



Faculty of Computer & Information Sciences

Ain Shams University

Department of Information Systems 2018/2019

Project: *Customizable Context-Aware Recommender System*

Supervised By:

- *Dr. Wedad Hussein*
- *T.A. Mariam Hassanein*

Team Members:

- *Nader Sayed Mahmoud*
- *Youssef Sayed Yehia*
- *Youssef Mohamed Ahmed*
- *Nourhan Osama Ahmed*
- *Huda Mohamed Hisham*

June 2019

Acknowledgment

We have to thank *Dr. Wedad Hussein* for her guidance and her Patience on Answering any questions we asked and also taking from her time to support us in the whole project till we complete it.

We would like to express our gratitude towards *TA. Mariam Hassanein* for her kind co-operation and her help in the project.

Finally, Thanks to our team in developing this project and also thanks to individuals who help us in completing this project that it would not have been possible to complete it without the kind support and help of them.

Abstract

Given the rapid changes on the online information and increase of their volume and complexity over time and the ways that people look for their needs are becoming old and time consuming. Therefore people need modern ways to search for their needs in a reasonable and saving time way. Recommender systems are used to recommend useful items to a user. Context information can be used to characterize the state of an item like Climate, Location, and Age, etc... for more accurate recommendations.

In our project, we developed a reusable component which recommends items to users based on his/her pervious ratings to a similar item and also based on a set of contexts. The systems use association rules to be able to deal with and understand any type of contexts. This feature makes the system in addition to being reusable, it makes it also customizable to any application.

Table of Contents

Ch1	Introduction.....	7
1.1	Motivation.....	7
1.2	Problem definition.....	7
1.3	Objective	8
1.4	Document organization.....	8
Ch2	Background.....	9
2.1	Recommender systems	9
2.2	Filtering techniques	9
	2.2.1: content-based filtering	10
	2.2.2: collaborative filtering.....	10
	2.2.3: Hybrid filtering.....	17
2.3	Recommender systems and their drawbacks.....	18
2.4	Context.....	19
	2.4.1:context types	19
	2.4.2: context acquisition.....	20
	2.4.3: context relevance and context selection.....	21
	2.4.4: context-aware datasets	21
	2.4.5:context in recommendation system.....	22
	2.4.6:context-aware recommendation system (CARS).....	22
	2.4.7:other challenges.....	23
2.5	Related Works.....	24
Ch3	Analysis and design.....	27
3.1	System overview.....	27
	3.1.1: System architecture.....	27
	3.1.2: Functional requirements.....	28
	3.1.3: Non-functional requirements.....	28
	3.1.4: System users.....	28
3.2	System analysis && design.....	29
	3.2.1: Use case diagram.....	29
	3.2.2: Class diagram.....	30
	3.2.3:Sequence diagram.....	31

Ch4	Implementation.....	33
4.1	Collect dataset.....	33
4.2	User-item matrix.....	34
4.3	Contexts generation.....	34
4.4	Association rules.....	35
4.5	Cosine similarity.....	37
	4.5.1: Cosine similarity steps.....	37
4.6	Initial recommendations.....	38
	4.6.1: Generate predictions.....	38
4.7	Final recommendations.....	38
	4.7.1: Check Matched Rules.....	38
	4.7.2: Final Recommendations.....	39
Ch5	User Manual.....	40
5.1	Overview.....	40
	.	
5.2	Library Usage.....	40
	5.2.1: how to include the library.....	40
	5.2.2: dealing with functions.....	40
Ch6	Conclusions And Future Work.....	44
6.1	Conclusions.....	44
6.2	Future work.....	44

References

List of figures

Fig.(2.1)	Filtering Techniques.....	9
Fig.(2.2)	Content Based filtering	10
Fig.(2.3)	Model-Based filter.....	16
Fig.(2.4)	Hybrid filtering.....	17
Fig.(2.5)	Static vs. Dynamic contextual changing	19
Fig.(2.6)	Common contexts for some domains.....	20
Fig.(2.7)	Context-aware dataset.....	22
Fig.(2.8)	Types of Context filtering.....	22
Fig.(2.9)	Tour planning interface.....	24
Fig.(2.10)	Recommend based on weather.....	25
Fig.(2.11)	Recommend based on mood.....	25
Fig.(2.12)	Cocare interface.....	26
Fig.(3.1)	System Architecture.....	27
Fig.(3.2)	Use case Diagram.....	29
Fig.(3.3)	Class Diagram.....	30
Fig.(3.4)	Sequence Diagram (request and access to database)	31
Fig.(3.5)	Sequence Diagram (generate initial recommendations)	32
Fig.(3.6)	Sequence Diagram (final recommendations)	32
Fig.(4.1)	Users Table.....	33
Fig.(4.2)	Items Table.....	33
Fig.(4.3)	Rating Table	33
Fig.(4.4)	Distinction Percentage Table	34
Fig.(4.5)	Joined Table	36
Fig.(4.6)	Similarity Matrix.....	37
Fig(5.1&5.2)	Main Interface.....	41

Chapter 1 – Introduction

1.1: Motivation:

With the ever-growing volume, complexity and availability of online information, recommender systems have been an effective key solution to overcome such information overload.

Recommendation systems help users find and select items (e.g., books, movies, restaurants) from the huge number available on the web, given a large set of items and a description of the user's needs.

Recent work in recommendation systems includes intelligent aides for filtering and choosing.

Recommendations provide a ranked list of items which predict what the most suitable products or services are, based on the user's preferences. To do so, they collect information from users regarding their preferences which are either explicitly expressed, such as ratings for products, or are inferred by interpreting the actions of the user.

1.2: Problem Definition:

Traditional recommender systems only consider the data of the users and their ratings to items. They offer the same recommendations to the user even with the change in the context of the recommendation. For example, the age of the user, his location, whether he is buying the gift for himself or another, time of recommendation etc.

This problem led to the emergence of Context-Aware Recommender Systems in which the context is any information that can be used to characterize the state of an item like Climate, Location, and Age.

By using the Context-Aware Recommender, recommendations are generating based on user's context.

1.3: Objective

The aim of our project is to develop a reusable component, implementing context aware recommender systems. The component is meant to be easily incorporated into a website or system.

This makes it easier for the user to be capable of recommending items to users based on a customizable set of contexts chosen by the site developer.

1.4: Document Organization:

The following chapters of this document are organized as follows:

- Chapter 2: Background

This chapter gives an overview of the scientific background behind the project. In this chapter we discuss the concept of recommender system, context and how to use them together (Context-aware recommender system).

- Chapter 3: Analysis and Design

This chapter presents the outcomes of the analysis and design phases of the project. It discusses the functional and non-functional requirements of the system and its architecture. This chapter also offers a detailed account of the functionalities and processes offered by the project expressed as UML diagrams.

- Chapter 4: Implementation

This chapter explains the technologies and tools used in the implementation of the system and the role of each of them in the project.

- Chapter 5: User Manual

The user manual offers a guideline on how to operate the system and make use of the different functionalities.

- Chapter 6: Conclusions and Future Work

The final conclusions of our work are presented in this chapter. Also, this chapter presents the possible directions for future work

Chapter 2 – Background

In this chapter we will introduce an overview of the scientific background behind the project.

2.1: Recommender Systems:

A recommender system or a recommendation system is a subclass of information filtering system that seeks to predict the "rating" or "preference" a user would give to an item.

"The goal of a recommender system is to generate meaningful recommendations to a collection of users for items or products that might interest them".

Recommender systems are utilized in a variety of areas including movies, music, news, books, research articles, search queries, social tags, and products in general. There are also recommender systems for experts, collaborators, jokes, restaurants, garments, financial services, life insurance, romantic partners (online dating), and Twitter pages.

Recommender systems typically produce a list of recommendations in one of two ways – through collaborative filtering or through content-based filtering (also known as the personality-based approach).

2.2: Filtering Techniques:

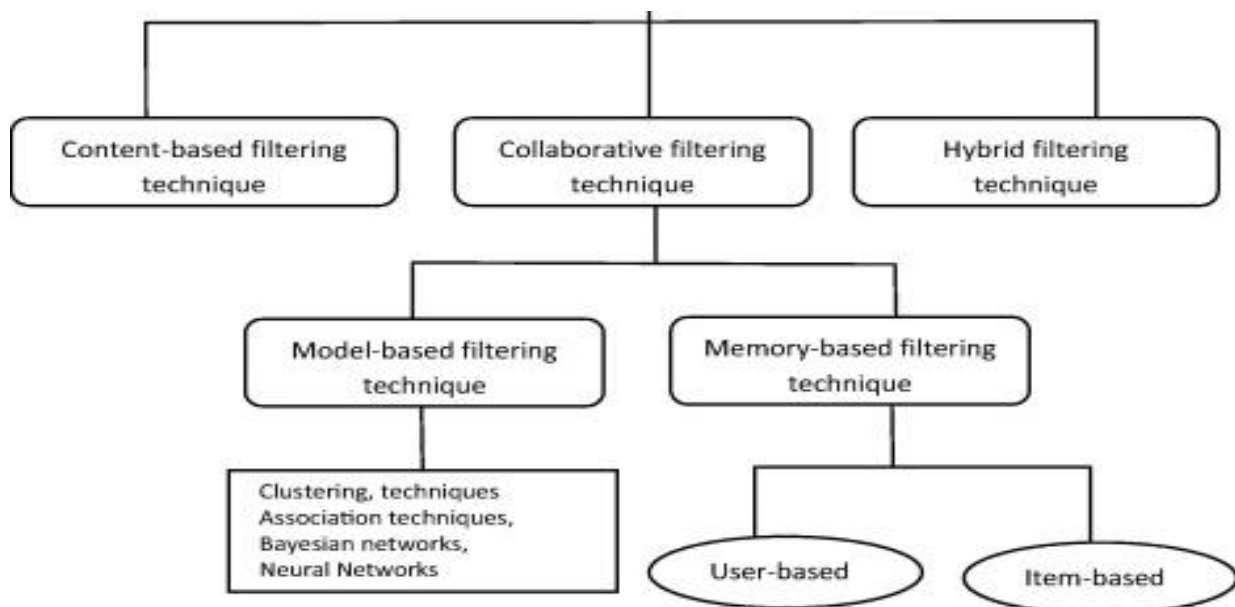


Figure (2.1) Shows the filtering techniques

2.2.1: Content based filtering

This type of filter does not involve other users if not us. Based on what we like, the algorithm will simply pick items with similar content to recommend us.

