An Empirical Study of Collaborative Filtering Algorithms For Building a Diet Recommendation System

Supervisor: Hossain Arif

Assistant Professor

Submitted By:

Ashique Mohaimin Ornab (13201080)

Sakia Chowdhury (14101252)

Seevieta Biswas Toa (14101003)

Department of Computer Science and Engineering
BRAC University

Submitted on: 26th December 2017

DECLARATION

We declare that, this thesis report is our own work and has not been submitted for any other degree or professional qualifications. All sections of the paper that use quotes or describe an argument or concept developed by another author have been referenced in the reference section.

_____

Hossain Arif

Supervisor

_____ _____ _____

Ashique Mohaimin Ornab | Sakia Chowdhury | Seevieta Biswas Toa

Author | Author | Author

ACKNOWLEDGEMENTS

All thanks to Almighty ALLAH, the creator and the owner of this universe, the most merciful, beneficent and the most gracious, who provided us guidance, strength and abilities to complete this research.

We are grateful to our supervisor, Hossain Arif, for his immense support and valuable ideas throughout the work. He made us to get out of our comfort zones and to push the limit. Without his guidance, it would have been impossible to get introduced to the amazing research subject.

We are thankful to Kalyan Banik, alumni of BRAC University for sharing his knowledge and experience with recommendation systems and Apache Spark.

Finally, we would like to express our sincere gratefulness to our beloved parents, brothers and sisters for their love and care.

ABSTRACT

The purpose of this research is to study the different techniques that can be approached in order to build a recommendation system. Here, we have analyzed the different approaches between two different collaborative filtering algorithms in perspective of a food diet recommendation system. A food recommendation system that will help people to choose their daily meal just the way we select movies to watch from suggestions in Netflix or add a friend in Facebook when the suggestion pops up in our home page. Sometimes people get bored of having the same food items on regular basis hence in order to help them get rid out of this monotonous lifestyle, we have proposed a diet recommendation system. In this paper, we first give you some basic information about what recommendation system is, and then we talk about the two collaborative algorithms and finally tell you what kind of approaches we have used to build a diet recommendation system.

Keywords: collaborative filtering, cosine similarities, matrix factorization, alternating least squares (ALS), recommendation system

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Motivation

We have noticed some people who have to follow strict diet suggested by their nutritionist in order to maintain a healthy life. Sometimes these people become bored of having the same type of meal over and over again. In order to change this monotonous food consumption we have came with an idea that will help them change their taste of food and also get rid out of the boredom of consuming the same kind of meal sets every day. We have implemented collaborative filtering algorithms to recommend with an alternative meal set for such people for their fixed meal set suggested by their nutritionists.

1.2 Workflow

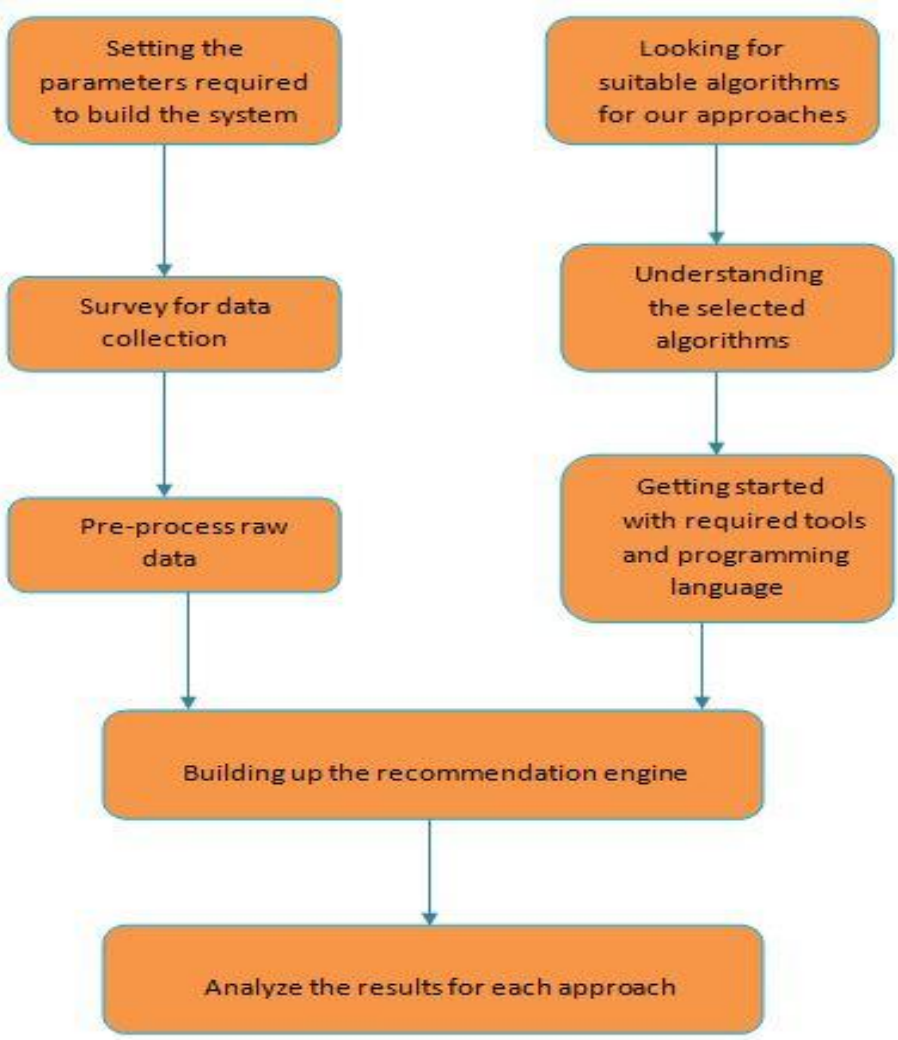


Figure 1.1: Workflow of our research

At first we have set the parameters that are essential for this recommendation system like meal sets names, ratings etc. Then we did a survey for collecting the food names and ratings from different groups of people, pre-processed the raw dataset to fit in our system. After that we have searched for techniques that can be applied to build the system and understood the algorithms. Finally we have merged all these together in order to build the recommendation system.

1.3 Thesis Orientation

Chapter 2 describes the similar works done by other authors.

Chapter 3 describes about recommendation system.

Chapter 4 describes about our proposed model.

Chapter 5 describes how the recommendation system works

Chapter 6 describes the result and analysis

Chapter 7 includes the conclusion and future works

CHAPTER 2

LITERATURE REVIEW

2.1 Literature Review

Xiaocheng Li et al. [1] proposed a recommender system which uses web-based data mining and the system recommends healthy meals to the users. This is a recommender system especially for health related issues that analyses health conditions and food intake of a user. They suppose that there is a website where users had been ordering different food items of different recipes and from the recommendation had collected all the users' information using data acquisition. They had use data mining algorithms like clustering, classification, association rules to extract the necessary user data. And finally they recommend a user a healthy diet using their personal information like their health condition etc.

This paper especially gave us an idea in building such recommendation engine that can help normal people to get out of their monotonous food habit and taste something different which is similar to that of their fixed food diet. They said they would have collected data of recipes from any kind of food ordering website. Since we could not find any relatable dataset for our approach we collected meal sets from different people. Likewise we have build up a model that suggests a user a different meal set for him or her based on his or her previous likeness on other meal sets.

Agapito et al. [2] proposed a personalized suggestion system that will monitoring a user's health profile and recommend him or her a healthy food meal. The profile of the user had been build based on real-time questionnaires by medical doctors and produced by users. The system has been build up by their self made algorithm which focuses on users' profile and their health status and suggest meals based on that. The result of this system is typical Calabrian foods are

suggested by DIETOS to the users and that in 3 ways: i) according to the user's health profile spontaneously foods are suggested; ii) showing the place on a map where the Calabrian foods are made; iii) displaying the nutritional properties for each products stored in the database, benefits and side effects on pathologies and specific health conditions. [3] This paper focused on designing on recommender system which assists the daily routinary diet selections depend on some nutrition guidelines. It takes the user's profile, food taken by and nutrition database and some additional knowledgebase. They have used a knowledge-based framework to build up the recommendation engine. It consists of two forms of knowledge ontology and rules. Ontology-based knowledge contains information about users' profiles and food structure details whereas rule-based knowledge represents decision model in order to show the recommendation system's results. They have used OWL to process ontology data and mapped those on a database using RDF model. The knowledge base data had been actually manipulated by Jena API. Here the system provides food recommendation based on a user's BMI and his or her personal preference of any food item or based on any health issues he or she has. The result given by this system is list of food items' names, type of dish, and process of type of preparing the food item, energy of the food item and ingredients of the food. This paper [4] focused majorly focus on two initial dimensions of food recommendations: calories intake by the person and the calories which is left unused for that person. Neha Gaur et al. proposed a model that used information taken from the university's students belonging to the age group 18-24.This information includes the basic details of the users like age, weight, height, lifestyle, disease (if any) and food taken by that user till evening. This information is given as input to calorie and BMI calculators for calculating the BMI and calorie consumed by that person, it is based on the information of a person defined above which will result in the. BMI Calculator-BODY MASS INDEX calculator is used to

calculate the total fat of the body on the basis of weight and height. After calculating these and taking into accounts the user's BMI, DNA and genetic disease - a suitable recommendation of food is given to the user that will be beneficial for his or her health.

Wahidah et al. proposed a personalized diet suggestion framework particularly for cancer patients to offer assistance patients oversee their everyday food intake [5]. The proposed framework coordinating the data mining strategies of case-based reasoning, rule-based reasoning and genetic algorithm. Case-based reasoning is utilized to recommend a set of count calories plans taken from the cases existing in the framework, while rule-based reasoning is utilized to channel out insignificant cases from the framework and select the most suitable case to be recommended to the understanding. The genetic algorithm procedure guarantees that the diet menus proposed are customized agreeing to each patient's individual wellbeing conditions. The yield of the diet plan framework is in the shape of a list of particular wholesome values to be taken, and a menu suggestion proposing genuine dishes for the understanding. The paper by Eghbali et al. [6] examined the human count calories issue in the fuzzy environment. The approach bargains with multi-objective fuzzy direct programming issue utilizing a fuzzy programming procedure for its arrangement. A few thinks about too proposed computation on count calories arranging or eat less issue cases. The research problem was formulated as a linear multi-objective fuzzy programming problem (MOFLPP) with mixed constraints where right hand side of the constraints are fuzzy numbers for which suitable solution is presented. Using Bellman and Zadeh's fuzzy decision-making process, the MOFLPP is converted into an equivalent crisp LPP. Then, it is solved by simplex method. Next, they also considered MOFLPP with coefficients of objective as well as constraint functions where right hand sides of constraints are Triangular Fuzzy Number(TFN). Converting the problem into an equivalent crisp non-linear

programming problem, it is also solved by fuzzy decisive set method. Moreover, Human Diet Problem was assumed in the form of a fuzzy linear programming with two objective functions. All coefficients were assumed as triangular fuzzy numbers in this model. The consideration by Sufahani and Ismail [7], was to extend the current information in menu arranging and count calories issues, fulfilling the dietary prerequisites and serve a assortment of nourishment serve each day. The main goal of this paper is to build a model that can provide affordable and proper nutritious meals for the boarding school children. The authors have implemented Delete-Shuffle algorithm for building up this system. The result of this model is a 7 days meal sets for the children which includes different food items and beverages for different days that will meet up the daily nutrients required for the children aged 13-18 years old.

2.2 What is a Recommendation System?

In any situation where a user interacts with a very large catalogues of items- these items can be products in Amazon, movies in Netflix, music from Pandora's catalogue or it can be news items from Google News. What really matters is that there are hundreds, thousands or millions of items which is a really large catalogue and the user is interacting with this catalogue. There are two ways in which users can interact with these kinds of large item catalogues.

The first is search - the user knows what they are looking for and the user searches for the precise item in the catalogue. But when the catalogue is too large, the user might not know well exactly what he or she is looking for. And this is where recommendations come in. The system recommends to the user certain items that they think the user will be interested in, based on what they know about the user.

Now why do we need such recommendations? The key that made recommendation so important and why recommendation system developed so much in the last 10 or 20 years is that we moved from an era of scarcity to an era of abundance. For example, when we used to go out for shopping 20 years ago and we went to a local retailer and we would find a certain number of items on the shelf of the local retailer. Even if the retailer is as large as Walmart, here shelf space is a scarce commodity for local retailers. It limits the number of items that the retailer can carry shelf space is expensive so a retailer can only carry a limited number of shelves. But once internet developed the web enables near-zero-cost dissemination of information about products. This means we can have many more items than ever before. There will not be any limitation for shelf space on the number of products. This is why the number of products available in Amazon is much more than any physical retailer. This gives rise to a phenomenon called the long tail.

Long tail means the most popular items available in both retail and online are more purchased by users than the items that are only available in online. But there can be millions of items in online with which the users may not be familiar with. The users may not be able to figure out what they actually need and this is where recommendation engines come.

Recommendation engines work for many items for examples books, movies, music etc and also for finding people in Facebook, LinkedIn, Twitter. There is an anecdote where you can know how recommendation engine is important. Many years ago there were a book published called ‘Touching the Void’. The book was not very popular and was only bought by few. And a few years after this book was published another book named ‘Into Thin Air’ which turned out to be very popular among people. Amazon noticed a few people who bought ‘Into Thin Air’ also bought ‘Touching the Void’ so they started recommending ‘Touching the Void’ to the people who already bought ‘Into Thin Air’. This made people to buy ‘Touching the Void’. Ultimately

‘Touching the Void’ became very popular among people [8]. And this is how recommendation system can enlighten people with gems which they are not familiar with or which is unknown to them.

2.3 Types of Recommendation Systems

The simplest and the oldest type of recommendation is ‘editorial or hand curated’ – this is the one where items are marked as favorites or essential items rated by the staff themselves in any website or even on home pages of most popular websites including any product websites you can see editorial picks that are picked by editorial staff to feature on the home page. The drawback of this recommendation system is that it is done entirely by the staff of the website and there is no input from the users of the website.

