Boğaziçi University
CMPE 492
FINAL REPORT

Project: A recommendation engine for cafés

Denizalp Kapısız, 2012400144
Gözde Berk, 2012400150

Advisor: Prof. Tunga Güngör

Table of Contents

A recommendation engine for cafés (June 2017)

GÖZDE BERK – DENİZALP KAPISIZ
Boğaziçi University

1. Introduction and Motivation

It goes without saying that filtering the abundant information provided to the users is a necessity in this era which makes recommendation systems necessary since a recommendation system is a subclass of information filtering system.[1] A recommender system or a recommendation system is a system that suggests some items such as food, movie, and music to some customers in an application. In other words, it estimates the "rating" or "preference".[2] There are many examples of such systems around the internet. For example, IMDb recommends similar movies to the movie that we look for there.[3] Another example could be YouTube, which lists some similar videos to the left of the page.[4] There are two point of views of this system. From the customer point of view, it is important because it gives the opportunity to experience some new materials that have similarity with the products he/she has been enjoyed. From the provider point of view, it is crucial because it presents a chance to make profit quick by saving time of the customer while he/she looks for an item.[5]

In our project, we will build a recommendation engine for cafés. In the ordering applications such as YemekSepeti, the system recommends some drinks if we do not buy some with the meal.[6] Furthermore, it lists the items that have been sold mostly on the top of the menu in order to speed up the purchase. When it comes to the outside, the employee brings a menu and we order some items. The fact that QR code has been used thoroughly by smartphones nowadays brings us the idea that we can use a smartphone to view menu and to make recommendations as in the ordering applications. In fact, those recommendations will be more reliable than the recommendation that the employee of the café makes because the latter wants to popularize all items in the menu, whereas the former is generated from the past orders of the café, most of which have been rated by some clients or via the information of the current customer.

Finally, our application will improve the sales of the cafés because total satisfaction among the clients will increase by our recommendations. In other words, we increase the probability of enjoy for a customer by narrowing down the choices in the menu. Therefore, the cafés benefiting our app will produce more items that are popular among the past clients, and improve the items that are less liked.

2. State of the Art

Throughout our work, we have encountered a plenty of research papers that have resolved the recommendation problem from many perspectives. Additionally, we discovered an app called QikServe that has a strong association with our project in terms of UI and categorization of the menu.[7]

First of all, collaborative filtering and knowledge-based approaches were used mainly in the research. The major point of the collaborative approach is that a customer will prefer those items that like-minded people prefer.[8] A collaborative filtering recommender system, therefore, makes prediction for a user based on the similarity between the interest profile of that user and those of other users.[9] [10] [11] For the latter approach, we can say that it generates recommendations to a client by consulting its knowledge base of the product domain, and then reasoning what products will best satisfy his/her requirements.[12]

Each filtering approach has both advantages and disadvantages. Knowledge-based filtering does not need prior knowledge and adapts easily to preference changes.[1] In spite of solving well-known cold start problem, knowledge- based approaches also have some drawbacks.[1] It requires knowledge engineering since the system is based on the knowledge of important features of the product.[1] The worst thing about knowledge-based filtering is that the recommendations are static.[1]

On the other hand, collaborative filtering is not only dynamic but also personalized in terms of recommendations resulting from individual past behavior.[1] As collaborative filtering is based on similarity between users or items, past behaviors of similar users are taken into consideration.[1] [13] Therefore, it is also social and active.[1] [13] Collaborative filtering has mainly two different approaches which are memory based and model based. The model approach deals with new users and new items easily.[13] [14] [15] Since, the memory based approach relies on similarity between users, the similarity rate should be calculated for each recommendation. On the other hand, the memory based approach relies on model of users or items based on the user preferences and only the model is queried in the model case and the model is smaller compared to the dataset which results in higher speed.[13] [14] [15] The best thing about memory-based collaborative filtering is simplicity.[15] Moreover, the database is updated easily with the new data.[13] Since there is limited amount of items in a menu, user-based and memory-based approaches are beneficial for our case. Regarding these benefits, we decided on user-based and memory-based collaborative filtering.

