Topic 2: Analysis of Movie Recommendation System for MovieLens Dataset

Group ID :13

Student Name	**Student Number**
Kxxxx Cxxx	12xxxx
Jxxx xxx	9xxxx
Sxx xxxx	1xxxx
Mohammad Emon	12794121

Table of Contents

Abstract

A recommendation system uses an algorithm to predict unknown information by utilizing available knowledge about the user and related items. Within e-commerce, recommendation systems help personalize the user experience, by displaying the items the user is most likely to be interested in, based on similar items that the user has been interested in or items that similar users have been interested in. The aim of this project, focuses on two aspects and methods of online recommendation systems, using the user and item-based collaborative filtering models. To complete this evaluation, a recommendation system is created and designed to generate a predicted rating for a movie from a user-based on their past ratings and ratings from users from similar tastes, then analyze and compare several different algorithms and prediction models that can be used to build the recommendation system. We take advantage of data from the MovieLens (ML) website datasets to help evaluate the accuracy of our models. However, from the analysis, it was observed that Item-based collaborative filtering (CF) have lower root mean square deviation (RMSD) compared to user-based CF evaluation performance which is calculated by average error between the actual rate and predicted rating. The project finds that item-based CF recommendation system is potentially suggested for Small to medium (SME) E-commerce and information filtering business in order to optimize and finding new products through online business so that customers expectations are met which also can generate more revenues.

Keywords: Recommendation System, Collaborative Filtering (CF), User-user CF, Item-item CF, Movielens (ML)

1.Introduction

Due to the substantial growth of e-commerce business, one of the most popular ways for businesses to help users interact with or discover items online is by building a recommendation system using an algorithm to analyze the user's preferences and other similar users tastes , in order to give the best possible recommendations of its products or services. It is increase and help the business generate more sales, revenue and enable users to find the items they like (Sarwar et al., 2001). There are many types of recommendation system models including content-based, demographic and collaborative filtering but in this paper, discuss only about the CF techniques and then the evaluation performance metrics is analyzed by RMSD and Recall. By dynamic, we mean a model that can run in online commerce business backend of the database and forecasts the rating or make preferences at a future point of time which helps and keep users engaged with the websites and apps.

The Explicit feedback data(rating) is used to make a preference model for the customers. Due to this offline algorithm techniques, it is suffering from ‘Cold start’ problem when new products are introduced in the system (Hu et al., 2008). There are other data types implicit feedback such as purchase history, search history etc. But in this project, the ML five-star rating datasets will be used to analyze the recommendation system. The main challenges are to improve the scalability, sparsity, quality performance of CF. In the figure below briefly showed the CF process.

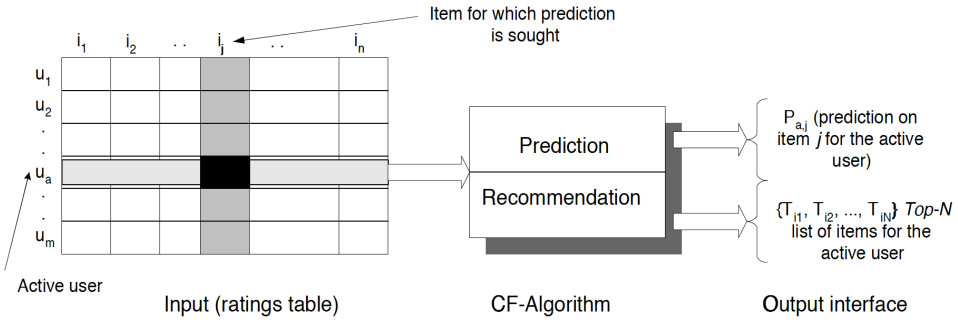

Figure 1: The CF process (Sarwar et al., 2001)

In this project, we address these problems by applying different approaches algorithms user-based and item-based for recommendation system.

In the following sections discussed about the rating prediction of the users or item-based on ML datasets using the collaborating filtering user and item-based techniques. Several sections are divided in this report to discuss more details of the project such as 'Section 2' review prior research , which is done related to CF, 'Section 3' discusses the brief summary of the ML datasets, Section 4 describes the details proposed algorithm techniques for this project and lastly, Section 5 describes the experimental results and the evaluation analysis using RMSD and Recall and lastly, make comparison with the obtained result to conclude.