So when we go beyond editorial recommendations the next simple thing we can do is ‘simple aggregates’. On many websites we can see a list of top ten or most popular or most recent for example if we go to YouTube we can see the most popular videos for instance. So these are simple aggregates which sort of take into account the user activity to make recommendations to other users. But these recommendations don’t depend on the user. They only depend on the aggregate activity of a lot of other users.

The third and the most interesting kind of recommendation to us is a recommendation that is tailored to individual users. For example, book recommendations tailored to our taste or movie recommendation based on the movies we have watched previously. And we will be focusing this kind of recommendations. We will be talking about recommendation system that works based on users’ previously likeness and unlikeness with other users. These types of recommendations are usually done by using Collaborative Filtering.

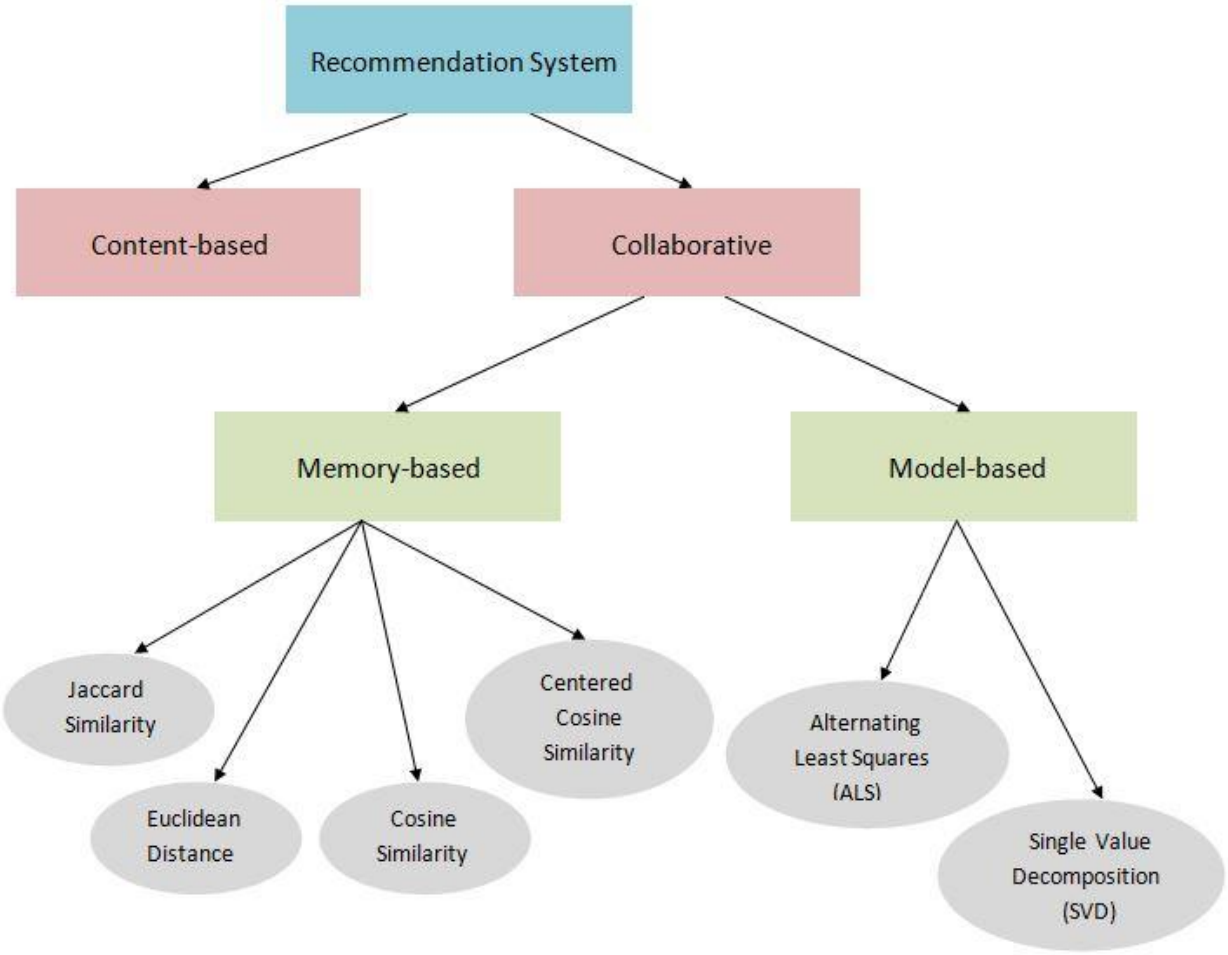


Figure 2.1: Types of Recommendations and their respective approaches

Recommender systems can be divided mainly into two types - Content Based and Collaborative Filtering. There is another type called Hybrid which is usually done by the combination of Content-based and Collaborative.

People have been sharing ideas and opinions for centuries and that is the origin of what we call Collaborative Filtering. It is a process that evaluates and filters people's likes and dislikes (opinions) for a particular product. Collaborative Filtering is a common and popular

recommendation algorithm that predicts and recommends based on past behavior of the system users or ratings given by them.

Collaborative Filtering recommends item based on user's attitude towards it. Content-based recommender systems focus on the attributes of the items and give you recommendations based on the similarity between them.

In our thesis, we have used user based collaborative filtering instead of content based filtering. Collaborative Filtering can be divided into Model-Based Collaborative Filtering and Memory-Based Collaborative Filtering. We have implemented Model-Based Collaborative Filtering using ALS and Memory-Based Collaborative Filtering using Cosine Similarity.

CHAPTER 3

PROPOSED WORK

3.1 Implementation of Memory-based Collaborative Filtering using Cosine Similarity

This section describes about our implementation of Memory-Based Collaborative Filtering using Cosine Similarity. And in the following sections we are going to discuss why we have chosen Cosine Similarity.

Memory Based Collaborative Filtering can be divided into two main sections-

1. User-User Collaborative Filtering

There are four simple steps to implement user-user collaborative filtering-

I. Take an UserX

II. Find similarity values between UserX and all the other users based on similarity of ratings. For example, in our user-item matrix User1 has rated bread|omlette a 4 and User3 has rated bread|omlette a 5. Hence bread|omlette is a healthy meal for both users. So we can assume that User1 and User3 are users with similar health issues and they will have a high similarity value.

III. Find similarity values between all users.

IV. Making predictions - estimate what UserX's rating of items would be that UserX did not rate. For example, in our user-item matrix User1 did not rate bread|fruit jam|green tea. So we can assume whether this meal set would be good for User1's health by using the ratings that other users gave to this meal set and the similarity value between those users and User1.

We have found user-user similarity values in our work. But we are yet to find the predictions. This is one of our future works.

2. Item-Item Collaborative Filtering

We have mainly implemented the Memory-Based Collaborative Filtering approach of our Recommender System using Item-Item Collaborative Filtering. We did this by-

I. Take an Item X (a meal set)

II. Find the similarity values between ItemX and all the other items based on similarity of ratings. For example in our User-Item Matrix, Item1101(bread|omlette) and Item1102(omlette|boneless chicken) are both rated by User1. User1 rated bread|omlette a 4 out of 5 and rated omlette|boneless chicken a 5 out of 5. Since the same user gave a SIMILAR RATING to both bread|omlette and omlette|boneless chicken, we can assume that these two meal sets are SIMILAR in terms of nutrition so they will have a high similarity value. Hence they can act as a substitute of one another.

III. Find similarity values between all the items.

IV. Lastly, take a meal set as an input from the user and recommend him/her the top similar meal sets which he/she can eat as a substitute. The output is arranged in a descending order from the MOST SIMILAR meal set to the LEAST SIMILAR meal set.

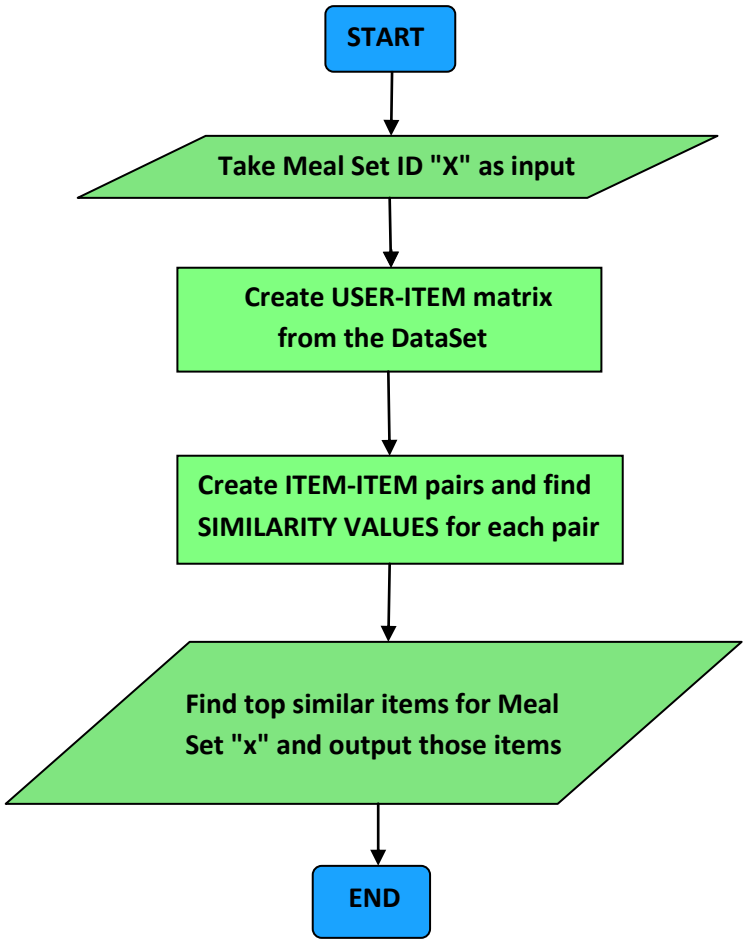


Figure 3.1: Flowchart of Memory-based Recommendation System

Creating the user-item matrix and finding the similarity values are the most important steps in the flowchart above. I have elucidated these two steps.

CREATING USER ITEM MATRIX

Firstly, we created a USER-ITEM MATRIX from our dataset. Since in our survey 10 users rated 30 items, our USER-ITEM MATRIX is a 10X30 matrix.

Meal Sets (Items)

Users	bread|omlette	omlette|boneless chicken	bread|fruit jam|green tea	bread|low fat milk|poached egg	cereal|low fat milk	cereal|low fat milk|strawberry	oats|milk	cereal|milk|almonds	bread|omlette|green tea	parata|mixed vegetable curry	rice|mixed vegetables	rice|chicken|lentils	vegetable oats	rice|beef|fish	rice|boiled vegetables|fish curry	chicken salad|green tea	chicken corn soup	flour bread|mixed vegetables|salad	rice|stir fry spinach|lentils	rice|boiled egg|mixed vegetables	oats|apple|milk	boiled vegetables|omlette	bread|boiled egg|orange juice	stir fry vegetables|orange juice	bread|milk|boiled egg	rice|chicken curry	cereal|milk|banana	corn soup|stir fry vegetables	flour bread|chicken curry	flour bread|mixed vegetables
	1101	1102	1103	1104	1105	1106	1107	1108	1109	1110	2201	2202	2203	2204	2205	2206	2207	2208	2209	2210	3301	3302	3303	3304	3305	3306	3307	3308	3309	3310
1	4	5																												
2					5			5					5	4			3		5			3	2			1		3			
3	5					2																								
4	1					4																								
5												2			1			5	3				5	3	1				2		3
6			5			4	1		1	5		1	5		1		3			3	5			3	2				3	
7			1			1	4	5		2				3		4		3	3			5	2	5	1	3		1		
8			2				5	3				2	4		2	2	2				4		1				5			1
9											1			5		5		3	5			3			2	2		2		2
10				2			5	2				3	3			2			4	1	4		3		5		1		5	

Figure 3.2: User-Item Matrix

In the figure above, ROWS are the users and COLUMNS are the items (Meal Sets). This matrix illustrates all the ratings given by each user to the meal sets they consume. The higher the rating, the healthier is the food for them.

For example, User1 gave Item1101 (bread|omlette) a 4 out of 5 and gave Item1102(omlette|boneless chicken) a 5 out of 5. All the ratings are in the range of 1 to 5.

FINDING SIMILARITY

This is the most significant part of our work.

In Figure 3.2, Item1101 and Item1102 are both rated by User1. User1 gave a 4 to Item1101 and gave a 5 to Item1102. So we can assume that both Item1101 and Item1102 are SIMILAR ITEMS since they are both rated closely by the same user. User1 liked both the items and hence gave a high rating.

On the other hand, Item1101 and Item1105 are both rated by users 3 and 4. User3 gave a 5 to Item1101 but only gave a 2 to Item1105. User4 gave a 4 to Item1105 but only gave a 1 to Item1101. User3 really liked Item1101 but really disliked Item1105. User4 really liked Item1105 but really disliked Item1101. Since both the users gave very dissimilar ratings to both the items, Item1101 and Item1105 must be very DISSIMILAR ITEMS.

Our primary goal is to find the best technique to capture this intuition. The first technique that we tried is Jaccard Similarity –

TECHNIQUE 1 : JACCARD SIMILARITY

In Jaccard Similarity, the items are illustrated as points inside two intersecting sets. In the User-Item matrix of Figure1, Item1101 and Item1102 are both rated by only 1 user and Item1101 and Item1105 are rated by 2 users.