In this case there will be less diversity in the recommendations, but this will work either the user rates things or not. If we compare this to the example above, maybe user B potentially likes dark comedy, but he/she will never know, unless he/she decides to give it a try autonomously, because this filter will only keep recommending dystopian movies or similar. Of course, there are many categories we can calculate the similarity on: in the case of movies we can decide to build our own recommender system based on genre only, or maybe we want to include director, main actors and so on.

These are not the only two types of existing filters, of course. There are also cluster or behavioral based for example, just to mention a couple more.

So far, I have mentioned many times the word similarity, but what is it, exactly? It doesn't really seem something we can quantify, but surprisingly it can be measured! Before jumping into a practical example of how to build a content-based system, let's briefly review the concept of cosine similarity. This is one of the metrics that we can use when calculating similarity, between users or contents.

2.2.2: Collaborative filtering

Is a method of making automatic predictions (filtering) about the interests of a user by collecting preferences or taste information from many users (collaborating). The underlying assumption of the collaborative filtering approach is that if a person *A* has the same opinion as a person *B* on an issue, *A* is more likely to have *B*'s opinion on a different issue than that of a randomly chosen person.

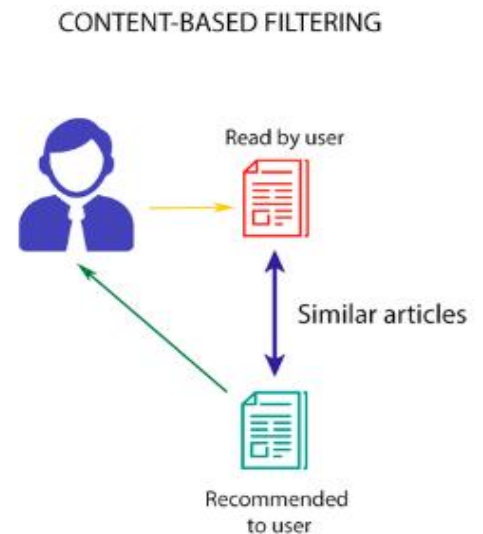
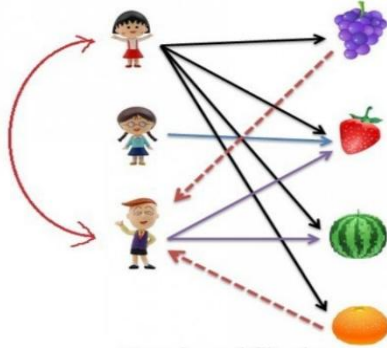
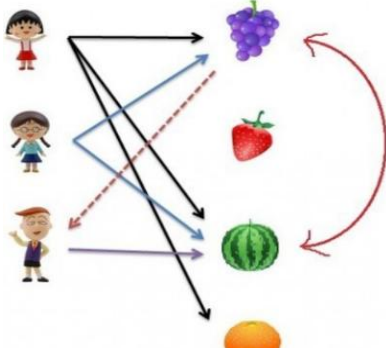


Figure (2.2)
Content-Based Filtering Example

2.2.2.1: Memory-based

The memory-based approach uses user rating data to compute the similarity between users or items. Typical examples of this approach are item-based and user-based top-N recommendations.

	<i>User-Based Filtering</i>	<i>Item-Based Filtering</i>
<i>Description</i>	It finds the X most similar users to you and use them for a basis of recommendation. In a threshold-based neighborhood, all users that fall within the threshold, i.e. are similar enough are used to provide recommendations.	Unlike user based collaborative filtering, item-based filtering looks at the similarity between different items, and does this by taking note of how many users that bought item X also bought item Y. If the correlation is high enough, a similarity can be presumed to exist between the two items, and they can be assumed to be like one another.
<i>Approach</i>		
<i>How similarity calculated</i>	<ol style="list-style-type: none"> 1- K-nearest neighbors 2- Euclidean distance 3- Pearson correlation 	<ol style="list-style-type: none"> 1- Cosine-Based Similarity 2- Correlation-Based Similarity 3- Adjusted Cosine Similarity 4- Jaccard distance
<i>Predication</i>	Predict the rating that user a will give to all items the k neighbors have consumed but a has not. We Look for the item j with the best predicted rating.	<ol style="list-style-type: none"> 1- Weighted Sum 2- Regression

2.2.2.2: Similarity and Predictions

The similarity measure is the measure of how much alike two data objects are. The similarity is subjective and is highly dependent on the domain and application. For example, two fruits are similar because of color or size or taste. Care should be taken when calculating distance across dimensions/features that are unrelated. The relative values of each element must be normalized, or one feature could end up dominating the distance calculation. **similarity is measured in the range 0 to 1 [0,1].**

2.2.2.2.1: Similarity measures:

Euclidean distance: is the most common use of distance. The Euclidean distance between two points is the length of the path connecting them. The Pythagorean theorem gives this distance between two points.

$$Sim(x, y) = d = \sqrt{\sum_{i=1}^n (X_i - Y_i)^2} \quad (2.1)$$

Pearson correlation: This method is used to find linear correlation between two vectors. PCC results a value between -1 and +1. -1 represents a negative co-relation while +1 represents high positive correlation. 0 value shows no relation sometimes called zero order correlation. For the user-based algorithm

$$PCC_Sim(u, v) = \frac{\sum_{i \in I} (r_{u,i} - \bar{r}_{u,I})(r_{v,i} - \bar{r}_{v,I})}{\sqrt{\sum_{i \in I} (r_{u,i} - \bar{r}_{u,I})^2 + \sum_{i \in I} (r_{v,i} - \bar{r}_{v,I})^2}} \quad (2.2)$$

Where $\bar{r}_{u,I}$ and $\bar{r}_{v,I}$ represents average rating of user u and user v respectively, for co-rated items represented by set I

Constrained Pearson correlation: Constrained Pearson correlation uses median value instead of average of rating co-rated by both users. Median value of scale [1-5] is 3. For the user-based algorithm.

$$CPCC_Sim(u, v) = \frac{\sum_{i \in I} (r_{u,i} - r_{mod})(r_{v,i} - r_{mod})}{\sqrt{\sum_{i \in I} (r_{u,i} - r_{mod})^2 + \sum_{i \in I} (r_{v,i} - r_{mod})^2}} \quad (2.3)$$

r_{mod} Median value for rating scale

Cosine similarity: is also most commonly used method in collaborative filtering in recommender systems. Cosine similarity finds how two vectors are related to each other using measuring cosine angle between these vectors. For the user-based algorithm, the major drawback with cosine similarity is that it considers null preferences as negative preference.

$$Cos_Sim(u, v) = \frac{\vec{R}_u \cdot \vec{R}_v}{|\vec{R}_u| \cdot |\vec{R}_v|} \quad (2.4)$$

Where “ \cdot ” represents dot product of two vectors. \vec{R}_u and \vec{R}_v are rating vectors of user u and v respectively.

Adjusted cosine similarity: Cosine similarity measuredoes not consider the scenario in which different users use different rating scale. Adjusted cosine similarity solves it by subtracting the average rating provided by the user u . Adjusted cosine similarity considers the difference in rating scale used by each user. Adjusted cosine similarity is slightly different from Pearson Correlation; Pearson Correlation considers the average rating of user u for co-rated the items. Adjusted cosine similarity subtracts the average rating of user u for all the items rated by user u . For the user-based algorithm.

$$Adjusted_Cos_Sim = \frac{\sum_{i \in I} (r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in I} (r_{u,i} - \bar{r}_u)^2 + \sum_{i \in I} (r_{v,i} - \bar{r}_v)^2}} \quad (2.5)$$

Where \bar{r}_i Average rating of user i for all items rated by user i itself.

Jaccard similarity: Jaccard similarity takes number of preferences common between two users into account. This does not consider the absolute ratings rather it considers number of items rated. Two users will be more similar, when two users have more common rated items.

$$Jaccard_Sim(u, v) = \frac{|I_u \cap I_v|}{|I_u \cup I_v|} \quad (2.6)$$

Where I_u is a set of items rated by user u and I_v is a set of items rated by user v .

