a business for a given period. The Linear Regressor was used to represent the overall trend and Gradient boosting (GBDT) Regressor was used to represent a finer and more detailed trend.

Fig. 14. Regression Result of Business '--9e1ONYQuAa-CB_Rrw7Tw'

Fig. 14 shows the regression result of the business with id '--9e1ONYQuAa-CB_Rrw7Tw', the red line represents the result of the Linear Regressor and the green line represents the result of GBDT Regressor. From the results, we can gather some information like the general tendency of ratings of the business, and it is showing the business is improving since the first day it opened and it has underwent trough and reached peak recently. In conclusion, this kind of businesses are making positive changes to itself.

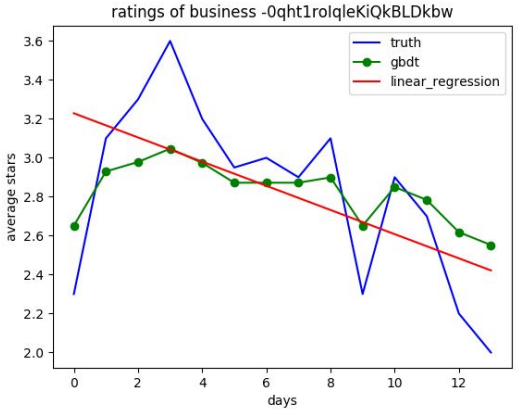

Fig. 15. Regression Result of Business '-0qht1roIqleKiQkBLDkbw'

Fig. 15 displays the regression result of the business with id '-0qht1roIqleKiQkBLDkbw', the red line represents the result of Linear Regressor and the green line represents the result of GBDT Regressor. From the results, we can get some information like the general tendency of ratings of the business is declining since the first day it opened and it reached its trough recently. In conclusion, this kind of businesses are making negative changes to itself, they should make more improvements in their services or products.

4.2 Rating Prediction

The objective of Rating Prediction is to capture what users are really interested in and satisfied with to improve the personalized recommendation.

4.2.1 A naive rating prediction method

To predict the ratings, we firstly split businesses into two classes, higher_average, and lower_average, which represents businesses which have an average rating higher than 3.5 or lower than 3.5. We chose higher or lower than 3.5 as the splitting criteria because it can split the businesses into two classes with similar quantities. Then we predicted the ratings based on the preference of users towards two different kinds of businesses.

```
drop view if exists business_higher_rating;
create view business_higher_rating as
   select id,stars from business where business.stars>3.5;

drop view if exists business_lower_rating;
create view business_lower_rating as
   select id,stars from business where business.stars<=3.5;

drop view if exists rating_prediction;
create view rating_prediction as
select * from
(select user_id,avg(review.stars-business_higher_rating.stars) avg_higher,count(review.business_id) number_higher, avg(review.stars) avg_stars_higher
from review ,business_higher_rating
where business_higher_rating.id = review.business_id
group by user_id) A
inner join
(select user_id,avg(review.stars-business_lower_rating.stars) avg_lower,count(review.business_id) number_lower, avg(review.stars) avg_stars_lower
from review ,business_lower_rating
where business_lower_rating.id = review.business_id
group by user_id) B
using(user_id);
```

Fig. 16. Query Statement for Rating Prediction

user_id	avg_higher	number_higher	avg_stars_higher	avg_lower	number_lower	avg_stars_lower
---udAKDsn0yQXmzbWQNSw	0.5	1	5.0000	0.5	1	4.0000
--1mPJZdSY9KluaBYAGboQ	1	1	5.0000	1.5	2	5.0000
--3oMd6gjXpAzhjLBrsVCQ	-0.5	1	4.0000	1	1	4.0000
--44NNdtngXMzsxyN7ju6Q	0	1	5.0000	1.5	1	5.0000
--6Ke7_IBBM6XAramtPoWw	1	1	5.0000	0.5	1	4.0000
--bk6oc1GSNnTZG-UakcfQ	0.8333333333333334	3	5.0000	1.75	2	5.0000
--f7c9Mlug2QcYA7aCzxYg	1	2	5.0000	2	1	5.0000
--gc9Xg3MPKCy3BPt1g_1A	0.25	2	5.0000	1.5	1	5.0000

Fig. 17. Result of Query Statement

Fig. 17 shows the result of querying average ratings of each user towards two kinds of businesses. avg_higher represents average ratings higher than that of businesses with an average rating higher than 3.5 and avg_lower represents average ratings lower than that of businesses with an average rating lower than 3.5.

