2019

Discovering E-commerce Sequential Data Sets and Sequential Patterns for Recommendation

Raj Bhatta
University of Windsor

Discovering E-commerce Sequential Data Sets and Sequential Patterns for Recommendation

By

Raj Bhatta

A Thesis
Submitted to the Faculty of Graduate Studies
through the School of Computer Science
in Partial Fulfillment of the Requirements for
the Degree of Master of Science at the
University of Windsor

Windsor, Ontario, Canada

2019

Discovering E-commerce Sequential Data Sets and Sequential Patterns for Recommendation

By

Raj Bhatta

APPROVED BY:

A. Sarker

Department of Mathematics & Statistics

S. Saad

School of Computer Science

C. Ezeife, Advisor

School of Computer Science

April 5, 2019

DECLARATION OF ORIGINALITY

I hereby certify that I am the sole author of this thesis and part of this thesis has been submitted to Big Data Analytics and Knowledge Discovery-DAWAK19 for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

ABSTRACT

In E-commerce recommendation system accuracy will be improved if more complex sequential patterns of user purchase behavior are learned and included in its user-item matrix input, to make it more informative before collaborative filtering. Existing recommendation systems that use mining techniques with some sequences are those referred to as LiuRec09, ChoiRec12, SuChenRec15, and HPCRec18**.** LiuRec09 system clusters users with similar clickstream sequence data, then uses association rule mining and segmentation based collaborative filtering to select Top-N neighbors from the cluster to which a target user belongs. ChoiRec12 derives a user's rating for an item as the percentage of the user's total number of purchases the user's item purchase constitutes. SuChenRec15 system is based on clickstream sequence similarity using frequency of purchases of items, duration of time spent and clickstream path. HPCRec18 used historical item purchase frequency, consequential bond between clicks and purchases of items to enrich the user-item matrix qualitatively and quantitatively. None of these systems integrates sequential patterns of customer clicks or purchases to capture more complex sequential purchase behavior.

This thesis proposes an algorithm called HSPRec (Historical Sequential Pattern Recommendation System), which first generates an E-Commerce sequential database from historical purchase data using another new algorithm SHOD (Sequential Historical Periodic Database Generation). Then, thesis mines frequent sequential purchase patterns before using these mined sequential patterns with consequential bonds between clicks and purchases to (i) improve the user-item matrix quantitatively, (ii) used historical purchase frequencies to further enrich ratings qualitatively. Thirdly, the improved matrix is used as input to collaborative filtering algorithm for better recommendations. Experimental results with mean absolute error, precision and recall show that the proposed sequential pattern mining-based recommendation system, HSPRec provides more accurate recommendations than the tested existing systems.

Keywords: Sequential pattern mining, collaborative filtering, historical recommendation system, sequence product recommendation, techniques for E-commerce recommendation

DEDICATION

I would like to dedicate this thesis to my parents (Mr. Durga Prasad Bhatta and Mrs. Saraswoti Bhatta), sisters (Mrs. Sajana Bhatta and Mrs. Sushila Bhatta), supervisor (Dr. Christie Ezeife) and my friends who have helped and supported to complete my graduate study at the University of Windsor.

ACKNOWLEDGEMENT

I would like to give my sincere appreciation to my parents and sisters for their continuous support and motivation throughout my graduate studies.

I would like to express my sincere gratitude to my advisor Prof. **Dr. Chrisite Ezeife** for her continuous support throughout my graduate study. She always provided me a chance to grow and further enhance research skills by providing a chance to participate and present a paper from our WODD lab to Big Data Analytics and Knowledge Discovery (**DAWAK 2018**) conference in Germany, Regensburg from 3rd of September 2018 to 7th of September 2018. Thank you so much for your valuable time to read all my thesis updates and providing me financial support through Research Assistantship (R.A.) throughout my study.

Besides my advisor, I would like to thank my thesis committee: Prof. **Dr. Animesh Sarker (**external reader**)**, Prof. **Dr. Sherif Saad (**internal reader**)** and Prof. **Dr**. **Asish Mukhopadhyay (**Chair**)** for their insightful comments and encouragement.

Finally, I would express my appreciations to all my friends and colleagues at the University of Windsor, especially Mrs. Sravya Vangala and Ms. Mahreen Nasir Butt for their support and encouragement. Thank you all.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF EQUATIONS

CHAPTER 1: INTRODUCTION

Recommendation systems provide a suggestion of items to the user in various decision-making processes such as what item to buy, what movies to watch, what music to listen to what online news to read **(Ricci, Rokach, & Shapira, 2011)**. The main goal of the recommendation system is to generate meaningful recommendations to a user for items that might interest them. One of the important applications of recommendation systems is in the e-commerce domain. Recommendation system in e-commerce helps to model the business process through analysis of customer requirements or their purchase behaviors **(Schafer, Frankowski, Herlocker, & Sen, 2007)**. Recommendation systems use data mining technologies such as classification, clustering, association rule mining, frequent pattern mining, and sequential pattern mining to generate a meaningful representation of user purchase data (**Han, Pei, & Kamber, 2011)**.

Traditionally, collaborative filtering was one of the most widely used recommendation technique, and it depends on explicit rating of items provided by users, but many users may not be ready to provide the items ratings. To resolve the rating problem, some implicit rating techniques **(Choi, Keunho, Yoo, Kim, & Suh, 2012)** derived from user behaviors (for example, purchases, clicks) across E-commerce and clickstream data analysis techniques (**Kim, Yum, Song, & Kim, 2005**), **(Liu, Lai, &Lee,2009)** are used. However, users purchase behaviors are always dynamic in nature and purchase of items may be different in each purchase. So, one of the main challenges in the field of recommendation system is to integrate sequential patterns of purchases with collaborative filtering because collaborative filtering finds closest neighbors between users or items without considering i) sequential purchase patterns ii) click and purchase behaviors iii) possible reasons for changes in user purchase habits. Various recommendation techniques such as collaborative filtering, content-based, and hybrid collaborative filtering approaches have been developed. While Collaborative filtering (CF) does not take into account the properties of the items but uses only the preference (rating or voting) provided by users for items, the content-based approach makes recommendation based on the user profiles (such as age, class) and product features (such as price, product attributes). These user or item features serve as contents that can be modeled to discover the relationship between different items similarity values using Vector Space Model such as Term Frequency Inverse Document Frequency (TF-IDF), or Probabilistic models such as Naïve Bayes Classifier, Decision Trees or Neural Networks extracted from those contents. Hybrid approach allows recommendation both collaborative filtering and content-based approach to be used for

recommendation and can serve to solve the cold start problem when there is no rating information for by a user on an item. However, such approaches suffer from a major drawback because they are not able to capture the E-commerce domain with sequential information of customer purchase behavior. Furthermore, sequential data may be available in a historical form, clickstream form. So, one of the main challenges in E-commerce recommendation is to generate the best recommendation suggestions from historical or clickstream sequential data to capture customer shopping behavior with respect to time.

1.1 Sequential Pattern

Sequential patterns are ordered set of items (events) that are occurring with respect to time (**Agrawal & Srikant, 1996**). A sequential pattern is denoted in the angular bracket (< >), and each itemset contains sets of items, where each item enclosed in parenthesis ( ) separated by commas represents a set of items purchased at the same time. For example, E-commerce sequential pattern < (Bread, Milk), (Bread, Milk, Sugar), (Milk), (Tea, Sugar)> means customer bought Bread and Milk together on first purchase, then bought Bread, Milk, and Sugar together on second purchase, then bought Milk on third purchase, and finally, bought Tea and Sugar together on fourth purchase. A sequential pattern with n-itemsets is called an n-events sequence. For example, if we consider only 2-itemsets, then we will have 2-events sequence such as <(Bread), (Milk)> or < (Bread), (Tea, Milk)>. Additionally, an item can occur at most once in an event (itemset) but can occur multiple times in different events (itemsets) within the same sequential pattern. Thus, the number of instances of items in a sequence is called the length of a sequence. For example, < (Bread, Milk), (Bread, Milk, Sugar), (Milk), (Tea, Sugar)> is 4-events sequence with length 8.

1.2 Sequential database

Sequence database is composed of a collection of sequences {$s_1, s_2, \ldots, s_n$} that are arranged with respect to time (**Han, Pei & Kamber, 2011**). A sequence database can be represented as a tuple **<SID, sequence-item sets>**, where SID: represents the sequence identifier and sequence-item sets specifies the sets in item enclosed in parenthesis ( ). For example, let us consider an example of E-commerce historical daily purchase data of grocery store as shown in **Table 1.1**, which contains **CustomerID** to represents a customer, **PurchasedItem** to represents a set of purchase items by customers and **Timestamp** to represents a time when purchased occurred.

CustomerID	PurchasedItem	Timestamp
01	Bread, Milk	13, Dec 2018 00:48:44
02	Bread	14, Dec 2018 1:48:44
01	Bread, Milk, Sugar	18, Dec 2018 10:48:44
02	Sugar, Tea	21, Dec 2018 09:48:44
01	Milk	19, Dec 2018 00:48:44
01	Tea, Sugar	22, Dec 2018 00:48:44

Table 1.1: E-commerce historical data

The daily sequential database created from historical data (**Table 1.1**) is present in **Table 1.2**, where SID represents the sequence identity. As we can see in **Table 1.2**, **SID(01)** contains <(Bread, Milk),(Bread, Milk, Sugar),(Milk),(Tea, Sugar)>, which means customer (01) first purchased Bread and Milk together then purchased Bread, Milk and Sugar together in second purchase and Milk in third purchase. Finally, Tea and Sugar together at last purchase.

SID	Sequences
01	<(Bread, Milk),(Bread, Milk, Sugar),(Milk),(Tea, Sugar)>
02	<(Bread),(Sugar, Tea)>

Table 1.2: Daily sequential database created from historical purchase

1.3 Sequential Pattern Mining

Sequential pattern mining algorithm (for example, Generalized Sequential Pattern (GSP) **(Agrawal & Srikant, 1996)**) discover repeating patterns (known as frequent sequences) from input E-commerce historical sequential database that can be used later to analyze the user purchase behavior by finding the association between items. In other words, it is a process of extracting sequential patterns whose support exceeds a predefined minimum support threshold. Formally, Given (i) a set of sequential records (called sequences) representing a sequential database D, (ii) a minimum support threshold (iii) a set of k unique items or events $I = \{i_1, i_2, \ldots, i_k\}$, the problem of mining sequential patterns is of finding the set of all frequent sequences S in the given sequence database D of items I at the given minimum support. The details of different types of sequential pattern mining algorithms are present in section **2.2**.

Example of sequential pattern mining using GSP algorithm

GSP (Generalized Sequential Pattern) is an Apriori-like sequential pattern mining algorithm (**Agrawal & Srikant, 1996**) which counts supports for each k-sequence in the candidate k-sequence (C_k) to find frequent k-sequence (F_k) after a pruning step to remove sequences not

meeting the Apriori property. The Apriori property is used to prune candidate sequential patterns whose subsets are not already frequent in earlier rounds as these patterns cannot be frequent and there is no need to scan the database for their support count. The GSP algorithm then generates candidate (k+1)-sequences from (F_k) sequences as F_k GSP-join F_k. The algorithm iterates between the candidate generate and prune step, and support count step until either a C_m or an F_n step generates an empty set. Details about the GSP-join operation are illustrated further through an example. Let us, consider daily sequential database (**Table 1.3**) as input, **minimum support**=2 and **candidate set (C_1)** = {A, B, C, D, E, F, G}.

SID	Sequences
1	<(A),(B),(FG),(C),(D)>
2	<(B),(G),(D)>
3	<(B),(F),(G),(A,B)>
4	<(F),(A,B),(C),(D)>
5	<(A),(B,C),(G),(F),(D,E)>

Table 1.3: Sequence database representing customer purchase

Step 1: Find 1- frequent sequence (L_1) to keep only sequence with occurrence or support count in the database greater than or equal to minimum support. For example, L_1= {<(A):4>, <(B):5>, <(C):3>, <(D):4>, <(F):4>, <(G):4>}.

Step 2: Generate candidate sequence ($C_{k=2}$) using $L_1 \bowtie_{GSPjoin} L_1$

To generate larger candidate set 2, use 1-frequent sequences found in step 1, which can be written as L ($_{k-1}$) $\bowtie_{GSPjoin}$ L ($_{k-1}$) and it requires every sequence (W_1) found in first L ($_{k-1}$) joins with other sequence (W_2) in the second, if subsequences obtained by removal of first element of W_1 and last element of W_2 are same. In our case, the possible 2-length candidate ($C_{k=2}$) sets generated using $\bowtie_{GSPjoin}$ are present in **Table 1.4.**

<(A),(A)>	<(A),(B)>	<(A),(C)>	<(A),(D)>	<(A),(F)>	<(A),(G)>
<(B),(A)>	<(B),(B)>	<(B),(C)>	<(B),(D)>	<(B),(F)>	<(B),(G)>
<(C),(A)>	<(C),(B)>	< (C),(C)>	<(C),(D)>	<(C),(F)>	<(C),(G)>
<(D),(A)>	<(D),(B)>	<(D),(C)>	<(D),(D)>	<(D),(F)>	<(D),(G)>
<(F),(A)>	<(F),(B)>	<(F),(C)>	<(F), (D)>	<(F),(F)>	<(F),(G)>
<(G),(A)>	<(G),(B)>	<(G),(C)>	<(G),(D)>	<(G),(F)>	<(G),(G)>
<(A,B)>	<(A,C)>	<(A,D)>	<(A,F)>	<(A,G)>	<(B,C)>
<(B,D)>	<(B,F)>	<(B,G)>	<(C,D)>	<(C,F)>	<(C,G)>
<(D,F)>	<(D,G)>	<(F,G)>			

Table 1.4: Candidate set (C_2) generated from L_1 GSP join L_1

Step 3: Find 2- frequent sequences (L_2) by counting occurrence of 2-sequences in candidate sequence (C_2) to keep only sequence with occurrence or support count in the database greater than or equal to minimum support. For example,

L_2=

<(A), (B)>	<(A, B)>	<(A), (C)>	<(A), (D)>	<(A), (F)>	<(A), (G)>
<(B), (C)>	<(B), (D)>	<(B), (F)>	<(B), (G)>	<(C), (D)>	<(F), (A)>
<(F), (B)>	<(F), (C)>	<(F), (D)>	<(G), (D)>		

Table 1.5: 2-frequent sequences

Step 4: Generate candidate sequence ($C_{k=3}$) using $L_2 \bowtie_{GSPjoin} L_2$

Use same candidate generation technique used in Step 2. An example of two sequences merged is present in **Table 1.6**.

W_1 Sequence	W_2 Sequence	Merged Sequence
<~~(A)~~,(B)>	<(B),~~(C)~~>	<(A), (B), (C)>
<(A), (B,C)>	<(B,C), (D)>	<(A), (B,C), (D)>

Table 1.6: Example to demonstrate merging of two sequences in GSP

Step 5: Find 3- frequent sequences (L_3) to keep sequences with occurrence or support count in the database greater than or equal to minimum support. For example, L3= {< (F), (C), (D)>, < (B), (G), (D)>, < (B), (F), (D)>, < (B), (C), (D)>, < (A), (G), (D)>, < (A), (F), (D)>, < (A), (C), (D)>}.

Step 6: Repeat process of candidate generation and pruning until result of candidate generate (C_k) and prune (L_k) for finding frequent sequence is an empty set.

Output: Finally, the output frequent sequences are union of L_1 U L_2 U L_3 U L_4

1-Frequent Sequences	2-Frequent Sequences	3-Frequent Sequences	4-Frequent Sequences
<(A)>, <(B)>, <(C)>, <(D)>, <(F)>, <(G)>	<(A), (B)>, < (A, B)>, <(A), (C)>, <(A), (D)>, <(A), (F)>, <(A), (G)>, <(B), (C)>, <(B), (D)>, <(B), (F)>, <(B), (G)>, <(C), (D)>, <(F), (A)>, <(F), (B)>, <(F), (C)>, <(F), (D)>, <(G), (D)>	<(F), (C), (D)> <(B), (G), (D)> <(B), (F), (D)> <(B), (C), (D)> <(A), (G), (D)> <(A), (F), (D)> <(A), (C), (D)> <(A), (B), (G)> <(A), (B), (F)> <(A), (B), (D)>	<(A), (B), (G), (D)> <(A), (B), (F), (D)>

Table 1.7: n-frequent sequences generated by GSP algorithm

1.4 E-commerce Data Types

1.4.1 E-commerce historical data

E-commerce historical data consists of a list of items clicked and/or purchased by a user over a specific period of time. A fragment of E-commerce historical database data is present in **Table 1.8** with schema {Uid, Click, Clickstart, Clickend, Purchase, Purchasetime}, where Uid represents User identity, Click represents a set of items clicked by a user, Clickstart and Clickend represent the timestamp when user started clicking item and when click is terminated. Furthermore, Purchase contains a set of items purchased by a user and Purchasetime represents timestamp when purchase happened.

Uid	Click	Clickstart	Clickend	Purchase	Purchasetime
1	1,2,3	2014-04-04 11:25:14	2014-04-04 11:45:19	1, 2	2014-04-04 11:30:11
1	7,5,3	2014-04-05 15:30:07	2014-04-05 15:59:36	3	2014-04-05 15:56:32
2	1, 4	2014-04-13 4:01:11	2014-04-13 4:30:15	1, 4	2014-04-13 04:04:34
2	1, 2,5, 6	2014-04-17 11:30:18	2014-04-17 11:50:19	1, 2,5, 6	2014-04-17 11:44:55
3	5	2014-04-23 11:00:05	2014-04-23 11:20:15	5	2014-04-23 11:06:37
4	6,6,7	2014-04-26 9:45:11	2014-04-26 10:20:13	6, 7	2014-04-26 10:06:37
5	1,5	2014-04-27 16:30:25	2014-04-27 16:45:45	?	

Table 1.8: E-commerce historical data

1.4.2 E-commerce clickstream data

Clickstream data represents the visitors' paths through E-commerce sites. A series of E-commerce pages visited by a user in a single visit is referred to as a session. Clickstream data in an E-commerce environment is a collection of sessions. Clickstream data can be derived from raw page requests (referred to as hits) and their associated information (such as timestamp, IP address, URL, status, number of transferred bytes, referrer, user agent, and, sometimes, cookie data) recorded in Web server log files (**Bucklin & Sismeiro, 2009**). Analysis of clickstreams shows how an E-commerce site is navigated and used by E-commerce users. In an E-commerce environment, clickstreams in online stores provide information essential to understanding the effectiveness of marketing and merchandising efforts, such as how customers find the store, what products they see, and what products they buy. Analyzing such information embedded in clickstream data is critical to improve the effectiveness of recommendation in online stores. An example of E-commerce Clickstream data is present in **Table 1.9**.

Session ID	Timestamp	ItemID	CategoryID
*****Ef4d7	2018-08-24T22:38:13+00:00	2145456502	3
*****Ef4d7	2018-08-24T20:38:12+00:00	21453650011	4
*****Ef4d7	2018-08-24T23:38:10+00:00	214536503	1
*****KM5M7	2018-08-24T22:38:14+01:00	2145775612	2
*****KM5M7	2018-08-24T22:38:14+03:03	2146627421	4
*****KM5M7	2018-08-24T22:38:14+04:05	214662742	6
*****KM5M7	2018-08-24T22:38:14+05:07	214825110	3

Table 1.9: Clickstream E-commerce data

The clickstream data given in **Table 1.9**, consists of session ID (*****Ef4d7, *****KM5M7) which represents user identity, timestamp (2018-08-24T23:38:10+00:00) represents the time when item visited, ItemID (2146627421, 214662742) represents the item visited by the user and CategoryID represents a category (e.g., milk belongs to dairy category) where items belong.

1.5 Consequential Bond (CB)

E-commerce data contains information's of clicks and purchases referred to as a consequential bond, and it is introduced by **Xiao and Ezeife, 2018** in their HPCRec18 system. The term consequential bond is originated from the concept that customer who will click some items will ultimately purchase an item from a list of clicks in most of the cases. For example, historical data present in **Table 1.8** shows that user 1 clicked items {1, 2, 3} and ultimately purchased {1, 2}; thus, there is a relationship between click and purchase.

1.6 Types of E-commerce Recommendation Systems

Based on how recommendations are made, recommender systems are usually classified into three categories:

1. Content-based filtering (CBF): It is based on the analysis of the attributes of items to generate predictions (**Ekstrand, Riedl & Konstan, 2011**). In other words, a recommendation is made based on the user profiles using features extracted from the content of the items the user has evaluated in the past. The CBF uses different types of models to find a similarity to generate a meaningful recommendation. The similarity could use Vector Space Model such as Term Frequency Inverse Document Frequency (TF-IDF) or Probabilistic models such as Naïve Bayes Classifier, Decision Trees or Neural Networks to model the relationships. The major

disadvantage of this technique is the need to have in-depth knowledge and description of the features of the items in the profile.

2. Collaborative filtering (CF): Collaborative filtering (CF) does not take into account the properties of the items but only the preference (rating or voting) provided by users for items **(Aggarwal & Charu, 2016)**. Thus, CF predicts rating of items using either a user-based or item-based approach. The user-based CF is based on the similarity between users, and items and item-based CF is based on the similarity between items and items. The similarity is computed by using one of the similarity measures such as (Cosine similarity, Pearson Correlation Coefficient and Jacquard similarity) then these similarity values are used to predict the unknown ratings of a user on an item using Top-N neighbors. The major problems of CF are cold start, sparsity, and scalability.
3. Hybrid filtering: Both CF and CBF have their benefits and demerits; therefore, if we combine both of them together, then the benefits of both can be used to overcome the demerits of others (**Kumar & Fan, 2015**). For example, CF provides recommendations using rating matrix now what happens when there is no rating given by a user (new user) then in such case the contents of user-item (CBF filtering) can be used with CF for recommendations.

1.7 Collaborative Filtering in E-commerce

Collaborative filtering makes a recommendation to a target customer based on the purchase behavior of customers whose preference is similar to a target customer. It is one of the widely used recommendation technique. Given a user-item rating matrix-R (such as **Table 1.10**),

	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6	Mean rating
User A	7	6	7	4	5	4	33/6
User B	6	7	?	4	3	4	24/5
User C	?	3	3	1	1	?	8/4
User D	1	2	2	3	3	4	15/6
User E	1	?	1	2	3	3	10/5

Table 1.10: User-item matrix for illustration of user based collaborative filtering

where a value of matrix is a rating $r_{u_j i_k}$, where u_j represents user j as in $\{u_1, u_2, \ldots, u_j\}$ and i_k represents item k as in $\{ i_1, i_2, \ldots, i_k\}$. Furthermore, the rating can be either an explicit rating or an implicit rating. Goal of collaborative filtering is to predict unknown rating r_{ui} of $user_u$ on $item_i$ through following four major steps **(Aggarwal & Charu, 2016)**:

1. Compute the mean rating for all user u_j using all of their rated items.
2. Calculate the similarity between the target user (v) and all other users u_j. Similarity can be computed with Cosine Similarity (v,u_j) or Pearson Correlation Coefficient function.
3. Find similar users of the target user (v) as Top-N users'.
4. Predict rating for the target user (v) for item i using only rating of v's Top-N peer group.

There are two types of collaborative filtering, user based and item based collaborative filtering. User-based collaborative filtering takes the ratings from similar users of the target user whereas item-based collaborative filtering considers the ratings from similar items of the target item.

1.7.1 User- based collaborative filtering

User-based collaborative filtering uses ratings of similar users for making a recommendation to a target user. The necessary algorithm of user-based collaborative filtering along with a running example is given below:

Input: user-item rating matrix R, containing $r_{u_j i_k}$, where u_j represents user j as in {u_1, u_2, . . . , u_j} and i_k represents item k as in { i_1, i_2, . . . , i_k}.

Output: Predicted ratings for previously unknown rating.