Mean squared differences: MSD does not consider number of common rating rather it considers absolute ratings. Various similarity measures have been proposed in combination of Jaccard similarity as JMSD, JPSS. Jaccard and MSD similarity can be combined to form another similarity measure method JMSD.

$$MSD(u, v) = \sum_{i \in I} (r_{u,i} - r_{v,i})^2 \quad (2.7)$$

$$MSD_Sim(u, v) = \frac{L - MSD(u, v)}{L} \quad (2.8)$$

Where L is a threshold

Pip similarity: PIP stands for Proximity, Impact, Popularity. Proximity factor calculates the arithmetic difference between two ratings, along with consideration of agreement or disagreement of ratings, giving penalty to ratings in disagreement. Two ratings occur in agreement if they lie on one side of the median of the rating scale.

$$PIP_Sim(u, v) = \sum PIP(r_{u,i} - r_{v,i}) \quad (2.9)$$

$$PIP(r_{u,i}, r_{v,i}) = Proximity(r_{u,i}, r_{v,i}) \cdot Impact(r_{u,i}, r_{v,i}) \cdot Popularity(r_{u,i}, r_{v,i}) \quad (2.10)$$

New heuristic similarity model (NHSM): NHSM similarity measure is combination of JPSS and User Rating Preference similarity measures. User Rating Preference similarity measure is based on mean and variance of the ratings of user. For the user-based algorithm, the value produced by NHSM ranges from 0 to 1. This similarity measure considers the fact that different users have different preferences scale and model's user preference based on mean and standard variance of user ratings.

$$URP_{Sim}(u, v) = 1 - \frac{1}{1 + \exp(-|\bar{r}_u - \bar{r}_v| \cdot |\sigma_u - \sigma_v|)} \quad (2.11)$$

$$\sigma_u = \sqrt{\sum_{i \in I} \frac{(r_{u,i} - \bar{r}_u)^2}{|I_u|}} \quad (2.12)$$

$$NHSM_Sim(u, v) = JPSS_Sim(u, v) \cdot URP_Sim(u, v) \quad (2.14)$$

Where \bar{r}_i Average rating of user I for all items rated by user and σ_u and σ_v are the standard variance of user u and v respectively

Spearman rank correlation: Spearman Rank Correlation uses ranks instead of ratings for calculating similarity. Spearman Rank Correlation does not work well for partial orderings. Weak orderings occur whenever there are at least two items in the ranking such that neither item is preferred over the other. If there is difference between user ranking ordering and system ordering. When the system ranks same rated items at different levels, then the Spearman correlation will be penalized for every pair of items rated same by the user.

$$Spearman_sim(u, v) = 1 - \frac{6 \sum_{h_0}^{n_i} d_h^2}{n_i(n_i^2 - 1)} \quad (2.15)$$

Where d_h difference in ranks of items h co-rated by both users. n_i is Number of items co-rated by both users.

Kendall's tau correlation: This is also rank based method to compute correlation. Kendall's correlation considers relative ranks instead of ratings for calculating similarity.

$$Kendal_Tau = \frac{N_c - N_d}{\sqrt{(N_c + N_d + T_r)(N_c + N_d + T_p)}} \quad (2.16)$$

Where N_c is number of concordant pairs items that the system predicts in the proper ranked order. N_d Is the number of discordant pairs that the system predicates in the wrong order. T_r is number of pairs of items that the system predicts the true ordering while T_p is the number of pairs of items in the predicted ordering.

2.2.2.2.2: Prediction:

Weighted Sum As the name implies, this method computes the prediction on an item i for a user u by computing the sum of the ratings given by the user on the items like i . Each rating is weighted by the corresponding similarity $s_{i,j}$ between items i and j . We can denote the prediction $P_{u,i}$ as

$$P_{u,i} = \frac{\sum_{all \text{ similar items } N} (S_{i,N} * R_{u,N})}{\sum_{all \text{ similar items } N} (|S_{i,N}|)} \quad (2.17)$$

Basically, this approach tries to capture how the active user rates the similar items. The weighted sum is scaled by the sum of the similarity terms to make sure the prediction is within the predefined range.

Regression This approach is like the weighted sum method but instead of directly using the ratings of similar items it uses an approximation of the ratings based on regression model. instead of using the similar item N 's "raw" ratings values $R_{u,N}$'s, this model uses their approximated values $R_{u,N}$ based on a linear regression model. If we denote the respective vectors of the target item i and the similar item N by R_i and R_N the linear regression model can be expressed as

$$\bar{R}'_N = \alpha \bar{R}_i + \beta + \epsilon \quad (2.18)$$

2.2.2.3: Model-based

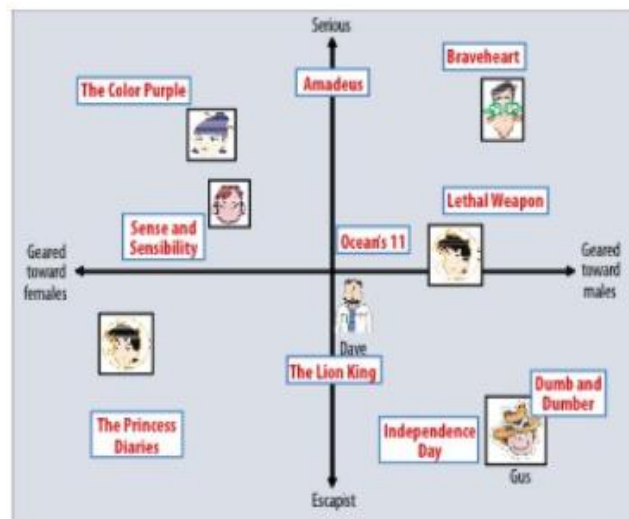


Figure (2.3) Model-Based Filtering

In this approach, models are developed using different data mining, machine learning algorithms to predict users' rating of unrated items.

There are many model-based CF algorithms.

Bayesian networks, clustering models, latent semantic models such as singular value decomposition, probabilistic latent semantic analysis, multiple multiplicative factor, latent Dirichlet allocation and Markov decision process based models.

There are several advantages with this paradigm. It handles the sparsity of the original matrix better than memory based ones. Also comparing similarity on the resulting matrix is much more scalable especially in dealing with large sparse datasets.

2.2.3: Hybrid filtering

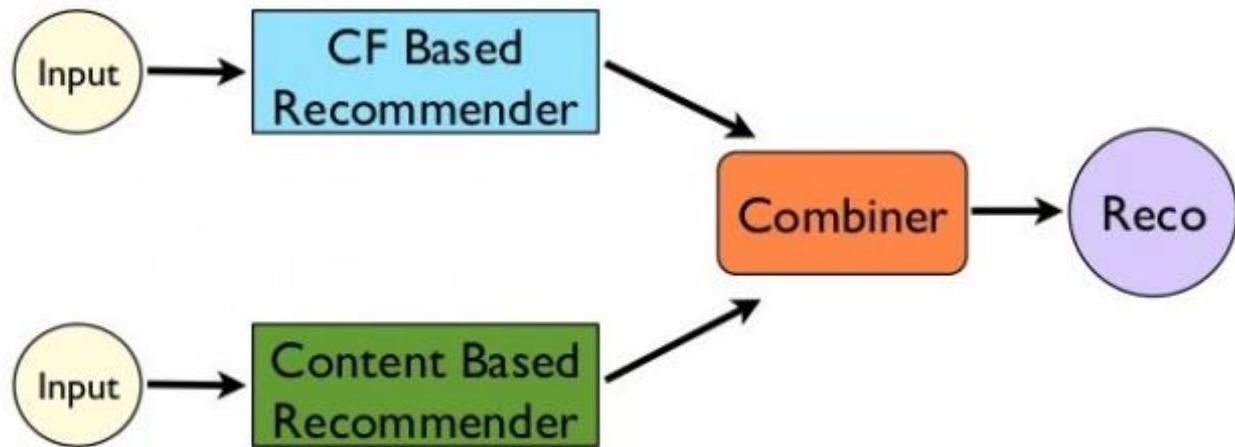


Figure (2.4) Hybrid Filtering

Hybrid filtering technique is a combination of Both content-based filtering and collaborative filtering have their strengths and weaknesses. Three specific problems can be distinguished for content-based filtering.

- Content description. In some domains generating a useful description of the content can be very difficult. In domains where the items consist of music or video for example a representation of the content is not always possible with today's technology.
- Over-specialization. A content-based filtering system will not select items if the previous user behavior does not provide evidence for this. Additional techniques must be added to give the system the capability to make suggestion outside the scope of what the user has already shown interest in.
- Subjective domain problem. Content-based filtering techniques have difficulty in distinguishing between subjective information such as points of views and humor.

2.3: Recommender Systems and their drawbacks:

A recommender system is a subclass of information filtering system that seeks to predict the "rating" or "preference" a user would give to an item. The goal of a recommender system is to generate meaningful recommendations to a collection of users for items or products that might interest them. Recommender systems are utilized in a variety of areas including movies, music, news, books, research articles, search queries, social tags, and products in general. There are also recommender systems for experts, collaborators, jokes, restaurants, garments, financial services, life insurance, romantic partners (online dating), and Twitter pages. Recommender systems typically produce a list of recommendations in one of two ways – through collaborative filtering or through content-based filtering.

But Sometimes the recommender systems don't recommend a good recommends which is not suitable due to time or user preferences. So, the context-aware recommended systems are introduced to solve this problem in which it take in consideration the current time, location, users preference, weather or companions and some other contexts. The context has solved the problem of not suitable recommendations and saves time for the user by helping them finding their needs rapidly.

2.4: Context:

One of the most cited definitions of context is the **definition** of AnindK. Dey, 2001 that defines context **as**:

any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.

How Contextual Factors Change	Knowledge of the RS about the Contextual Factors		
	Fully Observable	Partially Observable	Unobservable
Static	Everything Known about Context	Partial and Static Context Knowledge	Latent Knowledge of Context
Dynamic	Context Relevance Is Dynamic	Partial and Dynamic Context Knowledge	Nothing Is Known about Context

Figure (2.5) Static VS. Dynamic Contextual Changing

2.4.1: Context Types

Representative Context: Fully observable and Static.

Interactive Context: Non-fully observable and Dynamic.

Observed Context: Contexts are those variables which may change when a same activity is performed again and again.

Examples: Watching a movie: time, location, companion, etc.

Listening to a music: time, location, emotions, occasions, etc.

Party or Restaurant: time, location, occasion, etc.

Travels: time, location, weather, transportation condition, etc.