While collaborative filtering offers advantages, it has some disadvantages that must be considered beforehand. Main drawbacks are the dependency on human ratings, sparsity and cold start problem.[16] Hybrid approach which is a combination of different filtering techniques is used to solve these drawbacks in general. However, there is no

specific hybrid approach that addresses our case. Therefore, we designed our hybrid approach by combining user and memory-based collaborative filtering and some other techniques.

3. Methods

3.1 General Idea

Let the system has *n users* and *m items*. When the *user x* logs in to the system to order food or/and beverages, our aim is to find the most similar *k users* and recommend items to the *user x* accordingly. The idea is that there are items that are not rated by the *user x.* Therefore, these items are all candidates for recommendation. A rating is estimated for each unrated item via using the rating given for this item by the each of *k users*. The item with the highest estimated rating is recommended to the *user x.*

3.2 Representation

We have two classes which are *users* and *items.* Users have preferences as ratings for certain items.[17] These ratings are represented by a *utility matrix*.[17] Let us call the *nxm* utility matrix *R*. Each element *Rij* in the *R* represents the average rating for the *item j* given by the *user i*. Ratings are between 0 and 5. If the *user i* has not rated the *item j* yet, corresponding *Rij* will be empty.[17]

3.2.1 Menu Assumption

Items fall into five categories which are beverages, appetizers, entrée, main course and dessert. Items will be sorted in terms of categories. In addition, matrix indexing for items will be shaped according to these categories. For instance, first 10 items are appetizers then entrée etc. Categorization is necessary to recommend items from each category easily.

3.2.2 Past Orders Matrix

In order to represent the past orders, we have used a matrix. In this matrix, each row implies an order of a customer. For each order, ordered items are assigned as 1, the other items are assigned as 0. Items that are assigned as 1 have their rankings, the others have no ranking. We divided the columns of the matrix into three sections. First section takes the first column as customer's identification. The remaining columns are divided equally. The

first half corresponds to each item in the menu, and sorted according to the menu assumption. The second half corresponds to rating of each item.

3.3 Data Generation

We have preferred working with randomized data because the datasets in the literature involve too many features to handle. Furthermore, the data gives idea about the recommendation procedure. In order to generate the data, we specified some assumptions. However, we parameterized the number of customers, orders and items to analyze how the recommendation changes with the different parameters.

3.3.1 Assumptions

- There are 100 customers in the population. In order to pick a customer, we take a random number from uniformly distributed [1…100].
- There are 200 orders in total. In order to fill an order, we need to select some items from each category.
- From entrée and dessert, maximum 1 item must be selected each.
- From appetizers, main course and dessert, maximum 2 items must be selected each.
- For the selected items, we pick a random number from uniformly distributed [0…5].