Major steps of collaborative filtering using user-based neighborhood method (**Aggarwal & Charu, 2016**) are:

1. Compute the mean rating for all user u_j using all of their rated items.
2. Calculate the similarity between the target user (v) and all other users u_j. Similarity can be computed by Cosine Similarity or Pearson Correlation coefficient function as given in **Equation 1.1** and **Equation 1.2**.

$$\text{Cosine (u, v)}= \frac{\vec{u}.\ \vec{V}}{||u||.||v||} = \frac{r_{u1}\cdot r_{v1} + r_{u2}\cdot r_{v2} + \ldots + r_{un}\cdot r_{vn}}{\sqrt{{r_{u1}}^2+{r_{u2}}^2+\cdots+{r_{un}}^2} * \sqrt{{r_{v1}}^2+{r_{v2}}^2+\cdots+{r_{vn}}^2}}$$

Equation 1.1: Formula to Compute Cosine similarity

In **Equation 1.1**, r_{u1} represents rating of user u on item 1, and r_{v1} represents rating of user v on item 1 respectively.

$$\text{Pearson Correlation (u, v)}= \frac{\sum_{i\in I}(r_{ui}-\overrightarrow{r_u})*(r_{vi}-\overrightarrow{r_v})}{\sqrt{\sum_{i\in I}(r_{ui}-\overrightarrow{r_u})^2} * \sqrt{\sum_{i\in I}(r_{vi}-\overrightarrow{r_v})^2}}$$

Equation 1.2: Formula to compute Pearson Correlation coefficient

Where r_{ui} represents the rating given by user u on item i and $\overrightarrow{r_u}$ is mean rating of user u and formula to compute mean rating is present in **Equation 1.3**.

$$\text{Mean rating } (\overrightarrow{r_u}) = \frac{\sum_{i \in I} r_{ui}}{|Number\ of\ items|}$$

Equation 1.3: Equation to compute mean rating

3. Find similar users of the target user (v) as Top-N users'.
4. Predict rating of target user (v) for item i using only ratings of v's Top-N peer group.

Example of user based collaborative filtering

Let us consider user-item rating matrix (**Table 1.10**) as **input** and our goal is to predict a rating of User C on Item 1 using collaborative filtering.

Step 1: Compute the mean rating for User A, User B, User C, User D, and User E using all of their rated items

For, User1= 33/6=5.5, User 2=24/5=4.8, User 3=8/4=2, User 4=15/6=2.5 and User 5=10/5=2

Step 2: Compute similarity between User C and others users

The similarity between User C and all others users can be computed using Cosine similarity or Pearson-Correlation Coefficient. In our case, we have used Cosine similarity, which is present in **Equation 1.1**. For example, SIM (User A, User C) $=\frac{6*3+7*3+4*1+5*1}{\sqrt{6^2+7^2+4^2+5^2}*\sqrt{3^2+3+1^2+1^2}}$ = 0.956. Similarly, SIM (User B, User C) =0.981, SIM (User D, User C) =0.789 and SIM (User E, User C) =0.645.

Step 3: Select the Top-N (in our case N=2) neighbor of User C by comparing similarity

Select Top-N neighbor of User C by comparing Cosine similarity. In our case, User A and User B have the highest similarity with User C. So, they are selected as Top-N neighbors.

Step 4: Compute the raw rating value using Top-N users (User A and User B)

To compute raw rating, Top-N users rating on item are used. For example, Raw rating$_{\text{User-C, item1}}$ is calculated by using rating for of User A on Item 1 and rating of User B on Item 1.

Raw rating $_{\textbf{User-C, item 1}}$ = $\frac{7*0.956+6*0.981}{0.956+0.981}$ = 6.49

Raw rating $_{\textbf{User-C, item 6}}$= $\frac{4*0.956+4*0.981}{0.956+0.981}$ = 4

Step 5: Compute mean centric rating

From above raw ratings, we can see that Item 1 should be prioritized over item 6 to recommend to User C. Furthermore, the prediction suggests that User C is likely to be interested in both Item 1 and Item 6 to a greater degree than other items. Thus, mean centric rating needs to be computed to remove this biased. The mean centric rating helps to reduce the influence caused by high and low rating provided by users on items. For example, mean centric rating of User A on Item 1 is

computed by subtracting rating of User A on Item 1 and mean rating of User A (in our case, 7-5.5=1.5).

Mean centric rating $_{\textbf{User-C, item 1}}$=2+ $\frac{1.5 * 0.956 + 1.2 * 0.981}{0.956 + 0.981}$ = 3.35

Mean centric rating $_{\textbf{User-C, item 6}}$= 0.86

There are some fundamental issues with collaborative filtering; they are:

(1) **Cold start**: When new items or new users appear in the database, these items may not be rated by any users; thus, preferences of users' may be unknown.

(2) **Sparsity issue**: When known rating data takes only a very small proportion in the user-item rating matrix, for instance, the amount of products is usually billions in the real world and most of the users only purchased probably hundreds of them, which leads to confusing and compromised recommendations. To address the sparsity issues, in this thesis, we have used sequential patterns of click and/or purchase to derive a rule to provide the relationship between already clicks or purchased items and recommended items to fill the missing rating for an item to improve the user-item matrix quantitatively (providing possible value for the unrated item or 0 value item in user-item matrix).

(3) **Scalability issue**: As the numbers of users and products grow rapidly, the time complexity and space complexity issues become more prominent.

1.8 Goal of E-commerce Recommender System

1) Converting browser to buyer: In an E-commerce environment, large amounts of information are available, so it can be hard for a user to find the product they are looking for. Thus, recommender systems help consumers to find products they intend to buy **(Schafer, Konstan, & Riedl, 2010)**.

2) Increasing cross-sell products: Recommender systems can improve the cross-sales product ratio by suggesting additional products. In general, a recommender system suggests products based on the customer's cart and purchase history.

3) Building loyality between customer and vendors: E-commerce recommender systems use different data sources according to different profiles. Consumers repay these sites by returning to the ones that best match their needs. This relation is mutually beneficial for when consumers return to the site, as they experience a more accurate degree of personalization, thus strengthening the bond between the online store and the client.

1.9 Need of Sequential Purchase Data in E-commerce Recommendation

1) User purchase habit changes with time: Collaborative filtering (CF) methods make a recommendation to a target customer based on the purchase behavior of other customers whose preferences are similar to those of the target customer. Thus, CF cannot capture the changes in purchase behavior of the customer over time, and integrating sequential rule in E-commerce can capture the customer purchase behavior over time.

2) Integrating frequency, price factor in recommendation: Traditional collaborative filtering technique, only consider the rating of an item for making a recommendation. Only considering the rating factor cannot provide a good recommendation to users because user choice may depend on product quantity, price and overall profit gained from purchased.

3) Taking care of timing factor during E-commerce recommendation generation: In E-commerce, some users may purchase items regularly, while other users may purchase items irregularly. So, recommendation generation by considering irregular users may provide a wrong recommendation to regular users.

1.10 Data Mining

Data mining is a process of turning raw data into useful information. It is the process of knowledge discovery (KDD) from raw data (**Han, Pei, & Kamber, 2011**), (**Fayyad, Piatetsky-Shapiro, Smyth, & Uthurusamy, 1996**). The KDD process include 1) data selection (find necessary data), 2) data pre-processing (which integrates target data from various sources and cleans target data by removing noise and inconsistent data), 3) data transformation (which summarizes or aggregates the pre-processed data into appropriate forms), 4) pattern evaluation and knowledge interpretation (representation or visualization of these interesting patterns discovered). Some of unsupervised data mining techniques are clustering, association rule mining (derived from frequent pattern mining and sequential pattern mining), and supervised data mining techniques is classification. Data mining is closely related to the area of statistics called exploratory data analysis and also related to the subareas of artificial intelligence called knowledge discovery and machine learning but handles much larger data in an automated fashion with more focus on database algorithms. Machine learning algorithms focus on classifications and clustering with more simulations but not including association rule techniques. Data mining tools are built to be embedded into the business

data warehouse and to be understandable and usable by marketing professionals, while classic statistical tools cannot fulfill these objectives.

1.10.1 Clustering

Clustering is a process of grouping a set of related objects in such a way that objects in the same group are similar to each other (**Jain & Dubes, 1998**). It is an unsupervised data mining technique that can automatically divide the data into a set of clusters or groups of similar items. The K-means clustering (**Hartigan & Wong, 1979**) is one of the widely accepted clustering approaches in the field of data mining (**Steinbach, Karypis, & Kumar, 2000).** K-means clustering is used, when we have data without defined categories or groups and goal of this algorithm is to find groups in the data, with the number of groups represented by the variable K. The algorithm works iteratively to assign each data point to one of K groups based on the features that are provided. The K-means clustering algorithm consists of four major steps:

1. Randomly pick centroid from available objects. Let us consider, we do have **n** objects $\{I_1, I_2, I_3, ..., I_n\}$ and their attributes as $\{A_1, A_2,....A_n\}$ then, we can consider $\mathbf{(H_1, W_1)}$ as a centroid of objects considering height and weight as major attributes.
2. Calculate the distance between the centroid and other objects. The distance can be calculated using the Euclidean distance formula (**Equation 1.4**).

$$\mathbf{E.\ D}=\sqrt{(\mathbf{A}_H - \mathbf{H_1})\mathbf{2} + (A_w - \mathbf{W_1})\mathbf{2}}$$

Equation 1.4: Euclidean distance formula

Where, $\mathbf{X_H}$= Observation value of height, $\mathbf{H_1}$= Centroid value of cluster 1 for height, $\mathbf{X_w}$= Observation value of height, $\mathbf{W_1}$= Centroid value of cluster 1 for weight

3. Update centroid of each new cluster, by computing the average attributes of all object in a cluster.
4. Repeat step 1, 2 and step 3 until the centroids stop changing.

Example of K-means clustering

Let's consider input data set as given in **Table 1.11** and height and weight are two major attributes.

Height	Weight
185	72
170	56
168	60
179	68
182	72
188	77

Table 1.11: Input data to clustering algorithm

Step 1: Initialize cluster centroid

Let's consider, two centroids one containing minimum value of Height, Weight and another containing maximum value of Height, Weight as given in **Table 1.12.**

Cluster	Initial Centroid	
	Height	Weight
Cluster 1	185	72
Cluster 2	170	56

Table 1.12: Maximum and minimum cluster centroids

Step 2: Select objects value from input data and calculate Euclidean Distance from centroids

Once centroids (maximum, minimum) are fixed, select input value from input data and calculate Euclidean distance using **Equation 1.4**. Here, we are using (Height: 168, Weight: 60) as object value from input data.

Euclidian Distance from Cluster 1	Euclidian Distance from Cluster 2	Chosen cluster
$\sqrt{(168-185)^2+60-72^2} = 20.808$	$\sqrt{(168-185)^2+(60-72)^2} = 4.472$	Cluster 2

Table 1.13: Table showing computation of Euclidean distance

From Euclidean distance, we can see that record with (168, 60) is very close to cluster 2.

Step 3: Update centroid of each new cluster, by computing the average attributes of all objects in each cluster.

Cluster	Updated Centroid	
	Height	Weight
Cluster 1	185	72
Cluster 2	(170+168)/2=**169**	(56+60)/2=**58**

Table 1.14: Table showing update of centroid in new cluster in K-means method

Step 4: Repeat step 2 and step 3 until dataset is empty. The output created in our example is present in **Table 1.15**.

Objects	Cluster
{(185,72), (179,68), (182,72), (188,77)}	Cluster 1
{(170,56), (168,60)}	Cluster 2

Table 1.15: Cluster created by K-means method

1.10.2 Classification

Classification is used to classify an item in a set of predefined set of classes or groups. The paramount difference between classification and clustering is that classification is used in supervised learning technique where predefined labels are assigned to instances by properties; on the contrary, clustering is used in unsupervised learning, where similar instances are grouped, based on their features or properties (**Arabie, Phipps, & Soete, 1996**). The classification process involves the training set and testing set. The training dataset is used to train model, by pairing the input with expected output. Then, the same classification model is applied to the test data having unknown target class values, to check for its prediction accuracy. The classification by decision tree induction (**Apté, Chidanand, & Weiss, 1997**) is one of the most widely used classification technique. The decision tree has two types of nodes, decision node (which are internal nodes) and leaf node. A decision node specifies test (asks a question) on a single attribute. A leaf node indicates a class. To use the decision tree in testing, the tree top-down according to attribute values with given test instance until a leaf node.

Example of classification by decision tree

Let us consider the example data set as given in **Table 1.16** for classification and our main goal is to determine, whether a user is eligible for a credit card or not using the decision tree.

TID	AGE	JOB_STATUS	HOUSE_STATUS	CREDIT_SCORE	Credit Offer
1	Young	FALSE	FALSE	Fair	No
2	Young	FALSE	FALSE	Good	No
3	Young	TRUE	TRUE	Fair	Yes
4	Middle	TRUE	TRUE	Good	Yes
5	Middle	FALSE	TRUE	Excellent	Yes

Table 1.16: Dataset to be classified by the decision tree

Then, the decision tree to check credit card eligibility for this data set is present in **Figure 1.1**.

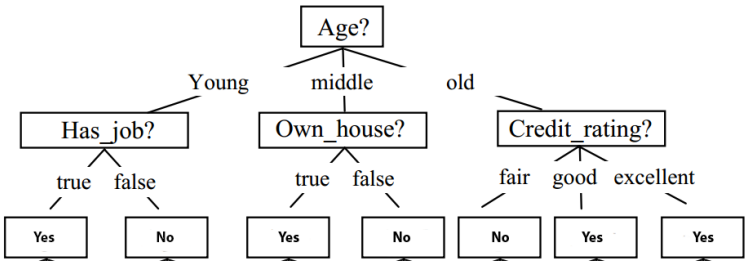


Figure 1.1: Decision tree for classification

In **Figure 1.1**, Age is the root node, which asks the question: what is the age of the applicant? It has three possible answers or outcomes, which are the three possible values of Age (Young, middle and old).

1.10.3 Association Rule

Association rules analysis is an unsupervised technique to discover how items are associated with each other (**Ma & Liu, 1998**). The association rule consists of two parts the lefthand side is called antecedent, and the righthand side is called consequent. Association rule is represented in the form X-> Y, where X and Y belong to a candidate set I= $\{i_1, i_2....i_n\}$ of n items. Association rule is performed in two stages i) finding all frequent patterns (itemsets) having support greater than or equal to minimum support ii) finding all rules from frequent patterns with confidence greater than or equal to minimum confidence. Association rule finds the relationship between the items in the rule. For example, Bread->Milk implies that if product Bread is bought customers also buy product Milk. The Apriori algorithm (**Agrawal & Srikant, 1994**) is a popular algorithm for association rule mining, and it works in two steps i) generate frequent itemsets ii) pruning the itemsets based on the user-defined support. Apriori algorithm takes a transactional database and output is frequent itemsets that satisfied minimum support. So, in the first step, support count of each item in the candidate set (C_1) is calculated, and those items that don't satisfy the minimum support are pruned and produced frequent set (L_1). In the next step, the candidate set (C_2) is produced by Apriori join method by **L_1 App-join L_1**. This process is iterative until can't produce more candidate set. In Association rule, confidence and support are two major factors, which can be computed by **Equation 1.5** and **Equation 1.6**.

$$\text{Support of item i} = \frac{(\text{number of occurences of i})}{(number\ of\ database\ transactions)}$$

Equation 1.5: Equation to compute support of itemset i

$$\text{Confidence of item i} = \frac{(\text{number of occurences of i})}{(number\ of\ occurances\ of\ the\ antecedent\ of\ the\ item\ i)}$$

Equation 1.6: Equation to compute confidence of itemset i

Example of association rule

Let us consider transactional data as shown in Table 1.17 as input, where candidate set (C_1) = {A, B, C, D}, minimum support=3, and our goal is to find frequent items to create possible association rules.

Transaction Id (TID)	Items
T_1	A,B,C,D
T_2	A,B,D
T_3	A,B
T_4	B,C,D
T_5	B,C
T_6	C,D
T_7	B,D

Table 1.17: Transactional data to mine by Apriori algorithm

Step 1: Find frequent item (L_1) from candidate set (C_1)

The principal step in Apriori process is to find frequent item by the counting occurrence of each item. The items that don't satisfy the minimum support count are pruned and produced frequent item (L_1). In our case, frequent item (L_1) = {A:3, B:6, C:4, D:5}.

Step 2: Generate candidate set (C_2) from frequent item (L_1) by Apriori join (**L_1 App-join L_1**)

We can generate a candidate set (C_2) by L_1 App-join L_1. Frequent item (L_1) can be joined only with an item that comes after it in frequent item (L_1). Which will give candidate set (C_2) = {AB, AC, AD, BC, BD, CD}.

Step 3: Find frequent item (L_2) from candidate set (C_2)

Frequent item (L_2) is obtained by following the same procedure as in step 1. We can count the occurrence of each item in candidate set (C_2), and infrequent items are removed to create frequent itemset (L_2) = {AB: 3, BC: 3, BD: 4, CD: 3}.

Step 4: Generate candidate set (C_3) from frequent item (L_2) by Apriori join (**L_2 App-join L_2**)

We can apply the same process as in step 2 to generate candidate set (C_3) by joining L_2 with L_2 using Apriori join and it produces candidate set (C_3) = {ABC, ABD, BCD}.

Step 5: Find frequent item (L_3) from candidate set (C_3)

None of the item in candidate set (C_3) satisfied minimum support. So, we need to stop here and join frequent item to get the final frequent item (L) $=L_1 \cup L_2=$ {A, B, C, D, AB, BC, BD, CD}.

1.11 Existing E-commerce Recommendation Systems

There are different kinds of E-commerce data such as historical or clickstream. The historical data represents the list of an item purchased by the users over the time, which may consist of several attributes such as transactional ID, category ID, product ID, purchased Time, rating and many more. Many researchers tried to predict the users interest to items by using the rating of items provided by users (**explicit rating** - rate or vote of items within the specified range with available rating or voting system) as principle parameter (**Sarwar, Karypis, Konstan & Riedl, 2001**), (**Herlocker, Konstan, Terveen & Riedl, 2004**) in collaborative filtering. An example of explicit user-item rating matrix is present in **Table 1.18.**

User/Item	Item1	Item2	Item3	Item4	Item5	Item6
User1	5	4	5	3	3	2
User2	4	3	?	2	3	2
User3	?	3	3	1	1	?
User4	1	2	2	3	3	3

Table 1.18: User-item rating matrix

On another side, every user may not provide the rating for the purchased items or may not purchase items once they clicked or placed inside a basket. So, to alleviate this problem, researcher finds a way of representing user-item purchased by binary information (**implicit rating**- rating derived from user's behaviors such as click, purchase), such as 1 for purchased/rated or 0 for non-purchased/unrated item. But, binary user-item matrix (**Table 1.19**) may be unable to provide information of click, basket placement, and purchase behavior.

User/Item	Item1	Item2	Item3	Item4	Item5	Item6
User1	1	1	1	1	1	1
User2	1	1	0	1	1	1
User3	0	1	1	1	1	0
User4	1	1	1	1	1	1

Table 1.19: User-item purchased matrix generated from rating information

So, many researchers worked on the implicit rating matrix using various approaches as given below:

1) **Probability based decision tree approach- KimRec05 (Kim, Yum, Song, & Kim, 2005):** It used a binary user-item matrix to visualize the click and purchase behavior of a user and made a non-purchased item (0) more informative in the user-item matrix by computing the probability of purchase after basket placement. This approach is based on forming a decision tree from user's behaviors such as searching, clicking to gives the proportion of users taking that path and its related probability. When new users arrive, it finds the right path based on basement placement probability. Finally, binary user-item rating matrix is filled with basket placement probability (as shown in Figure 1.2) to improve the user-item rating matrix before applying collaborative filtering. But KimRec05 is failed to provide: (a) Frequency of item purchased because a user may purchase the same item different number of times according to the time span (b) Unable to capture sequential purchase behavior (c) Fail to integrate E-commerce historical data.

	*CD*1	*CD*2	*CD*3	*CD*4
*Customer*1	1	0	1	
*Customer*2		1	0	0
*Customer*3	1	0		
*Customer*4	0		0	1
*Customer*5		0		1

(a) Conventional Recommender System

	*CD*1	*CD*2	*CD*3	*CD*4
*Customer*1	1	0.15	1	
*Customer*2		1	0.62	0.44
*Customer*3	1	0.15		
*Customer*4	0.82		0.44	1
*Customer*5		0.15		1

(b) Proposed Recommender System

Figure 1.2: Improved user-item matrix on the right and traditional matrix on the left

2) **Segmentation based approach- LiuRec09 (Liu, Lai & Lee, 2009):** This approach is based on forming a segmentation of user on the basis of Recency, Frequency, Monetary (RFM) using K-mean clustering method, where Recency is period since the last purchase, Frequency is a number of purchased and Monetary is the amount of money spent. Once the RFM segmentation is created, users are further segmented using transaction matrix. The transactions matrix captures the list of items purchased or not purchased by users over a monthly period in a given products list. From the transaction matrix, users' purchases are further segmented into T-2, T-1, and T, where T represents the current purchase and T-1 and T-2 represents two previous purchases. Finally, association rule mining is used to match and select Top-N neighbors from the cluster to which a target user belongs using binary choice and derive the prediction score of an item not yet purchased by the target

user based on the frequency count of the item scanning the purchase data of k-neighbors. The major drawbacks of LiuRec09 are: (a) It does not learn sequential purchase during user-item matrix creation (b) Utility of an item such as frequency and price are ignored during the recommendation generation.

3) **User transactions based preference approach- ChoiRec12 (Choi, Keunho, Yoo, Kim, & Suh, 2012):** Users are not always willing to provide a rating or they may provide a false rating. Thus, **ChoiRec12** developed the system that derives preference ratings from a users' transactional data by using the number of time $user_u$ purchased $item_i$ respect to total transactions. Once preference ratings are determined, they are used to formulate a user-item rating matrix for collaborative filtering. To make a better recommendation, they tried to use the purchase item but there is no evidence of sequential purchase patterns generated using a sequential pattern mining algorithm. To recommend purchase items to a target user, subsequences of a target user purchase items are matched with derived purchase items of all other users. If some patterns are matched, then importance on item is added by counting the support. Finally, items having the highest count are recommended to users. The main limitation of ChoiRec12 are: (a) User purchase patterns are not considered during user-item matrix creation. (b) No provision for recommending infrequent item. Thus, an example of the user-item matrix in ChoiRec12 recommendation system is represented as:

(a) Traditional Implicit Matrix

User / Item	*Item*1	*Item*2	*Item*3	*Item*4
*User*1	0	1	0	0
*User*2	1	0	0	1
*User*3	1	0	0	0

(b) ChoiRec12 user-item matrix

User / Item	*Item*1	*Item*2	*Item*3	*Item*4
*User*1	0	1.4	0	0
*User*2	2.9	0	0	3.8
*User*3	2.5	0	0	0

4) **Common interest based approach- SuChenRec15 (Su & Chen, 2015):** It is based on finding the common interest similarity (frequency, duration, and path) between purchase patterns of users to discover the closest neighbors. For the frequency similarity, it computes total hits occurred in an item or category with respect to a total length of users' browsing path. For duration similarity, it computes the total time spent on each category with respect to total time spent by users'. Finally, for path similarity, it uses the longest common subsequence comparison. Then, CF method is used to select the Top-N neighbor from three indicators. The major drawbacks of **SuChenRec15** are: (a) It requires domain knowledge

for categories, and only supports category level recommendations. (b) Fails to integrate sequential purchase pattern during formation of user-item rating matrix.

5) **Historical and clickstream based recommendation- HPCRec18 (Xiao & Ezeife, 2018):** **Xiao & Ezeife, 2018** proposed **HPCRec18** system, which normalizes the purchase frequency matrix to improve rating quality, and mines the session-based consequential bond between clicks and purchases to generate potential ratings to improve the rating quantity. Furthermore, **HPCRec18** used historical purchased frequency of item and enriched the user-item matrix from both quantity (finding the possible value for 0 rating) and quality (finding the more precise value for 1 rating) by using normalization of user-item purchase frequency matrix and using consequential bond between click and purchase. The major drawbacks of **HPCRec18** are: (a) User-item matrix frequency matrix is created by neglecting sequential pattern. (b) Sequential patterns are not used in the consequential bond.

User-item frequency matrix					**Normalized user-item frequency matrix**					**Rating matrix with predicted rating**				
User / Item	1	2	3	4	*User / Item*	1	2	3	4	*User / Item*	1	2	3	4
1	?	2	1	?	1	?	0.89	0.45	?	1	0.63	0.89	0.45	0.5
2	1	2	?	3	2	0.27	0.53	?	0.8	2	0.27	0.53	0.35	0.8
3	1	?	?	?	3	1	?	?	?	3	1	0.74	0.27	0.3

Table 1.20: Normalized user-item frequency matrix created by Xiao & Ezeife, 2018

SessionId	**Clicks**	**Purchases**
xc1csd…	<4,1,2>	<1,2,4>
df2nbf…	<3,5,2>	<5>
sd3fhs…	<5,2>	<2>
mk4gs…	<3,4,5>	<3,4>
gm5ca…	<1,5>	?