After getting the query result, we predicted the ratings by considering the preference of a user to those two classes of businesses, that is, when predicting the rating from a user to a business with an average rating lower than 3.5, we get the prediction by adding avg_lower and the truth average rating of this business. Similarly, when predicting the rating from a user to a business with an average rating higher than 3.5, we get the prediction by adding avg_higher and the truth average rating of this business. For example, when to predict the rating of user '---udAKDsn0yQXmzbWQNSw', the avg_higher is 0.5, then when predicting a business with average rating 3.2, the prediction is 3.2+0.5=3.7.

4.2.2 Rating Prediction based on Matrix Factorization

Matrix Factorization (MF) is another kind of method we applied for this project. The objective of MF is to solve two matrices P and Q and let the inner product of P and Q as R, the User-Item Rating Table, which is:

$$R_{U*I}=P_{U*K}*Q_{K*I}$$

We implemented SVD++ to do MF in this project.

4.2.2.1 Description of SVD++

SVD++ [6] makes some improvements by adding bias of each user and a bias of each item. Also, it considers the implicit preference or feedback of each user. Then the prediction function becomes much more complicated and contains more information. The prediction function is

$$\hat{r}_{ui} = \mu + bi + bu + q_i^T(p_u + \frac{1}{\sqrt{\|R_u\|}}\sum_{j\in R_u} y_j)$$

Given the prediction function (hypothesis) of the ratings, we can get the Cost Function to optimize, which is

$$\underbrace{arg\ min}_{p_u,q_i} \sum_{u,i}(r_{ui} - \mu - b_u - b_i - q_i^T p_u - q_i^T|R_u|^{-1/2}\sum_{j\in R_u} y_j)^2 + \lambda(||p_u||_2^2 + ||q_i||_2^2 + ||b_u||_2^2 + ||b_i||_2^2 + \sum_{j\in R_u} ||y_j||_2^2)$$

We used Stochastic Gradient Descent Algorithm (SGD) to optimize the Cost Function.

$$e_{ui} = r_{ui} - \hat{r}_{ui}$$
$$b_u \leftarrow b_u + \gamma \cdot (e_{ui} - \lambda \cdot b_u)$$
$$b_i \leftarrow b_i + \gamma \cdot (e_{ui} - \lambda \cdot b_i)$$
$$p_u \leftarrow p_u + \gamma \cdot (e_{ui} \cdot q_i - \lambda \cdot p_u)$$
$$q_i \leftarrow q_i + \gamma \cdot (e_{ui} \cdot (p_u + \frac{1}{\sqrt{\|R_u\|}}\sum_{j\in R_u} y_j) - \lambda \cdot q_i)$$
$$y_j \leftarrow y_j + \gamma(e_{ui} \cdot \frac{1}{\sqrt{\|R_u\|}} \cdot q_i - \lambda \cdot q_i)$$

4.2.2.2 Applying SVD++ to Yelp Dataset

```
drop table if exists user_business_stars;
create table user_business_stars as
select user_id, business_id, stars
from review inner join (select user_id
from review
group by user_id
having count(business_id)>=5) A
using(user_id);
```

Fig. 18. Query Statements to get User-Business Rating Table

We only considered users who have rated at least 5 different businesses to train the SVD++ model because users who didn't rate enough businesses cannot provide enough information. After adding this filtering condition, there are only 11736 users left in our dataset. Then inner join these 11736 users and the review table, we got 79017 reviews and by building the user-business rating matrix, we can get the input matrix of SVD++ Algorithm. For training/testing set splitting, we randomly chose 2 reviews from each users' reviews as the test set.