2. Related Works

In this section, some of the related works or prior research that have done previously using ML dataset and briefly summarized of the selected papers and then discuss about the ML dataset that using different models for recommendation system to compare and critical review for each of the papers.

2.1 A Scalable, Accurate Hybrid Recommender System:

According to the article 'A Scalable, Accurate Hybrid Recommender System (2014), state that the recommended system included three features which are collaborative filtering, content-based filtering, and demographic recommender systems. Proposal of this work to providing a particular cascading hybrid recommendation method with the demographic information about an item, feature and matching the rating, in order to reduce the data sparsity, scalability and resulting in the quality problem of the performance outcome of the recommender system. The author used ML and Film Trust datasets for evaluation, they predict the scores of each item that already active and use Mean Absolute Error (MAE) and Receiver Operating characteristic (ROC) to store the performance (Ghazanfar M. and Prugel-Bennet A., 2014).

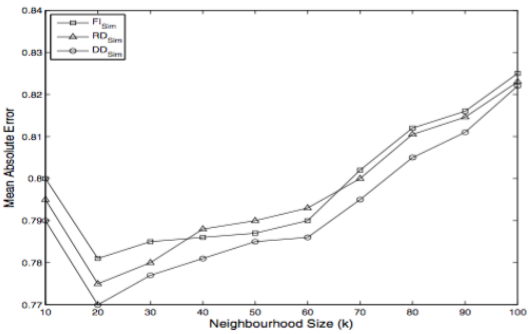

Figure 2: Determining the optimal value of neighbourhood size, k (MovieLens)
Adapted zanfar M. and Prugel-Bennet A., 2014

They varied the number of neighbors for an active user, from 0 to 100 and computed the corresponding MAE for FI, RD, and DD sim. The results are shown that for ML dataset MAE is minimum for k = 20. Also, the hybrid named Boosted RDF is used to building accurate and practical recommendation. By comparing the MAE, ROC Sensitivity, Coverage, and On-line Cost of the Proposed Algorithm with the others, for the FilmTrust dataset. The coverage of the algorithm is very low because it is very sparse, therefore the Boosted Demo feature is predicted that scalable and practical as its online cost is less or equal to the cost of other algorithms.

After randomly selected the rating for testing and training as 20% and 80% respectively. They also did compare the algorithm with other 7 differences and did turn all of the algorithms into the best mentioning parameters. With the evaluation of Item and user, the authors use a linear regression model for finding an approximation of the active user' rating for an item, which is shown the result having more accuracy for the recommended system.

2.2 AutoRec: Autoencoders Meet Collaborative Filtering

Krishna Menon A. et al (2015) discussed about the neural autoencoder (AutoRec) CF recommendation model to predict the user's preference for items using the ML and Netflix datasets. It was predicted that AutoRec has significant advantages compared to other neural CF approaches. The collaborative filtering is one of the models that will support the user-item to achieve advance information from the personalized recommendation. The author had attempted to proposing the AutoRec which is a new collaborative filtering model by based on the autoencoder paradigm in order to achieve the current state-of-art-method performance.

The author has designed an item-based(user-based) autoencoder for input into the low-dimensional talent area, and after that, the output space will reconstructure to predict the missing rating of the recommendation.

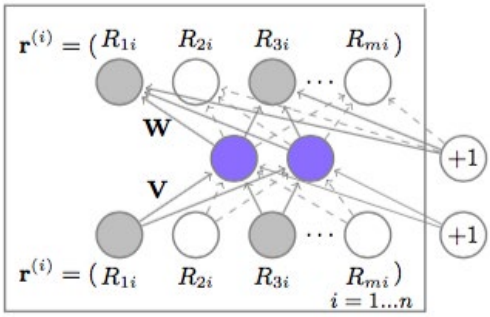

Figure 3: Item-based AutoRec model. Adapted from Krishna Menon A. et al (2015)

The above figure shown the AutoRec model of the item based that the author used an autoencoder for each Equation 1 to the set of vectors and two particulars modify. The model demonstrates the shaded nodes corresponding to input rating and connecting to weight that update for the output.