Table 1.21: E-commerce data containing consequential bond of click and purchase

1.11.1 Summary of some close existing E-commerce recommendation systems

Existing System	Methodology	Input Data	Limitation
LiuRec09 by **Liu, Lai, &Lee, 2009**	Users are first segmented by RFM. Once RFM segmentation is created, users are further segmented with transaction matrix. The transactions matrix contains binary purchase information of users over a month. From the transaction matrix, user's purchases are further segmented into T-2, T-1, and T, where T denotes current purchase and T-1 and T-2 represents two previous purchases. Finally, the association rule is used to match Top-N neighbors from the cluster to which a target user belongs using binary choice and derive the prediction score of an item not yet purchased by the target user with a frequency count of k-neighbors.	Minimum support, historical purchase data, and products list.	No provision for recommending infrequent items. Sequential pattern and frequency are not considered during recommendation.
ChoiRec12 by **Choi, Keunho, Yoo, Kim, &Suh, 2012**	Based on preference ratings from a users' transactional data by using the number of time $user_u$ purchased $item_i$ respect to total transactions. Once preference ratings are determined, they are used to formulate a user-item rating matrix for collaborative filtering. To a make better recommendation, they tried to use the purchase item, but there is no evidence of sequential purchase pattern generated using the sequential pattern mining algorithm.	Historical purchased, containing purchase date and list of purchased items.	It did not use user purchase sequential patterns in a user-item matrix. Furthermore, no provision for making a recommendation to infrequent users.
SuChenRec15 by **Su & Chen, 2015**	It is based on finding the common interest similarity (frequency, duration, and path) between purchase patterns of users to discover the closest neighbors. Frequency similarity is computed by counting total hits occurred in an item or category with respect to a total length of users' browsing path. Duration similarity is computed by considering total time spent on each category with respect to total time spent by users'. Finally, for path similarity is computed by counting the longest common subsequence.	Historical data containing the frequency of item, path, and duration.	It requires domain knowledge for categories, and only supports category level recommendations.
HPCRec18 by **Xiao & Ezeife, 2018**	Improved the quality of user-item matrix by normalizing the frequency of item purchase. Furthermore, they provided the purchase possibility of clicked but not purchased items by analysis of consequential bond.	User-item purchase frequency and clickstream data that contains information's of click and purchase	Unable to integrate sequential pattern during qualitative and quantitative analysis of user-item matrix.

Proposed HSPRec by **Bhatta, Ezeife & Butt, 2019**	HSPRec first generates an E-Commerce sequential database from historical purchase data using SHOD (Sequential Historical Periodic Database Generation). Then, mines frequent sequential purchase patterns before using these mined sequential patterns with consequential bonds between clicks and purchases to (i) improve the user-item matrix quantitatively, (ii) used historical purchase frequencies to further enrich ratings qualitatively. Thirdly, the improved matrix is used as input to the collaborative filtering algorithm for better recommendations.	Minimum support, historical purchase, and consequential bond of historical click and purchase	Unable to capture multi-database. No provision for infrequent user.

Table 1.22: Table showing existing E-commerce recommendations

1.12 Problem Definition

Given E-commerce historical click and purchase data over a certain period of time as input, the problem being addressed by this thesis is to find the frequent periodic (daily, weekly, monthly) sequential purchase and click patterns in the first stage. Then, these sequential purchase and click patterns can be used to make user-item matrix qualitatively (specifying level of interest or value for already rated items) and quantitatively (finding possible rating for previously unknown ratings) rich before applying collaborative filtering (CF) to improve the overall accuracy of recommendation.

1.13 Thesis Contribution

The main limitation of existing related systems such as (**HPCRec18, Xiao & Ezeife, 2018**) is that they treated the entire clicks and purchases of items equally and did not integrate frequent sequential patterns to capture more real-life customer purchase behavior sequence patterns inside consequential bond. Thus, in this thesis, we propose a system called Historical sequential pattern recommendation (HSPRec (**Figure 1.3**)) to discover frequent historical sequential pattern from click and purchase, so that discovered frequent sequential patterns are used to improve the consequential bond and user-item frequency matrix to improve recommendation. The detailed architecture of the HSPRec system is present in **Figure 3.1.**

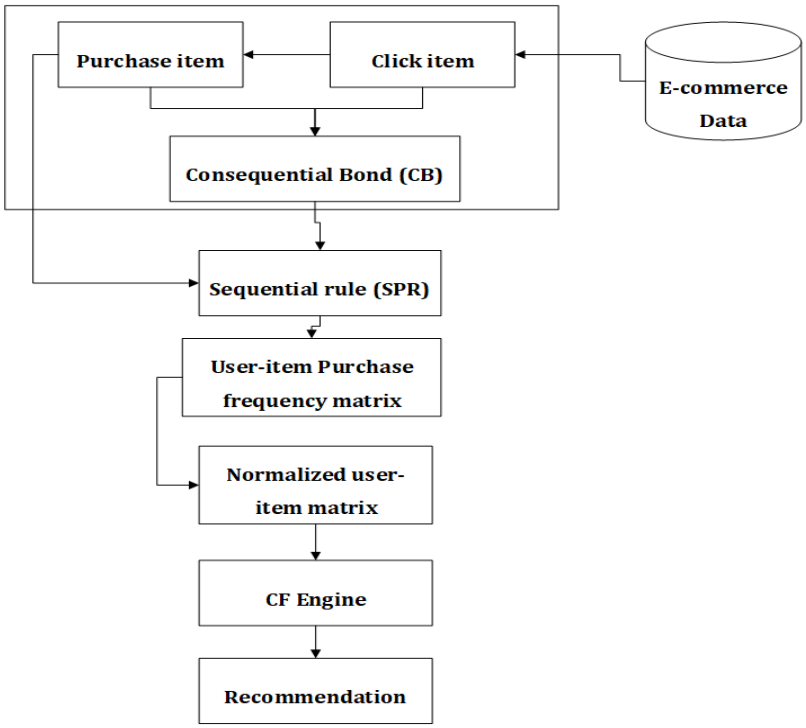


Figure 1.3: Historical sequential recommendation (HSPRec)

1.13.1 Thesis feature contributions

1. Using sequential pattern to improve user-item rating matrix quantitatively

In E-commerce, a user-item rating is very sparse. Thus, only the normalization of user-item frequency matrix is not sufficient to indicate the level of user-interest on an item. So, in this thesis, we are doing the analysis of historical sequential purchase patterns of a user to provide the relationship between already purchased items and recommended items to fill the missing rating for an item to improve the user-item matrix quantitatively (providing possible value for the unrated item or 0 value item in user-item matrix). The details process is provided in section 3.2.

2. Using sequential pattern to enhance consequential bond of click and purchase

In E-commerce click and purchase are two different types of events in E-commerce, and they are not synchronous even if they contain equal numbers of items. For example, <3, 5, 2, 3> and < (3), (5), (2, 3)> contain similar items but sequences of itemsets are different. Thus, integration of sequential patterns in the consequential bond is necessary to make it strong. The details process is present in section 3.2.

3. Discovering periodic (daily, weekly and monthly) sequential pattern

A weekly sequential pattern consists of a large number of items in itemsets compared to a daily sequential pattern to generate complex sequential rules. Thus, in this thesis, we have developed SHOD (Sequential Historical Periodic Database) algorithm (defined in section **3.2.1**) to discover daily, weekly and monthly sequential pattern by considering timestamp clicks and/or purchases items to enhance user-item purchase frequency matrix.

4. Improving the recommendation accuracy

We are using sequential patterns to improve user-item matrix qualitatively and quantitatively by processing with frequent clicks and/or purchases sequential patterns to generate a rich user-item matrix for CF algorithm, and experimental results show that our approach HSPRec performs better than tested existing related system.

1.12.2 Thesis procedural contributions

To make the specified feature contributions, this thesis proposes HSPRec system (**Algorithm 3.1**), which consists of following major steps:

1. Convert historical purchase information to user-item purchase frequency matrix by counting the number of items purchased by each user.
2. Create purchase sequential database from historical purchase by applying sequential historical periodic database (SHOD) algorithm present in **3.2.1**.
3. Apply a purchase sequential database to Sequential Pattern Rule (SPR) module present in **3.2.2**, to create frequent purchase sequences. Once frequent purchase sequences are found, use them to generate purchase sequential rules.
4. Apply purchase sequential rule in user-item purchase frequency matrix to improve quantity.
5. For each user, where click happened without a purchase such as for user 5 in Table 3.4, create a click periodic sequential database (Table 3.5) by neglecting purchase from the consequential bond. Then, input a click sequential database to Sequential Pattern Rule (SPR) (present in 3.2.2) module to get recommended items as the predicted purchase items.
6. Once purchased items are recommended to a user, then compute click and purchase similarity using Click and Purchase Similarity (CPS) module present in **3.2.3**.

7. Supply value of Click Purchase Similarity (CPS) to purchase patterns including recommended items from Sequential Pattern Rule (SPR) to create weighted purchase pattern.
8. Call Weighted Frequent Purchase Pattern Miner (WFPPM) present in **3.2.4** and apply weighted purchase patterns as input and compute the weight of each item present in purchase patterns.
9. Repeat the steps 5, 6, 7, and 8 if there are users without purchase otherwise, use computed rating to further enhance user-item purchase matrix and apply normalization function present in **3.2.5** to enhance user-item matrix before running collaborative filtering algorithm.

1.14 Outline of Thesis

CHAPTER 2: Discuss related E-commerce recommendation systems, different sequential pattern mining algorithms.

CHAPTER 3: Discusses the proposed E-commerce sequential dataset and sequential recommendation system and its related algorithms, methods.

CHAPTER 4: Discusses the experimental implementation for sequential recommendation system, required tools and technologies.

CHAPTER 5: Discusses about the future work and conclusion.

CHAPTER 2: RELATED WORK

2.1 E-commerce Recommendation Systems

2.1.1 E-commerce recommendation system based on navigational and behavioral patterns by Kim, Yum, Song, & Kim, 2005 (KimRec05)

Conventional recommendation approach represents user-item matrix using binary data (1/0) (1 represents purchased and 0 represents not purchased items) for making recommendation but **Kim, Yum, Song, & Kim, 2005** proposed clickstream approach that analyze searching, browsing, clicking, basket placement and purchasing data captured from the navigational and behavioral patterns of customers to estimates the preference levels of a customer for the products, which they clicked but not purchased using decision tree. So, major steps involved in this work are:

Step 1: Gather data related to the purchase, navigational, and behavioral patterns.

Navigational patterns include browsing, searching, product click, basket placement, and actual purchase, while behavioral patterns consist of the click ratio for a certain type of product, length of reading time spent on a specific product, number of visits to a specific product, printing, and bookmarking. So, data collected may be as shown in **Table 2.1.**

Parameters	**Description**
Click type	Binary variable: searching=1; browsing=0
Number of visits	Discrete variable
Length of reading time	Continuous variable (s)
Print status	Binary variable: printZ$_1$; no printZ$_0$
Bookmarking status	Binary variable: bookmarkingZ$_1$; no bookmarkingZ$_0$
Level 1 click ratio (genre)	Continuous variable defined for each product k clicked by customer i. Let j be the category (at Level 1) to which product k belongs. Then, Level 1 click ratio for product, k=(Total number of products clicked by customer i that belong to category j at Level 1)/(Total number of products clicked by customer i)
Level 2 click ratio (specific type)	Continuous variable defined for each product k clicked by customer i. Let j be the category (at Level 2) to which product k belongs. Then, Level 2 click ratio for product, k=(Total number of products clicked by customer i that belong to category j at Level 2)/(Total number of products clicked by customer i)
Basket placement status	Binary variable: basket placement=1; no basket placement=0
Purchase status	Binary variable: purchase=1; no purchase=0

Table 2.1: E-commerce data parameters and their description

Step 2: For each customer, the preference level of a product which is clicked but not purchased is estimated using three steps:

1. Estimation of the probability of purchase after basket placement (p)

$$P=\frac{\text{Total number of cases in which product is purchased}}{\text{Total number of cases in which product is placed in basket}}$$

For example, the total number of cases of product purchased is 5, and the total number of cases of product placed in the basket is 6 then p=5/6=0.83.

2. Estimate the probability *b* of placing a product after clicking it using decision tree (DT) analysis, logistic regression (LR) analysis, or artificial neural network (ANN).

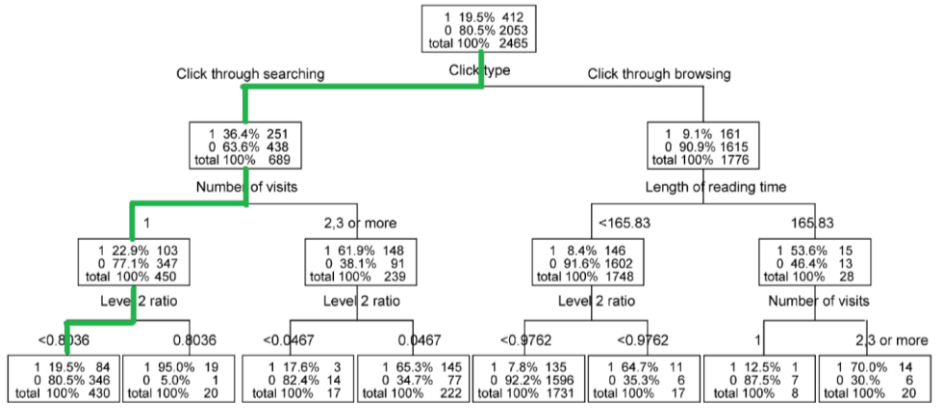


Figure 2.1: Decision tree to show click and basket placement probability

3. Determination of the preference level of a product which is clicked but not purchased by each customer is computed by using (p*b).

For example, preference level to place clicked item on the basket is 5/6*19.5%=0.161

Step 3: CF is performed using the preference levels data as input values, and the preference levels of a customer for the products not clicked are predicted**.**

Conventional recommender system that uses only the purchase status, where 0's (no purchase) and 1's (purchase) as input data but they are improved by integrating the probability of reaching the point of purchase is estimated for a product clicked by a customer as shown in Figure 2.2.

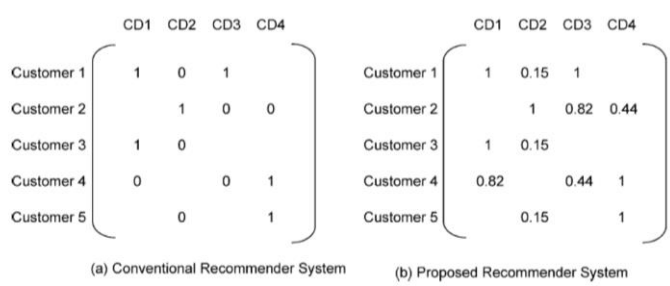


Figure 2.2: a) conventional recommendation system and b) Kim recommendation system

Therefore, enhanced rating matrix consisting of more information is used as input in collaborative filtering to provide the recommendation.

2.1.2 A hybrid of sequential rules and collaborative filtering for product recommendation by Liu, Lai, and Lee, 2009 (LiuRec09)

The sequential rule method considers the sequence of customers' purchase behavior over time but does not utilize the target customer's purchase data for the current period. So **LiuRec09** proposed a segmentation based method using Recency, Frequency, Monetary (RFM) segmentation using K-mean clustering method, where R is period since the last purchase, F is a number of purchased and M is the amount of money spent. Once the RFM segmentation is created, users are further segmented using transaction matrix. The transactions matrix captures the list of items purchased or not purchased by users over a monthly period in a given products list. From the transaction matrix, the users' purchases are further segmented into T-2, T-1, and T, where T represents the current purchase and T-1 and T-2 represents two previous purchases. Finally, association rule mining is used to select Top-N neighbors from the cluster to which a target user belongs using binary choice analysis and derive the prediction score of the item not purchased based on the frequency count of the item scanning the purchase data of k-neighbors.

Example of LiuRec09

Let us consider E-commerce historical data containing information of price, quantity and transaction time as given in **Table 2.2** as input.

Customer ID	Transaction Time	Product	Quantity	Price
C001	July 11 2017	Perfumes	1	$77183.60
C001	August 17 2017	Skincares	3	$4196.01
C001	September 14 2017	Dresses	4596	$33719.73
C002	July 15 2017	Perfumes	199	$4090.88
C002	August 13 2017	Shoes	59	$942.34
C002	September 25 2017	Skincares	1	$77183.60
C003	July 19 2017	Skincares	431	$251657.30
C003	August 22 2017	Perfumes	337	$94330..79
C003	September 18 2017	Kints	963	$91062.38
C004	July 24 2017	Perfumes	277	$66653.56
C004	September 18 2017	Dresses	5111	$65164.79
C005	September 18 2017	Perfumes	568	$65164.79
C005	July 13 2017	Shoes	2379	$65039.62

Table 2.2: E-commerce transactional data

Step 1: Customer clustering using (R) Recency, F (Frequency) and (M) Monetary value

To create a cluster of users using RFM value, user RFM is matched with predefined RFM range and RFM quartile values are assigned to users. Then, a final RFM score is computed using RFM quartile as shown in **Table 2.3**.

CID	Recency(R)	Frequency(F)	Monetary(M)	R_quartile	F_quartile	M_quartile	RFM Score
1	109	5	191.85	3	4	4	344
2	70	96	1054.43	4	4	4	444
3	130	3	67	2	3	3	233
4	74	1	15	4	1	1	411
5	214	1	19.92	1	1	2	112

Table 2.3: E-commerce data clustering using RFM value

Step 2: Create transaction matrix

Once RFM clusters of users are created, users' transaction (binary) matrix is created by analyzing the list of items purchased by users, where 1 represents purchased items and 0 represents not purchased items by a user. An example of transaction matrix created from historical E-commerce data (**Table 2.2**) is present in **Table 2.4**.

CID	Perfumes			Skincares			Knits			Dresses			Shoes		
	July	Aug	Sep	July	Aug	Sep	July	Aug	Sep	July	Aug	Sep	July	Aug	Sep
001	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0
002	1	0	0	0	0	1	0	0	0	0	0	0	0	1	0
003	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0
004	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0
005	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0

Table 2.4: transaction clustering of E-commerce data

According transaction matrix, customer (CID_{001}) purchased perfumes on July 2017, so value it set to 1 and it is set to 0 in August and so on. In given product list P= {Perfumes, Skincare, Knits, Dresses, Shoes}, the transactions of customer CID_{001} are, July={Perfumes}, Aug.={Skincares}, Sep.={Dresses}. Thus, dynamic customer profile of customer CID001 from July to September may be represented as CID_{001}, July = {1, 0, 0, 0, 0}, CID_{001}, Aug= {0, 1, 0, 0, 0}, CID_{001}, Sep = {0, 0, 0, 1, 0}.

Step 4: Transaction matrix clustering

The transaction clustering helps to locate the customer past transaction and present transaction. Furthermore, transaction cluster represents a group of transactions with a similar item purchased by users. For example, if we take customer CID_{001} transaction sequence then clusters may be different as given in **Table 2.5**.

Transaction Sequence	Cluster No (Ci)
100	Cluster 10 (C1) on the basis of RFM quantity
010	Cluster 3 (C2) on the basis of RFM
001	Cluster 9 (C3) on the basis of RFM
000	Cluster 1 (C4) on the basis of RFM

Table 2.5: Transaction sequence clustering

According to transaction sequence, the customer CID_{001} belonged to the tenth cluster in July and moved into the third cluster in August, thereafter reaching the ninth cluster in September. So behavior locus for CID_{001} is (10, 3, 9).

Step 4: Mining customer behavior from transaction clusters

To mine customer behavior according to purchase time, this work adopted association rule R_j for determining the most frequent pattern with confidence.

$$R_j = r_{j,T-l+1,\ldots\ldots,}r_{j,T-1} \rightarrow r_{j,T}(Support_j, Confidence_j)$$

Equation 2.1: Association rule to mine customer behavior in LiuRec09

Where rule R_j indicates that, if the locus of a customer is $r_{j,T-l+1},.,r_{j,T-1}$, then the behavior cluster for that customer is $r_{j,T}$ at time T. To illustrate this rule, let us consider the locus behavior of customer according to transaction sequence is as given in **Table 2.6**.

CID	Locus in 1st Trans(T-2)	Locus change in 2nd Trans(T-1)	Behavior cluster in Trans(T)
C001	10	3	9
C002	10	1	3
C003	3	10	4
C004	10	-	9
C005	1	-	10
C006	4	-	3
C011	9	3	?

Table 2.6: Customer transaction cluster

Thus, some of the possible association rules from customer transaction cluster (**Table 2.6**) are

Rule	T-2	T-1	T	Association Rule (locus at T-2,T-1→ locus at T)	Support	Confidence
1	10	-	9	10→ 9	0.28	0.5
2	3	10	4	3,10→ 4	0.14	1
3	10	3	9	10,3→ 9	0.14	1
4	10	1	3	10,1→ 3	0.14	1
5	1	-	10	1→ 10	0.14	1
6	4	-	3	4→ 3	0.14	1

Table 2.7: Association rules created from customer transaction cluster

According to the rule 1, if customer purchase behavior in time T-2 is in cluster 10 then his/her behavior will be in 9 clusters at time T , where support for (10→ 9)= 0.28 and confidence for (10→ 9)= 0.5

Step 5: The determination and match of the cluster sequences of target customers

The cluster locus of a target customer is compared with the association rules derived from other customers' loci, and then the best-matching locus is determined and multiplied by the support and confidence of the rule to derive the fitness measure using **Equation 2.2**.

$$M_y^x = \sum_{k=1}^{l-1} M_{y,T-k}^x, where\ M_{y,T-k}^x = \begin{cases} 1\ if\ C_{y.T-k=}r_{x,T-k} \\ 0 \quad otherwise \end{cases} * support_x * confidence_y$$

Equation 2.2: Formula to match target user purchase in LiuRec09

Step 6: Recommendation

Let $M(r_1)$ denote the most frequently purchased product at time T in Cluster. Similarly, $M(r_2)$ is ranked the next highest, and $M(r_N)$ is ranked the N_{th} highest. Then, the recommendation list for the target customer is given by $M(r_1)$, $M(r_2)$…$M(r_N)$.

2.1.3 A time-based approach to effective E-commerce recommender systems using implicit feedback by Lee, Park, & Park, 2008

Collaborative filtering is a widely used method of recommendations based on ratings of items provided by users. However, in e-commerce environments, it is very difficult to collect explicit rating. So, to alleviate the problem of explicit rating **Lee, Park, & Park, 2008** developed a recommender system using time-based implicit rating. The main work starts with constructing a pseudo rating matrix. The pseudo rating matrix contains binary value 1 for purchased item and 0 for not purchased item. After constructing a pseudo rating matrix, temporal information such as users purchase time, item launch time is incorporated into the pseudo rating matrix then values in the pseudo rating matrix are extracted from predefined rating function. In the end, the final user-item rating matrix is applied to collaborative filtering for a recommendation. So, major steps involved in this work are:

Step 1: Collect implicit feedback data

In this step, the item purchased date and the item launch time data are collected. The main motive behind collecting two kinds of information are: 1) most recent purchase reflect better user preference 2) recently launched items appeal more to users.

Step 2: Construct a pseudo rating matrix using temporal information

During pseudo rating matrix construction, we can simply assign 1 as a rating value when a user u purchased an item i. Let us consider pseudo rating matrix as present in **Table 2.8**.

	Item1	Item 2	Item 3	Item 4
User 1	1	0	1	1
User 2	0	1	1	0
User 3	1	0	0	1

Table 2.8: Pseudo rating matrix

Let us consider, LTime represents item launch time and PTime represents the time when user purchased item. Then integrate LTime and PTime in pseudo rating matrix as given in **Table 2.9.**

	Item1 (LTime1)	Item 2 (LTime2)	Item 3 (LTime3)	Item 4 (LTime4)
User 1	PTime1	0	PTime3	PTime4
User 2	0	PTime4	PTime5	0
User 3	PTime6	0	0	PTime7

Table 2.9:Pseudo rating matrix with temporal information

Then, define rating function w (p_i, l_j), where p_i represents purchased time and $\mathbf{l_j}$ represents launch time of a product. Then, convert pseudo rating matrix with temporal information is present in **Table 2.10**.