Customers	Main Course						Appetizer				Beverage				Dessert			Entrée			Ranks																			
user36	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	3																	3	
user29	0	1	1	0	0	0	1	1	0	0	0	0	1	0	0	0	0	0	1	0		4	1				4	3					5						1	
user83	1	1	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	1	0	0	3	1					4				5				0			5		
user56	1	1	0	0	0	0	1	1	0	0	0	0	0	1	0	1	0	1	0	0	0	5					3	2						5		0		2		
user22	1	0	1	0	0	0	0	0	0	1	0	0	1	0	0	1	0	0	0	0	0		4							4			5			1				
user16	0	0	1	1	0	0	0	1	1	0	1	0	0	0	1	0	0	1	0	0			1	4				1	0		5				2			5		
user73	1	1	0	0	0	0	1	1	0	0	1	0	0	0	1	0	0	0	0	0	1	2					2	4			4				2					
user88	1	0	0	0	1	0	0	0	0	1	1	0	0	0	0	0	1	0	1	0	2				1					4	3						2		1	
user54	0	1	0	0	1	0	1	1	0	0	1	0	0	0	1	0	0	1	0	0		3			5		4	2			2				5			2		
user28	1	0	0	0	1	0	1	0	1	0	1	0	0	0	1	0	0	0	1	0	1				2		3		2		3				1				3	
user72	1	1	0	0	0	0	1	1	0	0	1	0	0	0	0	1	0	1	0	0	5	1					3	2			2					1		5		
user20	0	0	1	1	0	0	1	1	0	0	0	0	0	1	0	0	1	0	0	0			1	0			0	4						1			4			
user19	1	0	0	1	0	0	1	1	0	0	1	0	0	0	0	0	0	0	0	1	2			4			5	3			5									0
user65	0	1	1	0	0	0	0	1	1	0	0	1	0	0	0	0	0	1	0	0		2	5					4	2			1						1		
user59	0	0	1	0	1	0	0	1	0	1	1	0	0	0	1	0	0	1	0	0			2		5			2		3	5				1			3		
user19	1	0	1	0	0	0	0	1	0	0	1	0	0	0	1	0	0	0	0	1	3		3					4			0				0					3
user25	1	0	1	0	0	0	0	1	1	0	0	0	1	0	0	0	1	1	0	0	1		0					1	4				4				0	5		
user86	1	0	0	1	0	0	0	1	0	1	1	0	0	0	1	0	0	1	0	0	1			2				4		5	1				5			3		
user47	0	1	0	1	0	0	1	1	0	0	0	1	0	0	1	0	0	1	0	0		0		3			4	4				4			2			3		
user32	0	0	1	0	1	0	1	0	0	0	1	0	0	0	0	1	0	1	0	0			4		1		3				2					5		4		
user58	1	0	0	1	0	0	0	1	1	0	0	0	1	0	1	0	0	0	1	0	5			2				4	4				0		5				0	
user54	0	1	1	0	0	0	0	0	0	1	1	0	0	0	0	1	0	1	0	0		3	5							0	2					4		1		
user77	0	1	1	0	0	0	1	1	0	0	1	0	0	0	0	0	0	1	0	0		4	4				3	0			1							4		
user57	0	0	0	0	0	0	0	0	1	0	0	0	0	1	1	0	0	0	1	0									1					5	0				0	
user33	1	1	0	0	0	0	0	0	1	1	1	0	0	0	0	0	1	0	1	0	3	4							1	0	4						3		5	
user71	0	0	0	1	0	1	0	1	0	0	0	0	0	0	1	0	0	0	0	1				2		0		5							1					0
user85	1	1	0	0	0	0	1	0	0	1	0	1	0	0	0	1	0	0	0	1	3	4					2			3		2				5				5
user16	1	0	0	0	1	0	0	0	1	1	0	0	1	0	0	1	0	0	0	1	1				5				4	5			3			0				3
user63	0	0	1	1	0	0	1	0	0	0	0	0	1	0	0	1	0	0	1	0			1	3			3						4			1			5	
user73	0	1	0	0	1	0	1	0	0	0	0	1	0	0	1	0	0	0	0	0		0			1		3					3			0					
user81	0	0	0	0	1	1	1	0	1	0	0	0	1	0	1	0	0	0	0	1					3	0	0		5				5		0					0
user4	0	1	1	0	0	0	0	1	1	0	1	0	0	0	1	0	0	1	0	0		2	1					0	1		1				5			1		
user11	1	0	0	0	0	1	1	1	0	0	0	0	0	1	1	0	0	1	0	0	5					1	4	3						4	2			3		
user38	0	1	0	1	0	0	0	1	1	0	1	0	0	0	0	1	0	0	0	1		0		5				3	1		3					0				3
user31	1	0	1	0	0	0	1	1	0	0	1	0	0	0	0	0	0	1	0	0	0		5				3	3			3							3		
user16	0	0	0	1	1	0	1	0	1	0	0	1	0	0	1	0	0	1	0	0				1	0		0		1			1			1			5		
user26	1	0	0	1	0	0	1	0	1	0	0	1	0	0	0	0	1	1	0	0	5			4			0		4			0					3	1		
user98	0	1	1	0	0	0	0	1	1	0	1	0	0	0	0	1	0	0	1	0		0	0					0	5		1					2			4	
user60	0	0	0	1	0	1	1	0	0	1	1	0	0	0	0	0	0	1	0	0				3		1	3			0	3							1		

For example, the data above was generated from a menu consisting of 3 entreés, 3 desserts, 4 beverages, 4 appetizers and 6 main courses. (note that this is just a part of the data because full version does not fit into the page.)