The author did compare the AutoRec and Restricted Boltzmann machine (RBM)-CF based on the ML 1M, 10M and Netflix dataset based on the Biased Matrix Factorization and Local Low-Rank. Finally, the experimental found that the AutoRec have some differences from the RBM-CF because the Autorec is unaccepting to r (possible ratings), less memory and less biased to the overfitting even require more parameters. It was achieved great accuracy using user or item based with AutoRec. The evaluation matrices using RMSE is reduced from 0.831 to 0.827 using the model.

3. Dataset

We use the ML datasets which contains data from the MovieLens website from the 7 months from 19th September 1997 to 22nd April 1998. ML is an online recommender system that recommends movies for users to watch based on their ratings and reviews of their previously watched movies. The data only contains users who had at least 20 ratings. The full dataset used contains 100,000 ratings provided from 943 users on 1682 movies.in

The data has been organized into 4 columns with each row containing user id, item id, rating and timestamp. Unique user id and item id values are given to different users and movies respectively, numbered consecutively from 1. The rating column contains rating, in integer values from 1 to 5, given by an individual user to a specific movie. The timestamp contains the time of the rating in unix time format. For the purposes of creating and evaluating a movie recommendation system, the ML data has been separated into training and test datasets in a 4 to 1 ratio split. Although, the majority number of the rating values are sparse.

4. Methods or Algorithm

4.1 User-User Filtering

The user-user filtering method recommends movies based on the ratings given by similar users. Similar users are calculated from the similarity of ratings of other movies that both users have rated. These techniques also known as ‘memory or user-based’ CF is the most popular and widely used in research, information and e-commerce business. The following section we briefly described the user-based CF computation process using ‘structured query language’ (SQL).

4.1.1 Correlation (Cosine Similarity)

The level of correlation between each user and the rest was calculated by cosine similarity.

$$\text{sim}(a, b) = \frac{a \cdot b}{\|a\|_2 \|b\|_2}$$

Figure 4: Cosine similarity (Dash et al., 2018)

In this study, we stored the source dataset into SQL Server database and queried to get the similarities between users. Firstly, we have tested the dataset “u1.base”. The following SQL illustrates how we obtained the cosine similarity table.

```
 WITH POW_RATINGS AS (
            SELECT USER_ID
             ,SUM(RATINGS * RATINGS) AS R2      ------------------(1)
            FROM U1_BASE
            GROUP BY USER_ID
)
SELECT X.USER_ID
    ,Y.USER_ID AS USER_ID2
    ,PX.R2 AS X2                ------------------(2)
    ,PY.R2 AS Y2                ------------------(3)
    ,SUM(X.RATINGS * Y.RATINGS) AS INNERPRODUCT                         ------------------(4)
    ,ROUND(SUM(X.RATINGS * Y.RATINGS)/SQRT(PX.R2 * PY.R2),4) AS SIMILARITY  ----------(5)
  INTO U1_SIMILARITY
  FROM U1_BASE AS X
  JOIN U1_BASE AS Y WITH(INDEX(IX_U1_BASE_01)) ON (X.ITEM_ID = Y.ITEM_ID)
  JOIN POW_RATINGS AS PX ON (PX.USER_ID = X.USER_ID)
  JOIN POW_RATINGS AS PY ON (PY.USER_ID = Y.USER_ID)
GROUP BY X.USER_ID
            ,Y.USER_ID
            ,PX.R2
            ,PY.R2
;
```

Firstly, (1) is the temporary view created to calculate “||a||2*||b||2” from the above formula. And this view was joined with the main table “U1_BASE” two times where PX.R2 in (2) is the result value of ||a||2, and PY.R2 in (3) is for ||b||2. And the part (4) gets the result value of “a*b”. Finally, we got the final cosine similarities in (5).

The following table (Table #) shows the result of the previous SQL query.