	Item1	Item 2	Item 3	Item 4
User 1	$w(p_1,l_1)$	0	$w(p_2,l_3)$	$w(p_3,l_2)$
User 2	0	$w(p_2,l_2)$	$w(p_1,l_3)$	0
User 3	$w(p_3,l_1)$	0	0	$w(p_3,l_3)$

Table 2.10: Pseudo rating matrix with rating function $w(p_i,l_j)$

Step 3: Extract value of rating function (w (p_i, l_j)) from predefined table

Used predefined rating function w present in **Table 2.11** to enrich pseudo rating matrix (**Table 2.10**). For example, w (p_1, l_1) is replaced by 0.7 in pseudo rating matrix. Similarly, w (p_2, l_3) is replaced by 2.3.

	old purchase (p_1)	middle purchase (p_2)	recent purchase (p_3)
old launch (l_1)	0.7	1.7	2.7
middle launch (l_2)	1	2	3
recent launch (l_3)	1.3	2.3	3.3

Table 2.11: Predefined rating function w

Step 4: Apply user-item rating matrix to collaborative filtering

By extracting the value of rating function from step 3, construct a user-item rating matrix as present in **Table 2.12**. Then, apply collaborative filtering to predict the missing rating of user on item.

	Item1	Item 2	Item 3	Item 4
User 1	0.7	?	2.3	3
User 2	?	2	1.3	?
User 3	2.3	?	?	3.3

Table 2.12: User-item rating matrix constructed from pseudo rating matrix

2.1.4 Recommender system based on click stream data using association rule mining by Kim, & Yum, 2011

Many recommendation systems only use the purchase data of users for e-commerce recommendation, while navigational and behavioral pattern data were not utilized. So, **Kim, Yum, Song, & Kim, 2005** developed a collaborative filtering technique based on navigational and behavioral patterns of customers. To improve the performance of the recommendation system, they developed a system that used association rule. In this system, they calculated the confidence levels between clicked products, between the products placed in the basket, and between purchased products, respectively, and then the preference level was estimated through the linear combination of the above three confidence levels. The major steps involved in this work are:

Step 1: Data collection and preparation

In this phase, all the navigational and behavioral patterns in e-commerce sites are collected. An example of this step is given in **Table 2.13**. The navigational patterns include browsing, searching, product click, basket placement, and actual purchase, while behavioral patterns consist of the click ratio for a certain type of product, length of reading time spent on a specific product, number of visits to a specific product, printing, and bookmarking to give the statics.

Case	Customer	CD	Clicktype	Timespent	No of visit	Basket placement	Purchase
1	1	CD_A	1	49	2	1	1
2	1	CD_B	1	15	1	1	0
3	2	CD_A	0	4	1	0	0
4	2	CD_C	0	6	1	0	0
5	2	CD_D	0	8	1	0	0
6	2	CD_E	1	12	1	1	1
7	2	CD_F	0	6	1	0	0

Table 2.13: E-commerce clickstream data

Step 2: Association rule mining

1. Identify all pairwise combinations of products that simultaneously appear in a transaction. Let us consider the minimum support is 2%, if the ratio of the number of clicks in which both CD_A and CD_B occur to the total number of transaction is more than 2% then CD_A and CD_B becomes the candidate of association rule.
2. For each pair (CD_i and CD_j, $i \neq j$) the corresponding support is calculated using

$$\text{Support} = P(U \cap V) = \frac{Number\ of\ transactions\ in\ which\ both\ U\ and\ V\ occur}{Total\ Number\ of\ transactions}$$

Equation 2.3: Equation to count support

For example, support (CD_A, CDB) =10/50 where product CD_A in case 1 and product CD_B in case 2 as given in **Table 2.13**. For 2.3, each pair whose support is greater or equal to a specified threshold (for example, 2%), calculate the lift values using following association rule lift. The lift of the rule ''U→V'' can be defined as:

$$\text{Lift} = \frac{P(V|U)}{P(V)} = \frac{P(U \cap V)}{P(U) * P(V)}$$

or

$$\text{Lift} = \frac{\text{Number of transactions in which both U and V occur} * \text{Total number of transactions}}{(\text{Number of transactions in which U occurs}) * (\text{Number of transactions in which V occurs}}$$

Equation 2.4: Equation to compute lift value

For example, lift between CD_A and CD_B = (10*50)/ (13*15) =2.56.

3. For each pair, whose lift is greater than a specified threshold (1 in our case) is selected for generating more elaborate association rules.
4. The association rule is generated on each of the product combinations and then, three confidence levels between clicked products, between the products placed in the basket, between purchased products are calculated.

Step 3: Confidence calculation

In this step, find the confidence level for both basket placement and purchase and use the higher confidence level for preference level.

Step 4: Making recommendation of Top-N list

For each phase (Click, Basket placement, Purchase), find the Top-N products ranked list by confidence level.

2.1.5 Combining collaborative filtering and sequential pattern mining for recommendation by Li, Niu, Chen, & Zhang, 2011

Collaborative filtering finds the similarity between users and items considering closet neighbors but user purchase choice is different with respect to time. Thus, collaborative filtering only is not sufficient to capture the time change purchase habit of users. To overcome the limitation of collaborative filtering and capture the sequential purchase behaviors of user **Li, Niu, Chen, & Zhang, 2011** developed an approach of recommending items using collaborative filtering and sequential pattern mining. From user-item rating matrix as input, at first, compute item-item similarity and predict Top-K items. Once Top-k items are found, items are arranged in descending order of their Top-K value to create a sequence of items purchased by users. Finally, apply a sequential pattern mining algorithm (for example GSP) on sequences of purchases to discover frequent sequential patterns.

Example

Let us consider the user-item rating matrix as given in **Table 2.14** as input.

	Item 1	Item 2	Item 3	Item 4	Item 5
User 1	1	?	1	1	1
User 2	1	2	?	?	2
User 3	3	?	3	?	3
User 4	?	4	?	1	?
User 5	?	5	1	2	?

Table 2.14: User-item rating matrix for Niu, Chen, & Zhang, 2011 recommendation system

Step 1: Compute mean centering user-item matrix

Mean centering of a user-item matrix helps a user with many ratings contributes less to any individual rating. So, mean centering user-item rating matrix (**Table 2.15**) is created by subtracting user's individual items rating from user mean rating as present in **Table 2.14**.

	Item 1	Item 2	Item 3	Item 4	Item 5	Mean rating
User 1	0	?	0	0	0	1
User 2	-0.66	0.34	?	?	0.34	1.66
User 3	0	?	0	?	0	3
User 4	?	1.5	?	-1.5	?	2.5
User 5	?	2.34	-1.66	-0.66	?	2.66

Table 2.15: Mean centering user-item rating matrix

Step 2: Find item-item similarity

Once the mean center user-item rating matrix is computed, then similarities between items are computed using similarity function such as Cosine similarity or Pearson Correlation Coefficient. An equation to compute the Pearson Correlation coefficient is present in **Equation 2.5**.

$$\text{SIM (i, j)}=\frac{\sum_{i\in I}(r_{u,i}-\overrightarrow{r_u})*(r_{u,j}-\overrightarrow{r_u})}{\sqrt{\sum_{i\in I}(r_{u,i}-\overrightarrow{r_u})^2}*\sqrt{\sum_{i\in I}(r_{u,j}-\overrightarrow{r_u})^2}}$$

Equation 2.5: Pearson Correlation coefficient to compute similarity

Where $r_{u,i}$ is the rating given to item i by user u, $\overrightarrow{r_u}$ is the mean rating of all the rating on item provided by user. The item-item similarity is computed with the help of Pearson Correlation Coefficient from user-item rating matrix (**Table 2.14**) is present in **Table 2.16**.

	Item 1	Item 2	Item 3	Item 4	Item 5
Item1	1.00	-0.9	0.78	-0.6	0.91
Item 2	-0.9	1.00	-0.8	0.32	-0.8
Item 3	0.78	-0.84	1.00	-0.3	0.51
Item 4	-0.6	0.32	0.39	1.00	-0.7
Item 5	0.9	-0.8	0.51	-0.7	1.00

Table 2.16: Item-item similarity of mean centered rating matrix 2.16

Step 3: Select items having highest similarities with the current item**.**

In this step, highest similar items of current item are selected, in our case, we can see that Item1 rating is 1.00 and other similar items to Item1 are Item3 and Item5 as present in **Table 2.17**.

	Item 1	Item 2	Item 3	Item 4	Item 5
Item1	1.00	-0.9	0.78	-0.6	0.91
Item 2	-0.9	1.00	-0.8	0.32	-0.8
Item 3	0.78	-0.84	1.00	-0.3	0.51
Item 4	-0.6	0.32	0.39	1.00	-0.7
Item 5	0.9	-0.8	0.51	-0.7	1.00

Table 2.17: Table showing similar item of current item

Step 4: Compute predicted rating for item_i by user_u**.**

The predictions for each user_u correlated with each item_i is present in **Equation 2.6**.

$$P_{u,i}=\frac{\sum_{t\in N}(SIM(i,k)*R_{u,k})}{\sum_{t\in N}(|SIM(i,k)|)}$$

Equation 2.6: Equation to compute predicted rating in item-item similarity

Where N represents the items i similar item sets and $R_{u,k}$ is the rating given to item k by user u and SIM(i,k) represents the similarity between item i and item k. In our case, predicted user-item rating matrix is present in **Table 2.18**.

	Item 1	Item 2	Item 3	Item 4	Item 5
User1	0.9	0	0.8	0	0.7
User 2	1.0	-2	1.1	1.0	0.5
User 3	2.8	**-3**	2.5	**-1**	2.3
User 4	0	0.5	0	2.0	0
User 5	0	1.0	0	1.9	0.2

Table 2.18: Prediction rating matrix

Step 5: Create sequence database by selecting item with highest value in descending order

Once the predicted rating is computed, all the items purchased by each user are arranged in descending order by considering a rating of a user on an item to create a sequence database. The main problem with this kind of sequences in a sequence database is the lack of actual item purchase order.

	Predicted Items
User1	$<I_3, I_1>$
User 2	$<I_3, I_4>$
User 3	$<I_1, I_3>$
User 4	$<I_2, I_4>$
User 5	$< I_2, I_4>$

Table 2.19: Sequence database created by using rating value in descending order

Step 6: Apply GSP algorithm on sequence database

Input: sequence database (**Table 2.19**), **minimum support**=2 and **candidate set (C_1)** = {I_1, I_2, I_3, I_4} and **algorithm**=GSP

1. Find 1- frequent sequence (L_1) to keep the only sequence with occurrence or support count in the database greater than or equal to minimum support count of 2. For example, L_1={$<I_1>$:2, $<I_2>$:2, $<I_3>$:3, $<I_4>$:3}
2. Generate candidate sequence ($C_{k=2}$) using L $(_1)$ $\bowtie_{GSPjoin}$ L$(_1)$
3. Pruning candidate set C $_{(K=2)}$ by testing the minimum support and remove infrequent items.
4. Repeat the process of candidate generation and pruning until the result of candidate generate (C_k) and prune (L_k) for finding frequent sequence is an empty set.
5. Output frequent sequence as union of L_1 U L_2 U…L_n .

1-sequence	2-sequences
$<I_3>$	$<I_3, I_4>$

Table 2.20: n-frequent sequence for Li, Niu, Chen, & Zhang, 2011 recommendation

From 2-sequence $<I_3, I_4>$, we can see that the recommendation track for U_2 is to first adopt I_3 and then I_4.

2.1.6 Implicit rating-based collaborative filtering and sequential pattern analysis for E-commerce recommendation by Choi, Keunho, Yoo, Kim, & Suh, 2012 (ChoiRec12)

Users are not always willing to provide a rating or they may provide a false rating. Thus, **ChoiRec12** developed the system that derives preference ratings from a users' transactional data by using the number of time $user_u$ purchased $item_i$ respect to total transactions. Once preference ratings are determined, they are used to formulate a user-item rating matrix for collaborative filtering. To make a better recommendation, they tried to use the purchase pattern but there is no evidence of sequential purchase pattern generated using sequential pattern mining algorithm. Furthermore, to recommend purchase pattern to a target user, a subsequence of target user purchase items are matched with derived purchase items of all other users. If some items are matched then importance on the item is added by counting the support. Finally, the items having the highest count are recommended to users.

Example of ChoiRec12

Let us consider the fragment of historical purchased data as given in **Table 2.21**, where only purchase time is provided as available information, and our main goal of recommendation is to recommend the suitable item to user T.

	Item 1	Item 2	Item 3	Item4	Item 5
	Date	Date	Date	Date	Date
User 1	01/01	-	01/02	01/03	-
User 2	01/01	-	01/02	01/03	01/04
User 3	-	01/01	01/02	-	01/03
User 4	01/01	01/02	01/03	-	-
User T	-	01/01	01/02	01/03	-

Table 2.21: Choi, Keunho, Yoo, Kim, & Suh, 2012 historical user-item matrix

Step 1: Compute implicit rating of all users on items

The implicit rating can be computed by: $Implicit\ Rating(u,i) = Round\ up(5 * RP(u,i))$

Where, RP(u,i) is the relative preference of user u on item i and it is defined as:

$$RP(u,i) = \frac{AP(u,i)}{\underset{c \in U}{Max}(AP(c,i))}$$

Where AP(u,i) is the absolute preference of user u on item i and it is defined as:

$$\text{AP}(u,i) = \frac{numbe\ of\ transaction\ of\ item\ i\ by\ user\ u}{total\ transactions\ of\ user\ u} + 1$$

In our case, user **1** purchased **item 1** one time out of three transactions. Thus, AP (user1, $item_1$) =1/3+1=1.3. Furthermore, RP(user1, $item_1$) =1.3/1.3=1. So, implicit rating= $RP * 5 = 5$.

In the same way, let us consider a user-item implicit rating matrix created from the historical data using above technique as given in **Table 2.22**.

	Item 1	Item 2	Item 3	Item 4	Item 5	Mean Rating
User 1	3	?	1	5	?	3
User 2	4	?	3	1	2	2.5
User 3	?	1	2	?	4	2.3
User 4	5	4	3	?	?	4
User T	?	4	3	2	?	3

Table 2.22: Implicit rating derived from user's transactions

Step 2: Calculate mean rating and user similarity based on the implicit rating

1. Mean rating computation: The mean rating is computed by adding all the rating of users on items with respect to total numbers of rating. So,

   Mean of rating User 1 = (3+1+5)/3=3, User 2 =2.5, User 3=2.3, User 4=4 and User T=3

2. Similarity computation: Compute similarities between users' using Cosine similarity, which is given as:

$$Cosine(T,b) = \frac{\sum_{i=1}^{m}(R_{T,i})(R_{b,i})}{\sqrt{\sum_{i=1}^{m}(R_{T,i})^2}\ \sqrt{\sum_{i=1}^{m}(R_{b,i})^2}}$$

   Where ($R_{T,i}$) denote the ratings of usersT on item i similarly ($R_{b,i}$) denotes the rating of user b on item i. for example, similarity between Target user and User 1 is SIM (T, User1) = 0.793 similarly, the similarity between SIM (T, User2) = 0.966, SIM (T, User3) = 0.89 and SIM (T, User4) = 1.

Step 3: Find Top-N closest neighbors of target user T

Once the similarity between the target user and other users are calculated, they are sorted based on similarity in descending order then Top-N users are selected as neighbors of the target user. In our case, the number of neighbors is set to 2, so, closest neighbors of target User T are User 2 and User 4.

Step 4: Calculate the CF-based predicted preference (CFPP)

Top-N neighbors are used to predict CF-based predicted preference of target user(T) on $item_i$ by the equation:

$$CFPP(T,i) = \overline{R_T} + \frac{1}{\sum_{b=1}^{k}|sim(T,b)|} * \sum_{b=1}^{N} TopN - sim(T,b) * (R_{b,i} - \overline{R_b})$$

Equation 2.7: CF-based predicted preference

Where, N denotes the number of user a's neighbors and sim (T, b) denotes the similarity between User T and User b, which is computed by cosine similarity. Finally, $\overline{R_T}$ and $\overline{R_b}$ represents the mean rating of User T and mean rating of User b. For example,

$$CFPP(T,item1)=\mu_{TUser}+\frac{(SIM(Tuser,user2)*Ruser2,item1-\mu user2+SIM(Tuser,user4)*Ruser4,item1-\mu user4)}{|SIM(Tuser,user2)+SIM(Tuser,user4)|}$$

$$=4.74$$

Similarly, CFPP (T, item2) =3.5, CFPP (T, item3) =3.2365, CFPP (T, item4) = 2 and CFPP (T, item5) =3

Step 5: Compute purchase item based score (SPAPP)

1. In this step, purchase information of each user placed according to purchase time except for target user. In our case, item purchased by each user are: **User1**: <Item1><Item3><Item4>, **User2**: <Item1><Item3><Item4><Item5>, User**3**: <Item2><Item3><Item5>, User**4**: <Item1><Item2><Item3>.
2. Find frequent single item pattern (L1): Let us consider minimum support as 0.5 then the frequent purchase item (L1) are {<item1>:0.75, <item2>:0.5, <item3>:1, <item4>:0.5, <item5>:0.5}
3. Generate larger candidate set (C_2): Use L1 Apriori join L1 to create larger candidates set (C_2) as present in **Table 2.23**.

Items	Count
<item1><item2>	0.25
<item1><item3>	0.75
<item1><item 4>	0.5
<item1><item5>	0.25
<item2><item3>	0.50
<item 2><item 5>	0.25
<item 3><item 4>	0.50
<item 3><item 5>	0.50

Table 2.23: possible list of 2-items generated from frequent purchase (L1)

4. Find 2-frequent items from C_2: Test candaidate set (C_2) with minimum threshold to create frequent L_2 items.

Frequent items (L_2) =

Item	**Count**
<item1><item3>	0.75
<item1><item 4>	0.5
<item2><item3>	0.50
<item 3><item 4>	0.50
<item 3><item 5>	0.50

Table 2.24: Frequent 2-item generated from candidate set (C_2)

5. Repeat the process of candidate generation (C_k) and pruning (L_k) until the candidate set is empty. In our case, frequent items are: **<Item1> <Item3> (0.75), <Item2> <Item3> (0.5), <Item3> <Item4> (0.5), <Item3> <Item5> (0.5), <Item1> <Item4> (0.5), <Item1> <Item3> <Item4> (0.5)**
6. Match subsequences of a target user purchase with derived purchased items by enumerating target user purchase item. In our case, purchase data of the target user T are **<Item2><Item3><Item4>**, then possible subsequences can be **<Item2>**, **<Item3>**, **<Item4>**, **<Item2><Item3>**, **<Item2><Item4>**, **<Item3><Item4>**, and **<Item2><Item3><Item4>**. For example, since the first item **<Item2>** appears in the starting part of the second frequent item (C_2) thus, **<Item3>** can be decided as candidate items to recommend with supports 1.
7. Calculating the pattern analysis based predicted preference (SPAPP): Pattern based predicted preference of $user_T$ on $item_i$ is computed by $SPAPP(T,i) = \sum_{S \in SUB} Support_s^i$, Where **SUB** denotes the set of all subsequences of $user_T$, and $Support_s^i$ denotes the support of $item_i$ from a subsequence s. for example, predicted preference of Target user on item 1. SPAPP (T, 1) = 0, similarly, SPAPP (T, 2) = 0, SPAPP (T, 3) = 0.75+0.5+0.5=1.25, SPAPP (T, 4) = 0.5+0.5+0.5=1.5, SPAPP (T, 5) = 0.5

Step 6: Integrate CFPP and SPAPP

In this step, CFPP and SPAPP are normalized to get N_CFPP and N_SPAPP, which is calculated by: $FPP(T,i) = \alpha * N\_CFPP(T,i) + (1-\alpha) * N\_SOAPP(T,i)$, Where α and 1- α are weights given to collaborative filtering and association rule to adjust value variations.

	CFPP	SPAPP	N_CFPP	N_SPAPP	FPP	Rank
Item 1	4.7455	0.7071	1	0	0.5	2
Item 2	3.5	0.9648	0.5463	0	0.273	5
Item 3	3.2365	0.8944	0.4504	0.8333	0.6419	1
Item 4	2	1	0	1	0.5	2
Item 5	3	0.333	0.3642	0.3333	0.3488	4

Table 2.25: Table showing integration of CFPP and SPAPP

Step 9: Recommend the item having highest rank

The item the having highest rank generated by adding collaborative filtering and association rule generated value is recommended to target user T. In our case, item 2 is recommended first then item 5.

2.1.7 Interest before liking: Two-step recommendation approaches by Zhao, Niu & Chen, 2013

It is based on matching user interest first then finding the high-quality item that a user will like. First, it uses a binary user model to represent users' interests from their rating values on an item as a measure of interest no matter whether the value is high or low. So, this work is based on matching users' interests at first, and then tries to find high-quality items that users will like. According to **Zhao, Niu & Chen, 2013**, a user can browse an item in the system, and can give rating after browsing; but there are overabundant items in the system and user may not be able to browse them all, thus, the rating behavior itself (regardless what the rating values are) is an indication of the user's interest, and this interest is extensible to similar items. Furthermore, the rating values represent how the user likes the rated item, that is, the quality of the item in the user's point of view, and this quality indication is only applicable to the rated item. In existing item-based CF, items with high predicted values are always recommended to users, and they try to recommend items that users may like directly. Differently, this work ignores the rating value in order to find items that match users' interests first.

Example:

Input: Let us consider user-item rating matrix as given in **Table 2.26**, where rating available in the range of 1-5 and ? represents the unrated rating for item by users.

User/item	Item 1	Item 2	Item 3	Item 4
User 1	5	?	3	2
User 2	?	?	5	4
User 3	5	4	?	?
User 4	5	5	4	?

Table 2.26: User-item rating matrix for Zhao, Niu & Chen, 2013 recommendation system

Step 1: Compute the mean rating of user

The mean rating is computed by adding all the rating of particular users on his rated items with respect to the total number of rating. For example, mean rating User 1 = (3+2+5)/3=3.3, for User 2 =4.5, for User 3=4.5, for User 4=4.33 as shown in table 2.27

User/item	Item 1	Item 2	Item 3	Item 4	Mean rating
User 1	5	?	3	2	3.3
User 2	?	?	5	4	4.5
User 3	5	4	?	?	4.5
User 4	5	5	4	?	4.33

Table 2.27: User-item matrix showing mean rating of users on items

Step 2: Represent user-interest using binary

Represent user-item rating matrix by binary information, where 1 represents rated item and 0 represents the unrated item. In our case, the user-item rating matrix (**Table 2.26**) is represented as a binary user-item matrix as shown in **Table 2.28**.

User/item	Item 1	Item 2	Item 3	Item 4
User 1	1	0	1	1
User 2	0	0	1	1
User 3	1	1	0	0
User 4	1	1	1	0

Table 2.28: User-item binary matrix showing rated and unrated items

Step 3: Normalize binary user-item rating

Some users may have a rating for several items and normalizing help a user with many ratings contributes less to any individual rating. Normalization of user rating on item can be performed as:

$$Normalized\ rating, R_i = \frac{r_i}{Magnitude}$$

Where

$$Magnitude = \sqrt{r1^2 + r2^2 + r3^2 + \cdots . rn^2}$$

For example, the normalized rating of user1 for item1=$1/\sqrt{1^2 + 1^2 + 1^2}$ =1/√3 =0.57. By using the same technique, normalized rating of each user on item is computed. In our case, normalized user-item binary rating matrix is present in **Table 2.29**.

User/item	Item 1	Item 2	Item 3	Item 4
User 1	0.57	0	0.57	0.57
User 2	0	0	0.70	0.70
User 3	0.70	0.70	0	0
User 4	0.57	0.57	0.57	0

Table 2.29: Normalized user-item rating matrix

Step 4: Form item-item similarity on normalized user-item matrix

Normalized user-item rating matrix is used to compute item-item similarity using Cosine similarity function. For example, similarity between Item 1 and Item 2 is, Sim(1,2) =$\frac{0.57*0.70+0.57*0.70}{\sqrt{0.57^2+0.57^2}*\sqrt{0.70^2+0.70^2}}$= 0.37, similarly, Sim(1,3)= 1, Sim(1,4)=1 as shown in **Table 2.30**.

item/item	Item 1	Item 2	Item 3	Item 4
Cosine(item1,j)	1	0.37	1	1
Cosine(item3,j)	1	0	1	1

Table 2.30: Table showing item-item similarity

Step 5: Predict rating and adjust rating of user on item using mean rating

Once item-item similarity is computed, then it is used to compute user rating on item for unrated item such as item having 0 value then it is further adjusted by using mean rating using formula: $r_{final}(u,i) = r(u,i) * mean(i)$, where, mean(i) is mean rating computed in step 1.