3.4 Obtaining the Utility Matrix *R*

In order to fill the utility matrix, we need to clear out how many distinct customers are in the past orders matrix, and find their average ratings for all items they have ordered.

Basically, we iterate over rows of the matrix. In each iteration, we certainly encounter one of the two cases; namely, the customer is in the utility matrix or not. In the first case, we add the customer to the utility matrix and assign his/her rankings to the corresponding items. In the second case, we find the row of corresponding customer, and assign the rankings as in the first case; however, if there is an item that has a rank before, then we assign the average of before and current ranks of that item.

3.5 Similarity Measure

The first important question is how to find the most similar *k users*. In other words, how to measure the similarity of users from the utility matrix *R* is one of the main problems.[17] Leskovec, Rajaraman & Ullman (2014) claim that there are mainly two distance measures which are Jaccard distance and Cosine distance.[17] In terms of ratings, one of the main differences between Cosine distance and Jaccard distance is that the former one deals with detailed ratings smoothly whereas the latter one loses important information. [17] In cosine similarity, blanks in *R* are treated as 0 and the cosine of the angle between two users are calculated.[17] "A larger (positive) cosine implies a smaller angle and therefore a smaller distance."[17]

In cosine similarity, missing ratings are assumed to be zero which treats as negative.[18] To eliminate this effect, ratings can be normalized by subtracting the row mean from the each rating in the row.[18] This method is called Centered Cosine similarity which also assumes the missing ratings to be zero but zero is neutral and average for this case because after normalizing, positive values depict like and negative values depict dislike.[18] It handles not only missing ratings but also tough raters and easy raters.[18] Centered Cosine similarity is also known as Pearson Correlation.[18]

3.6 Rating Estimation

The other important question is how to estimate the rating. Generally, average rating is calculated by taking *k users'* rating into consideration. In some cases, also similarity score is used while taking average. However, it ignores the similarity values found by Pearson Correlation and treats all of the *k users* same. Therefore weighting the average rating with the similarity values is a clever solution.[18]

In our cafe case, a user is more likely to taste an item in the menu more than once. Hence, the number of ratings given for an item also plays an important role. The number of ratings given is stored in a matrix with the same logic behind the utility matrix *R*. Let us call this *nxm* utility matrix *T*. Each element T_{ij} in the *R* represents the number of ratings given for the *item j* by the *user i*. If the *user i* has not rated the *item j* yet, corresponding T_{ij} will be zero. As a result, not only the similarity values but also the number of ratings given is taken into consideration. Dependency on human ratings is decreased by using this approach. After the ratings are found for each unrated item, the items with the highest ratings from each category are recommended to the user.

$$r_{xi} = (\sum_{y \in N} t_{yi} s_{xy} r_{yi}) / (k \sum_{y \in N} s_{xy})$$

r_{xi} : the estimated rating for *user x* and *item i*
s_{xy} : the similarity value between *user x* and *user y*
t_{yi} : the number of ratings given by *user y* to *item i*
N : the set of *k users* similar to *user x*

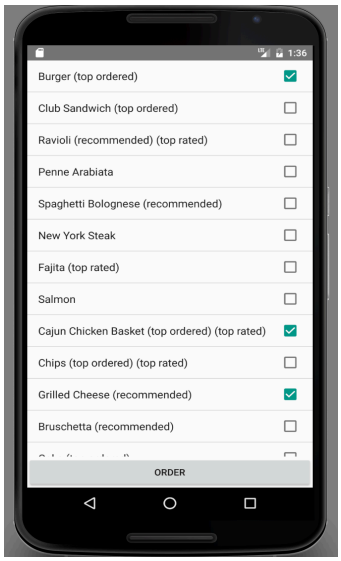
3.7 Overcoming The Drawbacks of Collaborative Filtering