USER_ID	USER_ID2	X2	Y2	INNERPRODUCT	SIMILARITY
1	1	2049	2049	2049	1
1	2	2049	616	109	0.097
1	3	2049	298	41	0.0525
1	4	2049	279	16	0.0212
...	...	...	...	...	...
...	...	...	...	...	...
943	939	2219	937	208	0.1442
943	940	2219	1388	423	0.241
943	941	2219	377	87	0.0951
943	942	2219	1483	331	0.1825
943	943	2219	2219	2219	1

Table 1: Cosine similarities between users in the dataset “u1.base”

Those similarities will be applied as variables to predict target users’ movie ratings in the following part.

4.1.2 User’s Movie Rating Prediction

Once we compute the most similar user-user using ‘cosine similarity’ formula, after it the most important step of CF system is generate the output interface of prediction. In order to predict the ratings of target users and movies, the ‘weighted average’ was implemented as in this method. The ‘weighted average’ method results the predictive value by reflecting similarities between users and calculating the average of the most similar users’ ratings. The formula for computing the ‘weighted sum’ mentioned below figure.

$$P_{u,i} = \frac{\sum_{\text{all similar items, N}} (s_{i,N} * R_{u,N})}{\sum_{\text{all similar items, N}} (|s_{i,N}|)}$$

Figure 5: Weighted averaged or prediction rate formula (Dash et.al., 2018)

The following SQL query describes how the predictive values are obtained in SQL Server database.

```
WITH U_AVG AS (
SELECT USER_ID, AVG(RATINGS) AVG_RATINGS ----------(6)
  FROM U1_BASE
 GROUP BY USER_ID
)
SELECT ORG.USER_ID
       , ORG.ITEM_ID
       , ORG.RATINGS
       , CASE WHEN ORG.PREDICTED = 0 THEN A.AVG_RATINGS ELSE ORG.PREDICTED END PREDICTED
FROM (SELECT T.USER_ID, T.ITEM_ID, T.RATINGS, COALESCE(P.W_AVG,0) PREDICTED
            FROM U1_TEST T                          ----------(1)
            LEFT JOIN (SELECT A.USER_ID             ----------(2)
                              , B.ITEM_ID
                        , SUM(SIMILARITY*B.RATINGS)/SUM(CASE WHEN B.RATINGS = 0 THEN 0
                                                             ELSE SIMILARITY END) W_AVG --------(3)
                        FROM (SELECT USER_ID
                                     , RK
                                     , USER_ID2  ----------(4)
                                     , SIMILARITY
                                FROM (SELECT EGO
                                             , DENSE_RANK () OVER (PARTITION BY USER_ID ORDER BY SIMILARITY
                                        DESC) RK
                                             , USER_ID2
                                             , SIMILARITY
                                      FROM U1_SIMILARITY
                                      WHERE USER_ID <> USER_ID2) A
                              WHERE RK <= 10 ----------(5) INPUT THE NUMBER OF NEIGHBOURS
                              ) A
                       JOIN U1_BASE B ON (B.USER_ID = A.USER_ID2)
                       GROUP BY A.USER_ID, B.ITEM_ID) P ON (T.USER_ID = P.USER_ID AND T.ITEM_ID = P.ITEM_ID)) ORG
JOIN U_AVG A ON (A.USER_ID = ORG.USER_ID)
;
```

In this section, the brief explanations for some parts of the query will be provided. Firstly, the testset “u1.test” (1) is the main table with the users’ ratings to movies. And this table is merged with another table containing each user’s weighted average ratings to each item (2). The weighted average ratings were calculated based on the top 10 similar users’ ratings (3). And the top 10 similar users were selected based on the similarities that we had in the previous “Correlation” part (4). We can choose the number of similar users to be involved in the weighted average process (5). In this study, ten most similar users are selected to get a weighted average value based on their similarities.
When the target user could not find anyone of Top 10 similar users who rated the target movie, the result of the weighted average formula will give zero value. To avoid this, we have replaced zero values with the average value of target user’s ratings (6).

The result of the SQL query is in showing below table.

USER_ID	ITEM_ID	RATINGS	PREDICTED
99	274	1	3
22	227	4	3
96	173	3	3
250	64	5	5
12	381	4	3
181	1017	1	3
86	327	4	4
...	...	...	...

Table 2: Predicted ratings by Weighted Average Method with Top 10 Similarities In User-User Collaborative Filtering

The columns “USER_ID”, “ITEM_ID” and “RATINGS” are the original data of “u1.test”. And the last column “PREDICTED” is the result value of the weighted average method.