In the following **Table (Figure 2.8)**, we present the common application domains and their incorporated contexts. Based on

the review, the incorporated contextual information in the RS can be viewed as either formalizing

or identifying a place for the users and/or the items (spatial context) or characterizing a possessed

attribute of the users and/or the items. The possessed attribute can either be for a period of time

(temporal context) or permanent (static context).

Spatial contexts are those contexts that formalize the geographical situation or environment of users and/or items such as their location.

Static contexts are those contexts that do not change over time, and which affect the recommendation

process such as gender, age, and identity etc.

Temporal Contexts contrary to static context, are those contexts that change over time and which are

dynamic in nature like user's mood, user's current goal and social relations etc.

Table. Domains of application with their incorporated contexts.

Domain of Application	Incorporated Contexts
Travel and Tourism	Time, companion, location, vicinity, current situation, social relations, intent, seasonality, nationality, budget, expertise.
Places	Current time, location, companion, distance to an available point of interest, intent, nationality, current activity, current weather, user's mood, social relations, personal preferences, social influence.
Multimedia	Who, when, where, what, location, time, crowd, mood, companion, social, mental stress, weather, orientation, age, sensory data, gender, a user profile.
e-Documents	Activity, technology, location, environment, background, device, URL, gender, time of the day, age, previous logs, ISBN, title, publisher, author, keyword, abstract, introduction, main idea, conclusion, paper type, language.
e-Commerce	URL, age, gender, location, category, vicinity, mood, seasonality, current budget, previous logs, time, mental stress, intent of purchase, store.
Others	Time, seasonality, sequentially, role, geographical location

Figure (2.6) Common contexts for some domains

2.4.2: Context Acquisition

How to Collect the context and user preferences in contexts?

- **User Surveys or Explicitly Asking for User Inputs:** Predefine context & ask users to rate items in these situations; Or directly ask users about their contexts in user interface.
- **Usage data:** The log data usually contains time and location (at least); User behaviors can also infer context signals.

2.4.3: Context Relevance and Context Selection

Apparently, not all of the context are relevant or influential.

- **User Surveys:**
 - e.g., which ones are important for you in this domain
- **Feature Selection:**
 - e.g., Principal Component Analysis (PCA)
 - e.g., Linear Discriminant Analysis (LDA)
- **Statistical Analysis or Detection on Contextual Ratings:**
 - Statistical test, e.g., Freeman-Halton Test
- Other methods: information gain, mutual information, etc.

2.4.4: Context-aware Data Sets

Some Public Data Set for Research Purpose:

- **Food**
 - AIST Japan Food
 - Mexico Tijuana Restaurant Data
- **Movies**
 - AdomMovie
 - DePaulMovie
 - LDOS-CoMoDa Data
- **Music**
 - InCarMusic
- **Travel**
 - TripAdvisor, South
 - Tyrol Suggests (STS)
- **Mobile**
 - Frappe

Frappe is a large data set, others are either small or sparse

2.4.5: Context in Recommendation System

- **Traditional RS:** Users \times Items Ratings
- **Contextual RS:** Users \times Items \times Contexts Ratings

Example of Multi-dimensional Context-aware Data set:

Context Dimension: *time, location, companion*

Context Condition: *Weekend/Weekday, Home/Cinema*

Context Situation: *{Weekend, Home, Kids}*

User	Item	Rating	Time	Location	Companion
U1	T1	3	Weekend	Home	Kids
U1	T2	5	Weekday	Home	Partner
U2	T2	2	Weekend	Cinema	Partner
U2	T3	3	Weekday	Cinema	Family
U1	T3	?	Weekend	Cinema	Kids

Figure (2.7) Context-aware dataset

2.4.6: Context-aware Recommendation System (CARS)

There are three ways to build algorithms for CARS

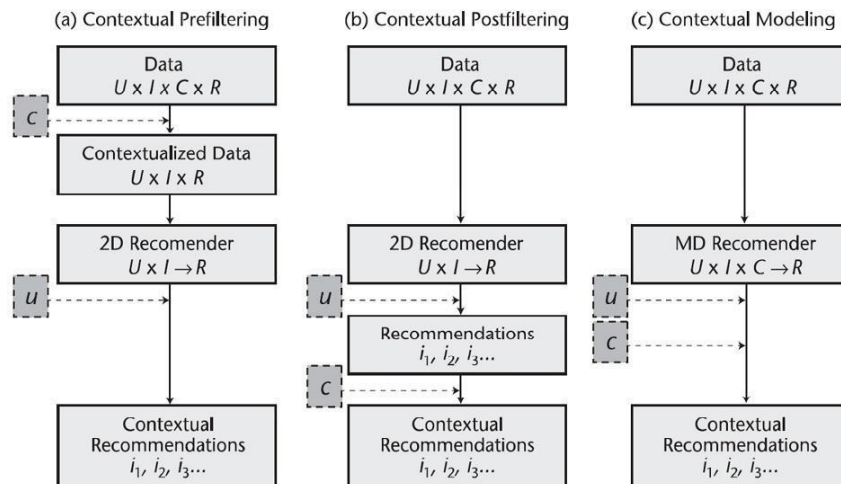


Figure (2.8) Types of context filtering

2.4.7: Other Challenges

There could be many other challenges in CARS:

Numeric Context

- **List of categorical Context:**
 - **Time:** morning, evening, weekend, weekday, etc
 - **Location:** home, cinema, work, party, etc
 - **Companion:** family, kid, partner, etc
- **How about numeric context:**
 - **Time:** 2016, 6:30 PM, 2 PM to 6 PM (time-aware recommendation systems)
 - **Temperature:** 12°C, 38°C
 - **Principle component by PCA:** numeric values
- **Other numeric values in context, how to develop CARS?**

Explanation

Recommendation Using context (Open Research):

- Similar thing could happen to context-aware recommendation system;
- How to use contexts to explain recommendations;
- How to design new user interface to explain;
- How to merge CARS with user-centric evaluations.

User Interface

New UI to collect context

Context Suggestion

A New Opportunity by CARS that enables new recommendation opportunities.

2.5: Related Works:

1. A Cold Start Context-Aware Recommender System for Tour Planning Using Artificial Neural Network and Case Based Reasoning^[1]:

- It is used in tour advising by considering the user interest for a given point of interest (POI), many additional contextual parameters such as user's own mobility, previous history, and the timing of a recommendation should be considered as important contextual information and they used artificial neural network and combine it with CBR to solve the cold start problem.



Figure (2.9) Tour Planning interface

2. A Context-Aware Recommender System for Personalized Places in Mobile Applications^[2]:

- The aim of the work is to make a context-aware recommender system, which recommends places to users based on the current weather, the time of the day, and the user's mood. This Context-aware Recommender System will determine the current weather and time of the day in a user's location. Then, it gets places that are appropriate to context state in the user's location. Also, Recommender system takes the current user's mood and then selects the location that the user should go. Places are recommended based on what other users have visited in the similar context conditions. Recommender system puts rates for each place in each context for each user. The place's rates are calculated by The Genetic algorithm, based on Gamma function. Finally, mobile application was implemented in the context-aware recommender system.

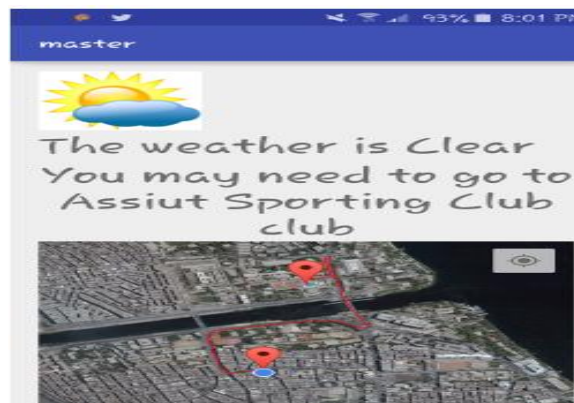


Figure (2.10) Recommend based on weather

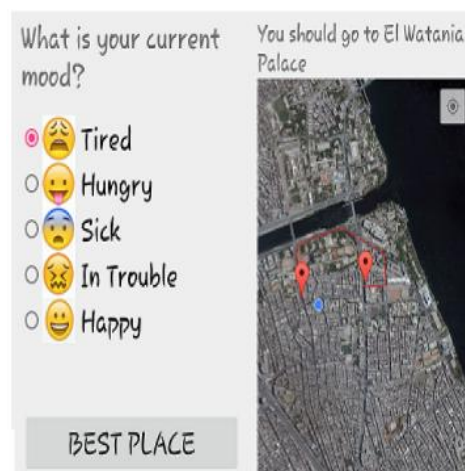


Figure (2.11) Recommend based on mood

3. Recommendation System based on CBR algorithm for the Promotion of Healthier Habits^[3]:

- Recommender systems in the health area have been proven as useful tools to help patient-oriented decision-making systems, promoting physical activity and disease prevention, in general, to improve health conditions through healthier habits. Health recommender systems (HRS) aim to promote health programs, to provide patients with relevant information, products or services, using knowledge about his/her personal health record systems.

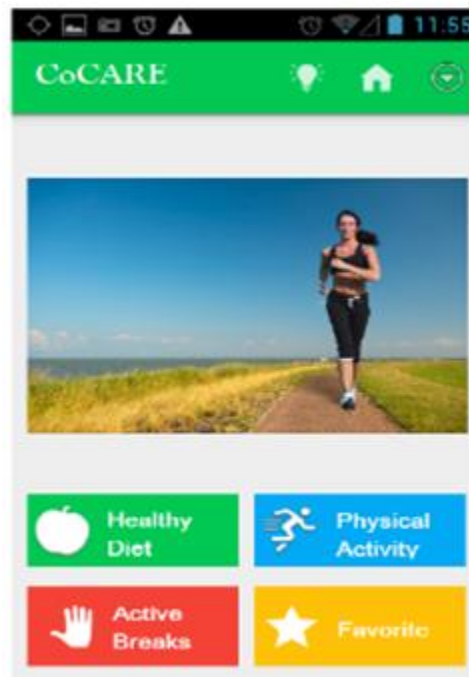


Figure (2.12) CoCare interface

Chapter 3- Analysis & Design

This chapter presents the analysis and design phases of the project. It discusses the functional and nonfunctional requirements of the system. This chapter also offers a detailed account of the functionalities and processes offered by the project expressed as a diagram.