Cold start and sparsity problems are the difficulties with collaborative filtering.[16] Since a menu in a cafe has limited amount of items, sparsity is not a problem in our case. In the case of cold start problem, users are divided into two: *active users* and *passive users*. A *passive user* becomes an *active user* after rating *p* items in the menu. If the the user is passive, average ratings for each unrated item is used via taking the column based averages of unrated items. The unrated item from each category with the highest average ratings are recommended. This top-rated items technique reflects highly rated items. Top-ordered items may also be recommended by using *T* matrix. The number of orders for each unrated item is used via taking the column sum of unrated items. The unrated item from each category with the highest number of orders are recommended. This top-rated items technique reflects popular items. Cold start problem is solved in this way.

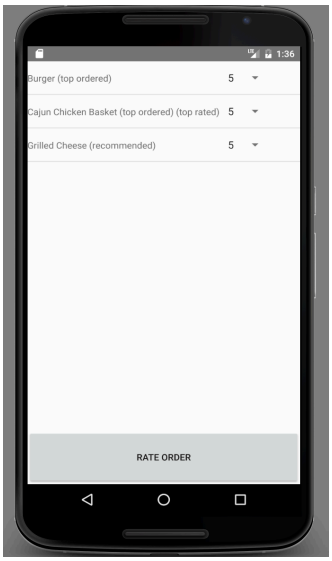
4. Application

Our recommendation system will be used in mobile devices. Each café will have its own QR code which will be read by our application from customer's smartphone. The QR code will open the menu and the customer will enter his/her order while getting recommendations.

In our case, we implemented a simple Android version of the recommendation engine. First, we scan the QR code, then we list the menu by denoting the recommendations on some items, finally we ask the user to rate the items he/she ordered.



Menu with recommendations



Rate the orders in the end

5. Results

In order to understand how a recommendation engine works accurately, one must observe how the recommendations are close to the desired. Mathematically, some accuracy metrics such as Mean Absolute Error(MAE) and Root Mean Squared Error(RMSE) could be defined to calculate the error between the predicted values and the actual values. This brings out two main issues; namely, what metric is used and how the actual values are determined.

Firstly, we preferred to use RMSE as the accuracy metric because it was used in the Netflix Prize contest to rate the proposed algorithms[19]. Furthermore, it indicates the large errors more clearly than MAE[20].

Secondly, there were two approaches to find the actual rating; randomly generate a rating between 0 and 5, or by using the central limit theorem, estimate a rating with the

mean and variance of the previous customers' ratings. The second approach is not reliable because the customer's rating becomes dependent to all other customers; however, in our recommendation, we estimate the rating with the similar customers to current customer. The first approach is not confident because it does not generate a stable rating, compared to the second approach.

5.1 Dataset

In order to handle the issue of determining the actual rating, we decided to work with some datasets. While determining the datasets, we considered that there must be a plenty of users with a plenty of ratings and the number of users must be much larger than the number of items rated. In the light of these criteria, we selected two datasets: Jester Dataset[21] and Random Dataset.

5.1.1 Jester Dataset

In this dataset, over the ratings (-10 to +10) of 100 jokes from 73421 users are collected[21]. However, we picked 1000 users those rated all 100 jokes to observe how the number of user and joke affects the error. From this point, we parameterized the number of users, jokes, and jokes to be removed in order to predict them. Furthermore, we converted the rating to the interval of 0-5. In order to prepare the dataset for the experiments, we removed randomly some jokes for each user. In the end, we conducted some experiments on the Jester Dataset.

5.1.2 Random Dataset

In this case, the dataset was generated randomly. In other words, we picked a random number in 0-5 as a rating for each item for each user. Then, we applied the same procedure, except converting, to the Random Dataset, as in the Jester Dataset, for the experiments.

5.2 Experiments

The experiments were conducted on the datasets explained above via four different measures with different parameters like number of users, total order, number of neighbours and number of recommendations etc. The number of tastes were not taken into consideration because there is no such data in the datasets. Moreover, we wanted to give more generalizable results in terms of applicability to different domains.