4.1.3 Evaluation

RMSD:

In this project we used the root mean square deviation (RMSD) is a commonly used statistical measure of accuracy for a prediction model which is average the difference between the actual and predicted values. The RMSD of predicted values for times(t) of a regression's dependent variable is computed for n different predictions as the square root of the mean of the squares of the deviations. The lower the RMSD the more accurately the recommendation prediction model will be. The formula is adopted from ‘UTS online’.

$$\mathrm{RMSD} = \sqrt{\frac{\sum_{t=1}^{n} (\hat{y}_t - y_t)^2}{n}}$$

Figure 6: Evaluation matrices formula

Recall:

Recall is another simpler measure to evaluate the results of prediction models. It just computes the number of true positive values over the total values, when comparing the actual and predicted values. Recall (R) is defined as the number of true positives (T_p) over the number of true positives plus the number of false negatives (F_n). Figure below showing the formula for calculating the ‘Recall’ (adopted from UTS online)

$$R = \frac{T_p}{T_p + F_n}$$

Figure 7: Recall evaluation

In our model, which predicts numeric values, the value R will be calculated as follows.

R = The number of times the actual and predicted ratings are equal / The number of total tests

Result:

For the evaluation of the previous model we developed, the root means square deviation (RMSD) and Recall were calculated on each distance between the actual value and predicted value. The SQL formula of each evaluation method is as follows.

-RMSD: SQRT(SUM(ABS(RATINGS-PREDICTED) *ABS(RATINGS-PREDICTED))/COUNT (*))

-RECALL: SUM (CASE WHEN RATINGS = PREDICTED THEN 1 ELSE 0 END)/(SUM(CASE WHEN RATINGS = PREDICTED THEN 1 ELSE 0 END) + SUM(CASE WHEN RATINGS <> PREDICTED THEN 1 ELSE 0 END))

We also tested the other train and test datasets including u2.base, u2.test, u3.base and u3.test. The result values of RMSD and RECALL for each dataset is in showing following table below.

	U1.TEST	U2.TEST	U3.TEST
RMSD	1.1342	1.1409	1.1565
RECALL	0.3964	0.3930	0.3894

Table 3: Evaluations of User-User Collaborative Filtering

4.2 Item-Item Filtering

Similarly, the method of item-item filtering looks for similar movies, which are calculated from the similarity of the ratings that are given to the movies. Also known as ‘model-based or item-based’ CF which improve in the case of scalability and sparse dataset. This approach produces more faster recommendations as it does not require to identify

similar users while making recommendations. The following section describes the computation techniques of 'item-item' CF.

4.2.1 Correlation (Cosine Similarity)

Like User-User Collaborative Filtering method, the cosine similarities were calculated between items. The word "USER_ID" was switched into "ITEM_ID" from the previous SQL, so that the similarities between items can be extracted.

```
WITH POW_RATINGS AS (
        SELECT ITEM_ID,
        SUM(RATINGS * RATINGS) AS R2
        FROM U1_BASE
        GROUP BY ITEM_ID
)
SELECT X.ITEM_ID
        ,Y.ITEM_ID AS ITEM2
        ,NX.R2 AS X2
        ,NY.R2 AS Y2
        ,SUM(X.RATINGS * Y.RATINGS) AS INNERPRODUCT
        ,ROUND(SUM(X.RATINGS * Y.RATINGS)/SQRT(NX.R2 * NY.R2),4) AS SIMILARITY
  INTO U1_SIMILARITY_I
  FROM U1_BASE AS X
  JOIN U1_BASE AS Y WITH(INDEX(IX_U1_BASE_01)) ON (X.USER_ID = Y.USER_ID)
  JOIN NORMS AS NX ON (NX.ITEM_ID = X.ITEM_ID)
  JOIN NORMS AS NY ON (NY.ITEM_ID = Y.ITEM_ID)
GROUP BY X.ITEM_ID
        ,Y.ITEM_ID
        ,NX.R2
        ,NY.R2
;
```

Following table shows the similarity of each set of items.