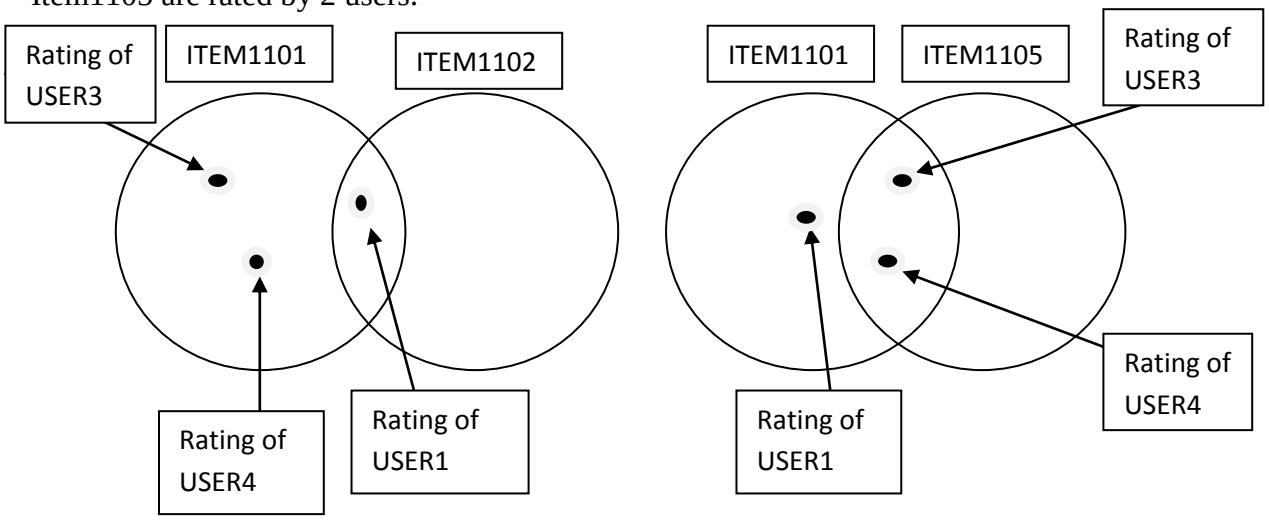


Figure 3.3: Jaccard Similarity

$sim(x,y) = |R_x \cap R_y| / |R_x \cup R_y|$

$sim(1101,1102) = 1/3$

$sim(1101,1105) = 2/3$

$sim(1101,1105) > sim(1101,1102)$

But this does not capture our intuition that we have established. We wanted to mathematically prove that Item1101 and Item1102 are more similar than Item1101 and Item1105. The problem with this approach is that it does not take the RATING VALUES in the calculation. It just takes the NUMBER OF USERS who has rated both the items. Item1101 and Item1105 have higher

similarity because these two items are rated by 2 users - User3 and User4. On the other hand, Item1101 and Item1102 have lower similarity because they are rated by only 1 user - User1.

The second technique we checked is by finding Euclidean Distance.

TECHNIQUE 2 : EUCLIDEAN DISTANCE

A very common way to find the distance between two points is known as Euclidean Distance. TWO ITEMS are considered as TWO POINTS in an n-dimensional space and the distance between them is the Euclidean Distance. The smaller the distance, the more similar the two items are. If the number of users is n then the points of the two items will be in n-dimensional space.

Firstly, I will explain how to calculate Euclidean Distance between two points on a 2 dimensional space. To do that, I have created a random 2X2 User-Item Matrix where two users have rated two items. Since there are two users, point of Item1101 and point of Item1102 are in 2 dimensional spaces (2 axes).

	Item1101	Item1102
USER 1	2	4
USER 2	1	5

Table 3.1: Euclidean User-Item Matrix

Below, I have plotted the graph of the above user-item matrix for visual representation.

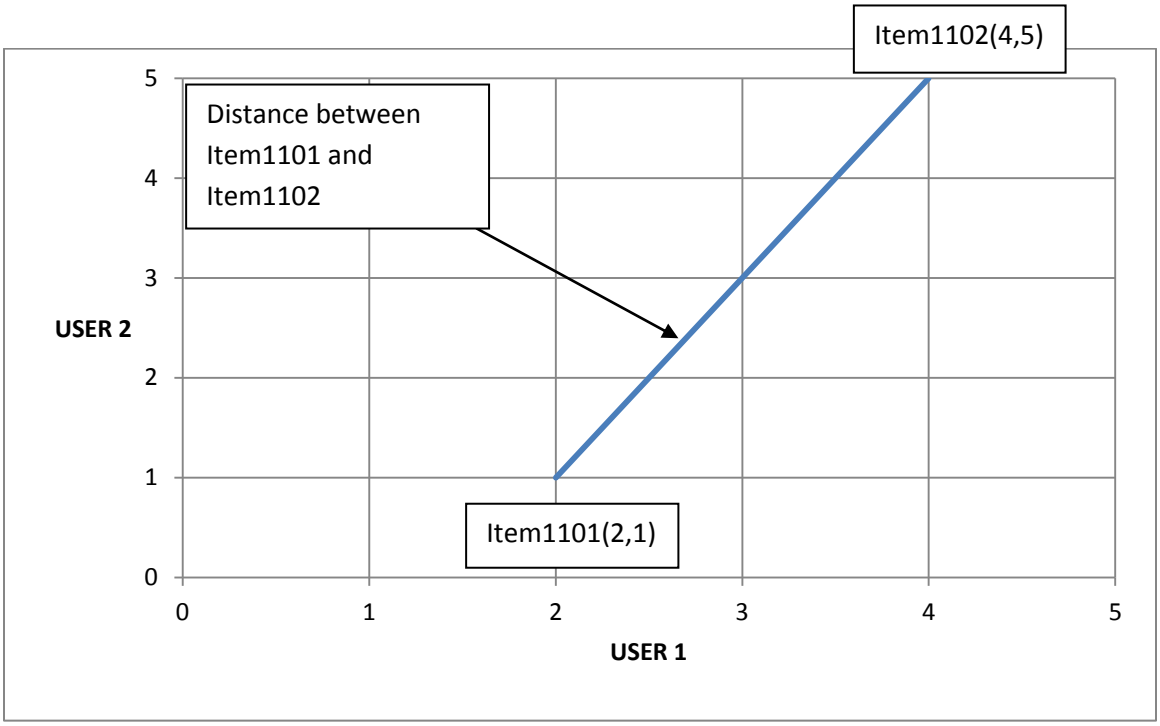


Figure 3.4: Euclidean User-Item Graph (plotted using Table 3.1)

This is the formula for finding Euclidean Distance -

$$dist = \sqrt{\sum_{k=1}^{n}(P_k - Q_k)2} \quad \text{..... (1)}$$

n = number of dimensions

P_k = kth dimension of Point P

Q_k = kth dimension of Point Q

Point Item1102 = (4,5)

Point Item1101 = (2,1)

Distance(Item1101,Item1102) = $\sqrt{(4-2)^2+(5-1)^2} = 2\sqrt{5}$

Now, I will delineate how to calculate the Euclidean Distance between two points in a n-dimensional space using our USER-ITEM Matrix in Figure 3.2.

For example in Figure 3.2, we want to find the Euclidean Distance between Item2206 and Item2207. I have provided the part of our User-Item Matrix in Figure 3.2 at the right showing the ratings of Item2206 and Item2207 for better illustration. Imagine Item2206 and Item2207 as two points in 10 dimensional spaces (as there are 10 users) or in a 10-dimensional graph with 10 axes.

Point Item2206 = (0,3,0,0,0,0,4,2,5,2)

Point Item2207 = (0,0,0,0,5,3,0,2,0,0)

Distance(Item2206,Item2207)

=

$$\sqrt{(0-0)^2+(3-0)^2+(0-0)^2+(0-0)^2+(0-5)^2+(0-3)^2+(4-0)^2+(2-2)^2+(5-0)^2+(2-0)^2}$$

$=2\sqrt{22}$

Finally, let us calculate whether Euclidean Distance captures our intuition that similarity(Item1101,Item1102) > similarity(Item1101,Item1105).

In case of Euclidean Distance, a small distance means more similar items and a large distance means less similar items.

I have provided the part of our User-Item Matrix in Figure 3.2 at the right with all the ratings of Item1101, Item1102 and Item1105.

	1101	1102	1105
1	4	5	
2			
3	5		2
4	1		4
5			
6			
7			
8			
9			
10			

Point Item(1101) = (4,0,5,1,0,0,0,0,0,0)

Point Item(1102) = (5,0,0,0,0,0,0,0,0,0)

Point Item(1105) = (0,0,2,4,0,0,0,0,0,0)

Distance(Item1101,Item1102) =

$$\sqrt{(4-5)^2+(0-0)^2+(5-0)^2+(1-0)^2+(0-0)^2+(0-0)^2+(0-0)^2+(0-0)^2+(0-0)^2+(0-0)^2}$$

$= 3\sqrt{3} = 5.196$

Distance(Item1101,Item1105) =

$$\sqrt{(4-0)^2+(0-0)^2+(5-2)^2+(1-4)^2+(0-0)^2+(0-0)^2+(0-0)^2+(0-0)^2+(0-0)^2+(0-0)^2}$$

$= \sqrt{34} = 5.831$

Distance(Item1101,Item1102) < Distance(Item1101,Item1105)

Therefore, Similarity(Item1101,Item1102) > Similarity(Item1101,Item1105)

Hence, it successfully captures our intuition.

Although it successfully captures our intuition but still we have implemented our Memory Based Collaborative Filtering using COSINE SIMILARITY not EUCLIDEAN DISTANCE.

TECHNIQUE 3 : COSINE SIMILARITY

Euclidean Distance finds the distance between two points. Cosine Similarity finds similarity by calculating the ANGLE between two VECTOR LINES. TWO ITEMS are considered as TWO

VECTOR LINES in an n-dimensional space and the ANGLE between them is the COSINE ANGLE. The smaller the angle, the more similar the two items are. If the number of users is n then the VECTOR LINES of the two items will be in n-dimensional space.

If you recall from trigonometry, the range of the cosine function goes from -1 to 1. Some important properties of cosine to recall:

1. Cosine(0°) = 1
2. Cosine(90°) = 0
3. Cosine(180°) = -1

With the cosine similarity, we are going to evaluate the similarity between two vectors based on the angle between them. The smaller the angle, the more similar the two vectors are.

Firstly, I will explain cosine similarity in a 2 dimensional space. To do that, I have created four random 2X2 User-Item Matrices where two users have rated two items. Since there are two users, vector line of Item1101 and vector line of Item1102 are in 2 dimensional spaces (2 axes).

MATRIX 1 -

	Item1101	Item1102
USER 1	5	2
USER 2	3	5

Table 3.2: Cosine User-Item Matrix-1

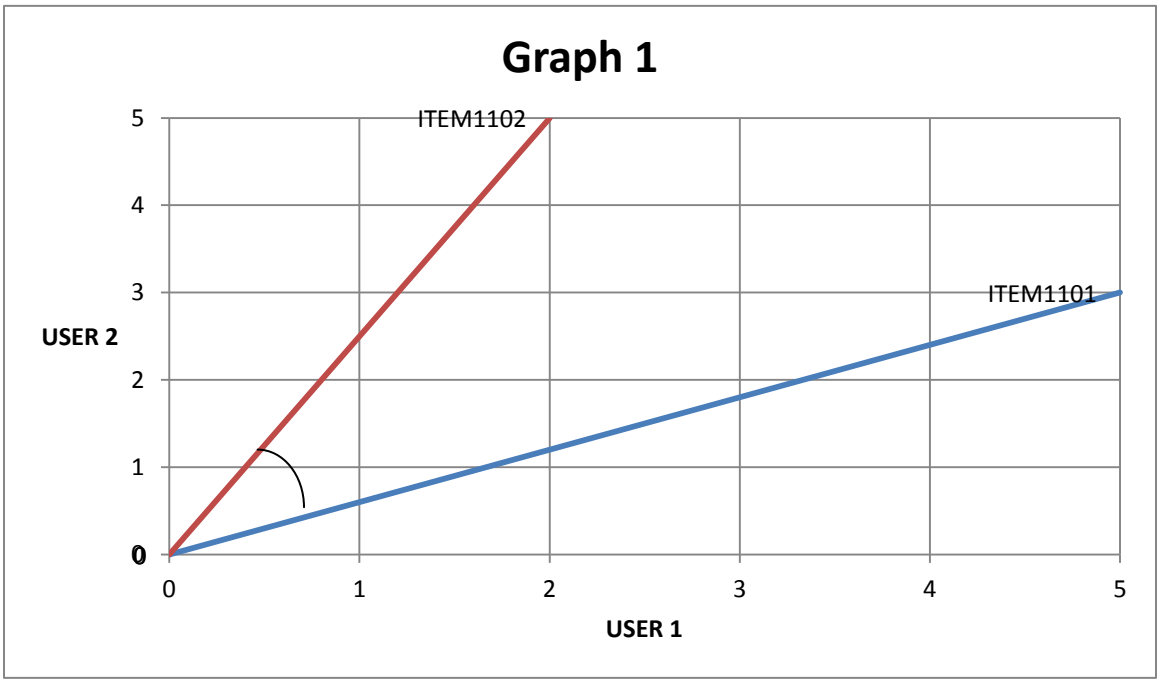


Figure 3.5: Cosine User-Item Graph – 1 (plotted using Table 3.2)

In Graph 1(Figure 3.5), User 1 gave a positive rating to both the items and User 2 gave a positive rating to both the items. If ITEM1101 and ITEM1102 are meal sets, then both the items are healthy for USER1 (that is why the user gave a positive rating). Both the items are also healthy for USER2 (that is why the user gave a positive rating). As a result, ITEM1101 and ITEM1102 must have similar nutritional values. Since both the users gave positive ratings to both the items, ITEM1101 and ITEM1102 must be VERY SIMILAR. That is why both the lines of ITEM1101 and ITEM1102 are pointing at the same direction and the angle between them is VERY SMALL. This angle is close to 0° so the cosine function of this angle is close to 1.