2.1.8 Discovering e-commerce interest patterns using click-stream data by Su & Chen, 2015 (SuChenRec15)

This approach is based on finding the common interest similarity (frequency, duration, and path) between purchase patterns of users to discover the closest neighbors. For the frequency similarity, it computes total hits in item or category with respect to the total length of the user's browsing path. For duration similarity, it computes the total time spent on each category with respect to total time spent by the user. Finally, for path similarity, it uses the longest common subsequence comparing the two click sequence groups of two users. By selecting Top-N similar users from three indicators, the CF method can use Top-N neighbor to improve the poor relationship between users in the rating matrix.

Step 1: Compute frequency of E-commerce webpage visit (indicator 1)

1. Compute the hits on item or category**:** The visiting frequency is calculated by counting the number of visits to category or item by users in a particular session. The hit consists of two parts as given in **Equation 2.8**.

$$hits_{user_i}^{category_j} = count(user_j, category_j) + \sum_{k=1}^{l} count(user_i, item_k^{category_j})$$

Equation 2.8: Formula to compute hits of user on item and category

Where, first part represents the count of category visited by the user and the second part represents the total items visited by the user, which belong to a particular category.

2. Utilize hits to compute frequency: Once hit count, the frequency is calculated as the ratio of category of hits to the length of the users browsing path as given **Equation 2.9**.

$$Frequency_{user_i}^{category_j} = \frac{hits_{user_i}^{category_j}}{(length(P_{user_i})}$$

Equation 2.9: Formula to compute frequency of hit

3. Formulate category-user frequency matrix**:** Once the frequency of user visit on category is calculated, it is used to form a category-user frequency matrix. An example of a category-user matrix with frequency characteristics is given in **Table 2.31**.

	Category1	Category 2	Category 3	Category 4
User 1	0.8	0.2	-	-
User 2	-	-	0.85	0.28
User 3	0.28	0.28	-	0.57
User 4	0.57	0.28	0.28	-

Table 2.31: User-category frequency matrix

4. Compute frequency similarity from user-category frequency matrix: User-based collaborative filtering is used to compute frequency similarity of a user-category matrix. There are many formulae available to compute similarity, some of the prominent are Cosine similarity and Pearson Correlation Coefficient, here we are using cosine similarity, which is given in **Equation 2.10**.

$$\mathrm{Sim}(user_u, user_v(Frequency)) = \frac{\sum_{i=1}^{m}(R_{u,i})(R_{v,i})}{\sqrt{\sum_{i=1}^{m}(R_{u,i})^2}\ \sqrt{\sum_{i=1}^{m}(R_{v,i})^2}}$$

Equation 2.10: Cosine similarity function to compute frequency similarity

Where, $(R_{u,i})$ denotes the ratings of $user_u$ on $item_i$ and $(R_{v,i})$ denotes the rating of $user_v$ on $item_i$.

Step 2: Compute duration of time spent on E-commerce webpages (Indicator 2)

1. Compute relative duration**:** The relative duration represent the total time spent by each user on each category with respect to the total time a user spent on each session. The relative duration is computed by dividing the $user_i$ spends time on category $category_j$ with respect to the total time spend by users on each session by visiting different category and item. **Equation 2.11** provides formula to compute relative duration

$$\text{Relative duration}_{\text{user}_i}^{\text{category}_j} = \frac{\text{Duration}_{\text{user}_i}^{\text{Category}_j}}{(\text{time}(\text{P}_{\text{user}_i})}$$

Equation 2.11: Formula to compute relative duration

Where nominator represents time spend by a user on category and denominator represents total time spend by a user on item and category on each session. For example, let us consider time spent by user on category as shown in **Table 2.32** then total time spent by User1 is 10+20=30 sec then relative duration of user 1 on category 1 is, $\text{Relative duration}_{\text{user}_1}^{\text{category}_1} = \frac{10}{30} = 0.33$

User	Category	Time in Sec
User1	Category 1	10 sec
User 1	Category 2	20 sec
User 2	Category 3	19 sec
User 2	Category 4	15 sec
User 3	Category 1	25 sec
User 3	Category 2	35 sec
User 3	Category 4	20 sec
User 4	Category 1	5 sec
User 4	Category 2	30 sec
User 4	Category 3	35 sec

Table 2.32: User spend time on category

2. Formulate user-category relative duration matrix: Use relative duration to form user-category matrix that represents a user's spent relative duration to the corresponding category as shown in **Table 2.33**.

	Category1	Category 2	Category 3	Category 4
User 1	0.33	0.66	-	-
User 2	-	-	0.55	0.44
User 3	0.31	0.43	-	0.25
User 4	0.07	0.42	0.5	-

Table 2.33: User-category relative duration matrix

3. Compute duration similarity between users**:** Use Cosine similarity function to compute the duration similarity between users using **Equation 2.12**.

$$\mathrm{Sim}(user_u, user_v(Duration)) = \frac{\sum_{i=1}^{m}(R_{u,i})(R_{v,i})}{\sqrt{\sum_{i=1}^{m}(R_{u,i})^2}\ \sqrt{\sum_{i=1}^{m}(R_{v,i})^2}}$$

Equation 2.12: Cosine similarity function to compute duration similarity

Where ($R_{u,i}$) denote the ratings of $user_u$ on item i similarly ($R_{v,i}$) denotes the rating of $user_v$ on item i.

Step 3: Compute users browsing path (Indicator 3)

The browsing path P_i {url_1, $url_{2.....}$ url_n} of $users_i$ is sequence of web pages browsed during a particular session. The browsing path indicates the users visited categories and items in a particular session. For example, **P_1{ctg1, $Item_1^1$, ctg2, $Item_2^1$, $Item_1^2$}** represents that user visit category **ctg1** then visit **$Item_1^1$** which belong to category 1 and after that visited category 2 and visited **$Item_2^1$** item and finally **$Item_2^2$** visited which belong to category 2. An example of the browsing path is shown in **Table 2.34**.

User	Item Browsing Path	Category Browsing Path
User1	P{ctg1, $Item_1^1$,ctg2, $Item_2^1$, $Item_2^2$}	CtgPath {ctg1,ctg2}
User 2	P{ ctg3, $Item_3^1$, $Item_3^2$, $Item_3^3$, ctg4, $Item_4^3$}	CtgPath {ctg3,ctg4}
User 3	P{ ctg4, $Item_4^1$, $Item_4^2$, ctg1, $Item_1^2$, ctg2, $Item_2^1$}	CtgPath {ctg4,ctg1,ctg2}
User 4	P{ ctg1, $Item_1^2$, ctg2, $Item_2^2$, $Item_1^1$,ctg3, $Item_3^1$}	CtgPath {ctg1,ctg2,ctg3}

Table 2.34: Users browsing path

1. Compute path similarity: Path similarities between two users compute the common path length divided by the maximal length. A common path is defined if two users visit the same categories in the same order. If there is more than one common path between two users, the longest one is used in the path similarity. The Equation to compute path similarity is provided in **Equation 2.13**.

$$\mathrm{Sim}(user_u, user_v(Path)) = Max(\frac{common(category_p, catgory_q)}{lenght(category_p, category_q)})$$

Equation 2.13: Equation to compute path similarity

Step 4: Compute total similarity using path, frequency and duration

Once frequency, path and duration similarity are computed, they are used to compute total similarity using **Equation 2.14**.

$$\mathrm{Sim}(user_u, user_v = \alpha * \mathrm{Sim}(user_u, user_v(Path)) + \gamma * \mathrm{Sim}(user_u, user_v(Frequency)) + \mathrm{B} * \mathrm{Sim}(user_u, user_v(Duration))$$

Equation 2.14: Equation to compute the total similarity

Where B,γ,α are used to adjust the weight of path, frequency and duration such that B+γ+α=1.

2.1.9 E-Commerce Product Recommendation Using Historical Purchases and Clickstream Data by Xiao & Ezeife, 2018 (HPCRec18)

In E-commerce, user-item rating matrices for collaborative filtering recommendation systems are usually binary and sparse which shows whether a user has purchased an item previously or not. Some existing recommendation system **Kim, Yum, Song, & Kim, 2005** uses decision tree, **Kim, & Yum, 2011** uses association rule mining and **Su & Chen,2013** use category based measurements from clickstream data to improve recommendations, however, these recommendation systems fail to integrate valuable information from historical purchases and consequential bond information between session-based clicks and purchases. Thus, **Xiao & Ezeife, 2018** proposes Historical Purchase with Clickstream recommendation system (HPCRec18), which normalizes the historical purchase frequency matrix to improve rating quality, and mines the session-based consequential bond between clicks and purchases to generate potential ratings to improve the rating quantity.

Example

Let's consider frequency and the consequential table containing clicks and purchases as shown in **Table 2.35** as input, where frequency table contains the number of time product purchased by a user, and the consequential table contains clicks and purchases on each session.

SessionId	UserId	Clicks	Purchases
1	1	1,2	2
2	1	3,5,2,3	2,3
3	2	2,1,4	1,2,4
4	2	4,4,1,2	2,4,4
5	3	1,2,1	1
6	3	3,5,2	

User\Item	1	2	3	4
1	?	2	1	?
2	1	2	?	3
3	1	?	?	?

Table 2.35: Consequential table on left and purchase frequency table on right

Step 1: Normalize the purchase frequency for each user on each item using the unit formula in a user-item purchase frequency table. The unit normalization function takes purchase frequency matrix as input and normalizes the frequencies into numbers between 0 and 1 using the unit vector formula as given in **Equation 2.15**.

$$Frequency\ normalization\ of\ user\ u\ on\ item\ i = \frac{item\ i}{\sqrt{item_1^2 + item_2^2 + item_3^2 + \cdots + item_n^2}}$$

Equation 2.15: Unit vector formula to normalize purchase frequency

For example, for user 2, the purchase vector is <1, 2, 0, 3>, so the normalized purchase frequency for user 2 on item 2 is $2/\sqrt{1^2 + 2^2 + 0^2 + 3^2}$=0.53. In the same way, we can get normalize frequency matrix as shown in **Table 2.36**.

Customer\Item	1	2	3	4
1	?	2	1	?
2	1	2	?	3
3	1	?	?	?

Normalized ➔

Customer\Item	1	2	3	4
1	?	0.89	0.45	?
2	0.27	0.53	?	0.8
3	1	?	?	?

Table 2.36: Non-normalized user-item matrix on left and normalized matrix on right

Step 2: For each session without purchase in the consequential table, compute click set similarity using Clickstream Sequence Similarity measurement (CSSM) function using the longest common subsequence rate.

Longest common subsequence rate LCSR (x, y) = (LCS (x, y))∕(max (|x|, |y|))

Equation 2.16: Longest common subsequence rate

$$LCS(X_i, Y_j) = \begin{cases} \emptyset & \text{if } i = 0 \text{ or } j = 0 \\ LCS(X_{i-1}, Y_{j-1}) \cap x_i & \text{if } x_i = y_j \\ \text{longest}(LCS(X_i, Y_{j-1}), LCS(X_{i-1}, Y_j)) & \text{if } x_i \neq y_j \end{cases}$$

Equation 2.17: Longest common sequence (LCS)

For example, there is no purchase information of session 6 for user 3 in the consequential table. So, let's compute the clickstream sequence similarity between session 6 and other session as given below:

CSSM between session 6 and session 1(<3, 5, 2>, <1, 2>) =0.37,
CSSM between session 6 and session 2 (<3, 5, 2>, <3, 5, 2, 3>) =0.845,
CSSM between session 6 and session 3 (<3, 5, 2>, <2, 1, 4>) =0.33,
CSSM between session 6 and session 4 (<3, 5, 2>, <4, 4, 1, 2>) = 0.245,
CSSM between session 6 and session 5 (<3, 5, 2>, <1, 2, 1>) =0.295

Step 3: Form a weighted transaction table using the similarity as weight and purchases as transaction records.

Purchase	<2>	<2,3>	<1,2,4>	<2,4,4>	<1>
1	0.37	0.845	0.33	0.245	0.295

Table 2.37: Weighted transactional table of purchase set created from consequential bond

Step 4: Call TWFI (Transaction-based Weighted Frequent Item) function, which takes a weighted transaction table, where weights are assigned to each transaction as input and returns items with weighted support in a given threshold. For example, let's consider minimum weighted support=0.1, then, we will have frequent weighted transaction table as shown in **Table 2.38**.

Purchase(Transaction records)	2	2,3	1,2,4	2,4,4	1
Weight	0.37	0.845	0.33	0.245	0.295

Table 2.38: Weighted frequent transaction table

Step 5: Calculate support to form a distinct item from set of all the transactions

Item	1	2	3	4
Support	2	4	1	3

Table 2.39: Support for item present in weighted frequent transaction table

Step 6: Compute the average weighted support for each item using (AWS=AW*support) ,where $AW = sum(weight)/support$). For example, AWS (1) =0.33+ 0.295=0.625, AWS (4) =0.33+ 0.245+0.245=0.82.

Item	1	2	3	4
AWS	0.625	1.97	0.845	0.82

Table 2.40: Weight for item present in purchase pattern

2.2 Sequential Pattern Mining Algorithms

2.2.1 **GSP** *(Generalized sequential pattern mining) algorithm*

GSP is an Apriori-based sequential pattern mining algorithm introduced by **Srikant & Agrawal, 1996**. The main step in the GSP algorithm is candidate generation (C_k) and pruning (L_k). To generate a candidate, we can use pair found in K-1th pass by merging. According to the algorithm, first sequence W_1 and second sequence W_2 can be merged, if subsequences obtained by removal of the first element of sequence W_1 and last element of sequence W_2 are same. In the second step, we need to prune candidate that contains a subsequence which is infrequent in K-1 pass. We need to iterate the process of candidate generation (C_k) and pruning (L_k) until a candidate set is empty. Finally, frequent sequences are the union of the entire list obtained so far.

Example of GSP algorithm.

Input: sequence database (**Table 2.41**), **minimum support**=2 and candidate set (C_1) = {A, B, C, D, E, F, G} and **algorithm**=GSP

SID	Sequences
1	<(A),(B),(FG),(C),(D)>
2	<(B),(G),(D)>
3	<(B),(F),(G),(A,B)>
4	<(F),(A,B),(C),(D)>
5	<(A),(B,C),(G),(F),(D,E)>

Table 2.41: Sequence Database representing customer purchase

Step 1: Find 1- frequent sequence (L_1) satisfying minimum support: Check the minimum support threshold of each singleton item and keep only sequences with occurrence or support count in the database greater than or equal to the minimum support count of 2. For example, (L_1) = {< (A):4>, <(B):4>, <(C):3>, <(D):4>, <(F):4>, <(G):4>}.

Step 2: Generate candidate sequence ($C_{k=2}$) using $L_1 \bowtie_{GSPjoin} L_1$

To generate larger candidate set 2, use 1-frequent sequence (L_1) found in step 1 to join itself using GSPjoin way, which can be written as $L_{(k-1)} \bowtie_{GSPjoin} L_{(k-1)}$ and it requires every sequence (W_1) found in first $L_{(k-1)}$ joins with other sequence (W_2) in the second if subsequences obtained by removal of the first element of W_1 and last element of W_2 are same. In our case, we are generating sequences with candidate 2, ($C_{k=2}$), which can generate 51 types of 2-length candidate set using Apriori algorithm as present in **Table 2.42**.

<(A),(A)>	<(A),(B)>	<(A),(C)>	<(A),(D)>	<(A),(F)>	<(A),(G)>
<(B),(A)>	<(B),(B)>	<(B),(C)>	<(B),(D)>	<(B),(F)>	<(B),(G)>
<(C),(A)>	<(C),(B)>	< (C),(C)>	<(C),(D)>	<(C),(F)>	<(C),(G)>
<(D),(A)>	<(D),(B)>	<(D),(C)>	<(D),(D)>	<(D),(F)>	<(D),(G)>
<(F),(A)>	<(F),(B)>	<(F),(C)>	<(F), (D)>	<(F),(F)>	<(F),(G)>
<(G),(A)>	<(G),(B)>	<(G),(C)>	<(G),(D)>	<(G),(F)>	<(G),(G)>
<(A,B)>	<(A,C)>	<(A,D)>	<(A,F)>	<(A,G)>	<(B,C)>
<(B,D)>	<(B,F)>	<(B,G)>	<(C,D)>	<(C,F)>	<(C,G)>
<(D,F)>	<(D,G)>	<(F,G)>			

Table 2.42: Candidate set (C2) generated from L1 GSP join L1

Step 3: Find 2- frequent sequences (L_2) by counting the occurrence of 2-sequences in candidate sequence (C_2) to keep the only sequence with occurrence or support count in the database greater than or equal to the minimum support. For example, L_2= {<(A), (B)>, <(A, B)>, <(A), (C)>, <(A), (D)>, <(A), (F)>, <(A), (G)>, <(B), (C)>, <(B), (D)>, <(B), (F)>, <(B), (G)>, <(C), (D)>, <(F), (A)>, <(F), (B)>, <(F), (C)>, <(F), (C)>, <(F), (D)>, <(G), (D)>}.

Step 4: Repeat process of candidate generation and pruning until the result of candidate generate (C_k) and prune (L_k) for finding frequent sequence is an empty set.

Output: Finally, the output frequent sequences as union of L_1 U L_2 U L_3 U L_4 U … L_k

1-Frequent Sequences	2-Frequent Sequences	3-Frequent Sequences	4-Frequent Sequences
<(A)>, <(B)>, <(C)>, <(D)>, <(F)>, <(G)>	<(A), (B)>, <(A, B)>, <(A), (C)>, <(A), (D)>, <(A), (F)>, <(A), (G)>, <(B), (C)>, <(B), (D)>, <(B), (F)>, <(B), (G)>, <(C), (D)>, <(F), (A)>, <(F), (B)>, <(F), (C)>, <(F), (D)>, <(G), (D)>	<(F), (C), (D)> <(F), (B, A)> <(F), (A, B)> <(B), (G), (D)> <(B), (F), (D)> <(B), (C), (D)> <(A), (G), (D)> <(A), (F), (D)> <(A), (C), (D)> <(A), (B), (G)> <(A), (B), (F)> <(A), (B), (D)>	<(A), (B), (G), (D)> <(A), (B), (F), (D)>

Table 2.43: n-frequent sequences generated by GSP from sequence database

2.2.2 PrefixSpan (Prefix-projected sequential pattern mining) algorithm

PrefixSpan algorithm **(Pei, et al, 2001)** proposed a new approach for finding the sequential pattern by avoiding generation of candidates. The algorithm is based on the creation of a projected database; the projected database is a set of sub-pattern in the original database that is suffixes of a pattern containing the prefix. The prefixSpan algorithm starts computing the patterns of size 1 that fulfill the frequency threshold in the database. Later, for each pattern of size 1, prefixSpan computes its projected database and find the patterns that fulfill the frequency threshold in the projected database. The pattern of size 1 grows concatenating it with each element of the pattern found in the projected database generating patterns of size 2; this process is recursive until the projected database is empty.

Example of prefixSpan algorithm

Let us consider sequence database as shown in **Table 2.44** as **input, Minimum support**=2**, Candidate sets**={A,B,C,D,E,F}

ID	Sequence
100	<(A),(A,B,C),(A,C),(D),(C,F)>
200	<(A,D),(C),(B,C),(A,E)>
300	<(E,F),(A,B),(D,F),(C), (B)>
400	<(E), (G), (A, F), (C), (B), (C) >

Table 2.44: Sequence input database for prefixSpan

Step 1: Count the support of singleton sequence

Check the minimum support threshold of each singleton item and keep only sequences with occurrence or support count in the database greater than or equal to the minimum support count of 2. In our case, we do have support for each singleton sequences as given in **Table 2.45**.

<(A)>	<(B)>	<(C)>	<(D)>	<(E)>	<(F)>	<(G)>
4	4	4	3	3	3	1

Table 2.45: support for singleton sequences

Step 2: Prune singleton sequences with specified minimum threshold

In this step, we need to prune sequence that does not satisfy the minimum support. In our case, minimum support is 2 and we can see that <(G)> doesn't satisfy minimum support, so we need to prune g from singleton sequence.

Step 3: Create a projected database by considering 1-frequent sequence from a sequential database

The next step is to divide search space into a set of projected databases according to the frequent

prefixes. For example, for each sequence of the sequence database (**Table 2.44**), the projected database of frequent 1 sequence <(A)> would consist of all the items that appear after the sequence <(A)> (that is) the projected database for <(A)> will consist of all the sequences with its prefix as <(A)>. **Table 2.46** gives the projected database for all the frequent 1 items.

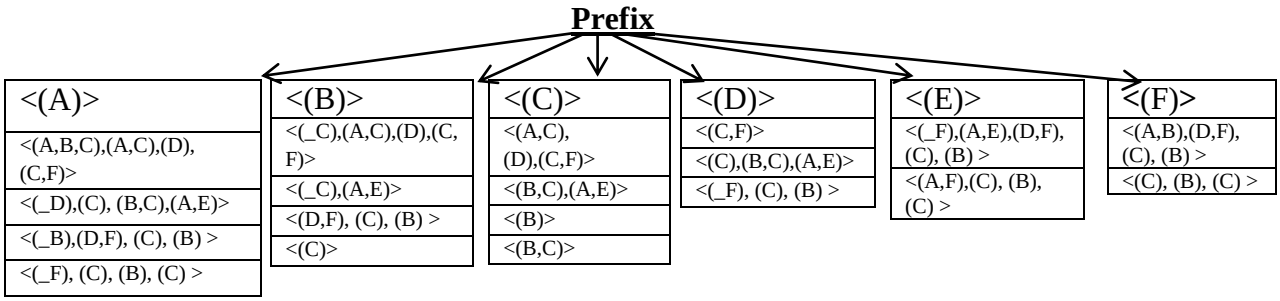


Table 2.46: Project database of sequence database

Step 4: Find frequent sequences from the projected databases and test with minimum threshold repeatedly until no projected database can be created

1. Find the sequence present in projected database. Let us consider projected database of <(D)> is present in **Table 2.47**.

<(D)>
<(C,F>
<(C), (B,C),(A,E)>
<(_F), (C),(B) >

Table 2.47: Projected database of sequence <(D)>

2. The projected database is scanned to find the frequent items in it. In our case, let's scan **Table 2.47** then we will find support as shown in **Table 2.48**. In our example, only <(B)> and <(C)> are frequent.

<(A)>	<(B)>	<(C)>	<(D)>	<(E)>	<(F)>	<(_F)>
1	2	3	0	1	1	1

Table 2.48: Frequencies of item presented in projected database of sequence <(D)>

3. Now, the projected database for sequence <(D), (B)> and <(D), (C)> are constructed using step 3. Furthermore, their respective projected databases are scanned to get the frequent items in their projected dbs.

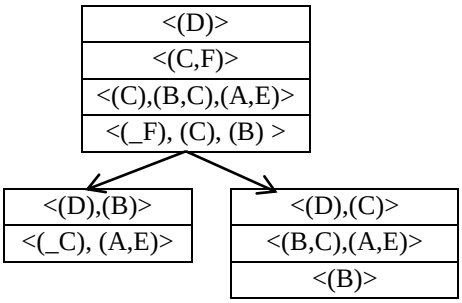


Table 2.49: project database of sequence <(D), (B)> and <(D), (C)>

4. Since item present in the projected database <(D), (B)> is infrequent. So, compute frequency of item present in the projected database <(D), (C)> and we can see only <(B)> is frequent.

<(B)>	**<(A)>**	**<(E)>**	**<(C)>**
2	1	1	1

Table 2.50: Frequencies of item present in projected database of sequence <(D), (C)>

5. Create the projected database of <(D), (C), (B)>. Since the projected database of <(D), (C), (B)> is empty. So, terminate the process.

<(D), (C), (B)>
Θ

Table 2.51: Projected database of sequence <(D), (C), (B)>

2.2.3 SPADE (Sequential Pattern Discovery using Equivalence classes) algorithm

SPADE algorithm was first introduced by **Zaki, 2001.** This algorithm is based on mining the subsequence by using vertical data format. The vertical data format consists of syntax: **<itemset: (Sequence_ID, event_ID)>** that means for each itemset we record sequence identifier and event identifier. The event identifier is also called as a timestamp. SPADE requires one scan to find frequent 1-sequences. To find candidate 2-sequence, we need to join all pairs of single items when they are frequent if they share the same sequence identifier and their event identifier follows the same sequential ordering and pattern are grown similarly. Support of K-sequence can be determined by joining the ID lists of K-1 sequences.