ITEM_ID	ITEM_ID2	X2	Y2	INNERPRODUCT	SIMILARITY
1	1	6143	6143	6143	1
1	2	6143	1168	958	0.3576
1	3	6143	791	682	0.3094
1	4	6143	2293	1402	0.3736
1	5	6143	816	523	0.2336
…	…	…	…	…	…
…	…	…	…	…	…
1682	1335	9	89	12	0.424
1682	1401	9	119	9	0.275
1682	1428	9	159	9	0.2379
1682	1597	9	53	9	0.4121
1682	1682	9	9	9	1

Table 4: Cosine similarities between items in the dataset "u1.base"

4.2.2 User's Movie Rating Prediction

In order to predict the ratings for the item-item collaborative filtering, the weighted average was also calculated by SQL query. We have used the same SQL formula with the user-user item filtering to get the weighted average values. In this case, ten most similar items are selected to get a weighted average value based on their similarities. The SQL query is as follows.

```
WITH U_AVG AS (
SELECT ITEM_ID, AVG(RATINGS) AVG_RATINGS
  FROM U1_BASE
 GROUP BY ITEM_ID
)
SELECT ORG.USER_ID, ORG.ITEM_ID, ORG.RATINGS, CASE WHEN ORG.PREDICTED = 0 THEN A.AVG_RATINGS ELSE
ORG.PREDICTED END PREDICTED
FROM (
SELECT T.USER_ID, T.ITEM_ID, T.RATINGS, COALESCE(P.W_AVG,0) PREDICTED
  FROM U1_TEST T
  LEFT JOIN (
SELECT A.EGO, B.USER_ID
            , ROUND(SUM(B.RATINGS)/COUNT(DISTINCT A.SIM_ITEM),0) S_AVG
            , COALESCE(ROUND(SUM(SIMILARITY*B.RATINGS)/NULLIF(SUM(CASE WHEN B.RATINGS = 0 THEN 0 ELSE
SIMILARITY END),0),0),0) W_AVG --Weighted Average
  FROM (SELECT EGO, RK, SIM_ITEM, SIMILARITY FROM (
            SELECT EGO, DENSE_RANK () OVER (PARTITION BY EGO ORDER BY SIMILARITY DESC) RK, SIM_ITEM,
SIMILARITY
            FROM U1_SIMILARITY_I
            WHERE EGO <> SIM_ITEM) A
            WHERE RK <= 10 --INPUT THE NUMBER OF NEIGHBOURS
            ) A
  JOIN U1_BASE B ON (B.ITEM_ID = A.SIM_ITEM)
GROUP BY A.EGO, B.USER_ID) P ON (T.ITEM_ID = P.EGO AND T.USER_ID = P.USER_ID)) ORG
JOIN U_AVG A ON (A.ITEM_ID = ORG.ITEM_ID)
;
```

The Table below shows the result set of prediction with this method.

USER_ID	ITEM_ID	RATINGS	PREDICTED
279	1	3	3
64	72	4	4
15	924	3	3
171	327	4	3
109	380	5	4
119	526	2	4
223	826	1	3
…	…	…	…

Table 5: Predicted Ratings by Weighted Average with Top 10 similarities in Item-Item Collaborative Filtering

4.1.3 Evaluation

RMSD and RECALL values based on the previous methodology are as shows following table.

	U1.TEST	U2.TEST	U3.TEST
RMSD	1.14	1.0898	1.0744
RECALL	0.43	0.4463	0.4486

Table 6: Evaluations of Item-Item Collaborative Filtering

5. Discussion

5.1 Comparison:

	U1.TEST	U2.TEST	U3.TEST	Average
RMSD	1.1342	1.1409	1.1565	1.1465
RECALL	0.3964	0.3930	0.3894	0.3929

Table 7: User-User Collaborative Filtering

	U1.TEST	U2.TEST	U3.TEST	Average
RMSD	1.14	1.0898	1.0744	1.1014
RECALL	0.43	0.4463	0.4486	0.4416

Table 8: Item-Item Collaborative Filtering

The above table suggests that item-based algorithm performs slightly better performance than user-based .We used different techniques computing 'user-user' and 'item-item' correlations to the 'cosine similarities' and used another different method is "Weighted sum' to obtaining the recommendations using the same k-nearest(10) neighbor approach. The bottleneck is that item-based algorithm is better because it is finds the relationship between the items first rather than relationship between users. Due to not only the huge number dataset, but also most of the values are sparse(null), the user-user algorithm is affected .On the other side, item-item approach find the items that are similar to other items user has liked which is computed as same as user-based techniques and that is increase the scalability and performance of the model due to the relation between the items are relatively static and used less computation(Sarwar et al., 2001).