3.1: System Overview

3.1.1: System architecture

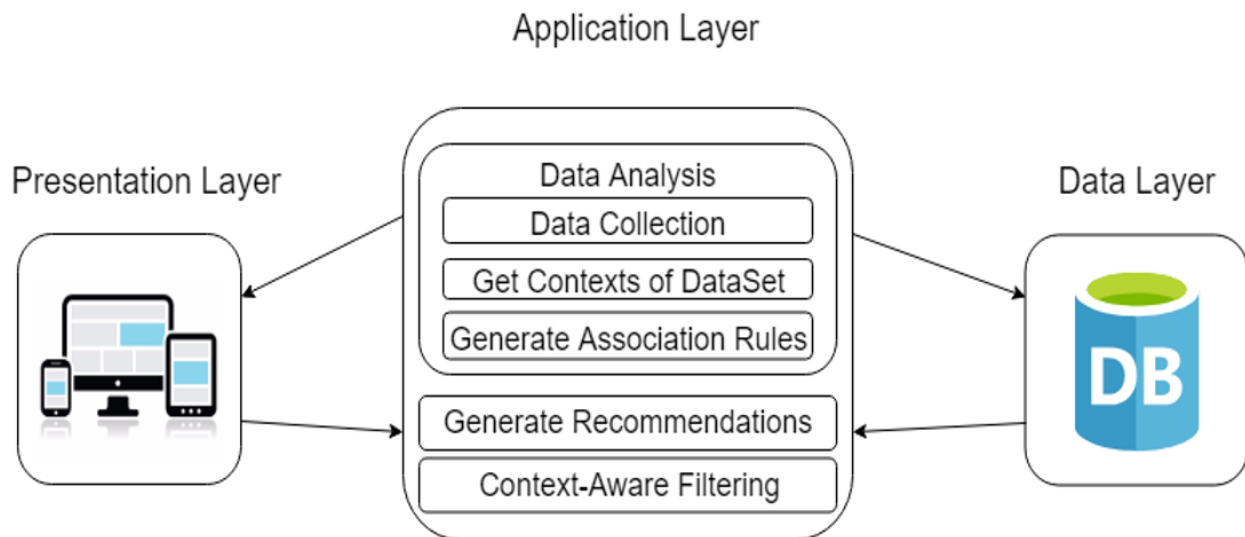


Figure3.1 System Architecture

Scenario:

- First, we ask to get access to website database and the site builder interacts with the interface to approve the access request by giving us the database name.
- Second, we access the database using its name and read all the data to get the table names and the columns names in the dataset and the site builder interact with the interface to specify the name of items data table , user data table , rating data table , primary and foreign keys in dataset , choose valid contexts to filter recommendations with it.
- Third, we get frequent item set, generate user-item matrix, get similarity between items.
- Fourth, we get top N recommendations based on user rating and the similarity between items.
- Fifth, we filter top N recommendations & unrated items based on the context.

3.1.2 Functional requirements:

- Get access to the website database & get dataset.
- Get all contexts related to dataset & recommend them to the site builder.
- Get frequent itemsets.
- Calculate similarity between items & use the to generate initial recommendations.
- Use contextual filtering to get final recommendations

3.1.3 Non-functional requirements:

- Usability requirements: The system is very easily operated and does not require special skills from its users.
- Adaptability requirements: As our system is able to fit its behavior according to changes in its environment or in parts of the system itself.
- Accessibility requirements: As the system can access any size of the data and can deal with big data.
- Interoperability requirements: As our system has an interface which is completely understood, to work with other products or systems, at present or in the future, in either implementation or access, without any restrictions.

3.1.4 System Users:

A. Intended Users:

1- Site Builder

B. User Characteristics:

- 1- Knowledge on data mining***
- 2- Know how to deal with databases***
- 3- Know how to deal with our system***

3.2: System Analysis & Design

3.2.1: Use case diagram

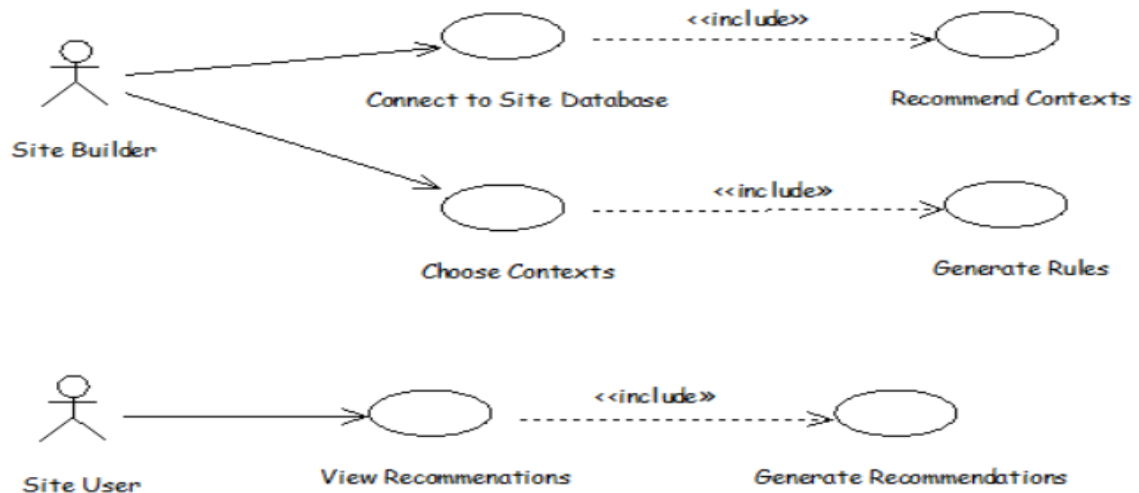


Fig. 3.2 Use Case Diagram

Use Case	Connect to Site Database
Brief Description	Site builder approve the request to access the database
Precondition	Request Sent
Post condition	Site builder provide our system the database name
Flow of Event	Primary flow: - Request sent to the site builder - Site builder approve it - Our system will access the database and retrieve its data Error flows: - Request not sent - Server down.

Use Case	View Recommendations
Brief Description	The top recommendations appear to the user
Post condition	There is top N recommendations for the user
Flow of Event	Primary flow: - Recommendations are displayed Alternate flows: No recommendations found

Use Case	Choose Contexts
Brief Description	All the contexts are displayed on the screen with their corresponding similarity percentage and the site builder choose the needed ones.
Precondition	Check that site builder has chosen any context.
Flow of Event	Primary flow: <ul style="list-style-type: none"> - The contexts displayed. - Site builder chose some of them. Error flows: <ul style="list-style-type: none"> - No context chosen.

3.2.2: Class diagram

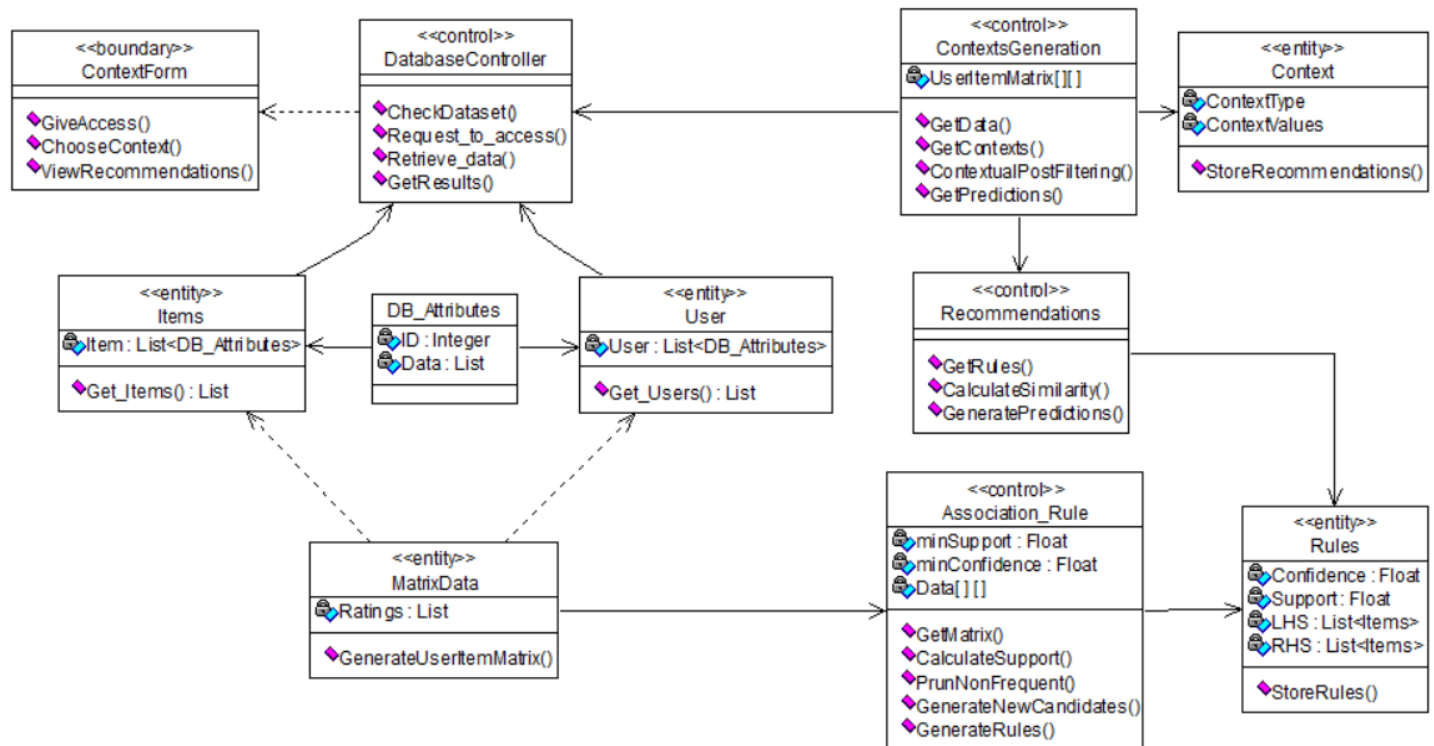


Fig. 3.3 Class Diagram

3.2.3: Sequence diagrams:

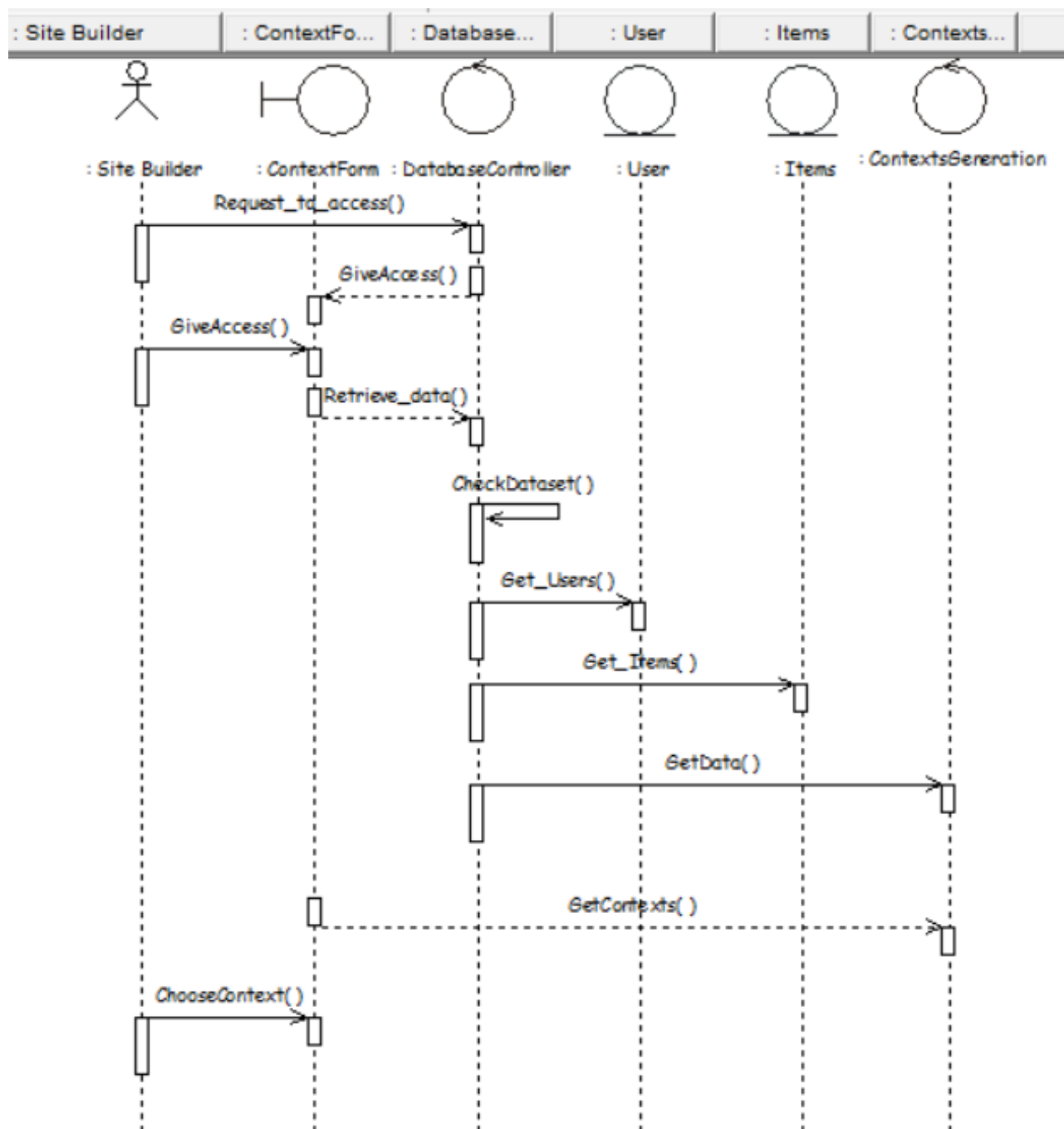


Fig. 3.4 Sequence Diagram (request and get access to database)

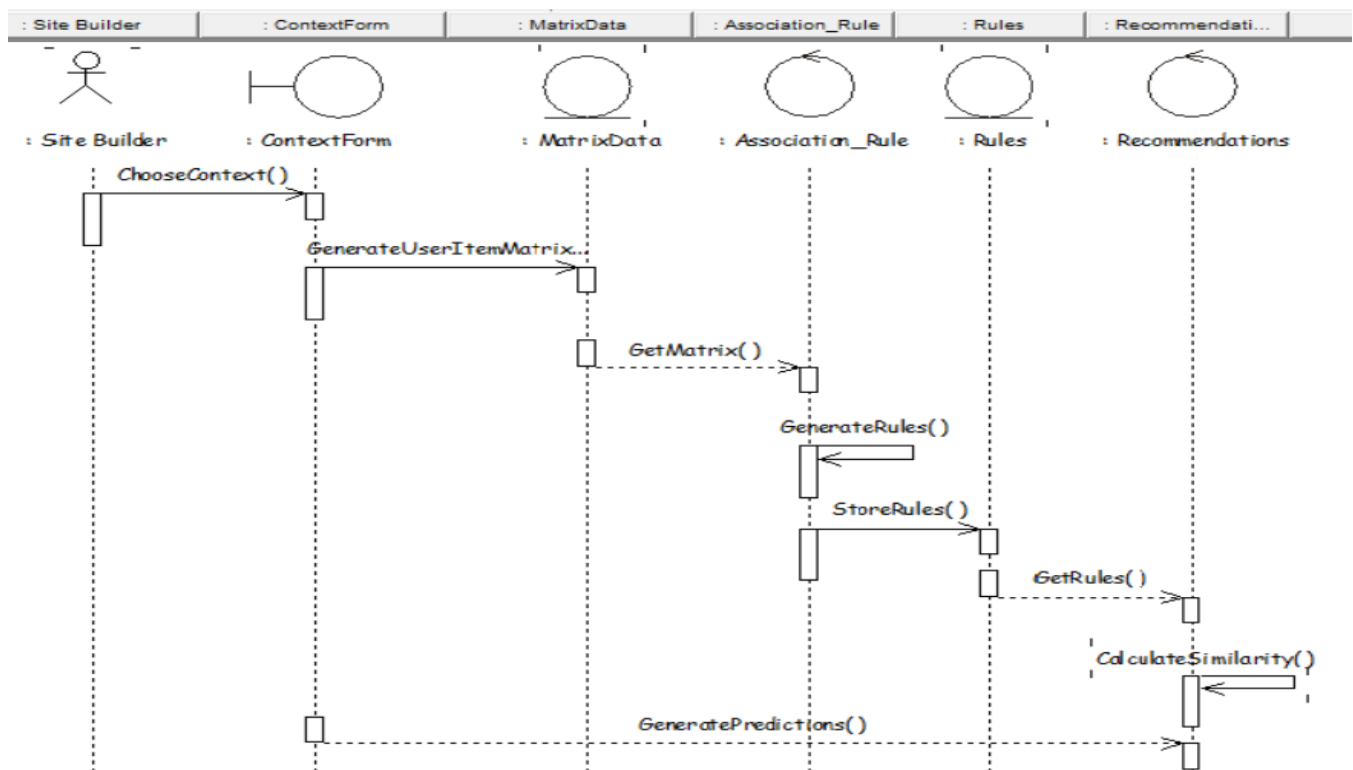


Fig. 3.5 Sequence Diagram (generate initial recommendations)