Example of SPADE

Let us consider, sequential database (**Table 2.52**) as **input, minimum support**=2 and **candidate set** = {A, B, C, D, E, F, G, H}

Sequence ID	Sequence
1	<(C,D),(A,B,C),(A,B,F) ,(A,C,D,F)>
2	<(A,B,F),(E)>
3	<(A,B,F)>
4	<(D,G,H),(B,F),(A,G,H)>

Table 2.52: Input sequential database for SPADE

Step 1: Find frequent singleton sequence

Keep only sequences with occurrence or support count in the database greater than or equal to the minimum support count of 2. In our case, C, E G and H are infrequent. So, frequent singleton sequences are {<A>:4, <B>:4, <D>:2 and <F>:4}.

Step 2: Convert the sequence database into vertical data format

Vertical data format contains the item present in the sequence database by their sequence id and events ids. The events id helps to determine the sequence of events. So, if we take one sequence <(CD) (ABC) (ABF) (ACDF)> from sequence database **Table 2.52** then we can see that CD is considered as one event, ABC is considered as another event and so on. So, the vertical data format of the sequential database in our case presented in **Table 2.53**.

Sequence ID(SID)	Event ID(EID)	Itemset
1	10	(C,D)
1	15	(A,B,C)
1	20	(A,B,F)
1	25	(A,C,D,F)
2	15	(A,B,F)
2	20	(E)
3	10	(A,B,F)
4	10	(D,G,H)
4	20	(B,F)
4	25	(A,G,H)

Table 2.53: Vertical data format of sequence database

Step 3: List frequent singleton sequences along with their sequence ID (SID) and event ID (EID)

List frequent singleton sequences from **Table 2.53** with sequence ID (ID) and event ID (EID) separately so that they can be used to generate the larger sequence. For example, we can see that item A is present in event {15, 20, 25, 15, 10, 25}.

A	
SID	EID
1	15
1	20
1	25
2	15
3	10
4	25

B	
SID	EID
1	15
1	20
2	15
3	10
4	20

D	
SID	EID
1	10
1	25
4	10

F	
SID	EID
1	20
1	25
2	15
3	10
4	20

Table 2.54: Frequent 1-sequence with event ID and item ID

Step 4: Generate 2-frequent sequences by joining all pairs of single item from step 2

To find candidate 2 frequent sequences, we need to join all pairs of single items when they are frequent and if they share same sequence identifier (SID) and event identifier (EID) follows a same sequential ordering as present in **Table 2.55**.

B	
SID	EID
1	15
1	20
2	15
3	10
4	20

D	
SID	EID
1	10
1	25
4	10

SID join →

SID	EID(D)	EID(B)
1	10	15
1	10	20
4	10	20

Table 2.55: Process of generating 2-frequent sequences in SPADE

Step 5: Repeat the process of joining and pruning until frequent sequences are present in vertical database.

Output: n-frequent are collection of {1,2….n} frequent sequences

Frequent 1-sequence

Item	Support
A	4
B	4
D	2
F	4

Frequent 2-sequences

Item	Support
AB	3
AF	3
B-> A	2
BF	4
D->A	2
D->B	2
D-> F	2
F->A	2

Frequent 3-sequences

Item	Support
ABF	3
BF->A	2
D->BF	2
D->B->A	2
D->F->A	2

Frequent 4-Sequences

Item	Support
D->BF->A	2

Table 2.56: n-frequent sequences generated by SPADE algorithms

CHAPTER 3: PROPOSED SYSTEM TO GENERATE SEQUENCE DATASET FOR E-COMMERCE RECOMMENDATION

There are many reasons for generating a sequential dataset from different E-commerce source such as historical, clickstream. In E-commerce, historical information of products purchased online for each online store is stored in transactional databases; sequential purchase behavior of the user cannot be identified from transactional data without using a sequential pattern mining method. So, without analyzing historical sequential data from E-commerce environment, we cannot provide the proper recommendation to the user such as: finding the next possible item for user A, if user A purchased laptop last month from BestBuy **(BestBuy, 2019)** or Amazon **(Amazon, 2019)**.

Additionally, collaborative filtering finds users' closest neighbor to generate matching recommendations. However, what people want from recommender systems is not whether the system can predict rating values accurately, but recommendations that match their interests according to time span. Thus, E-commerce recommendation system accuracy will be improved if more complex sequential patterns of users' historical purchase behavior are learned and included in the user-item matrix to make it quantitatively and qualitatively rich before applying collaborative filtering.

3.1 Problem Definition

Given E-commerce historical click and purchase data over a certain period of time as input, the problem being addressed by this thesis is to find the frequent periodic (daily, weekly, monthly) sequential purchase and click patterns in the first stage. Then, these sequential purchase and click patterns can be used to make user-item matrix qualitatively (specifying level of interest or value for already rated items) and quantitatively (finding the possible rating for previously unknown ratings) rich before applying collaborative filtering (CF) to improve the overall accuracy of recommendation.

3.2 Proposed Historical Sequential Recommendation- (HSPRec) System

The major goal of the proposed Historical (H), Sequential (SP), Recommendation (Rec)- HSPRec is to mine frequent sequential pattern from E-commerce historical data to enhance a user-item rating matrix from discovered patterns. Thus, HSPRec takes minimum support, historical click and purchase database containing consequential bond as input to generate rich user-item matrix as output as shown in **Algorithm 3.1**.

Algorithm 1: HSPRec (Historical sequential recommendation)

Input: minimum support (s), historical user-item purchase frequency matrix (M), consequential bond (CB), historical purchase database (DB), historical click database (CDB)

Output: user-item purchase frequency matrix (M_2)

Intermediates: historical sequential purchase database (SDB), weighted purchase pattern (WP), historical sequential click database (SCDB), rule recommended purchase items (RPI), each user u's rating of item i in the matrices is referred to as r_{ui}.

1. : purchase sequential database (SDB)← SHOD (DB) using Algorithm 2 present in **section 3.2.1.**
2. : user-item purchase frequency matrix (M_1) ← M modified with Sequential Pattern Rule (SDB) using **section 3.2.2.**
3. : **for** each user u do
4. : weighted purchase pattern for user u, (WP_u) ← null;
5. : end
6. : **for** each user u **do**
7. : **if** u has both click and purchase sequences **then**
8. : compute Click Purchase Similarity CPS (click sequence, purchase sequence) from SCDB and SDB using **section 3.2.3.**
9. : weighted purchase patter for user u, (WP_u) ← CPS (click sequence, purchase sequence) using **section 3.2.3**;
10. : **else**
11. : rule recommended purchase items (RPI) ← Sequential Pattern Rule (SCDB) using **section 3.2.1;**
12. : weighted purchase patter for user u, (WP_u) ← CPS (click sequence, purchase sequence) using **section 3.2.3**;
13. : **end**
14. : rating of item i by user u (r_{ui}) ← weighted purchase patter for user u, (WP_u);
15. : M_2 ← M_1 modified with rating r_{ui}
16. : **end**

Algorithm 3.1: Historical sequential recommendation (HSPRec) system

Steps in the proposed HSPRec system:

Step 1: Convert historical purchase information (present in **Table 3.9**) to user-item purchase frequency (present in **Table 3.1**) by counting the number of each purchased by a user. For example, User 2 purchased item 1 and item 2 twice and purchased item 3, item 4, item 5 and item 6 only once.

User/item	1	2	3	4	5	6	7
A2HD75EMZR8QLN (User1)	1	1	1	?	1	1	1
A1026QJYJTVE5T (User2)	2	2	1	1	1	1	?
A1026RERIHUK3C (User3)	1	1	?	?	1	1	?
A0130ZI3HIT9N5V (User4)	?	1	?	?	?	1	1
A31ZC98HM9C4LP (User5)	?	?	?	?	?	?	?

Table 3.1 :User-item purchase frequency matrix created from historical data

Step 2: Create a daily purchase sequential database (**Table 3.2**) of customer purchase (**Table 3.9**) by applying the sequential historical periodic database (SHOD) generation algorithm presented in section **3.2.1**.

SID	Purchase sequence
1	< (1,2), (3), (6), (7), (5)>
2	<(1, 4), (3), (2), (1, 2, 5, 6)>
3	<(1), (2), (6), (5)>
4	<(2) , (6, 7)>

Table 3.2: Daily purchase sequential database

For example, User 2 daily purchase sequence is < (1, 4), (3), (2), (1, 2, 5, 6)>, which shows User 2 purchased item 1 and item 4 together on the same day and purchased item 3 on the next day then purchased item 2 on another day and finally, purchased items 1, 2, 5 and 6 together on the next day.

Step 3: Input daily purchase sequential database (**Table 3.2**) to Sequential Pattern Rule (SPR) module present in section **3.2.2** to generate sequential rule from frequent purchase. For example,

1-frequent purchase sequences = {< (1)>, < (2)>, < (3)>, < (5)>, < (6)>, < (7)>}

Some of 2-frequent purchase sequences= {< (6), (5)>, < (3), (6)>, < (3), (5)>, < (2), (7)>, < (2), (6)>, < (2), (5)>}

Some of 3-frequent purchase sequences= {< (2), (6), (5)>, < (1), (6), (5)>, < (1), (3), (6)>, < (1), (3), (5)>, < (1), (2), (6)>}

Thus, some of the possible sequential purchase pattern rules based on frequent purchase sequences are:

(a) 1, 5 → 3 , (b) 2, 6 → 1 , (c) 2, 6 → 5

Where, rule (a) states that if user purchases item 1 and item 5 together then the user will purchase item 3 in next purchase, which will be applied in case of User 3 in user-item purchase frequency matrix.

Step 4: Reconstruct user-item purchase frequency matrix by using purchase sequential rule

Rule (a) is applied in case of User3, rule (b) and rule c are applied in case of User4. Thus, enhanced user-item purchase frequency matrix is present in **Table 3.3**.

User/item	1	2	3	4	5	6	7
User1	1	1	1	?	1	1	1
User2	2	2	1	1	1	1	?
User3	1	1	1	?	1	1	?
User4	1	1	?	?	1	1	1
User5	?	?	?	?	?	?	?

Table 3.3: Enhanced user-item purchase frequency matrix

As we can see from enhanced user-item purchase frequency matrix (**Table 3.3**), there is no purchase information for User5. Thus, to find the purchase information of User5, we are going to analyze the consequential bond of click and purchase by considering sequential patterns. Let us consider, historical click and purchase as present in **Table 3.4**.

UID	Clicks sequence	Purchases sequence
1	<(1,2,3), (7,5,3), (1,6), (6), (1,5)>	< (1, 2), (3), (6), (7), (5)>
2	<(1,4), (6,3), (1,2), (1,2,5,6)>	<(1,4), (3), (2), (1, 2, 5, 6)>
3	<(1,5), (6,5,2), (6), (5)>	<(1), (2), (6), (5)>
4	<(2,7), (6,6,7)>	<(2) , (6, 7)>
5	<(1,5)>	?

Table 3.4: Consequential bond of sequence of click and purchase

Step 5: For each user, where clicks happened without purchases such as for user 5 in **Table 3.4**, create a click periodic sequential database (**Table 3.5**) by neglecting purchase from the consequential bond. Finally, input a click sequential database to Sequential Pattern Rule (SPR) (present in **3.2.2**) module to get the recommended item as the predicted purchase item. In our case, we have to find the click sequential rule which will recommend purchase item when the user purchased item 1 and item 5 together and let's further consider item 1 and item 3 are recommend to User 5 from Sequential Pattern Rule (SPR) (present in **3.2.2**).

SID	Click sequence
1	<(1,2,3), (7,5,3), (1,6), (6), (1,5)>
2	<(1,4), (6,3), (1,2), (1,2,5,6)>
3	<(1,5), (6,5,2), (6), (5)>
4	<(2,7), (6,6,7)>
5	<(1,5)>

Table 3.5: Click sequential database

Step 6: Once the purchased item is recommended for a user (where, the click has happened without purchase), compute click and purchase similarity using Click and Purchase Similarity (CPS) module present in **3.2.3**.

Step 7: Supply CPS value to purchase pattern including a recommended item from Sequential Pattern Rule (SPR) to create weighted purchase pattern (**Table 3.6**).

Purchased Sequence	CPS
< (1, 2), (3), (6), (7), (5)>	0.624
<(1, 4),(3), (2), (1, 2, 5, 6)>	0.834
<(1), (2), (6), (5)>	0.636
<(2) , (6, 7)>	0.67
<(1) , (3)>	0.5

Table 3.6: Weighted purchase patterns

Step 8: Input weighted purchase pattern (**Table 3.6**) to Weighted Frequent Purchase Pattern Miner (WFPPM) present in section **3.2.4** to calculate the weight for each frequent individual item based on its occurrence in weighted purchase patterns. In our case, R_1= 0.68, R_2=0.71, R_3=0.65, R_4=0.834, R_5=0.698, R_6=0.691, R_7=0.647.

Step 9: Repeat steps 4, 5, 6 7 and 8, if there are more users without purchase, otherwise assign computed item weight to enhance user-item purchase frequency matrix (**Table 3.3**).

User/item	1	2	3	4	5	6	7
User1	1	1	1	?	1	1	1
User2	2	2	1	1	1	1	?
User3	1	1	1	?	1	1	?
User4	1	1	?	?	1	1	1
User5	0.68	0.71	0.65	0.834	0.698	0.691	0.647

Table 3.7: Quantitatively rich user-item purchase frequency matrix

Step 9: Normalize quantitatively rich user-item purchase frequency matrix (**Table 3.7**) using unit normalization formula present in section **3.2.5** to provide the level of user's interest on item

between 0 and 1 as shown in Table 3.8. We can see, that normalized quantitatively rich user-item matrix (Table 3.8) is less sparse compared to initial user-item purchase frequency matrix (Table 3.1).

User/item	1	2	3	4	5	6	7
User1	0.40	0.40	0.40	?	0.40	0.40	0.40
User2	0.57	0.57	0.28	0.28	0.28	0.28	?
User3	0.44	0.44	0.44	?	0.44	0.44	?
User4	0.44	0.44	?	?	0.44	0.44	0.44
User5	0.37	0.39	0.35	0.45	0.38	0.38	029

Table 3.8: Normalized enrich user-item purchase frequency matrix

3.2.1 HSPRec: Periodic Sequential Database Generation Module

The proposed sequential (S), historical (H), periodic (O), database (D) - (SHOD) generation module takes historical (click or purchase database) data as input and produce periodic (daily, weekly, monthly) sequential (click or purchase) database as output as present in Algorithm 3.2.

Algorithm 2: SHOD (Sequential historical periodic database) System

Input: historical click and/or purchase data
Output: periodic (daily, weekly, monthly) sequential database
Intermediates: Tuserid=temporary userid, Ttimestamp=temporary timestamp, -I: end of itemset, and –S: end of sequence

```
1.  : historical.txt ← extract userid, productid, timestamp from historical data
2.  : read first line from historical.txt and store userid, timestamp into temporary variable (Tvar)
3.  : for all user N ϵ historical.txt do
4.  :     If (userid==Tvar.userid)
5.  :             Tdur ← timestamp - Tvar.timestamp
6.  :                     If (Tdur <=24 hrs)
7.  :                             add item to daily-sequence-database.txt and goto step 3
8.  :                     Else
9.  :                             add -I to indicate end of itemset and goto step 3
10. :                     If (Tdur <=168 hrs)
11. :                             add item to weekly-sequence-database.txt and goto step 3
12. :                     Else
13. :                             add -I to indicate end of itemset and goto step 3
14. :                     If (Tdur > 672 hrs)
15. :                             add item to monthly-sequence-database.txt and goto step 3
16. :                     Else
17. :                             add -I to indicate end of itemset and goto step 3
18. :   Else (userid! =Tvar.userid)
19. :    add -I and -S after item to indicate end of itemset and sequence and update Tvar.userid and goto
    step 3
```

20. : End **if**
21. : End **for**

Algorithm 3.2: Algorithm to create sequential historical periodic database

Example to create daily sequence database

To explain the SHOD algorithm step by step, let us consider historical purchase data (**Table 3.9**) as input, where Uid represents user identity, Productid represents product identity, Product represents name of the product and Purchastime represents timestamp when purchased occurred.

Uid	Productid	Product	Purchasetime
A2HDEMZR8QLN	B003UYU16G	1	2014-04-04 11:30:11
A2HDEMZR8QLN	B003MYU66K	2	2014-04-04 13:25:19
A2HDEMZR8QLN	B00A9NE84C	3	2014-04-05 15:56:32
A2HDEMZR8QLN	B00EG0C20G	6	2014-04-06 16:18:26
A2HDEMZR8QLN	B000SAUVC4	7	2014-04-07 18:59:21
A2HDEMZR8QLN	B000GAYQTU	5	2014-04-08 21:19:55
A1026QJYJTVE5T	B003UYU16G	1	2014-04-13 04:04:34
A1026QJYJTVE5T	B003KYK18C	4	2014-04-13 06:05:39
A1026QJYJTVE5T	B00A9NE84C	3	2014-04-15 09:34:37
A1026QJYJTVE5T	B003MYU66K	2	2014-04-17 13:54:48
A1026QJYJTVE5T	B003UYU16G	1	2014-04-17 11:44:55
A1026QJYJTVE5T	B003UYU16G	2	2014-04-17 11:45:50
A1026QJYJTVE5T	B000GAYQTU	5	2014-04-17 11:46:52
A1026QJYJTVE5T	B00EG0C20G	6	2014-04-17 11:47:54
A1026RERIHUK3C	B003UYU16G	1	2014-04-20 10:02:53
A1026RERIHUK3C	B008PF1YPW	2	2014-04-21 12:07:15
A1026RERIHUK3C	B00EG0C20G	6	2014-04-22 17:10:28
A1026RERIHUK3C	B000GAYQTU	5	2014-04-23 10:06:37
A0130ZI3HIT9N5V	B008PF1YPW	2	2014-04-25 10:06:37
A0130ZI3HIT9N5V	B00EG0C20G	6	2014-04-26 10:06:37
A0130ZI3HIT9N5V	B000SAUVC4	7	2014-04-26 11:07:38
A31ZC98HM9C4LP	?	?	?

Table 3.9: Historical E-commerce purchase data

Step 1: Read the first line of record from historical purchase data (historical.txt in our case) and store userid, timestamp into a temporary variable. For example, let's store first line from **Table 3.9** into variable as:

Tuserid= A2HDEMZR8QLN, Ttimestamp=2014-04-04 11:30:11.

Step 2: Read another line from the historical database and check recently read userid with userid stored in a temporary variable (Tuserid). If userid is same, compute the difference between the last time the same user made a purchase and the current purchase time user is making a purchase and goto step 3 else goto step 4.

Step 3:

1. If the time difference between the two products is less than 24 hours add itemID to itemset in daily.txt file. In our case, the purchased time difference between two products {1**,** 2} purchased by user {Tuserid= A2HD75EMZR8QLN} is less than 24 hrs. So, add two items to itemset in daily.txt

   1, 2

2. If the time difference between purchased items is more than 24 hours add –I to indicate the end of itemset and add itemID after -I. For example,

   1, 2 **–I** 3

Step 4: If user identity is not similar, then add –I and -S after item to indicate the end of itemset and sequence and goto step 2 by updating temporary variable.

Step 5: Repeat step2, Step 3 and Step 4 until the historical database is empty. In our case, the daily sequential database using step2, Step 3 and Step 4 is shown in **Table 3.10**.

SID	UID	Purchase sequence
1	A2HD75EMZR8QLN	1, 2 **-I** 3 **-I** 6 **-I** 7 **-I** 5 **–I -S**
2	A1026QJYJTVE5T	1, 4 **–I** 3 **-I** 2 **-I** 1, 2, 5, 6 **–I -S**
3	A1026RERIHUK3C	1 **–I** 2 **–I** 6 **-I** 5 **–I -S**
4	A0130ZI3HIT9N5V	2 **–I** 6, 7 **–I -S**

Table 3.10:Sequential database created from historical transactional data

Which is alternatively represented as shown in **Table 3.11**, where angular bracket < > indicates sequence and ( ) contains item set purchased on same day.

SID	SID	Purchase sequence
1	A2HD75EMZR8QLN	< (1, 2), (3), (6), (7), (5)>
2	A1026QJYJTVE5T	<(1,4), (3), (2), (1, 2, 5, 6)>
3	A1026RERIHUK3C	<(1), (2), (6), (5)>
4	A0130ZI3HIT9N5V	<(2) , (6, 7)>

Table 3.11: Alternative representation of daily purchase sequential database

3.2.2 HSPRec: Sequential Pattern Rule (SPR) Module

Sequential Pattern Rule (SPR) is based on the use of frequent sequential pattern created from the periodic sequential database. Thus, input of SPR is periodic historical sequential (click or purchase) database and output is recommended rule using the following major steps:

1. **Frequent sequence generation**: It generates frequent sequences from sequential database by using GSP algorithm present in section **2.2.1.** Let us consider **input**= **Table 3.5**, **minimum support**=2, **candidate set (C_1)** = {1, 2, 3, 4, 5, 6, 7} and **algorithm**= **GSP** (as defined in section **2.2.1**). **Output:** frequent sequences, here, we are including some of frequent sequences.

   1-frequent purchase sequences = {< (1)>, < (2)>, < (3)>, < (5)>, < (6)>}

   Some of 2-frequent purchase sequences= {< (1), (2)>, < (1), (3)>, < (3), (6)>, < (5), (6)>}

   Some of 3-frequent purchase sequences= {< (1), (2), (6)>, < (3), (1), (5)>, < (1), (5), (6)>}

2. **Rule generation**: Represents frequent sequences in the form of $U_{click} \rightarrow U_{purchase}$, where the left-hand side of rule refers to a set of clicked items, while the right-hand side refers to a set of recommended items for purchase. The sequential rule for recommendation is inspired by work done by **pitman & Zankar, 2010** using sequential pattern. Furthermore, to verify the validity of rule, confidence of rule is defined as in **Equation 3.1**.

$$confidence(U_{click} \rightarrow U_{purchase}) = \frac{Support(U_{click} \cup U_{purchase})}{Support(U_{click})}$$

Equation 3.1: Sequential Pattern Rule generated from n-frequent sequences

   Here some of the rules from frequent click sequences are:

   (a) (1,5)→ (3), (1) with 50% confidence

   (b) (1), (5)→ (6), (5) with 50% confidence

3. **Rule selection:** Let's say, we are only interested in rule that satisfy following criteria:

   1) At least 2 antecedents

   2) Confidence >= 50%

   3) Select one rule having highest confidence value.

   In our case, rule (a) (1,5) → (1),(3) is selected for User 5, which states that, user is recommended with item 1 and item 3, when user purchased item 1 and item 5 together.

3.2.3 HSPRec: Click Purchase Similarity (CPS) Module

To compute the CPS similarity between click sequence and purchase sequence of each user, we have used sequence similarity and frequency similarity of the two sequences.

Sequence similarity (LCSR): It is based on using longest common subsequence rate (**LCSR**) (X, Y) = $\frac{LCS(X,Y)}{max(|X|,|Y|)}$. In our case, **X** represents click sequence and Y represents purchase sequence and LCS is defined in **Equation 3.2**.

$$\mathbf{LCS(X_i, Y_j)} = \begin{cases} \emptyset & if\ i = 0\ or\ j = 0 \\ LCS(X_{i-1}, Y_{j-1}) \cap X_i & if\ x_i = y_i \\ longest\left(LCS(X_i, Y_{j-1}), LCS(X_{i-1}, Y_j)\right) & if\ x_i \neq y_i \end{cases}$$

Equation 3.2: Sequence similarity function

In our case, **X** represents click sequence and Y represents purchase sequence

Frequency similarity (FS): First, form the distinct set of items from both click and purchase sequential patterns and count number of items occurring in each sequence to form vector specifying the number of times a user clicks or purchased a particular item then apply **Equation 3.3** to click and purchase vectors.

$$Cosine(X,Y) = \frac{X_1 * Y_1 + X_2 * Y_2 + \ldots + X_n * Y_n}{\sqrt{X_1{}^2 + X_2{}^2 + \cdots + X_n{}^2} * \sqrt{Y_1{}^2 + Y_2{}^2 + \cdots + Y_n{}^2}}$$

Equation 3.3: Cosine similarity function

Thus, **CPS(X, Y)** =α*LCSR(X, Y) +β*Cosine (X,Y), where α+β=1, 0<α, β<1, where α and β are weight to balance the two sequence similarity and frequency similarity.