5.2.1 Accuracy Measures

Root Mean Square Error, Reverse Root Mean Square Error, Group Binary Accuracy and Pairwise Binary Accuracy were used as error measures.

The error between the actual ratings of the items to be recommended according to our algorithm and their desired ratings was calculated by using common RMSE formula. The accuracy was found by subtracting the RMSE from one.

The error between the ratings of the items to be recommended according to the real data and their actual ratings according to our algorithm was calculated by using common RMSE formula. Since the items to be recommended were found by using the real values, this measure is the reverse of the previous one. Therefore, we called it as Reverse RMSE. The accuracy is again found by subtracting the Reverse RMSE from one.

Actually the aim of the project was not to find the most accurate ratings. The aim was to find the most accurate recommendations. Therefore, we developed another accuracy measure which focuses on the true recommendation rather than its rating. For instance, if an item is found to be recommended with the best rating of 3.0 among all other candidates and the desired rating of this item is 4.0 but actually this item is also has the best rating according to the real values in the dataset, our algorithm is successful regardless of its rating.

Let us say that the number of items to be recommended is *r*. The *r* items to be recommended is found by using the actual ratings and also the desired ratings. We have *r* items which were founded via actual ratings and *r* more items which were founded via desired ratings. The intersection of these two sets are the true recommendations. When the intersection is subtracted from the items founded by actual ratings, false recommendations are obtained. Finally, Group Binary Accuracy was found by dividing the number true recommendations to *r*.

Moreover, from the idea of the Group Binary Accuracy, we developed another accuracy measure. We called it Pairwise Binary Accuracy. The main idea is the same with Group Binary Accuracy but the pairwise comparison of the items in the two item sets were used instead of their intersection. If the items are same, they are labeled as true recommendations. Otherwise, false recommendations. However, this was not a significant and efficient measure because the items to be recommended to the users are not labeled with its rank in our application.

For instance, we want to recommend seven items. According to the our algorithm, items with id 2, 7, 8, 21, 24, 13 and 19 which is ordered in terms of the ratings are recommended. According to the desired ratings, items with id 8, 2, 7, 5, 24, 20 and 13 which is ordered in terms of the ratings should be recommended.

In RMSE, items with id 2, 7, 8, 21, 24, 13 and 19 and their ratings are used. In Reverse RMSE, items with id 8, 2, 7, 5, 24, 20, 13 and their actual ratings are used.

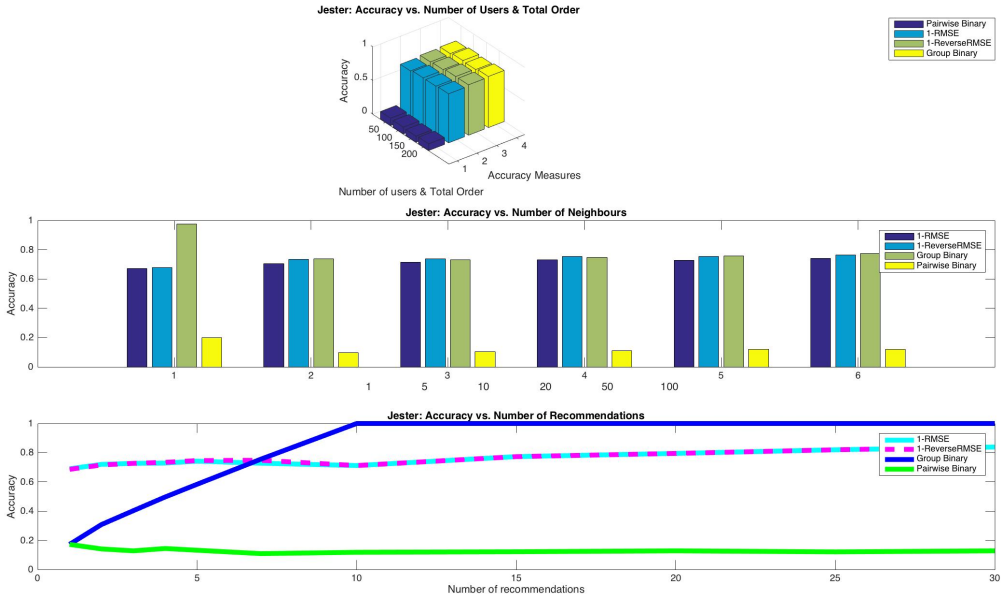
In Group Binary Accuracy, the intersection of two sets which contains items with id 2, 7, 8, 24 and 13 is labeled as true recommendations. So, accuracy is 5/7. In Pairwise Binary Accuracy, only the item with id 24 matches. So, the accuracy is 1/7.