MATRIX 2 -

	Item1101	Item1102
USER 1	5	-3
USER 2	3	5

Table 3.3: Cosine User-Item Matrix-2

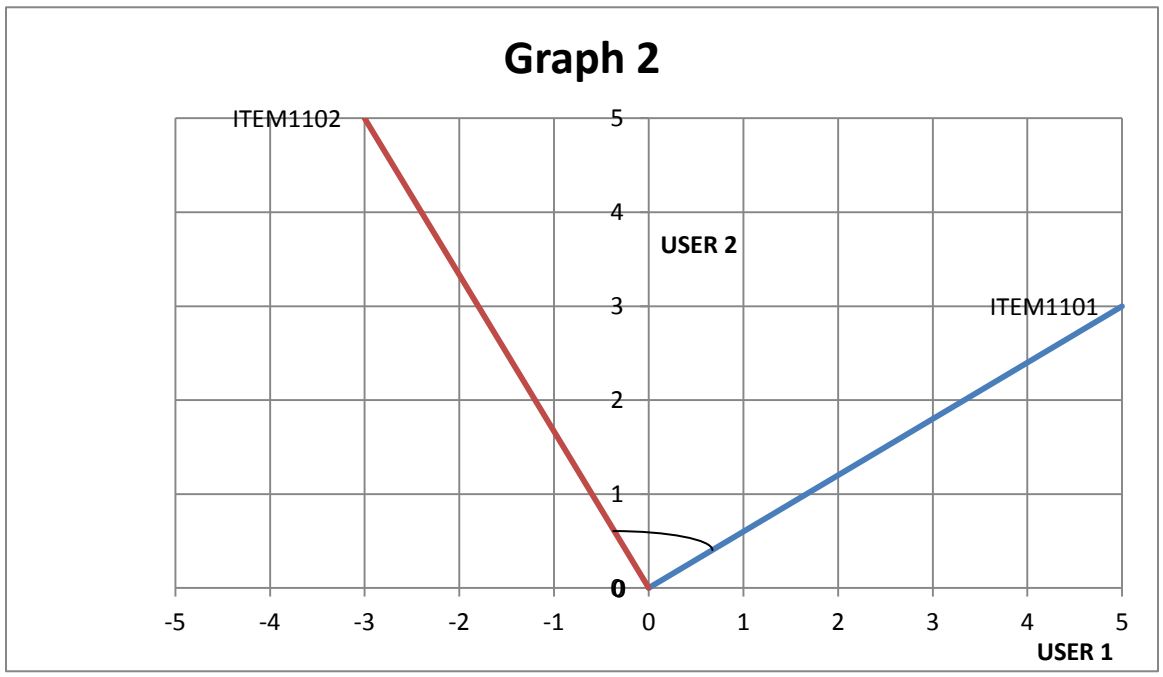


Figure 3.6: Cosine User-Item Graph – 2 (plotted using Table 3.3)

In Graph 2(Figure 3.6), USER2 gave both the items a positive rating. But USER1 gave ITEM1101 a positive rating and ITEM1102 a negative rating. If ITEM1101 and ITEM1102 are meal sets, then both the items are healthy for USER2 (that is why the user gave a positive rating to both of them). ITEM1101 is healthy for USER1 (that is why the user gave a positive rating) but ITEM1102 is unhealthy for USER1 (that is why the user gave a negative rating). As a result,

ITEM1101 and ITEM1102 do not have as similar nutritional values as the items in Matrix 1 (Table 3.2). Since one user gave positive rating to both the items and another user gave a positive rating to one item but a negative rating to the other item, these two items are LESS SIMILAR THAN THE ITEMS IN GRAPH 1 (Figure 3.5). That is why, the lines of ITEM1101 and ITEM1102 in Graph 2 (Figure 3.6) are not pointing at the same direction and the angle between them is larger than the angle between the items in Graph 1 (Figure 3.5). This angle is close to 90° so the cosine function of this angle is close to 0.

MATRIX 3 -

	Item1101	Item1102
USER 1	5	-3
USER 2	3	-5

Table 3.4: Cosine User-Item Matrix-3

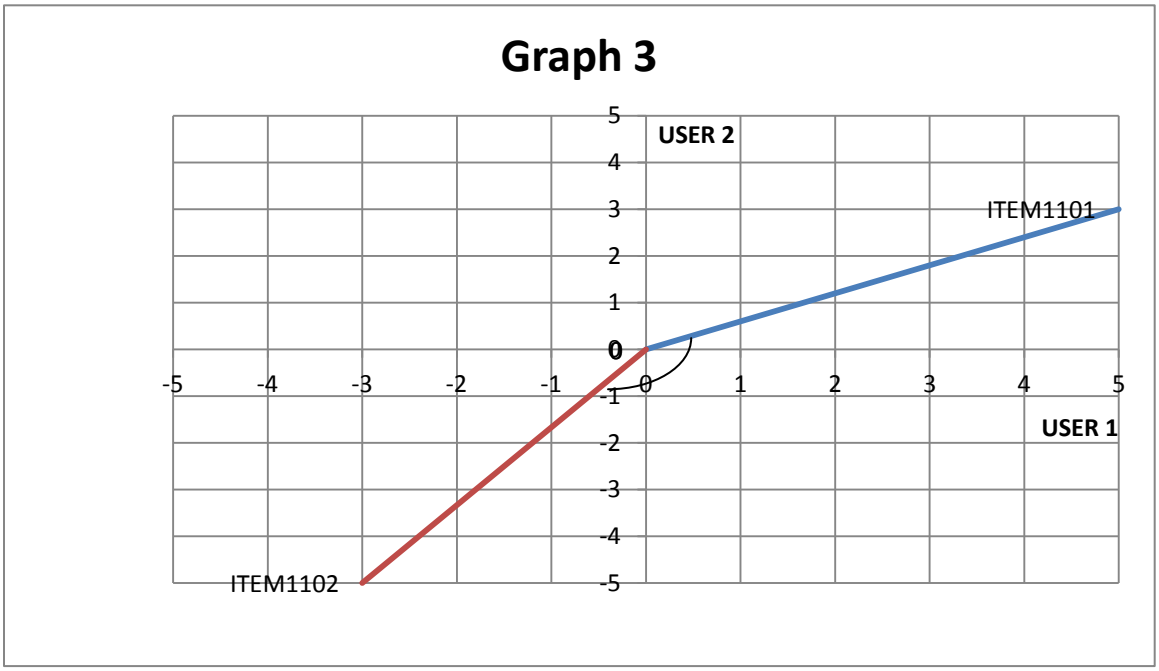


Figure 3.7: Cosine User-Item Graph – 3 (plotted using Table 3.4)

In Graph 3(Figure 3.7), USER1 gave a positive rating to ITEM1101 but a negative rating to ITEM1102. USER2 gave a positive rating to ITEM1101 but a negative rating to ITEM1102. If ITEM1101 and ITEM1102 are meal sets, ITEM1101 is healthy for USER1 (that is why the user gave a positive rating) but ITEM1102 is unhealthy for USER1 (that is why the user gave a negative rating).

ITEM1101 is healthy for USER2 (that is why the user gave a positive rating) but ITEM1102 is unhealthy for USER2 (that is why the user gave a negative rating). As a result, ITEM1101 and ITEM1102 must have VERY DISSIMILAR NUTRITIONAL VALUES. Since both the users liked an item and disliked the other item, both the items must be VERY DISSIMILAR. That is why the lines are almost pointing in the opposite direction and the angle between them is VERY LARGE (larger than the angle in both Graph1 and Graph2).

This angle is close to 180° so the cosine function of this angle is close to -1.

We have stated all our users to give a rating within a range of 1 to 5. So our user-item matrix does not have any NEGATIVE RATING. As a result, the range of our ANGLE is between 0° to 90° and the range of our COSINE FUNCTION (or SIMILARITY VALUES) is between 0 to 1.

cos(0°) = 1 = HIGHEST SIMILARITY VALUE

cos(90°) = 0 = LOWEST SIMILARITY VALUE

In our implementation, Matrix 2(Table 3.3) and Matrix 3(Table 3.4) is not possible because they have negative ratings. So I will explain the calculation of cosine similarity using Graph 1 (Figure 3.5).

The cosine similarity between two items i_m and i_b is calculated by -

$$s_u^{cos}(i_m, i_b) = \frac{i_m \cdot i_b}{\|i_m\| \, \|i_b\|} = \frac{\sum x_{a,m} x_{a,b}}{\sqrt{\sum x_{a,m}^2 \sum x_{a,b}^2}} \quad \ldots\ldots\ldots \quad (2)$$

where,

i_m = Item m

i_b = Item b

$x_{a,m}$ = rating x of USER a on ITEM m

$x_{a,b}$ = rating x of USER a on ITEM b

Now let us apply this formula on our random MATRIX 1(Table 3.2) and GRAPH 1 (Figure 3.5).

I have shown the matrix and the graph again -

MATRIX 1 (same matrix as shown before in this chapter) -

	Item1101	Item1102
USER 1	5	2
USER 2	3	5

Table 3.2: Cosine User-Item Matrix (same matrix as shown before in this chapter)

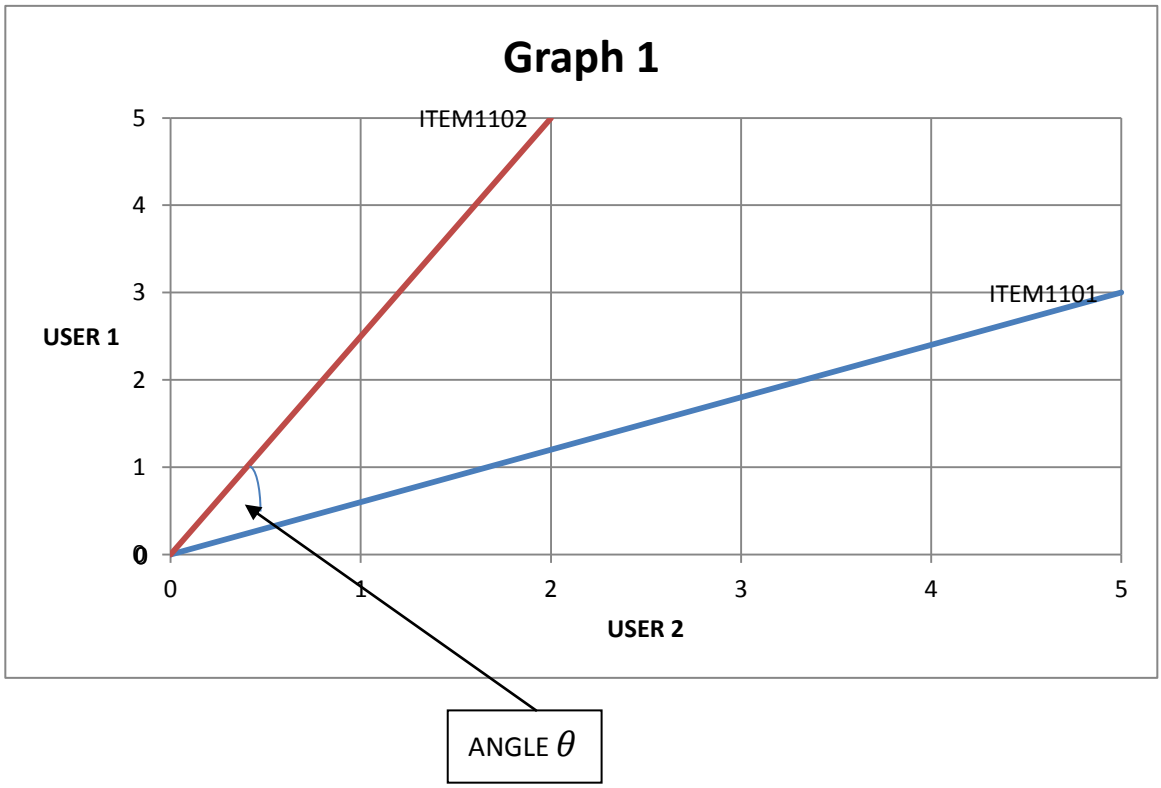
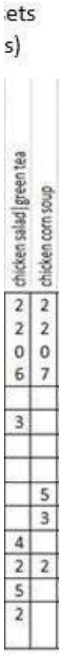


Figure 3.5: Cosine User-Item Graph – 1 (same graph as shown before in this chapter)

similarity(ITEM1101,ITEM1102) = Cos(θ) = $\frac{(5\times2)+(3\times5)}{\sqrt{(5^2+3^2)\times(2^2+5^2)}}$ = 0.7962

Now, I will delineate how to calculate the cosine similarity between two vector lines in an n-dimensional space using our USER-ITEM Matrix in Figure 3.2.

For example in Figure 3.2, we want to find the Cosine Similarity between Item2206 and Item2207. I have provided the part of Figure 3.2 at the right showing the ratings of Item2206 and Item2207 for better illustration. Imagine Item2206 and Item2207 as two vector lines in 10 dimensional spaces (as there are 10 users) or in a 10-dimensional graph with 10 axes.



Item2206 = (0,3,0,0,0,0,4,2,5,2)

Item2207 = (0,0,0,0,5,3,0,2,0,0)

cos(Item2206,Item2207) =

$$\frac{(0\times0)+(3\times0)+(0\times0)+(0\times0)+(0\times5)+(0\times3)+(4\times0)+(2\times2)+(5\times0)+(2\times0)}{\sqrt{(0^2+3^2+0^2+0^2+0^2+0^2+4^2+2^2+5^2+2^2)\times(0^2+0^2+0^2+0^2+5^2+3^2+0^2+2^2+0^2+0^2}}$$

= 0.08520286457

Finally, let us calculate whether Cosine Similarity captures our intuition that similarity(Item1101,Item1102) > similarity(Item1101,Item1105).

I have provided the part of our USER-ITEM MATRIX in Figure 3.2 at the right with all the ratings of Item1101, Item1102 and Item1105. In case of Cosine Similarity, a value of 0 means least similar and a value of 1 means most similar. In our user-item matrix, all our ratings are positive. So all the item-item cosine similarity values will be in the range between 0 and 1.

	1101	1102	1105
1	4	5	
2			
3	5		2
4	1		4
5			
6			
7			
8			
9			
10			

Item(1101) = (4,0,5,1,0,0,0,0,0,0)

Item(1102) = (5,0,0,0,0,0,0,0,0,0)

Item(1105) = (0,0,2,4,0,0,0,0,0,0)

cos(Item1101,Item1102) =

$$\frac{(4\times5)+(0\times0)+(5\times0)+(1\times0)+(0\times0)+(0\times0)+(0\times0)+(0\times0)+(0\times0)+(0\times0)}{\sqrt{(4^2+0^2+5^2+1^2+0^2+0^2+0^2+0^2+0^2+0^2)\times(5^2+0^2+0^2+0^2+0^2+0^2+0^2+0^2+0^2+0^2)}}$$

= 0.6172133998

cos(Item1101,Item1105) =

$$\frac{(4\times0)+(0\times0)+(5\times2)+(1\times4)+(0\times0)+(0\times0)+(0\times0)+(0\times0)+(0\times0)+(0\times0)}{\sqrt{(4^2+0^2+5^2+1^2+0^2+0^2+0^2+0^2+0^2+0^2)\times(0^2+0^2+2^2+4^2+0^2+0^2+0^2+0^2+0^2+0^2)}}$$

= 0.4830458915

cos(Item1101,Item1102) > cos(Item1101,Item1105)

Therefore, Similarity(Item1101,Item1102) > Similarity(Item1101,Item1105)

Hence, it successfully captures our intuition.