Example of CPS (click sequence, purchase sequence)

To compute CPS similarity between click sequence (X) = < (2, 7), (6, 6, 7)> and purchase sequence (Y) =< (2), (6, 7)>**,** we have to follow following steps:

1. Compute the longest common subsequences, LCS(X, Y) between click and purchase sequence. For example, LCS (< (2, 7), (6, 6, 7)>, < (2), (6, 7)>) is 3 because of common subsequence (2), (6, 7).
2. Find the maximum number of item occurring in click or purchase sequence as Max(X,Y). In our case, Max(X, Y) is 5.
3. Compute sequences similarity of click (X) and purchase (Y) sequence as LCS(X,Y)/Max(X,Y)=3/5=0.6.

4. Compute the frequencies of items in click and purchase sequences. In our case, we have format [(item): number of occurrences]. So, frequency count of click is: [(2):1, (6):2, (7):2]. Similarly, frequency count of purchase is: [(2):1, (6):1, (7):1].
5. Then, use the Cosine similarity function in **Equation 3.3** to get the frequency similarity between click sequence (X) and purchase sequence (Y) as Cosine(X, Y). In our case, Cosine(X, Y) =0.96.
6. The Click Purchase Similarity of user click and purchase sequence CPS(X,Y)=0.8*0.6+0.2*0.96, where α =0.8 and β=0.2.

This CPS(X, Y) can be used as weight or probability that user u will purchase the entire sequence as shown in **Table 3.12.**

3.2.4 HSPRec: Weighted Frequent Purchase Pattern Miner (WFPP) Module

Weighted Frequent Purchase Pattern Miner (WFPPM) takes weighted purchase sequences as input (present in **Table 3.12**) and generate frequent items with weight(u's rating of item i in the matrices referred to as r_{ui}) present in purchased patterns under the user specified minimum threshold as output. So major steps of WFPPM are:

Purchase sequence	CPS
< (1, 2), (3), (6), (7), (5)>	0.624
<(1, 4),(3), (2), (1, 2, 5, 6)>	0.834
<(1), (2), (6), (5)>	0.636
<(2) , (6, 7)>	0.67
<(1) , (3)>	0.5

Table 3.12: Weighted purchase pattern

1. Count support of item: Count the occurrence of items presented in weighted purchase pattern (Table 3.17). For example, {support (1): 5, support (2): 5, support (3): 3, support (4): 1, support (5): 3, support (6): 4, support (7): 2}
2. Calculate the weight of individual item: Compute weight of individual item from weighted purchase pattern (**Table 3.12**) using **Equation 3.4**.

$$R_{item\ i} = \frac{\sum_{i=1}^{n} CPS \in item_i}{support(item_i)}$$

Equation 3.4: Formula to compute weight in WFPPM

For example, $R_{item\ 1} = \frac{0.624+0.834+0.834+0.636+0.5}{5} = 0.68$.

3. Test weight with minimum support threshold**:** Define the minimum threshold rating, here in our case, minimum threshold=0.2. So, all items are frequent.

3.2.5 HSPRec: User-item Matrix Normalization

Normalization in the recommendation system helps to predict the level of interest of user on an item. Thus, the normalization function takes the user-item frequency matrix as input and provide the level of user interest between 0 and 1 using the unit vector formula (**Equation 3.5**).

$$Normalization(r_{ui}) = \frac{r_{ui}}{\sqrt{{r_{ui_1}}^2 + {r_{ui_2}}^2 + \cdots {r_{ui_n}}^2}}$$

Equation 3.5: Unit normalization function

3.3 Architecture of Proposed System

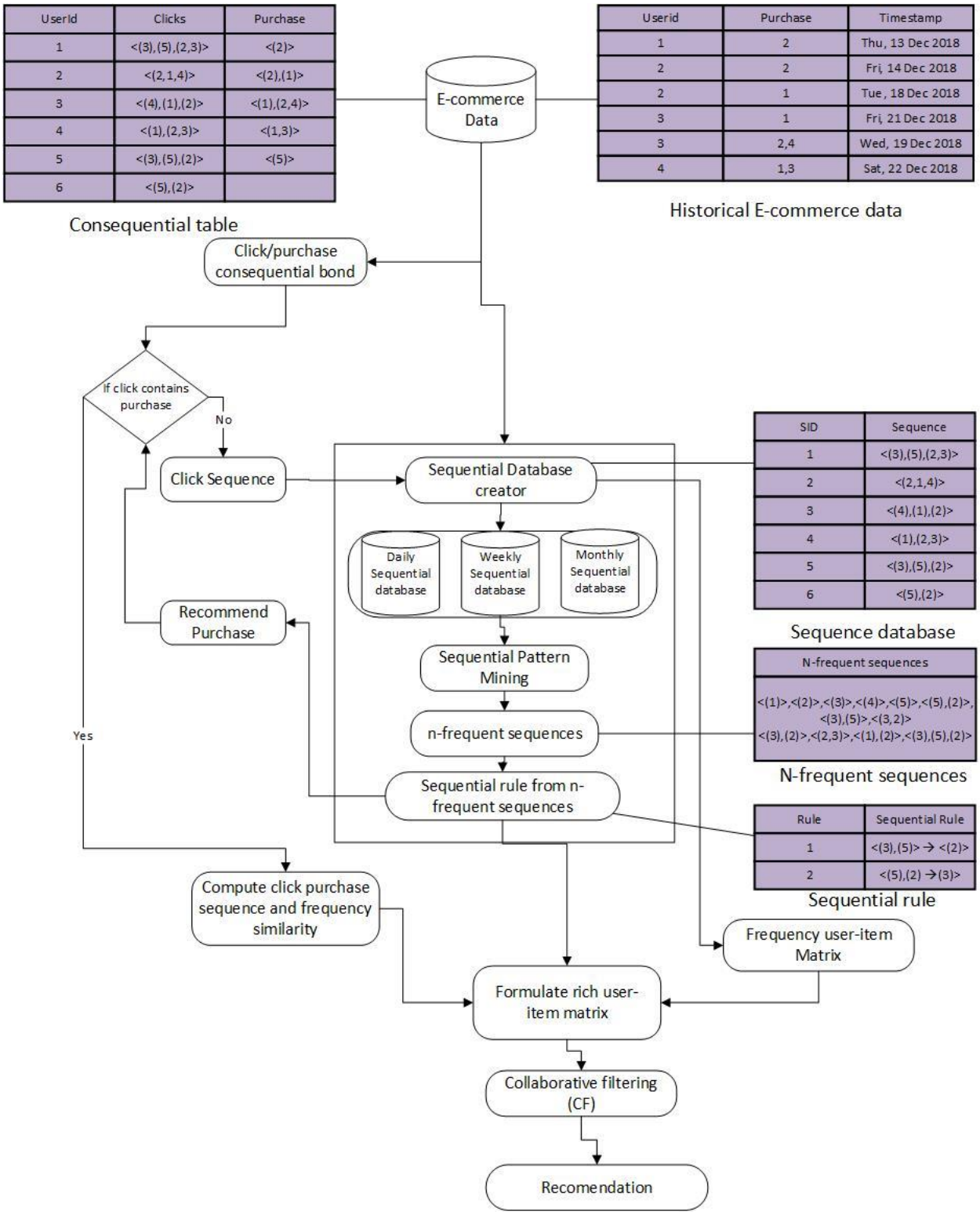


Figure 3.1: Architecture of HSPRec showing modules and flow

3.4 Example of HPCRec VS HSPRec system

Input historical click: Let us consider the historical click table as shown in **Table 3.13**, which contains click items, click start time and click end time.

Userid	Click items	Clickstart	Clickend
User 1	Cheese, Butter , Milk, Butter, Cream, Cheese	2017.06.05.13.23.30	2017.06.05.13.43.00
User 1	Honey, Cream, Butter	2017.06.06.09.00.34	2017.06.06.09.50.20
User 2	Cheese, Honey, Bread, Milk, Cream	2017.06.05.18.53.19	2017.06.05.19.33.14
User 2	Milk , Cheese, Cheese, Milk	2017.06.06.19.53.19	2017.06.06.20.33.13
User 3	Cheese, Cream, Honey, Butter	2017.06.05.19.33.14	2017.06.05.19.50.16
User 4	Cheese, Milk	2017.06.05.19.33.14	2017.06.05.19.53.19

Table 3.13: Historical Click data

Input historical purchase: Let us consider the historical purchase table as shown in **Table 3.14**, which contains a list of items purchased by a user over the specified time.

Userid	Purchase items	timestamp
User1	Cream, Butter, Milk	2017.06.05.13.38.00
User1	Honey, Butter	2017.06.06.09.40.20
User2	Milk, Cream, Honey	2017.06.05.19.23.14
User2	Milk, Honey, Cheese	2017.06.06.20.23.13
User3	Butter, Cheese	2017.06.05.19.40.16
User 3	Cheese, Honey	2017.06.06.10.40.16
User4	?	2017.06.05.19.43.19

Table 3.14: Historical purchase data

Consequential bond: Let us consider the consequential bond of clicks and purchases, which is created from using historical click (**Table 3.13**) and historical purchase (**Table 3.14**) as shown in **Table 3.15**.

Userid	Click	Purchase
1	Cheese, Butter , Milk, Butter, Cream, Cheese, Honey, Cream, Butter	Cream, Butter, Milk Honey, Butter
2	Cheese, Honey, Bread, Milk, Cream, Milk , Cheese, Cheese, Milk	Milk, Cream, Honey, Milk, Honey, Cheese
3	Cheese, Cream, Honey, Butter	Butter, Cheese, Cheese, Honey
4	Cheese, Milk	?

Table 3.15: Consequential table from click and purchase historical data

User-item purchase frequency matrix: Let us consider user-item purchase frequency matrix created from historical purchase data as present in **Table 3.16**, where the number indicates, the number of times item purchase by a user. For example, User 1 purchased butter 2 time, Honey 1 time and so on.

User/item	Milk	Bread	Butter	Cream	Cheese	Honey
User 1	1	?	2	1	?	1
User 2	2	?	?	1	1	2
User 3	?	?	1	?	2	1
User 4	?	?	?	?	?	?

Table 3.16: User-item frequency matrix from purchase historical data

3.4.1 Xiao & Ezeife, 2018 (HPCRec18)

Step 1: Normalize the user-item frequency matrix (**Table 3.16**) using the normalization function. Then, we will get normalized user-item frequency matrix as shown in **Table 3.17**.

User/item	Milk	Bread	Butter	Cream	Cheese	Honey
User 1	0.37	?	0.75	0.37	?	0.37
User 2	0.63	?	?	0.31	0.31	0.63
User 3	?	?	0.40	?	0.81	0.40
User 4	?	?	?	?	?	?

Table 3.17: Normalized user-item frequency matrix

Step 2: As we can see, there is no purchase information of user 4. So, select click item without purchases from consequential bond (**Table 3.15**) and compute similarity with other click using Clickstream Sequence Similarity Measurement (CSSM) function defined by **Xiao & Ezeife, 2018** to fill the information. For example, let's take click X= **{Cheese, Milk}** performed by user4 and Y= **{Cheese, Butter, Milk, Butter, Cream, Cheese, Honey, Cream, Butter}** by user 1.

1. Calculate LCSR(X,Y)= $\frac{\text{common(X,Y)}}{\text{max(X,Y)}} = \frac{2}{9} = 0.22$
2. Calculate FS(X, Y) = $cosine(\{1,1\},\{1,0,2,2,1,3\})$ = 3/11.28=0.26; where X= {Milk:1, Bread:0, Cream:0, Cheese:1, Honey:0, Butter:0} and Y={Milk:1, Bread:0, Cream:2, Cheese:2, Honey:1, Butter:3 } are frequency of product present in X and Y
3. Use α and β as parameters to balance the sub-sequence similarity and frequency similarity, where 0<α, β<1, α+β=1. α and β will be determined from the training dataset. So if set α=0.8, β=0.2, Sim (X, Y) =0.8*0.26+ 0.2*0.22=0.252.
4. Assign calculated similarity weight to purchase item set for a user with whom similarity is computed to create a weighted transactional table and 1, Step 2 and Step 3 for other users. In our case, weighted transactional table is as shown in **Table 3.28**.

Userid	Purchase	Weight
1	{Cream, Butter, Milk, Honey, Butter}	0.252
2	{Milk, Cream, Honey, Milk, Honey, Cheese}	0.36
3	{Butter, Cheese, Cheese, Honey}	0.27
4	?	

Table 3.18: Weighted transactional table

Step 3: Use TWFI function defined by **Xiao & Ezeife, 2018** to calculate weighted frequency for items.

1. Calculate support for item present in weighted transaction table: Form a distinct item set from transactions weighted transactional table and find the support for each item. For example, <Milk:3, Cream:2, Cheese:3, Honey:4, Butter:3>
2. Compute the Average Weighted Support (AWS) using formula: (AWS) = $\frac{sum(weight)}{support}$.

   For example, AWS for Milk is=sum (0.252+0.36+0.36)/3= 0.324. Similarly, AWS (Cream) = 0.306, AWS (Cheese) = 0.3, AWS (Honey)=0.31, AWS (Butter) = 0.258
3. Use minimum weighted threshold to test the Average Weighted Support (AWS). In our example, let us consider minimum weight=0.3 and we can see that all AWS meet minimum threshold.

Step 4: Use the weight of item to fill missing information. In our case, let use for user 4 then user-item frequency matrix looks like as given in **Table 3.19**.

User/item	Milk	Bread	Butter	Cream	Cheese	Honey
User 1	0.37	?	0.75	0.37	?	0.37
User 2	0.63	?	?	0.31	0.31	0.63
User 3	?	?	0.40	?	0.81	0.40
User 4	0.324	?	0.258	0.306	0.3	0.31

Table 3.19: Quantitatively rich normalized user-item frequency matrix

3.4.2 Example of purposed HSPRec

Step 1: Create a user-item frequency matrix from historical purchase. In our case, the user-item frequency matrix created from historical purchase (Table 3.14) is present in Table 3.21.

User/item	Milk	Bread	Butter	Cream	Cheese	Honey
User 1	1	?	2	1	?	1
User 2	2	?	?	1	1	2
User 3	?	?	1	?	2	1
User 4	?	?	?	?	?	?

Table 3.20: User-item frequency matrix created from historical purchase

Step 2: Convert historical purchase to the sequential database using section 3.2.1. The sequential database can be constructed by considering the period of time (day, week, and month). Here in our case, let's construct purchase sequential database from historical purchase information as present in Table 3.20.

SID	Purchase sequence
1	< (Cream, Butter, Milk),(Honey, Butter)>
2	<(Milk, Cream, Honey),(Milk, Honey, Cheese)>
3	<(Butter, Cheese), (Cheese, Honey)>
4	?

Table 3.21: Daily purchase sequential database created from historical transaction data

Step 3: Create frequent sequential purchase pattern from daily sequential database using GSP algorithm. In our case possible purchase sequential rule from frequent purchase sequences are

Rule No	Sequential rule
1	Milk, Butter→ Cheese
2	Cream, Cheese → Milk
3	Cheese, Honey → Cream
4	Honey → Cream
5	Honey → Milk

Table 3.22: Sequential rule created from n-frequent sequences

From rule 3, we can conclude that, user will purchase Honey if user purchased Cheese

Step 4: Fill purchase information in user-item frequency matrix using sequential purchase rule.

User/item	Milk	Bread	Butter	Cream	Cheese	Honey
User 1	1	?	2	1	**1**	1
User 2	2	?	1	1	1	2
User 3	**1**	?	1	**1**	2	1
User 4	?	?	?	?	?	?

Table 3.23: Rich user-item frequency matrix created with help of sequential rule

Step 5: As we can see in **Table 3.23** that there is no purchase information of user 4. To find purchase information of user 4, we have to analyze the relationship between click and purchase. Furthermore, the sequence of click and purchase play important role in product selection. So, rather than analyzing click pattern we have to find the relationship of click and purchase pattern considering their sequence using the following steps:

1. Form click sequential database from the consequential bond. Here, we are creating a daily sequential database but it is also possible to create a weekly and monthly sequential database to create more complex click sequential rule.

SID	Click
1	<(Cheese, Butter , Milk, Butter, Cream, Cheese), (Honey, Cream, Butter)>
2	<(Cheese, Honey, Bread, Milk, Cream), (Milk , Cheese, Cheese, Milk)>
3	<(Cheese, Cream, Honey, Butter)>
4	<(Cheese, Milk)>

Table 3.24: Sequential database created from consequential table

2. Use sequential pattern mining algorithm on user click sequence: Create n-frequent click sequential pattern from click sequential database using the GSP algorithm. In our case some of the n-frequent click sequences are:

   1- Sequences = {< (Milk)>, < (Cheese)>, < (Cream)>, < (Butter)>, < (Honey)>}

   2- Sequences = {< (Milk, Cheese)>, < (Butter, Cheese)>, < (Honey, Butter)>}

   3- Sequences = {< (Cheese, Cream, Milk)>, < (Cream, Cheese, Milk)>}

3. Create sequential rule from n-frequent click sequential pattern using Sequential Pattern Rule (SPR) present in section **3.2.2**. Here in our case possible sequential rule from n-frequent sequences are from click sequences are

Rule No	Sequential rule
1	Cheese, Milk→ Cream
2	Cream, → Cheese
3	Butter → Honey

Table 3.25: Sequential rule created from n-frequent sequences

4. Recommend item from the click sequential rule, where the user clicks but does not purchase anything. For example, there is no purchase for click sequence < (Cheese, Milk)> thus item < (Cream)> is recommended from the sequential rule

Userid	Click	Purchase	Recommend item
1	<(Cheese, Butter , Milk, Butter, Cream, Cheese), (Honey, Cream, Butter)>	<(Cream, Butter, Milk), (Honey, Butter)>	
2	<(Cheese, Honey, Bread, Milk, Cream), (Milk , Cheese, Cheese, Milk)>	<(Milk, Cream, Honey), (Milk, Honey, Cheese)>	
3	<(Cheese, Cream, Honey, Butter)>	<(Butter, Cheese), <(Cheese, Honey)>	
4	<(Butter, Bread, Cream, Cheese, Honey, Butter)>	?	< (Cream)>

Table 3.26: Recommend item for click when purchase is not happened

Step 6: Compute Click Purchase Pattern (CPS) similarity using frequency and sequence of click and purchase pattern using section 3.2.3. If there is no purchase along with click item, then use the recommended item. For example, let's take click (X) = **{< (Cheese, Butter, Milk, Butter, Cream, Cheese)>, < (Honey, Cream, Butter)>}** by user 1 and purchase (Y) = {< **(Cream, Butter, Milk), (Honey, Butter)**>}.

i. Calculate LCSR(X,Y)= $\frac{|\text{common}(X,Y)|}{\max(|X|,|Y|)} = \frac{5}{9} = 0.55$

ii. Calculate FS(X, Y) = $cosine(\{2,1,1,1\},\{1,0,2,2,1,3\})$ = 10/10.21=0.97; where X= {Milk:1, Bread:0, Cream:2, Cheese:2, Honey:1, Butter:3} and Y={Milk:1, Bread:0, Cream:1, Cheese:0, Honey:1, Butter:2 } are frequency of product present in X and Y

iii. Use α and β as parameters to balance the sub sequence similarity and frequency similarity, where 0<α, β<1, α+β=1. α and β will be determined from training dataset. So if set α=0.8, β=0.2, CPS-Sim (X, Y) =0.8*0.55+ 0.2*0.97=0.634.

Userid	Click	Purchase	Recommend item	CPS Similarity
1	<(Cheese, Butter , Milk, Butter, Cream, Cheese), (Honey, Cream, Butter)>	<(Cream, Butter, Milk), (Honey, Butter)>		0.634
2	<(Cheese, Honey, Bread, Milk, Cream), (Milk , Cheese, Cheese, Milk)>	<(Milk, Cream, Honey), (Milk, Honey, Cheese)>		0.516
3	<(Cheese, Cream, Honey, Butter)>	<(Butter, Cheese), <(Cheese, Honey)		0.562
4	<(Butter, Bread, Cream, Cheese, Honey, Butter)>	?	< (Cream)>	0.198

Table 3.27: CPS similarity using click and purchase

Step 7: Assign Click Purchase (CPS) similarity value to the purchase patterns present in the consequential bond. The weighted purchase pattern in our case is present in **Table 3.28**.

Purchase	CPS Similarity
<(Cream, Butter, Milk), (Honey, Butter)>	0.634
<(Milk, Cream, Honey), (Milk, Honey, Cheese)>	0.516
<(Butter, Cheese), <(Cheese, Honey)	0.562
< (Cream)>	0.198

Table 3.28: Weighted purchase patterns

Step 8: Assign weighted purchase patterns to Weighted Frequent Purchase Pattern Miner (WFPP) module present in section **3.2.4** and compute a weight for item present in weighted purchase pattern using formula: $R_{itemi} = \frac{\sum_{i=1}^{n} CPS\ containing\ item_i}{Support\ (item_i)}$

i. Count support of item:

Item	Milk	Cream	Cheese	Honey	Butter
Support	3	3	3	4	3

Table 3.29: Support for item present in weighted purchase patterns

ii. Calculate rating for individual item:

$R_{milk} = \frac{0.634+0.516+0.516}{3}$ =0.55

$R_{Cream} = \frac{0.634+0.516+0.198}{3}$ =0.44

$R_{Cheese} = \frac{0.516+0.562+0.562}{3}$ =0.54

$R_{Honey} = \frac{0.634+0.516+0.516+0.198}{4}$ =0.46

$R_{Butter} = \frac{0.634+0.634+0.562}{3}$ =0.61

Step 9: Use the weight of item to make user-item matrix rich. In our case, rich user-item purchase frequency matrix is shown in **Table 3.30**.

User/item	Milk	Bread	Butter	Cream	Cheese	Honey
User 1	1	?	2	1	**1**	1
User 2	2	?	1	1	1	2
User 3	**1**	?	1	**1**	2	1
User 4	0.55	?	0.61	0.44	0.54	0.46

Table 3.30: Rich user-item purchase frequency matrix

Step 10: Normalize rich user-item purchase frequency matrix

User/item	Milk	Bread	Butter	Cream	Cheese	Honey
User 1	0.35	?	0.70	0.35	**0.35**	0.35
User 2	0.60	?	0.30	0.30	0.30	0.60
User 3	**0.35**	?	0.35	**0.35**	0.70	0.35
User 4	0.48	?	0.53	0.38	0.47	0.40

Table 3.31: Quantitatively rich purchase user-item purchase frequency matrix

CHAPTER 4: EXPERIMENTAL EVALUATION AND ANALYSIS

We have used user-based collaborative filtering to evaluate the performance of recommendation systems. The historical data is converted into user-item matrices with (Choi12Rec, HPCRec18, and HSPRec) algorithms before applying collaborative filtering. We have used the Pearson Correlation Coefficient (PCC) to test user-based collaborative filtering, Furthermore, 80% of data is used in training and 20% of data is used in testing the performance. To evaluate the performance of the recommendation system, we have used a different number of users and nearest neighbors using three different evaluation parameters (a) mean absolute error (MAE) (b) precision and (c) recall with LibRec (**Guo, Zhang, Sun, & Yorke-Smith, 2015**) library available in Java.

4.1 Historical Purchase Dataset Selection

For historical purchase E-commerce data, we have used data available from Amazon (http://jmcauley.ucsd.edu/data/amazon/). The Amazon data sets consist of 23 different categories such as **Books, Electronics, Home and Kitchen, Sports and Outdoors, Cell Phones and Accessories, Grocery and Gourmet Food and many more.** The Data contains 142.8 million transactional records spanning May 1996 - July 2014. The fragment of historical purchase Amazon dataset is provided in **Figure 4.1**.

Data format: {**userID, asin, overall, purchaseTime**}

{"userID": "A2HD75EMZR8QLN", "asin": "0700099867", "overall": 1.0, "purchaseTime": "07 9, 2012" }
{"userID": "A3UR8NLLY1ZHCX", "asin": "0700099867","overall": 4.0, "purchaseTime": "06 30, 2013"}.

Figure 4.1: Historical purchase data (Amazon data)

Where, users are identified by userID and products are identified by asin and user provided rating on an item is represented by overall. Furthermore, purchaseTime provides timestamp when purchased occurred.