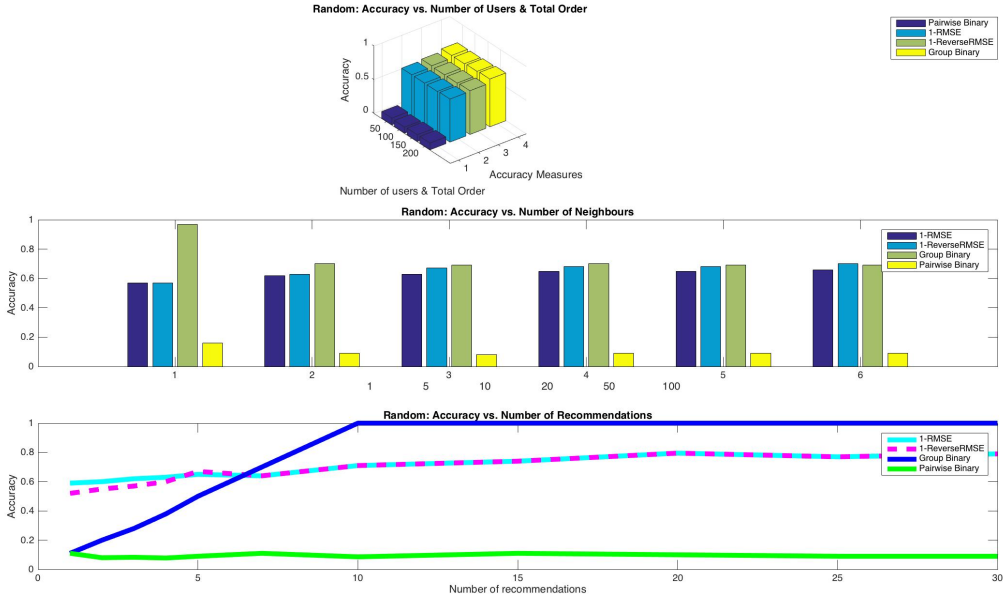
It is clearly seen that, Pairwise Binary Accuracy is not a good measure for our case but we also tested our algorithm with this measure to see its behavior with different parameters and to give idea about the case of reporting the ranks of the recommendations to the users. As it is stated before, we did not provide the predicted ratings of the recommended items to the user. Therefore, RMSE and Reverse RMSE is not vital but gives a general idea regarding the performance. It is clearly seen that the most important and vital measure is Group Binary Accuracy.

5.2.2 What has been achieved?

Here are the base values of the parameters:
Number of Users = 200
Total Number of Items = 30
Total Order = 200
Number of Neighbours = 10
Number of Recommendations = 7

Number of users & total number of items (each user was assumed to order once), total order, number of neighbours and number of recommendations were altered separately. Four different accuracies on two different datasets were calculated and the results were achieved.





It is seen that Group Binary Accuracy becomes 1.0 after 10 recommendations. RMSE and Reverse-RMSE becomes equal after 10 recommendations. This is so important because the equality depicts that all of the recommendations were true but the ratings were a little bit deviated. We did not use the value of 10 as the base case since we wanted to see the effect of the other parameters to the Group Binary Accuracy. As the number of users and total order increase, accuracies increase but there is no significant change. The number of neighbours is correlated with the accuracy. In addition, Pairwise Binary Accuracy is low as it was expected.

6. Conclusion and Discussion

In this project, the aim was recommending items accurately. The performance of our algorithm was measured with two different datasets and four different accuracy measures.