I have provided the output of the similarity values of our program below.

Figure 3.8: cos(Item1101,Item1102)

As you can observe in Figure 3.8, cos(Item1101,Item1102) = 0.6172133998

as we have shown in our calculation.

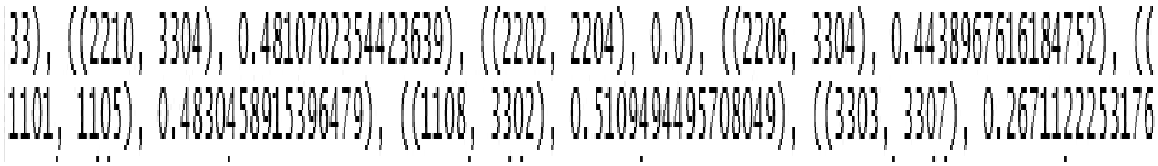


Figure 3.9: cos(Item1101,Item1105)

As you can observe in Figure 3.9, cos(Item1101,Item1105) = 0.4830458915 as we have shown in our calculation.

Our output has the similarity values of all the item-item pairs.

I have provided the final output of our program below.

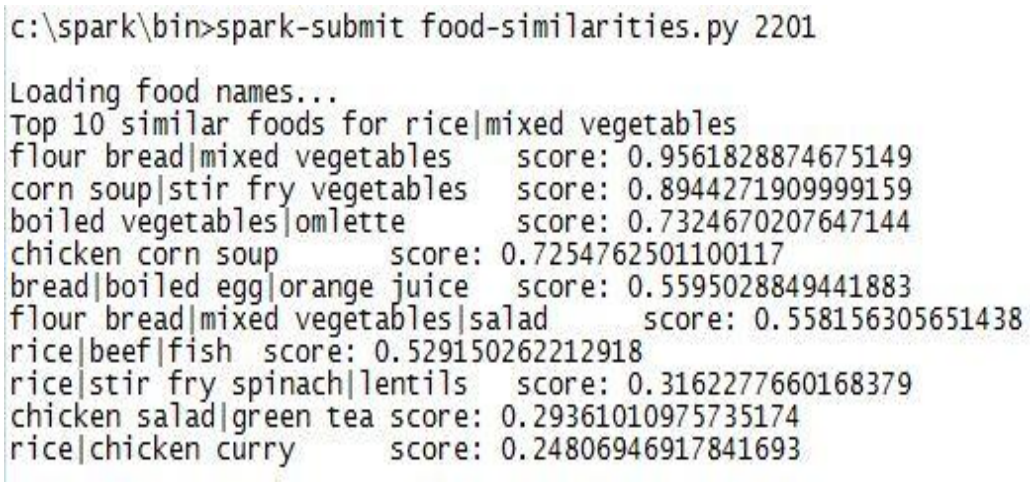


Figure 3.10: Top 10 similar foods of Item2201 rice|mixed vegetables

Figure 3.10 is the final output of our program with Item2201(rice|mixed vegetables) as the input. In the figure, cos(rice|mixed vegetables, flour bread|mixed vegetables) = 0.9561828874675149 which is the highest similarity value. This means that the nutritional values of rice|mixed vegetables and flour bread|mixed vegetables is very similar. As the result, the user can consume flour bread|mixed vegetables instead of rice|mixed vegetables as they both have very similar nutritional values. The output produces 10 most similar meal sets in the descending order of their similarity values. This means that flour bread|mixed vegetables is most similar to rice|mixed vegetables and rice|chicken curry is least similar to rice|mixed vegetables.

Although we successfully captured our intuition that Similarity(Item1101,Item1102) > Similarity(Item1101,Item1105) using Cosine Similarity but the Similarity(Item1101,Item1102) is only marginally greater than Similarity(Item1101,Item1105).

cos(Item1101,Item1102) = 0.6172133998

cos(Item1101,Item1105) = 0.4830458915

cos(Item1101,Item1102) > cos(Item1101,Item1105) but not by that much.

Although we can clearly observe from Figure 3.2 (the part of it is given to the right) that Item1101 and Item1102 is much more similar than Item1101 and Item1105. Because User1 gave very similar ratings to both Item1101 and Item1102 but User3 and User4 gave very dissimilar ratings to Item1101 and Item1105.

	1101	1102	1105
1	4	5	
2			
3	5		2
4	1		4
5			
6			
7			
8			
9			
10			

The problem here is the fact that cosine similarity assumes missing ratings to be zero. For example, 7 users did not rate Item1101, 9 users did not rate Item1102 and 8 users did not rate Item1105. Cosine Similarity assumes all these ratings to be 0.

This is why cos(Item1101,Item1102) > cos(Item1101,Item1105) but not by an extensive number.

There is a better approach of finding similarity values known as Centered Cosine Similarity. We did not implement our program using Centered Cosine Similarity but we will do it in future. This will help us attain a more accurate system where our intuition of Similarity(Item1101,Item1102) > Similarity(Item1101,Item1105) can be captured more accurately.

3.2 Implementation of Model-based Collaborative Filtering using Alternating Least Squares (ALS)

The model based collaborative system became popular after the Netflix opened a million competitions in 2006. In this competition, many people used model based approach as their strategy. Finally, the winning team used many strategies but the main algorithm was model based.

This algorithm recommends items by making a user rating model [16]. Model based CF take an approach to predict and infer from the existing ratings. The model building process includes usage of different machine learning techniques like, Bayesian Network, clustering, association rules and many more. In our system, we used a machine learning technique called matrix factorization.

Matrix Factorization using Alternating Least Square (ALS):

The two primary fields of Collaborative Filtering are the neighborhood model and the latent factor model. The neighborhood method is an item based approach which computes relations between items or between users. This method evaluates user preference of an item depending on how the user rated the neighboring items.

On the other hand, latent factor model is a method that characterizes both the user and item to explain the ratings. However, some of the best latent factor models are based on matrix factorization. Algorithms like matrix factorization performs better than the neighborhood approach therefore, we chose this.

A popular approach to build a recommendation system using collaborative filtering is through matrix factorization method. The rudimentary idea is that vectors of latent factors are inferred from ratings' pattern and matrix factorization characterizes users and items by this. The higher the correspondence between the users and items, the greater the chance of recommendation. The main aim of matrix factorization is to fill the missing ratings of the ratings matrix (shown below). What it seeks to do is, leverage ratings from similar users and fill the blank cells with predictive ratings. In our thesis, we have used matrix factorization algorithm and for training the matrices, Alternating Least Square method is used. An example of matrix factorization is given below.

	Item W	Item X	Item Y	Item Z
User A		5.0		
User B	5.0		4.0	3.0
User C			5.0	3.0
User D	2.0	3.0	2.0	

Rating Matrix

=

A	1.2	0.8
B	0.7	0.9
C	1.0	0.8
D	0.9	1.5

User Matrix

W	X	Y	Z
1.7	1.2	0.6	0.8
0.9	1.0	0.4	0.8

Item Matrix

Figure 3.11: Example of Matrix Factorization (user id as input)

In the above Figure 3.11, the rating matrix consists of the item(meal sets) ratings rated by each user in a system. In matrix factorization, the rating matrix is factorized into two matrices, user and item matrix, which expose the latent/hidden factors.

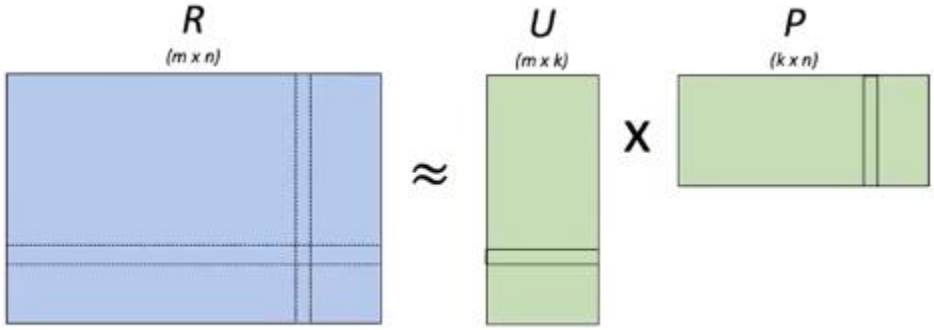
For example, if we consider building a movie recommendation system, the rating matrix will have users who watched and rated movies and the items will be movie names. Now, the users who rated the movies have varieties of reasons to like or dislike a particular movie. A user might like a movie because of its genre i.e. action, romantic, rom-com movies, but this is unknown. During factorization, this genre becomes the latent factor and it is chosen randomly. This latent factor is known as the rank. If there are two (2) random factors then rank becomes 2. Therefore, it is called LR (Low rank) matrix factorization. Matrix Factorization is a machine learning approach and we used ALS algorithm for this.

User ID as input for our recommendation system:

The system which we have implemented is a diet recommendation and the recommendations can be done by both user id and type id as input. When user id is given as input, user can rate items and get recommendations accordingly. For example, as shown in Figure 4.11, A B C D are users. Whereas, W (cereal|milk|almonds), X (vegetable oats), Y(chicken salad|green tea) and Z (flour bread, mixed vegetables|salad) are meal sets/food items. User B gave 5.0 to item W, 4.0 to Y and 3.0 to Z but did not rate item X. However, user D gave 2.0 to item W, 3.0 to item X and 2.0 to item Y but did not rate item Z. Notice that there is a similarity between users B and D, they both rated items W and Y. The similar items between these users are W and Y. Similarly, all the users rated some food items but not all. That is why some values are missing. To make recommendations, we need all the ratings and the highest rated food will be recommended to a particular user.

This is where matrix factorization and ALS comes into play. ALS takes the rating matrix and factorizes it into user and food item matrix where if the two matrices are multiplied back together

will produce an approximation of the original rating matrix. It is based on the properties of users and food items. These properties are the latent factors which are randomly chosen and we will never know what these are. The latent factors for users might be age, height, sex, disease etc. The latent factor for items can be type of food, liquid or solid etc. Based on this some values will be predicted and the user and item matrix will be filled by the computer randomly; this is an art of Machine learning.



$$Error_{ij} = \sum w_{ij} \cdot (R_{ij} - u_i \times p_j^T) + \lambda(\| U \|_2 + \| P \|_2) \quad \text{..... (3)}$$

Figure 3.12: Matrix Factorization (R=U*P) and Error equation

Figure 3.12 shows that the basic formula is R=U*P, R is our rating matrix, U is the user matrix and P is the food item matrix from Figure 3.11. After filling in the matrices with random numbers, an error formula (shown above) is used to calculate the error term and then alternating back and forth between matrix U (User matrix) and P (Food item matrix), ALS iteratively adjust the matrices to decrease the error. It continues to alternate between them until the error is minimized. Once this is completed, the matrices are multiplied back together. After this, the Rating matrix(R) is seen to be filled completely i.e. all the blank cells are filled with ratings. The item with highest rating becomes a recommendation for a user.

	Item W	Item X	Item Y	Item Z
User A		5.0		
User B	5.0		4.0	3.0
User C			5.0	3.0
User D	2.0	3.0	2.0	

ALS (user id as input)

	Item W	Item X	Item Y	Item Z
User A	**3.67**	5.0	**4.98**	**4.87**
User B	5.0	**3.45**	4.0	3.0
User C	**4.18**	**4.22**	5.0	3.0
User D	2.0	3.0	2.0	**3.23**

Figure 3.13: Filled in Matrix after ALS (user id as input)

In Figure 3.13, it is shown how the matrix will be filled in. The values in blue are calculated by ALS. Notice that, user B did not rate X but user D did, based on that rating, item X is recommended to user B is calculated using ALS. For user B item X (3.45) becomes recommendation. Similarly, for user D item Z (3.23) becomes a recommendation. The items

which user B rated did not show in the recommendation list i.e. this technique excludes the items rated by any particular user and recommend the items which have higher rating. The existing rating is only used to train the matrices. Therefore, the items with high ratings become recommendations.