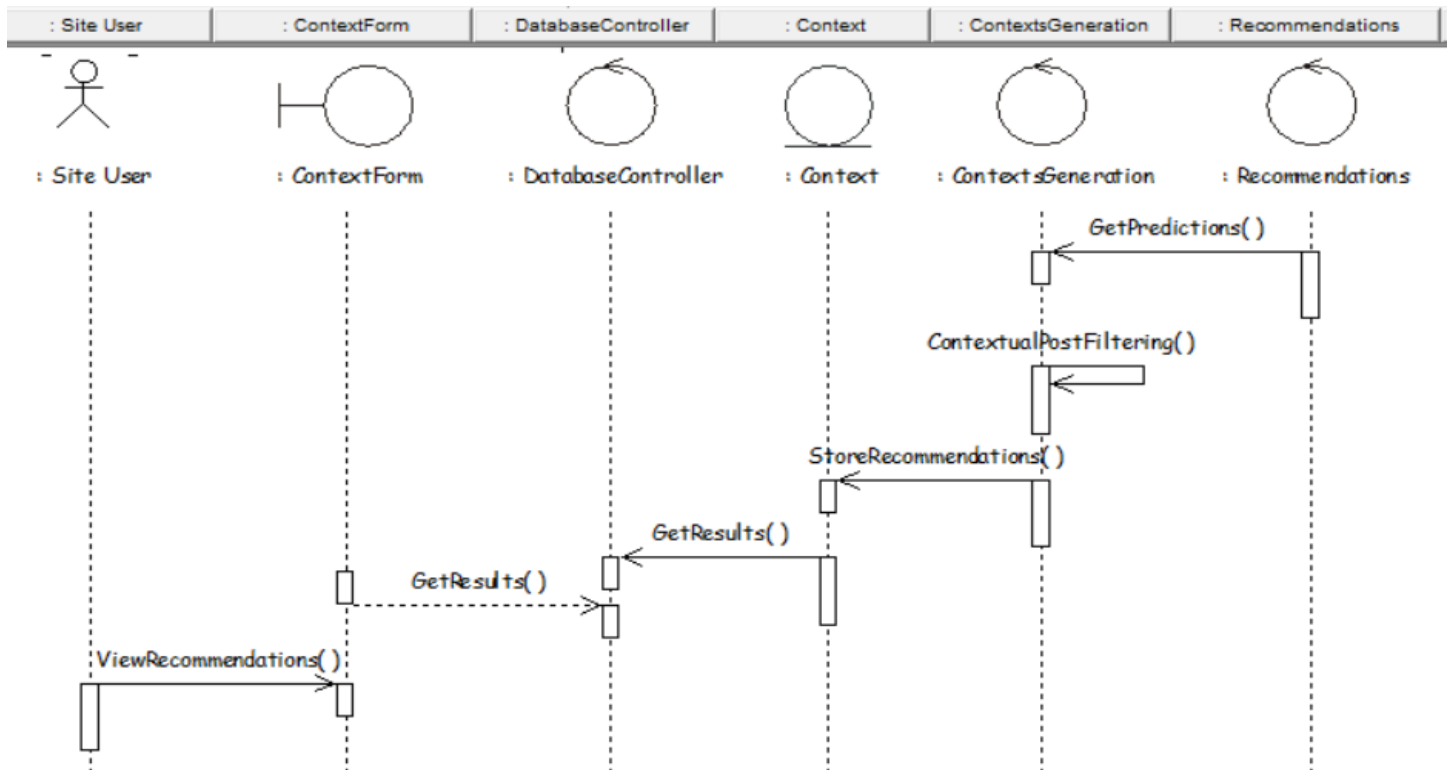


Fig. 3.6 Sequence Diagram (final recommendations)

Chapter 4–Implementation

This chapter explains the technologies and tools used in the implementation of the system and the role of each of them in the project and a simple case study for clarifying.

4.1: Collect Dataset

➤ **Retrieve_data function:**

- Firstly, the site builder has to enter the database name and the name of (items table, users table ,ratings table) and then identify the primary key for each table (user, item, rating) then identify the foreign keys.
 - We put items data in a list using Set_Items function.
 - We put users data in a list using Set_Users function.

UserID	Name	Age	Gender
1	Ahmed	20	Male
2	Ali	40	Male
3	Mona	65	Female
4	Jack	20	Male
5	Tala	40	Female

Figure (4.1) Users Table

HotelID	HotelName	Location	HotelRate
1	XYZ	Alexandria	5*
2	ABC	North Coast	3*
3	WER	Hurghada	4*
4	TYX	Luxor	5*
5	OLP	Alexandria	3*

Figure (4.2) Items Table

U_ID	I_ID	Rate	Season
1	1	8	Summer
2	2	3	Summer
3	2	9	Summer
1	5	4	Summer
3	3	10	Winter
5	4	2	Winter
4	4	10	Winter

Figure (4.3) Rating Table

4.2: User-Item Matrix

➤ GenerateUserItemMatrix:

- This function is used to formulate the user item matrix to help us in the collaborative filtering by putting all users with the given rate to a certain item so, as we calculate the similarity between items rated by the same user.

	U1	U2	U3	U4	U5
I1	8	-1	-1	-1	4
I2	-1	3	9	-1	-1
I3	-1	-1	10	-1	-1
I4	-1	-1	-1	10	-1
I5	-1	-1	-1	2	-1

4.3: Contexts Generation

- By using GetContexts function we will get all the names of columns and tables on the database that by using SQL statement.
- GetContexts function return list of column names.
- Calculate distinction percentage by using calculatePercentage function by
 - Get all distinct Items in column by using SQL statement.
 - Get number of rows in table by SQL statement.
 - Then divide distinct item by number of rows then multiply it by 100.
- calculatePercentage function return list of float number.

Context Name	#Distinct	#Values	Distinction Percentage
UserID	5	5	100%
Name	5	5	100%
Age	3	5	60%
Gender	2	5	40%
HotelID	5	5	100%
HotelName	5	5	100%
Location	4	5	80%
HotelRate	3	5	60%
U_ID	5	7	71.4%
I_ID	5	7	71.4%
Rate	6	7	85.7%
Season	2	7	28.6%

Figure (4.4) Distinction Percentage Table

4.4: Association Rules

- We are applying the association rules using Apriori algorithm in order to get relations between items and context.
- Apriori algorithm is applied by getting the frequency of all distinct items and calculate frequency for each of them if the frequency greater than or equal the minimum support so it is said to be frequent and non-frequent items is pruned.
- Create join table between all table in dataset by using CreateJoin function.
- Create Join function take parameters (names of item table, user table, ratings table, common column between table user and rating and common column between table item and rating).
- New table called ASSR is Created in the database.
- Then use the new table to create frequent item of length one by using SQL Statement by
 - Use each column in Table ASSR and get all the distinct records in each column if the column contain numbers it will be discretized by using the bin function which divide numbers into intervals each interval has min 10% of data , then write them in file called candidate1.txt.
 - Then by using SQL statement calculate how many times it repeated in table if it bigger than or equal min support write it in file called frequent1.txt
 - All of this done by function CreateCandidate1ANDFreq1 this function needs only the min support as a parameter.
- Then use file of frequent1 to get frequent of length-2 by
 - Firstly, generate candidate2 that by concatenate every line in file frequent1 with other lines that haven't same name of column
 - Then by SQL statement calculate how many times it repeated in table if it bigger than or equal min support write it in file called frequent2.txt.
 - All of this by function CreateCandidate2ANDFreq2 this function needs only the min support as a parameter.
- Then filter frequent of length-2 by context by scanning file frequent2.txt if line have any of chosen contexts and attribute from table item and attribute from table user or rating then write it in file called FinalRules.txt by using function FilterByContext

- Then get ten trending items by using function trending
 - This function need items table name and column of item name and min support and name of column items primary key
 - Arrange items based on their average rate
 - Then filter them by comparing then check if they are frequent if yes write its id in file called TrendingItem.txt.

HotelName	Location	HotelRate	Name	Age	Gender	Rate	Season
XYZ	Alexandria	5*	Ahmed	20	Male	8	Summer
OLP	Alexandria	3*	Ahmed	20	Male	4	Summer
ABC	North Coast	3*	Ali	40	Male	3	Summer
ABC	North Coast	3*	Mona	65	Female	9	Summer
WER	Hurghada	4*	Mona	65	Female	10	Winter
TYX	Luxor	5*	Jack	20	Male	10	Winter
TYX	Luxor	5*	Tala	40	Female	2	Winter

Figure (4.5) Joined Table

Rule 1: (Location=Alexandria) → (Season=Summer)

Rule 2: (Location=North Coast) → (Season=Summer)

Rule 3: (Location=Hurghada) → (Season=Winter)

Rule 4: (Location=Luxor) → (Season=Winter)

4.5: Cosine similarity:

Is a type of similarity measures that finds how two vectors are related to each other using measuring cosine angle between these vectors. Most commonly used method in collaborative filtering in recommender systems.

Which calculated as following: -

$$\text{Cos_Sim}(u, v) = \frac{\vec{R}_u \cdot \vec{R}_v}{|\vec{R}_u| \cdot |\vec{R}_v|} \quad (4.1)$$

Where “.” represents dot product of two vectors. \vec{R}_u And \vec{R}_v are rating vectors of user u and v respectively.

The probability must be between: -

$$0 \leq \text{Cos}_{\text{Sim}(u,v)} \leq 1 \quad (4.2)$$

➤ **Calculate Similarity function: -**

- Calculate similarity between two items (item based), So :-
 1. Get an item (t_1) from database && its rates and the next item (t_2) && its rates.
 2. For each rate in (t_1) r_{1i} , first-rate += (r_{1i}) * (r_{1i}), which $i=0, 1, 2, \dots$
 3. For each rate in (t_2) r_{2i} , second-rate += (r_{2i}) * (r_{2i}), which $i=0, 1, 2, \dots$
 4. If the user of two item is the same, Total rate += $r_{1i} * r_{2i}$
 5. Repeat step 3 till calculate all rates.
 6. Calculate equation: -

$$\text{Result} = \frac{\text{Total rate}}{\sqrt{\text{first-rate}} * \sqrt{\text{second-rate}}} \quad (4.3)$$

7. Repeat step 2 until calculate similarity between (t_1) and (t_j), which $j=3, 4, 5$
8. Repeat step 1 for each item.
9. Save similarity in file.

	I1	I2	I3	I4	I5
I1	1	0	0	0.98	1
I2		1	0.569	0.36	0
I3			1	0	0
I4				1	0
I5					1

Figure (4.6) Similarity Matrix

4.6: Initial Recommendations:

➤ GeneratePredictions: -

- *This function generates file contains top 10 recommendations for each user if exists.*

Steps: -

1. After calculating Cosine Similarity between each two items $t_{i,j}$ based on their rates, We got a similarity file contains the following form for each line: $t_i \quad t_j \quad \text{CosSim}(i,j)$ that We use to get initial recommendations.
2. For each user U_k who rated some items t_i we get their corresponding $\text{CosSim}(i,j)$ as a *rating metric* between U_k and t_j .
3. Then by descending Order, we sort those *rating metrics* for each U_k .
4. Thus, for each U_k we got the desired **TOP10** initial recommendations ranked by $\text{CosSim}(i,j)$ metric.