4.2 Dataset Evaluations

We used our historical dataset in user-based collaborative filtering to evaluate its performance with respect to MAE, precision, and recall. The data is modified into the intermediate form, which means when the value is larger than the minimum threshold; this value would be set to one (highest rating). When the value is less than the threshold, this value would be set to zero (lowest rating) and finally, user-item rating matrix is provided to collaborative filtering using Librec.

4.2.1 Evaluation parameters

Mean absolute error (MAE): MAE measures the average of the errors in a set of predictions. It's the average over the test sample of the absolute differences between prediction and actual rating.

$$MAE = \frac{\sum_{i=1}^{n} |actual\_rating - predicted\_rating|}{n}$$

Thus, higher mean absolute errors mean, less efficient for accurate rating predication and lower mean absolute errors means highly efficient for accurate rating prediction. For example, let us take an example of rating and computation of mean absolute error.

Item	Actual rating	Predicted rating	Absolute rating error
Item 7	2	4.9	2.9
Item 5	5	4.5	0.5
Item 10	4	4.3	0.3
Item 2	2	3.6	1.6
Item 2	3	3.4	0.4
Item 1	4	2.3	1.7
MAE			7.4/6=1.23

Table 4.1: Actual rating and predicted rating user-item matrix

In this case, mean absolute error is 1.23 and we can see the variation in actual rating and predicted rating. The fragment of code to implement the MAE in java for our experiment is:

```
public double evaluateMAE(User_item testMatrix, RecommendedList recommendedList)
{
 double mae = 0.0,testSize = 0;
 Iterator<MatrixEntry> testMatrixIter = testMatrix.iterator();
 Iterator<UserItemRatingEntry> recommendedEntryIter = recommendedList.entryIterator();
 while (testMatrixIter.hasNext()) {
 if (recommendedEntryIter.hasNext()) {
 MatrixEntry testMatrixEntry = testMatrixIter.next();
 UserItemRatingEntry userItemRatingEntry = recommendedEntryIter.next();
 if (testMatrixEntry.row() == userItemRatingEntry.getUserIdx()
                        && testMatrixEntry.column() == userItemRatingEntry.getItemIdx()) {
 double realRating = testMatrixEntry.get();
 double predictRating = userItemRatingEntry.getValue();
 mae += Math.abs(realRating - predictRating);
                    testSize++;
                }}}
        return mae / testSize;
}
```

Let us consider the confusion matrix as shown in **Table 4.2**

	Purchased	Not purchased
Recommended (relevant)	**TP** (Recommended and purchased)	**FP** (Recommended and not purchased)
Not recommended (Not relevant)	**TN** (Not recommended and purchased)	**FN** (Not recommended and not purchased)

Table 4.2: Confusion matrix for recommendation system

Precision: Determines the fraction of relevant items retrieved out of all items in the recommendation system. Let us consider, TP represents the fraction of items that user is interested with and FP represents the fraction of items that user is not interested with, then precision is defined as:

$$Precision = \frac{TP}{TP + FP} = \frac{interested\ item}{all\ recommended\ item}$$

Suppose that, our precision at the nearest neighbor (10) in a Top-10 recommendation problem is 40%. This means that 40% of the recommendations we make are relevant to the user. For example, let us consider, we are recommended with item7, item5 and item10 at Top-10 neighbors from user-item matrix (**Table 4.1**) and the user is interested with only item10 then precision=1/(1+2)=0.33. Java implementation of precision in our case is present in **Figure 4.3.**

```
public double evaluatePrecision(User_item testMatrix, RecommendedList recommendedList)
{
 double precision = 0.0, numHits = 0;
 int userNum = testMatrix.numRows();
 for (int userID = 0; userID < userNum; userID++) {
 Set<Integer> testSetByUser = testMatrix.getColumnsSet(userID);
 if (testSetByUser.size() > 0) {
 List<ItemEntry<Integer,Double>> recommendListByUser =

recommendedList.getItemIdxListByUserIdx(userID);
 int topK = this.topN <= recommendListByUser.size() ? this.topN : recommendListByUser.size();
 for (int indexOfItem = 0; indexOfItem < topK; indexOfItem++) {
 int itemID = recommendListByUser.get(indexOfItem).getKey();
 if (testSetByUser.contains(itemID)){
                        numHits++;
                                    }}
                precision += numHits / (this.topN + 0.0);
            }}
        return precision;
}
```

Figure 4.3: Function to compute precision

Recall: Determines the fraction of relevant items retrieved out of all relevant items in the recommendation system. Let us consider, TP represents the fraction of relevant items that user is interested with and FN represents the fraction of relevant items that user is not interested with, then precision is defined as:

$$Recall = \frac{TP}{TP + FN} = \frac{relevant\ recommended\ item}{all\ relevant\ items}$$

Suppose that, we computed recall at the nearest neighbor (10) and found it is 40% in our Top-10 recommendation system. This means that 40% of the total number of the relevant items appear in the top-k results. For example, let us consider we are interested with an actual rating greater or equal to 3.5 in user-item matrix (**Table 4.1**) then relevant items are {item5, item10 and item1} and

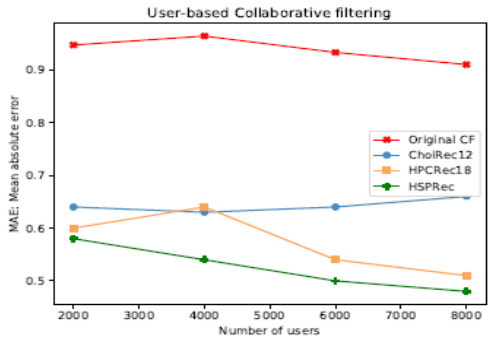
recommended item at Top-10 neighbors are {item7, item5, item10}. Thus, intersection of recommended and relevant items are {item5, item10} =2. Thus, recall=2/2+1= 0.66. Java implementation of recall in our case is present in **Figure 4.4**.

```
public double evaluateRecall(User_item testMatrix, RecommendedList recommendedList)
{
  double totalRecall = 0.0, numHits = 0;
  int userNum = testMatrix.numRows();
  int nonZeroNumUsers = 0;
  for (int userID = 0; userID < userNum; userID++) {
  Set<Integer> testSetByUser = testMatrix.getColumnsSet(userID);
  if (testSetByUser.size() > 0) {
  List<ItemEntry<Integer, Double>> recommendListByUser =
                                    recommendedList.getItemIdxListByUserIdx(userID);
 int topK = this.topN <= recommendListByUser.size() ? this.topN : recommendListByUser.size();
  for (int i = 0; i < topK; i++) {
  int itemID = recommendListByUser.get(i).getKey();
  if (testSetByUser.contains(itemID)) {
                        numHits++;
                    }}
                totalRecall +=  numHits / (testSetByUser.size() + 0.0);
                nonZeroNumUsers++;
            }}
        return nonZeroNumUsers > 0 ? totalRecall / nonZeroNumUsers : 0.0d;
}
```

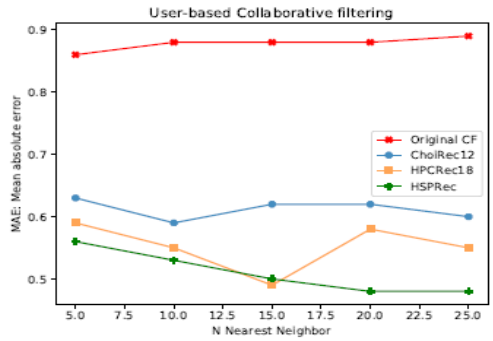
Figure 4.4: Function to compute recall

4.2.2 Result evaluation and analysis

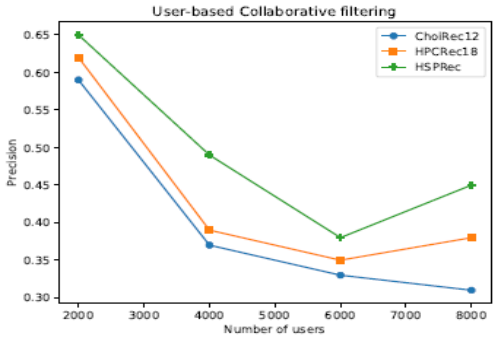
First, we applied user-based collaborative filtering on explicit rating available on Amazon historical data then we saw that performance is very low. Then, we implemented choiRec12 (**Choi, Keunho, Yoo, Kim, & Suh, 2012**) with derive implicit rating and got better result compared to original collaborative filtering. Furthermore, we implemented HPCRec18 (**Xiao & Ezeife, 2018**) and found a better result than choi12Rec. Finally, we implemented the historical sequential recommendation (HSPRec18), with the help of purchase frequency matrix at first. Then, we discovered frequent sequences of purchase data to create sequential rules and used sequential rules to enhance user-item matrix and applied to collaborative filtering and found better result compared to choiRec12 and HPCRec18.



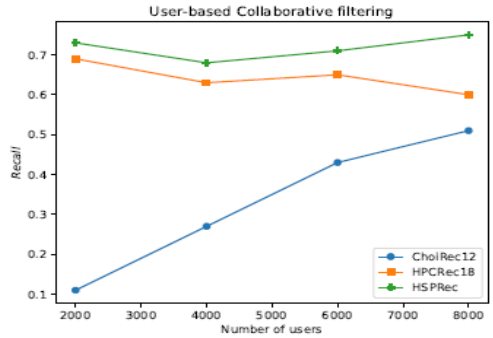
(a) User-based collaborative filtering



(b) User-based collaborative filtering with Top-N



(c) Precision in user-based collaborative filtering



(d) Recall in user-based collaborative filtering

Figure 4.5: Evaluation of HSPRec with respect to precision, recall and mean absolute error

4.2.3 Accuracy evaluation using precision

Recommendation system	Top-N	Neighbors	Number of users	Recommendation No	Precision	Relevant item	Percentage
ChoiRec12	10	10	2000	2090	0.59	1233	58%
			4000	3880	0.37	1435	37%
			6000	5647	0.33	1863	32%
			8000	7772	0.31	2409	30%
HPCRec18	10	10	2000	2050	0.62	1271	62%
			4000	4032	0.39	1572	38%
			6000	5857	0.35	2049	34%
			8000	8655	0.38	3288	37%
HSPRec	10	10	2000	2130	0.65	1394	64%
			4000	4156	0.49	2036	48%
			6000	6039	0.38	2294	37%
			8000	8938	0.45	4022	44%

Table 4.3: Precision evaluation with respect to different number of users

4.3 Complexity Analysis

4.3.1 Time complexity analysis of HSPRec algorithm

Our HSPRec is composed of several modules (SHOD (Sequential Historical Periodic Database), SPR (Sequential Pattern Rule), CPS (Click Purchase Similarity), WFPPM (Weighted Frequent Purchase Pattern Miner), and Matrix normalization); thus, we are going to discuss the time complexity of HSPRec with respect to specified modules.

1. Time complexity analysis of SHOD algorithm

In our case, SHOD algorithm starts with input historical.txt as the main input. But, SHOD algorithm is functional with the input from the relational database such as MySQL, SqlServer, and Oracle. So, time complexity in the worst case is,

0(n) - Time complexity to form historical.txt database

C- Time complexity to update temporary variable

0(n) - Time complexity to form sequential database

Thus, total time complexity in worst case is, 0 (n) +C+0 (n) = 0 (n)

2. Time complexity of Click Purchase Similarity (CPS) module

The CPS module takes the click sequence and purchase sequence of each user as input. Thus, the time complexity required to compute click and purchase similarity for n users is O (n^2).

3. Time complexity of Weighted Frequent Purchase Pattern Miner (WFPPM) module

Weighted purchase pattern miner takes weighted purchase patterns (purchase sequences with assigned weight) as input. Thus, counting the sum of the weight of item and support of the item in each purchase sequence requires O (n^2).

4. Time complexity of Sequential Pattern Rule (SPR) module

SPR module contains the General Sequential Pattern (GSP) mining algorithm. Thus, the time complexity of this module depends on the following factors:

(a) Support threshold: General Sequential Pattern (GSP) mining algorithm is based on the minimum support threshold to generate frequent sequences. Thus, lowering the support threshold often results in the production of more frequent sequences.

(b) Number of transactions: GSP algorithm makes repeated scanning of the dataset. Thus, run time increases with a large number of transactions.

(c) Average transaction width: Each transaction in the dataset contains a different number of items. So, the time complexity depends on the average transaction width.

(d) Number of items: A large number of items require more space to store the support count of items resulting in more time complexity.

4.4 Implementation and Coding

- Operation system: Windows 10 Unlimited
    - RAM: 16 GB
    - CPU: 3.6 GHz
    - System type: 64-bit Operating System, x64 based processor
- Integrated Development Environment:
    - Eclipse Java EE IDE for Web Developers
        - Version: Oxygen.1a Release (4.7.1a)
        - Build id: 20171005-1200
    - PyCharm
        - 2018.3
- Platform:
    - Java SE Development Kit
        - Version: 1.8.0_65
    - Python
        - 3.7.0
- Project manage tool: Apache Maven
    - Version: 3.5.3

CHAPTER 5: CONCLUSION AND FUTURE WORK

Many recommendation system neglect sequential pattern during recommendation. Thus, to verify the necessity of sequential pattern in recommendation, we generated a sequential pattern from historical E-commerce data and feed them into collaborative filtering to make user-item matrix rich from quantity and quality perspective. Furthermore, after evaluation with different systems, we got better result with a sequential pattern based recommendation.

Thus, some of the possible future works are:

(a) Finding more possible way of integrating sequential pattern to collaborative filtering.

(b) Incorporating multiple data sources based sequential pattern with different data schema, and make recommendations based on the overall data set.

(c) Finding the more possible way of integrating sequential pattern in user-item matrix from online data.

REFERENCES

Abdullah, N., Xu, Y., Geva, S., & Chen, J. (2010, December). Infrequent purchased product recommendation making based on user behavior and opinions in E-commerce sites. In *Data Mining Workshops (ICDMW), 2010 IEEE International Conference on* (pp. 1084-1091). IEEE.

Aggarwal, C. C. (2016). An introduction to recommender systems. In *Recommender Systems* (pp. 1-28). Springer.

Agrawal, R., Mehta, M., Shafer, J. C., Srikant, R., Arning, A., & Bollinger, T. (1996, August). The Quest Data Mining System. In *KDD* (Vol. 96, pp. 244-249).

Agrawal.R and R. Srikant (1994) Fast algorithms for mining association rules. In Pro of the 20th Int'l Conf. on Very Large Databases (VLDB '94), Santiago, Chile, June 1994

Agrawal, R., & Srikant, R. (1995, March). Mining sequential patterns. In *Data Engineering, 1995. Proceedings of the Eleventh International Conference on* (pp. 3-14). IEEE.

Altman, N. S. (1992). An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, *46*(3), 175-185.

Apté, C., & Weiss, S. (1997). Data mining with decision trees and decision rules. *Future generation computer systems*, *13*(2-3), 197-210.

Arabie, P., & De Soete, G. (1996). *Clustering and classification*. World Scientific.

Ayres, J., Flannick, J., Gehrke, J., & Yiu, T. (2002, July). Sequential pattern mining using a bitmap representation. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 429-435). ACM.

Balabanović, M., & Shoham, Y. (1997). Fab: content-based, collaborative recommendation. *Communications of the ACM*, *40*(3), 66-72.

Basu, C., Hirsh, H., & Cohen, W. (1998). *Recommendation as classification: Using social and content-based information in recommendation*. Proceedings of the 15th National Conference on Artificial Intelligence, 714 – 720.

Bhatta, R., Ezeife, C. I., Butt, M. (2019). Mining Sequential Patterns of Historical Purchases for E-commerce Recommendation. Submitted to International Conference on Big Data Analytics and Knowledge Discovery.

Bucklin, R. E., & Sismeiro, C. (2009). Click here for Internet insight: Advances in clickstream data analysis in marketing. *Journal of Interactive Marketing*, *23*(1), 35-48.

Bergroth, L., Hakonen, H., & Raita, T. (2000). A survey of longest common subsequence algorithms. In *Proceedings Seventh International Symposium on String Processing and Information Retrieval. SPIRE 2000* (pp. 39-48). IEEE.

Ben-Shimon, D., Tsikinovsky, A., Friedmann, M., Shapira, B., Rokach, L., & Hoerle, J. (2015). Recsys challenge 2015 and the yoochoose dataset. *Proceedings of the 9th ACM Conference on Recommender Systems*, (pp. 357-358).

Cho, Y. H., & Kim, J. K. (2004). *Application of Web usage mining and product taxonomy to collaborative recommendations in e-commerce*. Expert systems with Applications, 26(2), 233-246.

Choi, K., Yoo, D., Kim, G., & Suh, Y. (2012). A hybrid online-product recommendation system: Combining implicit rating-based collaborative filtering and sequential pattern analysis. Electronic Commerce Research and Applications, 11(4), 309-317.

Deshpande, M., & Karypis, G. (2004). Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems (TOIS)*, *22*(1), 143-177.

Ezeife, C. I., Lu, Y., & Liu, Y. (2005, August). PLWAP sequential mining: open source code. In *Proceedings of the 1st international workshop on open source data mining: frequent pattern mining implementations* (pp. 26-35). ACM.

Ekstrand, M. D., Riedl, J. T., & Konstan, J. A. (2011). Collaborative filtering recommender systems. *Foundations and Trends in Human–Computer Interaction*, *4*(2), 81-173.

Fayyad, Usama M., Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy. "Advances in knowledge discovery and data mining." (1996).

Guo, G., Zhang, J., Sun, Z., & Yorke-Smith, N. (2015, June). LibRec: A Java Library for Recommender Systems. In *UMAP Workshops* (Vol. 4).

Han, J., Pei, J., & Kamber, M. (2011). *Data mining: concepts and techniques*. Elsevier.

Hartigan, J. A., & Wong, M. A. (1979). Algorithm AS 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, *28*(1), 100-108.

Herlocker, J. L., Konstan, J. A., Terveen, L. G., & Riedl, J. T. (2004). Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, *22*(1), 5-53.

Herlocker, J. L., Konstan, J. A., & Riedl, J. (2000, December). Explaining collaborative filtering recommendations. In *Proceedings of the 2000 ACM conference on Computer supported cooperative work* (pp. 241-250). ACM.

Hoffman, T., & Puzicha, J. (1999). *Latent class models for collaborative filterin*g. Proceedings of the 16th International Joint Conference on Artificial Intelligence, 688 – 693.

Hu, Y., & Panda, B. (2004, March). A data mining approach for database intrusion detection. In *Proceedings of the 2004 ACM symposium on Applied computing* (pp. 711-716). ACM.

Jain, A. K., & Dubes, R. C. (1988). Algorithms for clustering data.

Kim, Y. S., Yum, B. J., Song, J., & Kim, S. M. (2005). Development of a recommender system based on navigational and behavioral patterns of customers in e-commerce sites. *Expert Systems with Applications*, *28*(2), 381-393.

Kim, Y. S., & Yum, B. J. (2011). Recommender system based on click stream data using association rule mining. *Expert Systems with Applications*, *38*(10), 13320-13327

Konstan, J. A., Miller, B. N., Maltz, D., Herlocker, J. L., Gordon, L. R., & Riedl, J. (1997). GroupLens: applying collaborative filtering to Usenet news. *Communications of the ACM*, *40*(3), 77-87.

Koren, Y. (2009, June). Collaborative filtering with temporal dynamics. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 447-456). ACM.

Kosala, R., & Blockeel, H. (2000). Web mining research: A survey. *ACM Sigkdd Explorations Newsletter*, *2*(1), 1-15.

Kumar, N. P., & Fan, Z. (2015). Hybrid user-item based collaborative filtering. *Procedia Computer Science*, *60*, 1453-1461.

Lee, T. Q., Park, Y., & Park, Y. T. (2008). A time-based approach to effective recommender systems using implicit feedback. *Expert systems with applications*, *34*(4), 3055-3062.

Liu, D. R., Lai, C. H., & Lee, W. J. (2009). A hybrid of sequential rules and collaborative filtering for product recommendation. *Information Sciences*, *179*(20), 3505-3519.

Li, Y., Niu, Z., Chen, W., & Zhang, W. (2011, December). Combining collaborative filtering and sequential pattern mining for recommendation in e-learning environment. In International Conference on Web-Based Learning (pp. 305-313). Springer, Berlin, Heidelberg.

Lin, W., Alvarez, S. A., & Ruiz, C. (2000). *Collaborative recommendation via adaptive association rule mining*. Proceedings of the WEBKDD

Linden, G., Smith, B., & York, J. (2003). Amazon. com recommendations: Item-to-item collaborative filtering. IEEE Internet computing, 7(1), 76-80.

Mabroukeh, N. R., & Ezeife, C. I. (2010). A taxonomy of sequential pattern mining algorithms. *ACM Computing Surveys (CSUR)*, *43*(1), 3.

Ma, B. L. W. H. Y., & Liu, B. (1998, August). Integrating classification and association rule mining. In *Proceedings of the fourth international conference on knowledge discovery and data mining*.

Melville, P., Mooney, R. J., & Nagarajan, R. (2002). Content-boosted collaborative filtering for improved recommendations. *Aaai/iaai*, *23*, 187-192.

Montgomery, A. L., Li, S., Srinivasan, K., & Liechty, J. C. (2004). Modeling online browsing and path analysis using clickstream data. *Marketing science*, *23*(4), 579-595.

O'Connor, M., & Herlocker, J. (1999, August). Clustering items for collaborative filtering. In *Proceedings of the ACM SIGIR workshop on recommender systems* (Vol. 128). UC Berkeley.

Pitman, A., & Zanker, M. (2010, December). Insights from applying sequential pattern mining to e-commerce click stream data. In *Data Mining Workshops (ICDMW), 2010 IEEE International Conference on* (pp. 967-975). IEEE.

Pei, J., Han, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., & Hsu, M. C. (2001, April). Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *icccn* (p. 0215). IEEE.

Ricci, F., Rokach, L., & Shapira, B. (2011). Introduction to recommender systems handbook. In *Recommender systems handbook* (pp. 1-35). springer US.

Russell, S., Norvig, P., & Intelligence, A. (1995). A modern approach. *Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs*, *25*(27), 79-80.

Siciliano, R., & Conversano, C. (2005). Decision Tree Inudction. In *Encyclopedia of Data Warehousing and Mining* (pp. 353-358). IGI Global.

Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2000, October). Analysis of recommendation algorithms for e-commerce. In *Proceedings of the 2nd ACM conference on Electronic commerce* (pp. 158-167). ACM.

Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001, April). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web* (pp. 285-295). ACM.

Srikant, R., & Agrawal, R. (1996, March). Mining sequential patterns: Generalizations and performance improvements. In *International Conference on Extending Database Technology* (pp. 1-17). Springer, Berlin, Heidelberg.

Schafer, J. B., Frankowski, D., Herlocker, J., & Sen, S. (2007). Collaborative filtering recommender systems. In *The adaptive web* (pp. 291-324). Springer, Berlin, Heidelberg.

Schafer, J. B., Konstan, J. A., & Riedl, J. (2001). E-commerce recommendation applications. *Data mining and knowledge discovery*, *5*(1-2), 115-153.

Su, Q., & Chen, L. (2015). A method for discovering clusters of e-commerce interest patterns using click-stream data. electronic commerce research and applications, 14(1), 1-13.

Steinbach, M., Karypis, G., & Kumar, V. (2000, August). A comparison of document clustering techniques. In *KDD workshop on text mining* (Vol. 400, No. 1, pp. 525-526).

Xiao, Y., & Ezeife, C. I. (2018, September). E-Commerce Product Recommendation Using Historical Purchases and Clickstream Data. In *International Conference on Big Data Analytics and Knowledge Discovery* (pp. 70-82). Springer, Cham.

Yun, U., & Leggett, J. J. (2006, September). WSpan: Weighted Sequential pattern mining in large sequence databases. In *Intelligent Systems, 2006 3rd International IEEE Conference on* (pp. 512-517). IEEE.

Zaki, M. J. (2001). SPADE: An efficient algorithm for mining frequent sequences. *Machine learning*, *42*(1-2), 31-60.

Zhao, X., Niu, Z., & Chen, W. (2013). Interest before liking: Two-step recommendation approaches. *Knowledge-Based Systems*, *48*, 46-56.

VITA AUCTORIS

NAME	Raj Bhatta
PLACE OF BIRTH	Gorkha, Nepal
YEAR OF BIRTH	1990
EDUCATION	Whitefield Higher Secondary School, Kathmandu, Nepal (2007 - 2009)
	Tribhuvan University, Kathmandu, Nepal (2010 - 2014)
	University of Windsor, Ontario, Canada (January, 2017 – April, 2019)