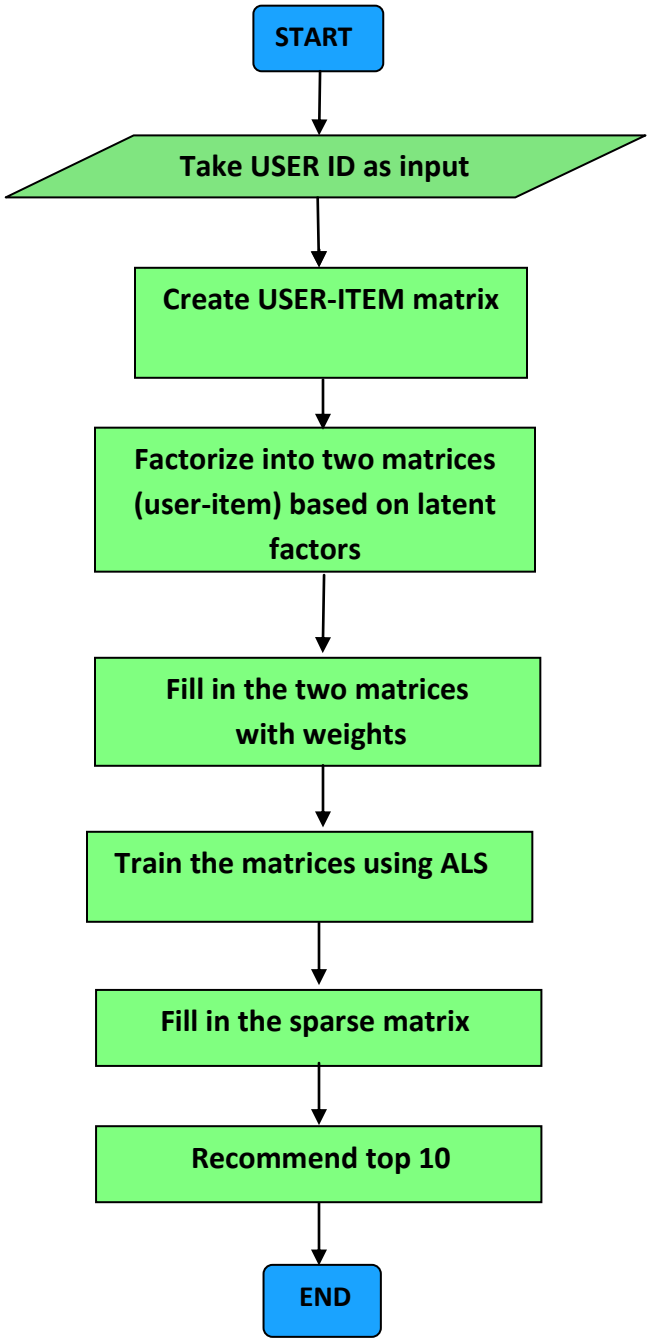


Figure 3.14: Flowchart of Model-based Recommendation System (user id as input)

Type ID as input for our recommendation system:

For our system, we considered three types of meals 1) Breakfast 2) Lunch and 3) Dinner. We have set unique type ids' for each set of meals. For instance, all the dinner meals are of type id 921-930. If the input is the type id of any meal like if the input is 922 then the recommendations will be done based on the type id of 921-930 which are dinner meal sets only. However, the meal set for type id (922) will not be included in recommendations. Considering only the dinner meal sets recommendations will be shown. This narrows down and makes the recommendation more specific for new and existing users. For example, a new user who did not rate any items can also see recommendations based on the type id. It is done exactly like the user id as input approach, except the items in the rating matrix depends on the type id input. For example if type id 927 is given which is an id of dinner meal then all the items will be of the dinner meal set. An example is shown below.

	Item M	Item N	Item O	Item P
User A	5.0			3.0
User B	3.0	2.0		
User C		3.0		
User D	4.0		3.0	

Rating matrix

=

A	0.2	1.8
B	1.7	0.9
C	0.9	0.4
D	0.5	1.2

User matrix

M	N	O	P
1.1	0.2	0.8	1.8
0.8	1.2	1.6	0.6

Item matrix

Figure 3.15: Example of Matrix Factorization (type id as input)

Similarly, like user id as input, rating matrix is formed. Then it is factorized into two matrices, user matrix and item matrix (i.e. meal set matrix which depends on the type id given in input) shown in Figure 3.15. As explained before, ALS fills in the blanks and recommendation is done after that.

	Item M	Item N	Item O	Item P
User A	5.0			3.0
User B	3.0	2.0		
User C		3.0		
User D	4.0		3.0	

ALS (type id as input)

	Item M	Item N	Item O	Item P
User A	5.0	**3.77**	**3.98**	3.0
User B	3.0	2.0	**3.51**	**4.70**
User C	**4.43**	3.0	**3.65**	**4.98**
User D	4.0	**4.66**	3.0	**3.58**

Figure 3.16: Filled in Matrix after ALS (type id as input)

Figure 3.16 shows the filled in matrix after ALS is performed. For example, a user entered 921 as input, type id of dinner, then the user will be given recommendations specifically from dinner meal sets.

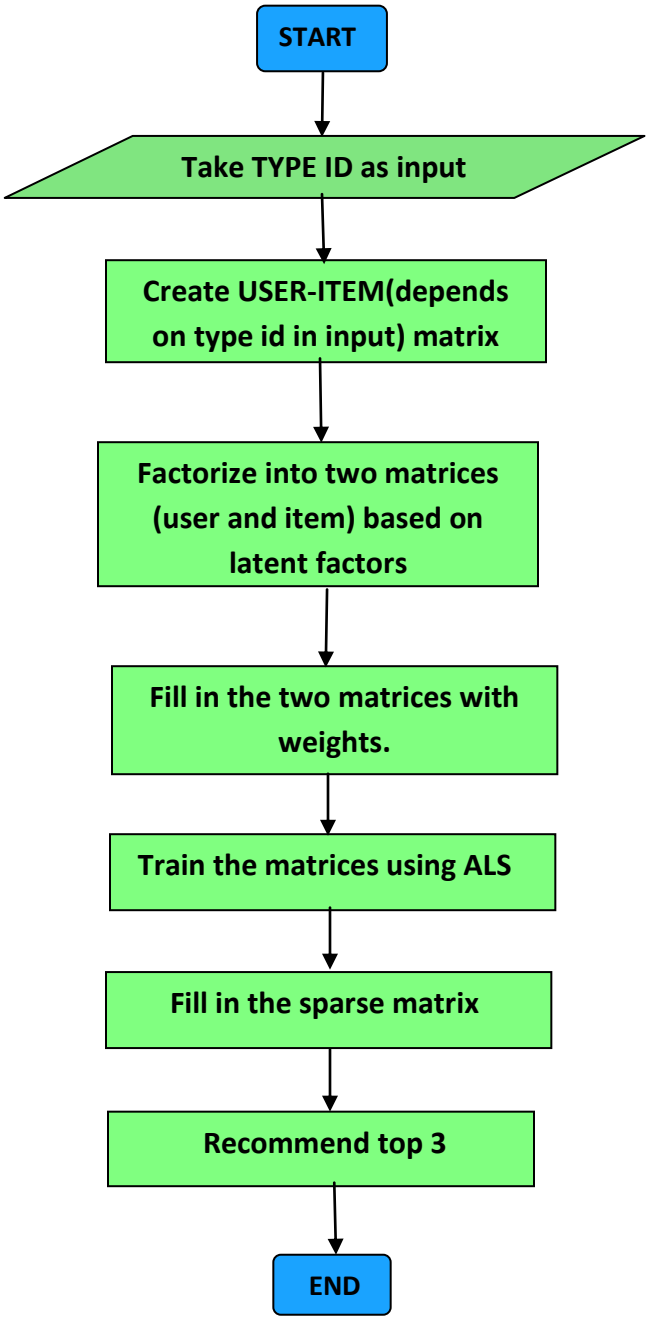


Figure 3.17: Flowchart of Model-based Recommendation System (type id as input)

Alternating Least Square (ALS) Algorithm

Initialize P with small random numbers

for E times **do**

Compute the U that minimizes the error for fixed P

Compute the P that minimizes the error for fixed Q

end

E=Number of iterations.

Another approach could have been used which is known as the Singular Value Decomposition (SVD). It is a very well established technique used to identify latent factors while retrieving information. When SVD is applied in collaborative filtering domain, the rating matrix is also factorized.

However, conventional SVD is not defined when the information is not complete. What it does is, it fills the missing data with zeros. This leads to inaccurate predictions especially when the matrix is highly sparse (blank). Therefore, we chose to use Matrix Factorization instead of Singular Value Decomposition [15]. Also, Matrix Factorization is faster than SVD.

3.3 Data Collection

The data required for our research is not available in public domain. So we had to do a public survey for collecting the required data. The response of the survey was not good enough but we could manage to get good range of ratings for thirty meal sets from ten users.

We have collected the list of meal sets from people that are suggested by their personal nutritionists and the ratings of each meal set for each user was also given by their nutritionists. The ratings scale is from 1-5. The higher the rating of an item means it is the healthier for a user.

CHAPTER 4

SYSTEM INFORMATION

4.1 System Information

Memory-based Collaborative Filtering Using Cosine Similarity

In this case, both a new user and an existing user who has rated meal sets previously will be beneficial. If you want to get a good bunch of diet suggestion then just provide the meal set i.e. FoodID for which you want a similar food. For this approach a good set of meal sets is recommended to the user based on similarity of taste on different meal sets rated by different users.

Model-based Collaborative Filtering Using Alternating Least Square (ALS)

We have implemented Alternating Least Square (ALS) algorithm in two ways:

1) The user will give his or her UserID as an input in order to get the recommendation for meal set.

In this case, an existing user can log in to his or her profile and can rate any existing meal set or a new meal set. He or she can also get suggestion of meal sets by providing his or her UserID to get top recommendations for meal set. Here the recommendations will come based on his or her previous ratings on meal sets matched with other users' choice.

The user will not get any specific type of meal set for example, meal set only for breakfast or lunch or dinner.

A new user who hasn't rated any meal sets before will not get any kind of recommendation from this approach. He or she must rate few meal sets before getting any diet recommendation.

2) The user will give a TypeID as an input in order to get a specific recommendation for meal set.

In this case, both an existing user and a new user will get advantage of getting recommendations of meal sets. The user only need to enter the TypeID of his fixed meal set to get the specific alternative meal set for breakfast, lunch, dinner.

For example, if a user enters a TypeID that is in the range of breakfast meal set then he or she will get recommendations only for breakfast meal set. This brings a more specific diet recommendation in order to change your daily diet plan.

4.2 Tools Used

Apache Spark

We have used Apache Spark to utilize Spark MLlib which is a machine learning library that consists of different algorithms and utilities. We have implemented Alternating Least Square (ALS) algorithm for our recommendation system using MLlib.

Apache Spark is a big data processing framework which enables us to use some of the most popular tools to implement big data related tasks. Spark runs on the top of Hadoop Distributed File System (HDFS) infrastructure to provide enhanced and additional functionality.

Resilient Distributed Dataset (RDD), a multi dataset which is readable and dispersed over a cluster of machines that maintains any kind of fault. RDD is an architectural foundation of Apache Spark.

In our recommendation system implementation we have used RDD of Spark so that the huge dataset can be handled in a proper way. The same dataset that is used a multiple times doesn't require to be run from the hard disk rather the dataset is kept in memory which increases the speed of our engine. Spark is also useful for processing huge dataset in a convenient way.

For our recommendation system we have used RDD map() function to convert the dataset into key-pair values.

For both the algorithms' implementation we have used map(). We have made a key-pair between userID and FoodID, Rating.

Spark runs on Java Virtual Machine (JVM) environment and is written in Scala Programming Language. It also supports languages like Java, Python etc for building up applications. We have built up our recommendation engine using Python Programming Language.

4.3 Hardware Specification

We have implemented our work in a Windows Operating System with a RAM of 4GB and 64-bit OS.

Apache Spark normally works best in 8GB RAM. Since our dataset wasn't that large 4GB was good enough to work with.

CHAPTER 5

RESULT AND ANALYSIS

5.1 System output

For cosine similarities approach, we get top 10 food recommendations as an alternative for a specific meal set. The recommended meal sets can be a very good food that can be replaced by the user's existing fixed meal set suggested by his or her personal nutritionist.

```
c:\spark\bin>spark-submit food-similarities.py 2201

Loading food names...
Top 10 similar foods for rice|mixed vegetables
flour bread|mixed vegetables    score: 0.9561828874675149
corn soup|stir fry vegetables   score: 0.8944271909999159
boiled vegetables|omlette       score: 0.7324670207647144
chicken corn soup       score: 0.7254762501100117
bread|boiled egg|orange juice   score: 0.5595028849441883
flour bread|mixed vegetables|salad      score: 0.558156305651438
rice|beef|fish  score: 0.529150262212918
rice|stir fry spinach|lentils   score: 0.3162277660168379
chicken salad|green tea score: 0.293610109757351 74
rice|chicken curry      score: 0.24806946917841693
```

Figure 5.1: Cosine similarities output

For the first approach of ALS we get the top 10 food recommendations for a user who hasn't yet consumed the meal sets recommended yet but has similar choices with other users.

```
c:\spark\bin>spark-submit food-als-input-user-id.py 8

Loading food names...

Top 10 recommendations for user ID 8:
oats|milk  score 4.977815906763 89
cereal|milk|banana  score 4.957016308295558
vegetable oats  score 3.9904579362164734
oats|apple|milk  score 3.9891894117898232
cereal|milk|almonds  score 2.971070602355025
rice|chicken|lentils  score 2.0076163269128826
bread|fruit jam|green tea  score 1.9913363771694286
chicken salad|green tea  score 1.9897783116558676
chicken corn soup  score 1.9817899511876291
rice|boiled vegetables|fish curry  score 1.9771210298904378
```

Figure 5.2: ALS output in case for a user

In the second approach of ALS, we specifically implemented the method to suggest users specific meal sets for each time of the day. For instance, if a user wants to know the alternative of any meal set for breakfast or lunch or dinner then our system will only suggest him or her the meal sets for only breakfast or lunch or dinner.

```
c:\spark\bin>spark-submit food-als-input-type-id.py 907

Loading food names...

Top 3 recommendations for breakfast:
oats|milk score 4.664277638349256
cereal|low fat milk|strawberry score 2.1641762032839438
cereal|milk|almonds score 1.6505724674963838
```

Figure 5.3: ALS output for meal set of breakfast

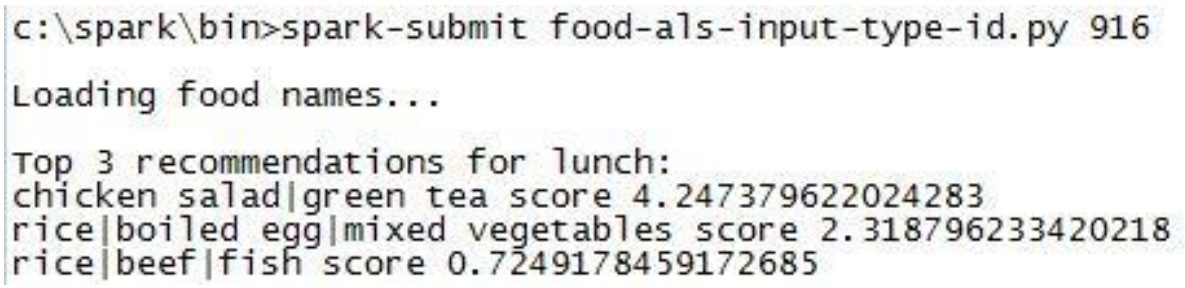


Figure 5.4: ALS output for meal set of lunch

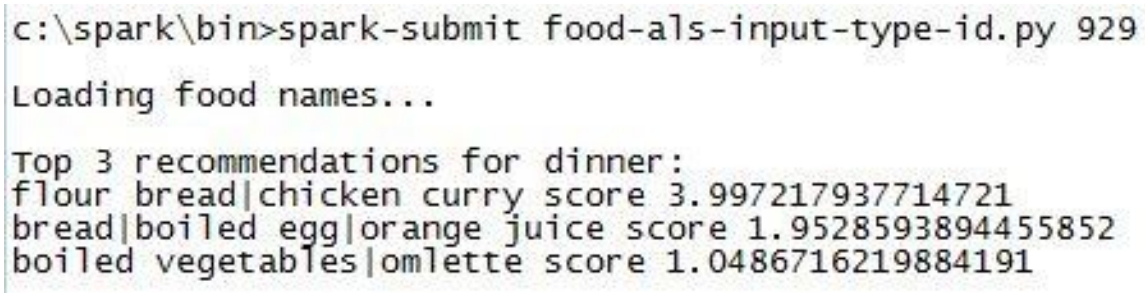


Figure 5.5: ALS output for meal set of dinner

In this approach the recommendation we got are only top three because of the limited size of our dataset.