Furthermore, based on the above insights, we can develop a movie recommendation system with user's ID. we built a SQL procedure by using Item-item model, which displays five movies that the target user will like the most. In this implementation, the dataset "u.data" was used to obtain similarities and train the model.

```
ALTER PROCEDURE PR_GET_USER_RATINGS @target_user_id int AS
WITH U_AVG AS (
SELECT ITEM_ID, AVG(RATING) AVG_RATINGS --replace the value 0 with user's average rating
```

```
 FROM U_DATA
 GROUP BY ITEM_ID
)
SELECT ORG.USER_ID, ORG.ITEM_ID, ORG.MOVIE_TITLE, CASE WHEN ORG.PREDICTED = 0 THEN A.AVG_RATINGS
ELSE ORG.PREDICTED END PREDICTED
FROM (
SELECT @target_user_id USER_ID, T.MOVIE_ID ITEM_ID, T.MOVIE_TITLE, COALESCE(P.W_AVG,0) PREDICTED
 FROM U_ITEM T
 LEFT JOIN (
SELECT A.ITEM_ID, B.USER_ID
            , ROUND(SUM(B.RATING)/COUNT(DISTINCT A.ITEM_ID2),0) S_AVG
            , COALESCE(ROUND(SUM(SIMILARITY*B.RATING)/NULLIF(SUM(CASE WHEN B.RATING = 0 THEN 0 ELSE
SIMILARITY END),0),0),0) W_AVG --Weighted Average
 FROM (SELECT ITEM_ID, RK, ITEM_ID2, SIMILARITY FROM (
            SELECT ITEM_ID, DENSE_RANK () OVER (PARTITION BY ITEM_ID ORDER BY SIMILARITY DESC) RK, ITEM_ID2,
SIMILARITY
            FROM U_SIMILARITY_I
            WHERE ITEM_ID <> ITEM_ID2) A
            WHERE RK <= 10 --INPUT THE NUMBER OF NEIGHBOURS
            ) A
 JOIN U_DATA B ON (B.ITEM_ID = A.ITEM_ID2)
GROUP BY A.ITEM_ID, B.USER_ID) P ON (T.MOVIE_ID = P.ITEM_ID AND T.MOVIE_ID = P.USER_ID)
WHERE NOT EXISTS (SELECT 'X' FROM U_DATA BB WHERE USER_ID = @target_user_id AND BB.ITEM_ID = T.MOVIE_ID)
) ORG
JOIN U_AVG A ON (A.ITEM_ID = ORG.ITEM_ID)
ORDER BY USER_ID, PREDICTED DESC
;
```

Figure below illustrates steps of how the procedure “PR_GET_USER_RATINGS” displays five recommended movies.