4.7: Final Recommendations:

➤ checkMatchedRules: -

- *This function is used to approve if the recommendation is matching based on the context or not.*

Steps: -

1. Get Item data
2. Get user data
3. Make them as one record
4. Search for a rule in FinalRules.txt that is matched with this record
5. Then return a Boolean value true if matched and false if not

➤ FinalRecommendations: -

- *This function is used to assign recommendations to the users by*
 1. Filtering the initial recommendations which exist in top10.txt using FinalRules.txt to filter it according to user and item context if the user has 10 items as recommendations so it is the final recommendations for him then we go to step 6 if not, we go to the step 2.
 2. Using TrendingItem.txt we filter them by context as we filter the initial recommendations if the user has 10 items as recommendations, so it is the final recommendations for him then we go to step 6 if not, we go to the step 3.
 3. Using UnRated.txt we filter them by context as we filter the initial recommendations if the user has 10 items as recommendations, so it is the final recommendations for him then we go to step 6 if not, we go to the step 4.
 4. Using NoRates.txt we filter them by context as we filter the initial recommendations if the user has 10 items as recommendations, so it is the final recommendations for him then we go to step 6 if not, we go to the step 5.
 5. If we finish all the previous steps and any user has no recommendations so we will put the trending items as final recommendations to him
 6. After finishing that we have a dictionary contains userID as a key and list of itemIDs so we write this into a FinalRecommendations.txt

Case study continuing: - when we have a user who need a hotels recommendations so we refer to the items similarity matrix to see the similar items to which he have rated previously and get the most similar items as initial recommendations for him then filter them using FinalRules.txt to have final recommendations.

Chapter 5–User Manual

5.1: Overview:

Our product is a reusable component in form of a library executed asdls to be included in any project and call certain functions and giving them some parameters as we will introduce how to use them in the following sections and our output is displayed in files which contain the user id and the items which is recommended for him.

5.2: Library Usage:

5.2.1: How to include the library: -

1. In Visual Studio, click Project, and then Add Reference.
2. Click the Browse tab and locate (ClassLibrary3.dll).
3. Click OK.
4. Add this to your code "using ClassLibrary3;"

5.2.2: Dealing with functions: -

1. Firstly, the site builder must type the database name in the text box so, as we can access it and get all table names and column names and you have to call the constructor of DatabaseController class so as to formulate the SQL connection string.
 - i. DatabaseController db = new DatabaseController(Database name, Path of folder to put the output files on it);

2. After typing the database name all the combo boxes will be filled with the database columns or tables and the list box on the left will be filled with the distinction similarity for each column. To fill list box, you have to call those function as shown below

- i. ContextsGeneration context = new ContextsGeneration();
- ii. List<String> colum = context.GetContexts();
- iii. List<float> percentage = context.calculatePercentage();

Please fill in all the desired data

Enter Database Name Choose Name of Users Table

Choose Name of Items Table Choose Name of Ratings Table Set minimum support

User Table PK

Items Table PK Item Name Column

FK of Items in Rating Table

FK of Users in Rating Table

Column name of Ratings

BookAuthor,82.2%
BookTitle,100%
ImageURLL,100%
ImageURLM,100%
ImageURLS,100%
ISBN,100%
Publisher,56.8%
YearOfPublication,2.2%
BookRating,7%
ISBN,82%
UserID,35%
Age,19.2429%
Location.88.2%

Clear Chosen

Click To Start

Figure (5.1) main interface

3. Then fill all the boxes using boxes and click start button
 - i. When we choose the table names so the combo boxes of the pk and fk is filled with the columns included in the table with the given name
 - ii. Minimum support is used in Apriori algorithm to get frequent items
 - iii. When we click on any of the contexts it will be added into the list of contexts used by the system

Please fill in all the desired data

Enter Database Name Choose Name of Users Table

Choose Name of Items Table Choose Name of Ratings Table Set minimum support

User Table PK

Items Table PK Item Name Column

FK of Items in Rating Table

FK of Users in Rating Table

Column name of Ratings

BookAuthor,82.2%

BookTitle,100%

ImageURLL,100%

ImageURLM,100%

ImageURLS,100%

ISBN,100%

Publisher,56.8%

YearOfPublication,2.2%

BookRating,7%

ISBN,82%

UserID,35%

Age,19.2429%

Location,88.2%

YearOfPublication

Publisher

Location

Age

Figure (5.2) main interface

4. Now all the desired data is successfully filled.

5. You must call those function with this sequence to generate recommendations

a. Retrieve Data in Class Database Controller

Example: -

- i. DatabaseController db = new DatabaseController();
- ii. db.Retrieve_data(users, items);

b. Then to formulate the rules you must use Association Rules class

Example: -

- i. AssociationRules Apriory = new AssociationRules();
- ii. Apriory.CreateJoin(item table name, user table name, rating table name, fk of user in rating, fk of item in rating, pk of user, pk of item);
- iii. Apriory.CreateCandaite1ANDFreq1(minSupport);
- iv. Apriory.CreateCandaite2ANDFreq2(minSupport);
- v. Apriory.FilterByContext(list of contexts chosen by site builder, Apriory.numfile);
- vi. Apriory.trending(item table name, item name, minSupport, pk of item);

c. Then to generate user item matrix you must use

Example: -

- i. MatrixData matrix = new MatrixData();
- ii. matrix.GenerateUserItemMatrix(Rating table name , Rating column name, fk of user in rating, fk of items in rating);

d. Then to get the recommendations

- i. Recommendations recommmed = new Recommendations();
- ii. recommmed.CalculateSimilarity(Rating Table name, Rating column name, fk of user in rating, fk of item in rating);
- iii. recommmed.GeneratePredictions();
- iv. recommmed.FinalRecommendations(User table name, pk of user, Item table name, pk of items);

Chapter 6–Conclusion and Future work

6.1: Conclusions:

The final conclusions of our work are presented in this chapter.

Recommender systems are a powerful new technology for extracting additional value for a business from its user databases. Recommender systems open new opportunities of retrieving personalized information on the Internet.

In context aware recommender system, the delivery of many software and software entities is affected by different context information. The achievement of business goals can be improved.

Our system is a reusable component, implementing context aware recommender systems so the component is meant to be easily incorporated into a website or system. The system based on customizable set of contexts so makes it easier for the user to be capable of recommending items to users.

6.2: Future Work:

Implement visualization techniques which can increase understanding of the input and output of a recommender system.

Implement clustering techniques to improve the scalability of the recommender system.

References: -

1. Zahra Bahramian, Rahim Ali Abbaspour, and Christophe Claramunt “**A Cold Start Context-Aware Recommender System for Tour Planning Using Artificial Neural Network and Case Based Reasoning**”, *Mobile Information Systems*, 2017.
2. SohaA.El-MoemenMohamed ,Taysir Hassan A.Soliman , Adel A.Sewisy “**A Context-Aware Recommender System for Personalized Places in Mobile Applications**”, (*IJACSA*) *International Journal of Advanced Computer Science and Applications*, 7(3), 2016.
3. Gineth M Ceron-Rios, Diego M Lopez-Gutierrez,BelenDaz-Agudo, and Juan A. Recio-Garca“**Recommendation System based on CBR algorithm for the Promotion of Healthier Habits**”, In *Proceedings of the ICCBR 2017 Workshops*, 2017.
4. Agarwal RC, Aggarwal CC, Prasad VVV (2001) A tree projection algorithm for generation of frequent item sets. *J Parallel Distrib Compute* 61(3):350–371 CrossRefzbMATH
5. Aggarwal CC (2014) An introduction to Frequent Pattern Mining. In: Aggarwal CC, Han J (eds) *Frequent Pattern Mining*. Springer, Basel, pp 1–14
6. Aggarwal CC, Bhuiyan MA, Hasan MA (2014) Frequent Pattern Mining algorithms: a survey. In: Aggarwal CC, Han J (eds) *Frequent Pattern Mining*. Springer, Basel, pp 19–64
7. Agrawal R, Srikant R (1994) Fast algorithms for mining association rules. In: Paper presented at the proceedings of the 20th international conference on very large data bases, Santiago
8. B. Shumeet, R. Seth, D. Sivakumar, Y. Jing, J. Yagnik, S. Kumar, D. Ravichandran, M. Aly, Video suggestion and discovery for YouTube: taking random walks through the view graph, in: *International Conference on World Wide Web*, 2008, pp. 895–904.
9. E. Brynjolfsson, Y.J. Hu, M.D. Smith, Consumer surplus in the digital economy: estimating the value of increased product variety at online booksellers, *Manage. Sci.* 49 (11) (2003) 1580–1596.
10. J.S. Breese, D. Heckerman, C. Kadie, Empirical analysis of predictive algorithms for collaborative filtering, in: *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, 1998, pp. 43–52.
11. F. Cacheda, V. Carneiro, D. Fernández, V. Formoso, Comparison of collaborative filtering algorithms: limitations of current techniques and proposals for scalable, high-performance recommender system, *ACM Trans. Web* 5 (1) (2011) 1–33.

12. M.J. Pazzani, D. Billsus, Content-based recommendation systems, The Adap. Web (2007) .
13. H. Junming, C. Xueqi, G. Jiafeng, S. Huawei, Y. Kun, Social recommendation with interpersonal influence, ECAI 10(2010) 601–606.
14. <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6152957/>
15. <https://towardsdatascience.com/how-to-build-from-scratch-a-content-based-movie-recommender-with-natural-language-processing-25ad400eb243>
16. <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6171847/>