Pairwise binary accuracy does not reflect the real performance of our system. So, its low accuracy is not important for us. Group binary accuracy is the most important one because it measures true recommendation and it has the highest accuracy rates. Remaining ones measures the accuracy of rating estimation. Furthermore, the shortcoming of the collaborative filtering was eliminated. The final result is satisfying regarding our aim.

Our project has an economical potential in terms of increasing the overall sales for a café by increasing the sales of most seller items and improving the least sold items. To some extent, it may accelerate the fact that the cafés will be segmented in terms of their popular items. Let us suppose we have two cafés; namely café A and B. In time, café A becomes famous for its Turkish coffee and café B for its Latté. Here, our app may decrease that time so that both cafés could sell more items on the remaining time.

In addition, simulating a QR code on the tables in the café to activate the menu and ordering items from a smartphone is exciting and innovator. Also, the customer will not need any other application for recommendation.

7. Future Work

iOS version may be developed and payment solution may be added which will create self-order and self-pay application with recommendations. In this case, the need for waiters will decrease. One day the cafes without checkout lines and waiters may spread, who knows. Nothing is impossible. Think Amazon Go.[22]

8. References

1. Dwivedi, Prerna; Chheda, Nikita. “A Hybrid Restaurant Recommender” International Journal of Computer Applications; New York55.16 (2012).
2. Ricci, Francesco; Rokach, Lior; Shapira, Bracha; Kantor, Paul B. “Recommender Systems Handbook” Springer (2011).
3. http://www.imdb.com
4. https://www.youtube.com
5. Dietmar Jannach and Gerhard Friedrich. “Tutorial: Recommender Systems” *International Joint Conference on Artificial Intelligence, Beijing, August 4, 2013.*

6. https://www.yemeksepeti.com
7. https://www.qikserve.com
8. Tran, Thomas. "Combining collaborative filtering and knowledge-based approaches for better recommendation systems." *Journal of Business and Technology* 2.2 (2007): 17-24.
9. Billsus, Daniel, and Michael J. Pazzani. "Learning Collaborative Information Filters." *Icml*. Vol. 98. 1998.
10. Greening, Dan R. "Collaborative filtering for Web marketing efforts." *Recommender Systems, Papers from the 1998 Workshop*. 1994.
11. Resnick, Paul, et al. "GroupLens: an open architecture for collaborative filtering of netnews." *Proceedings of the 1994 ACM conference on Computer supported cooperative work*. ACM, 1994.
12. Burke, Robin. "Integrating knowledge-based and collaborative-filtering recommender systems." *Proceedings of the Workshop on AI and Electronic Commerce*. 1999.
13. http://www.cs.carleton.edu/cs_comps/0607/recommend/recommender/collaborativefiltering.html
http://www.cs.carleton.edu/cs_comps/0607/recommend/recommender/memorybased.html
http://www.cs.carleton.edu/cs_comps/0607/recommend/recommender/modelbased.html
14. J.S. Breese, D.Heckerman, and C.Kadie. "Empirical analysis of predictive algorithms for collaborative filtering." In Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, 1998.
15. Pennock, David M.; Horvitz, Eric; Lawrence, Steve; Giles, C. Lee. "Collaborative Filtering by Personality Diagnosis: A Hybrid Memory- and Model-Based Approach." Uncertainty in Artifical Intelligence Proceedings 2000: 473-480.
16. Su, Xiaoyuan; Taghi, M. Khoshgoftaar. "A Survey of Collaborative Filtering Techniques." Advances in Artificial Intelligence Volume 2009 (2009)
17. Leskovec, Jure; Rajaraman, Anand; Ullman, Jeff. "Mining of Massive Datasets." Cambridge University Press (2014).
18. http://www.mmds.org/
19. http://www.netflixprize.com/community/topic_1537.html
20. https://www.kaggle.com/wiki/RootMeanSquaredError
21. Eigentaste: A Constant Time Collaborative Filtering Algorithm. Ken Goldberg, Theresa Roeder, Dhruv Gupta, and Chris Perkins. Information Retrieval, 4(2), 133-151. July 2001.
22. https://www.amazon.com/b?node=16008589011