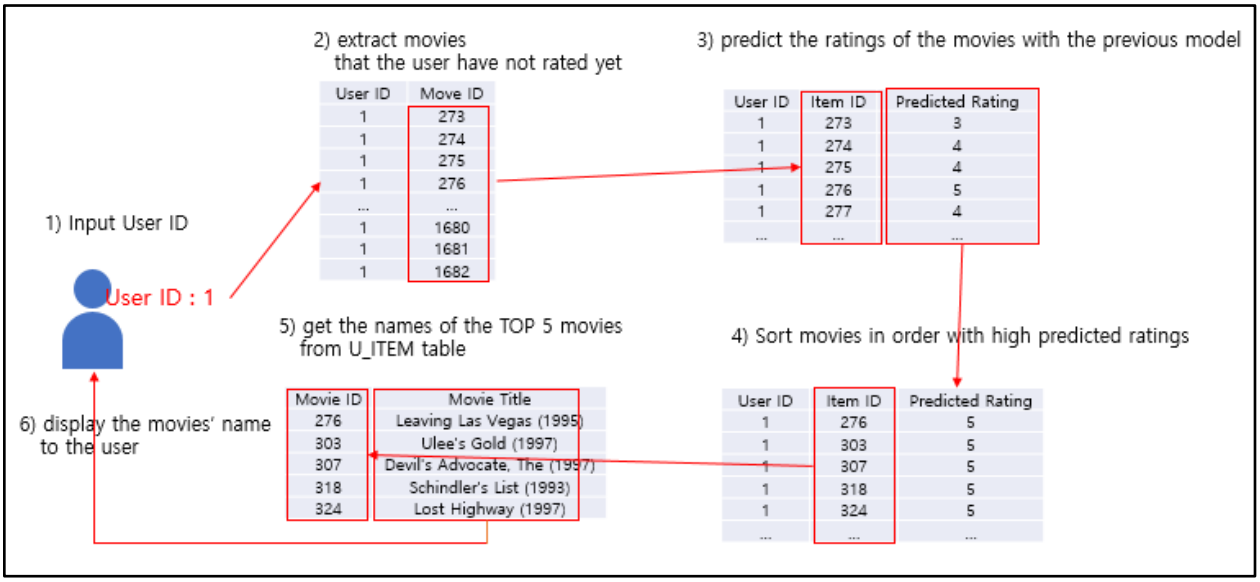

Figure 8: The structure of “PR_GET_USER_RATINGS”

6. Conclusion

In this paper, we have developed and presented an algorithm for movie recommendation system for predicting the ratings of users which never seen before. We have carried out experiments using user-item based CF approaches and then validation by applying RMSD and Recall on ML 100k dataset'.

Furthermore, we have used SQL language on a ML dataset. A potential reduction results have been obtained through these experiments and the item-based CF algorithms outperforms better than user-based CF aspects. This algorithm gives to see the tables of datasets in order to view the 'users and items rating' respect to predicted rating. The item-based CF hold the promise of CF and it can be scalable to large dataset and produce extreme quality recommendation system. And we also used other combination of 'training and test' dataset which is by comparison reduces the prediction rate error slightly. The major limitation of this project is that the recommendation system is implemented using on 100k datasets instead of 20M due to lack of computer power such as RAM and speeds which require 40-50GB of RAM. And other sophisticated, intuitive way of using python due to lack of experience in that programming language.

Our future work will be to make machine learning "support vector machine" algorithm and 'deep neural network' techniques to get more accurate results. Another plan, using "matrix Factorization' algorithm to 'features to item' way to predicting the best movie recommendation system using 20M of ML datasets.

References

1. Dash, A., lin, J., Handali, N.,2018. ‘Recommendation Systems for the Netflix Network’, Stanford@edu,USA.

2. Ghazanfar M. and Prugel-Bennet A., 2014, ‘A Scalable, Accurate Hybrid Recommender System’, School of Electronics and Computer Science University of Southampton Highfield Campus, SO17 1BJ, United Kingdom.

3. Hu, Y., Koren, Y. and Volinsky, C., 2008, December. Collaborative Filtering for Implicit Feedback Datasets. In *ICDM* (Vol. 8, pp. 263-272).
4. Korbut,D. 2017,’Recommendation System Algorithms’, Viewed 1 May 2019,<https://blog.statsbot.co/recommendation-system-algorithms-ba67f39ac9a3>.

5. Krishna Menon A., Sanner S., Sedhain S. and Xie L., 2015, 'AutoRec: Autoencoders Meet Collaborative Filtering, Australian National University, Australia.

6. Sarwar, B.M., Karypis, G., Konstan, J.A., Riedl, J., 2001. ‘Item-based collaborative filtering recommendation algorithms. *Www*, *1*, pp.285-295.

7. Zhang, Y. 2019. ‘Lecturer Slides: Recommendation’, UTS Online Subject 42913, PowerPoint Presentation, UTS, Sydney, Viewed 18 May 2019, <https://online.uts.edu.au/bbcswebdav/pid-3389220-dt-content-rid-45838799_1/courses/42913-2019-AUTUMN-CITY/Week5_recommendation.pdf>