5.2 Graphical Representation

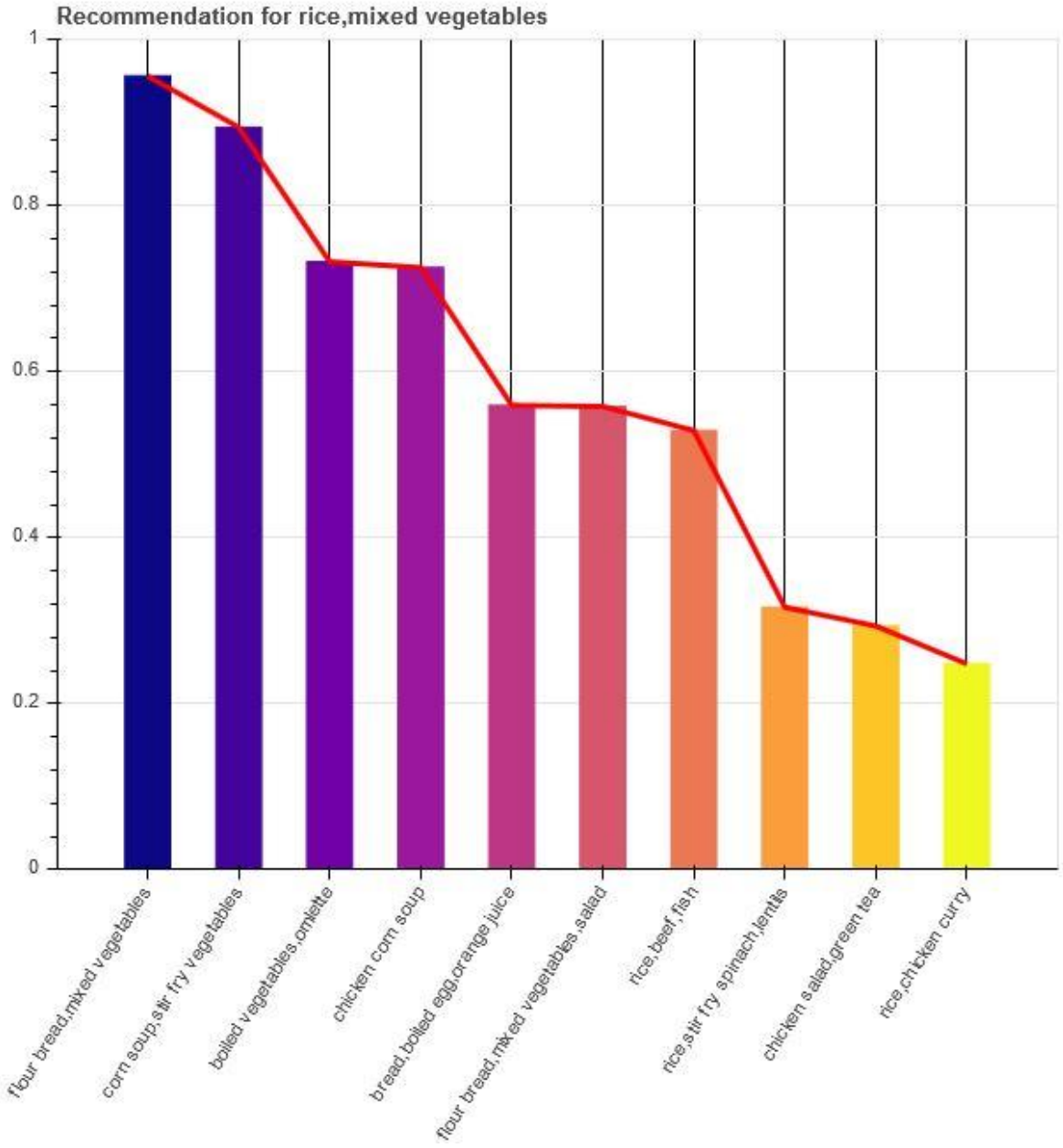


Figure 5.6: Graphical representation of Cosine Similarities approach output

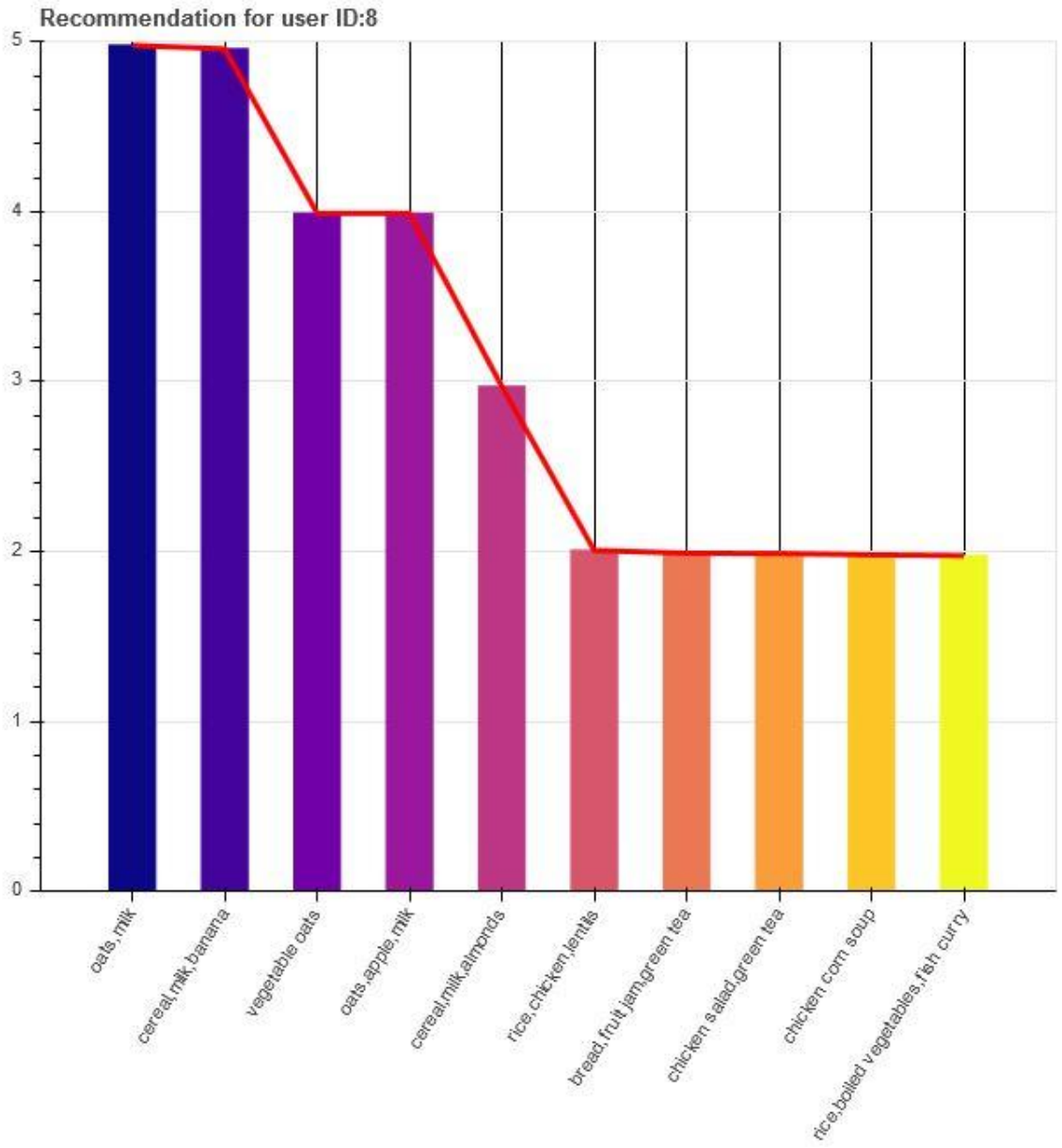


Figure 5.7: Graphical representation of first approach (user id as input) output of ALS

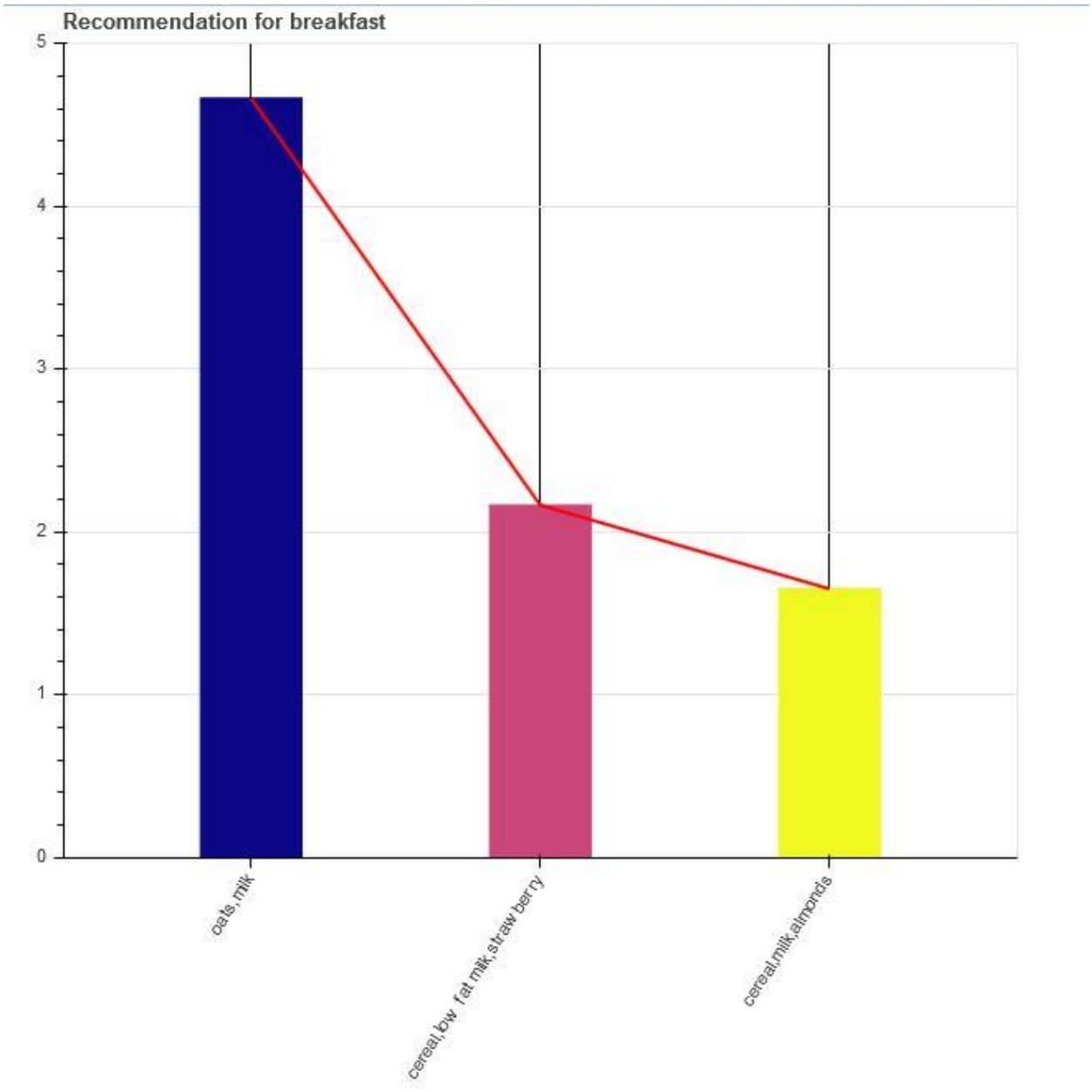


Figure 5.8: Graphical representation of second approach (type id as input) output of ALS for breakfast meal sets