4.2.2.3 Analysis of SVD++

The Mean Absolute Error (MAE) on the testing set is 0.82 using SVD++. One advantage of SVD++ is easy to implement. Then, a sparse user-business matrix will be decomposed into two matrices, and that reduces the dimensionalities and solve the problem of the sparse matrix to some extent as well, and it will result in the reduction of calculation complexity in the end. Last, using SVD++ can reflect an implicit preference of users.

For its disadvantages, it's time-consuming to train and it's impossible to predict the rating of a business which is new to Yelp.

5.0 SQL Security and Priviledges

A privilege is the right to perform a particular action or to perform an action on any schema objects of a particular type. For security reasons, we only grant specific privileges to trusted users. Otherwise, The abuse of privilege may cause database corruption. Therefore, we should divide users into different groups and grant privileges only to users who require that privilege to accomplish the necessary work.

The roles we created are shown in Fig. 19, and we will give some explanation of the creation of each user.

```
# Create user 'Critiques' and grant
CREATE USER 'Critiques'@'localhost' IDENTIFIED BY 'qwert';
-- grant select on yelp_db.* to Critiques;
GRANT INSERT ON yelp_db.review TO Critiques;

# Create user 'Business_analysts'' and grant
CREATE USER 'Business_analysts'@'localhost' IDENTIFIED BY 'qwert';
GRANT CREATE VIEW ON yelp_da.* TO Business_analysts;

# Create user 'Developer' and grant
CREATE USER 'Developer'@'localhost' IDENTIFIED BY 'qwert';
GRANT DELETE,INSERT, UPDATE ON yelp.* TO Developer;

# Create user 'Admin' and grant
CREATE USER 'Admin'@'localhost' IDENTIFIED BY 'qwert';
GRANT ALL PRIVILEGES ON yelp.* TO admin;

#Select Operations
GRANT SELECT ON yelp_db.* TO public;
```

Fig. 19. SQL Statements of user creation

5.1 Critiques

Critiques are allowed to browse search results and write reviews for places they visit. Thus, we grant SELECT privilege for all tables and INSERT privilege for REVIEW table to Critiques.

5.2 Business Analysts

Business Analysts are allowed to browse search results and access to creating extra views on the database schema. Thus, we grant SELECT privilege for all tables and CREATE VIEW for all tables to Business Analysts.

5.3 Developers

Developers are allowed to browse search results and write reviews, create new tables, perform data cleaning and indexing. Thus, we grant INSERT, UPDATE and DELETE privilege for all tables to Developers.

5.4 Database Admin

Database Admin is allowed to have full access to the database. Thus, we grant ALL privilege for all tables to Database Admin.

References

1. Yelp Fact Sheet*, Yelp*, retrieved May 9, 2017
2. G. Eason, B. Noble, and I.N. Sneddon, “On certain integrals of Lipschitz-Hankel type involving products of Bessel functions,” Phil. Trans. Roy. Soc. London, vol. A247, pp. 529-551, April 1955.
3. Veregin, H. Data quality parameters. Geographical Information Systems, pp. 177–190, 1999. Retrieved from http://www.geos.ed.ac.uk/~gisteac/gis_book_abridged/files/ch12.pdf
4. Koren, Y., Bell, R., Volinsky, C.: Matrix Factorization Techniques for Recommender Systems. Computer. 42, 30-37 (2009).
5. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Application of dimensionality reduction in recommender system-a case study. Minnesota Univ Minneapolis Dept of Computer Science. (2000).
6. Zheng, X., Luo, Y., Sun, L., Chen, F.: A New Recommender System Using Context Clustering Based on Matrix Factorization Techniques. Chinese Journal of Electronics. 25, 334-340 (2016).