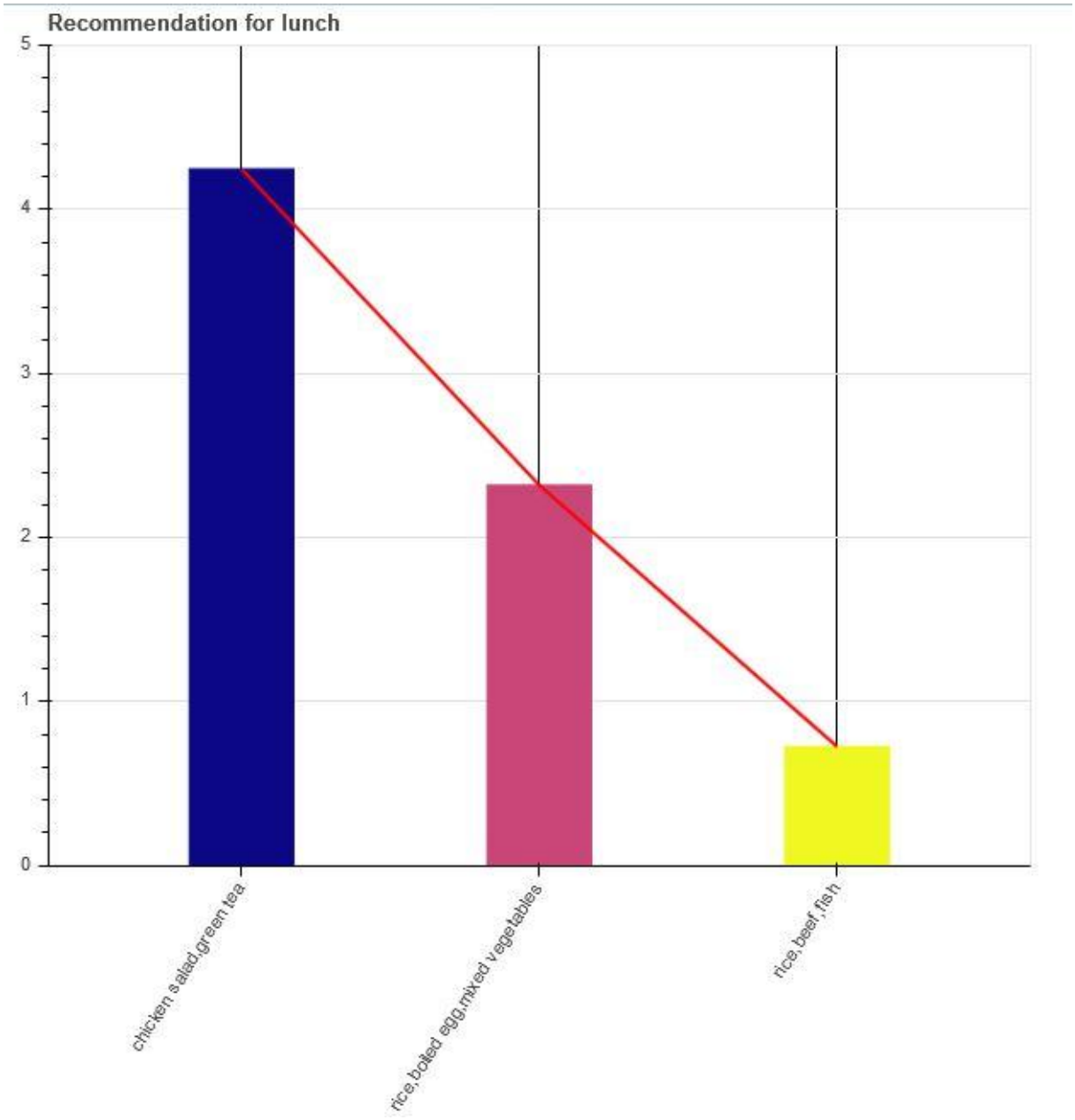


Figure 5.9: Graphical representation of second approach output (type id as input) of ALS for lunch meal sets

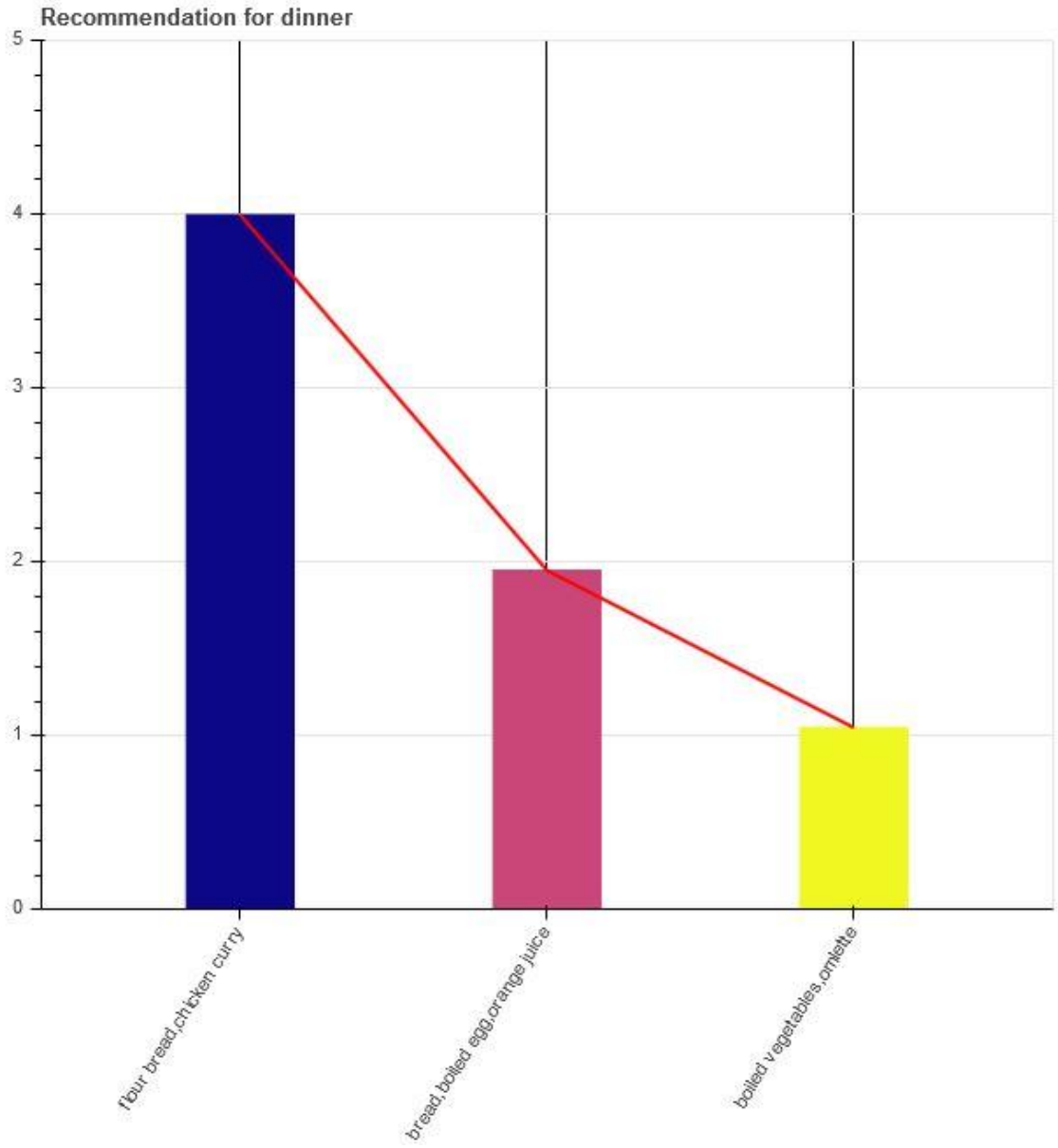


Figure 5.10: Graphical representation of second approach output (type id as input) of ALS for dinner meal sets

5.3 Analysis

Memory-based recommendation system mainly focuses on how much an item (meal set) is liked by the users. It figures out the similar ratings of an item (meal set) pair and checks whether they are similar meals or not. This means if there is no similarity of ratings between any pairs of meal sets then the system will give an average recommendation for the meal set alternative to what the user is looking for. So there must be such dataset that has at least one similar meal set for a given meal set that is both the meal sets must be given similar values by at least one user. Here no matter how poor or good ratings are given by the users to the meal sets it is not taken into account. This means if two meal sets are given a rating of 1 by the same user then these two meal sets are very similar. But if one meal set is rated a 1 and another meal set is rated a 5 then these are very dissimilar meal sets thus in such case we will get average recommendation for a meal set. Even if a meal set is rated very high by the users if that meal set doesn't have any similar rating with other meal sets then the recommendation of alternative meal sets will be very poor. Moreover, if a meal set is rated by only one user and the user has not rated any other meal sets then the recommendation system will have no similar meal sets for that meal set and therefore no recommendation for that meal set. The memory-based recommendation system using cosine similarity also assumes that the rating of a meal set is zero if it has not been rated by any user. So it finds very dissimilarity among items which are rated and which are not rated. Thus, again makes poor recommendation for a given meal set.

On the other hand model-based recommendation system basically focuses on the values of the ratings given by the users. The machine learning technique matrix factorization uses the existing ratings of meal sets to predict the ratings of other meal sets that have not been rated by the users. These predicted ratings are fully done by the system and there is no scope of any error if the

ratings on the meal sets given by the users are absolutely meaningful and correct. But these prediction ratings are also dependent on the parameters of our dataset. In a case where an item X is rated a 5.0 by user A, a 4.0 by user B and a 1.0 by user C. But D hasn't rated item X. Now when the values of the ratings are computed for training and getting the predicted ratings sometimes the lowest rating value compared with other ratings of an item is not considered. Here the rating value 1.0 might not be computed for training the existing values and computing the predicted ratings. This can make the recommendation very inefficient sometimes thus scaling is really necessary in such cases.

Another big factor that works for both the approach is how the users rate the meal sets. If there is any wrong or misleading rating on the meal sets despite the meal sets being very good for health then also there is a possibility of getting wrong or poor recommendation for meal sets. So the dataset must be arranged in such a way that there is no scope of incorrect or false ratings on the meal sets.

These are the basic differences that can be found between the two types of recommendation systems.

But we have implemented the memory-based recommendation system based on item-item(meal sets) similarities and the model-based recommendation system in two ways:1) an user will get recommendations of meal sets if previously he has rated other meal sets and based on that he will get an alternative meal set; 2) here the user ratings works implicitly, any user new or existing can get a recommendation for a meal set but in specific type compared to the implementation of memory-based recommendation system.

Here we cannot exactly compare between these two types of recommendation system since one is memory-based where it finds the predicted ratings implicitly by finding out the similarity values whereas, the model-based recommendation system uses a machine learning algorithm to find out the predicted ratings. But we can conclude that the model-based recommendation system works best since there is no scope of assuming an unrated meal set a zero that minimizes the dissimilarity among different meal sets which happens in case of memory-based recommendation system using cosine similarity.

CHAPTER 6

CONCLUSION

6.1 Conclusion

To conclude, we tried to build up a diet recommendation system by using two different Collaborative Filtering algorithms i.e. Cosine Similarity and Alternating Least Square (ALS). Each of these approaches were different in their ways and techniques but our main aim was to help users get rid of monotonous meals they consume in their daily lives. This recommendation engine will hopefully help those people who have a boring, fixed eating pattern and make their food habit an interesting one by helping them to choose variety of meal sets from the preferences of other users.

6.2 Limitation

The main limitation was data collection. As we have said earlier the dataset we needed to implement our idea is not available in public domain. As a result we had to collect it by doing survey. We have also approached some authors of various publications but there was no response.

6.3 Future Work

As we have said in Chapter 4.1, we haven't yet implemented cosine similarities to find the predicted rating of each meal sets by a user. We will do this in our next step. Also there is a better approach of finding similarity values known as Centered Cosine Similarity which we will implement for the memory-based recommendation system in order to get better accuracy of outputs and compare it with the outputs of the model-based recommendation system.

And finally after comparison we will choose the best one out of these two and use it for making an android application that will be suitable for all kind of users from youngsters to aged people and help them get meal sets of different tastes as an alternative for their fixed meal sets.

REFERENCES

[1] X. Li, Xinliu, Z. Zhang, Y. Xia, S. Qian, "Design of Healthy Eating System based on Web Data Mining", WASE International Conference on Information Engineering 2010.

[2] Agapito G., Calabrese B., Guzzi P. H., Cannataro M., Simeoni M., Car´e I., Lamprinoudi T., Fuiano G., Pujia A., "DIETOS: a recommender system for adaptive diet monitoring and personalized food suggestion", Fourth International IEEE Workshop on e-Health Pervasive Wireless Applications and Services 2016.

[3] N. Suksom, M. Buranarach, Y. M. Thein, T. Supnithi, P. Netisopakul, "A Knowledge-based Framework for Development of Personalized Food Recommender System", 2010

[4] N. Gaur, A. Singh **"**Recommender System - Making Lifestyle Healthy Using Information Retrieval", 2016.

[5] W. Husain, L. J. Wei, S. L. Cheng, N. Zakaria , "Application of Data Mining Techniques in a Personalized Diet Recommendation System for Cancer Patients"

[6] H. Eghbali, M. A. Eghbali, A. V. Kamyad, "Optimizing Human Diet Problem Based on Price and Taste Using Multi-Objective Fuzzy Linear Programming Approach", vol. 2, no. 2, pp. 139-151, 2012.

[7] S. F. Sufahani, Z. Ismail, "Planning a Nutritious and Healthy Menu For Malaysian School Children Aged 13–18 Using 'Delete-reshuffle Algorithm' in Binary Integer Programming", *J. Appl. Sci.*, vol. 15, no. 10, pp. 1239-1244

[8] Wired.com, 'The Long Tail', 2004. [Online]. Available: https://www.wired.com/2004/10/tail/ . [Accessed : 23- November- 2017].

[9] J. Leskovec, A. Rajaraman, J. D. Ullman, "Mining of Massive Datasets"

[10] G. Adomavicius , A. Tuzhilin, "Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions", IEEE Transactions on Knowledge and Data Engineering, Vol. 17, No. 6, June 2005

[11] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen, Collaborative Filtering Recommender Systems, 2007.

[12] M. D. Ekstrand, J. T. Riedl and . A. Konstan, Collaborative Filtering Recommender Systems, 2011.

[13] Bugra.github.io, 'Alternating Least Squares Method for Collaborative Filtering', 2014. [Online]. Available: https://bugra.github.io/work/notes/2014-04-19/alternating-least-squares-method-for-collaborative-filtering/. [Accessed: 10- October- 2017].

[14] T. Yu, O. J. Mengshoel, A. Jude, E. Feller, J. Forgeat, N. Radia, "Incremental Learning for Matrix Factorization in Recommender Systems", 2011.

[15] Y. Koren, R. Bell and C. Volinsky, "Matrix Factorization techniques For Recommender Systems", 2011.

[16] Claudio Adrian Levinas, "An Analysis of Memory BasedCollaborative Filtering Recommender Systems with Improvement Proposals", 2014.