3-10-2021

Improving E-Commerce Recommendations using High Utility Sequential Patterns of Historical Purchase and Click Stream Data

Komal Virk
University of Windsor

Improving E-Commerce Recommendations using High Utility Sequential Patterns of Historical Purchase and Click Stream Data

By

Komal Virk

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

2021

Improving E-Commerce Recommendations using High Utility Sequential Patterns of Historical Purchase and Click Stream Data

By

Komal Virk

APPROVED BY:

E. H. Kim

Department of Physics

A. Biniaz

School of Computer Science

C. Ezeife, Advisor

School of Computer Science

January 15, 2021

DECLARATION OF ORIGINALITY

I hereby certify that I am the sole author of this thesis. 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

Recommendation systems not only aim to recommend products that suit the taste of consumers but also generate higher revenue and increase customer loyalty for e-commerce companies (such as Amazon, Netflix). Recommendation systems can be improved if user purchase behaviour are used to improve the user-item matrix input to Collaborative Filtering (CF). This matrix is mostly sparse as in real-life, a customer would have bought only very few products from the hundreds of thousands of products in the e-commerce shelf. Thus, existing systems like Kim11Rec, HPCRec18 and HSPRec19 systems use the customer behavior information to improve the accuracy of recommendations. Kim11Rec system used behavior and navigations patterns which were not used earlier. HPCRec18 system used purchase frequency and consequential bond between click and purchased data to improve the user-item frequency matrix. The HSPRec19 system converts historic click and purchase data to sequential data and enhances the user-item frequency matrix with the sequential pattern rules mined from the sequential data for input to the CF. HSPRec19 system generates recommendations based on frequent sequential purchase patterns and does not capture whether the recommended items are also of high utility to the seller (e.g., are more profitable?).

The thesis proposes a system called High Utility Sequential Pattern Recommendation System (HUSRec System), which is an extension to the HSPRec19 system that replaces frequent sequential patterns with use of high utility sequential patterns. The proposed HUSRec generates a high utility sequential database from ACM RecSys Challenge dataset using the HUSDBG (High Utility Sequential Database Generator) and HUSPM (High Utility Sequential Pattern Miner) mines the high utility sequential pattern rules which can yield high sales profits for the seller based on quantity and price of items on daily basis, as they have at least the minimum sequence utility. This improves the accuracy of the recommendations. The proposed HUSRec mines clicks sequential data using PrefixSpan algorithm to give frequent sequential rules to suggest items where no purchase has happened, decreasing the sparsity of user-item matrix, improving the user-item matrix for input to the collaborative filtering. Experimental results with mean absolute error, precision and graphs show that the proposed HUSRec system provides more accurate recommendations and higher revenue than the tested existing systems.

Keywords: Data mining, Sequential pattern mining, Collaborative filtering, High utility pattern mining, E-commerce recommendation systems.

DEDICATION

I would like to dedicate this thesis to my parents, supervisor, internal and external readers and my friends who have helped and supported to complete my graduate study at the University of Windsor.

ACKNOWLEDGEMENTS

My sincere appreciation goes to my parents Mr. Nishan Singh Virk and Mrs. Palwinder Kaur Virk. Your love, faith and words of encouragement gave me the extra energy to see this work through.

I would like to express my sincere gratitude to my advisor Prof. Dr. Christie Ezeife for her continuous support throughout my graduate study. Thank you for keeping your patience with me and investing your valuable time in reading all my thesis updates, feedbacks, and providing me with continuous feedbacks on my work and financial support through Research Assistantship (R.A.) positions supported from her grants from funding agencies such as NSERC.

Besides my advisor, I would like to thank my thesis committee members: Dr. Eugene Kim (my external reader), Dr. Ahmad Biniaz (my internal reader) and Dr. Xiaobu Yuan (thesis defense chair) for accepting to be my thesis committee, despite their tight schedules and their insightful comments and encouragement is highly appreciated.

Finally, I would express my appreciation to all my friends and colleagues at the University of Windsor, for their advice, encouragement, and support throughout the duration of this work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF EQUATIONS

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

Recommendation Systems (RS) 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**)**. Recommender systems (Agrawal, 2016) have evolved into a fundamental tool for helping users make informed decisions and choices, especially in the era of big data in which customers must make choices from many products and services. A lot of RS models and techniques have been proposed and most of them have achieved great success. Among them, the content-based RS and collaborative filtering RS (Melville, Mooney & Nagarajan, 2002) are two major ones. Recommendation systems use data mining technologies such as classification, clustering, association rule mining, frequent pattern mining, and sequential pattern mining (Han, Pei & Kamber, 2011) to generate a meaningful representation of user purchase data.

Traditionally, collaborative filtering is the most used technique in recommendation systems and it has as its input, user item rating matrix containing the explicit ratings given to the items (products) by the users. Most of the users do not give ratings to the products because there are too many products and each user may have interacted with only a very small percentage of products. So, the user-item rating matrix becomes sparse. To resolve this issue, (Choi, Yoo, Kim, & Suh, 2012**)** used the other available information about the purchase and customer behavior (such as basket placement, clicks etc). The clickstream analysis techniques like KimRec05 (Kim, Yum, Song, & Kim, 2005) and LiuRec09 (Liu, Lai &Lee, 2009) proved that these behaviors are dynamic in nature and purchase of items could be different in each purchase. The introduction explains importance of recommendation systems, mostly used recommendation techniques (Content based and Collaborative filtering), data mining and its techniques to gather useful information from the available e-commerce datasets.

1.1 Why we need E-commerce Recommender System

In addition to the common goals of recommendation systems, (Schafer, Frankowski, Herlocker & Sen, 2007) gave three important goals for recommendation systems in E-commerce as follows:

1. *Increase the sale of the products*: This is probably the most important reason for popularity of commercial RS, i.e., to be able to sell an additional set of items compared to those usually sold without any kind of recommendation. The increase in the sales can be achieved if the recommended items are likely to suit the user's needs and wants.
2. *Sell more diverse items:* Another major role of a RS is to enable the user to select items that might be hard to find without a precise recommendation. For instance, in a movie RS such as Netflix, the service provider is interested in renting all the movies in the catalogue, not just the most popular ones.
3. *Increase the loyalty between user and customer*: A well designed RS can also improve the experience of the user with the site or the application. The user will find the recommendations interesting, relevant and, with a properly designed human-computer interaction, they will also enjoy using the system.

1.2 Types of Recommendation systems (RS)

1.2.1 Content Based Filtering (CBF)

Content Based Filtering is a domain-dependent algorithm and it emphasizes more on the analysis of the attributes of items to generate predictions. In this technique, 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 similarity techniques to generate meaningful recommendations among items to recommend (Pazzani, Muramatsu & Billsus, 1996). It could use similarity techniques such as Vector Space Model such as Term Frequency Inverse Document Frequency (TF-IDF) for information retrieval (Musto, 2010) to model the relationship between different documents within a corpus. If the user profile changes, CBF technique still has the potential to adjust its recommendations within a very short period. The major disadvantage of this technique is the need to have an in-depth knowledge and description of the features of the items in the profile.

1.2.2 Collaborative Filtering (CF)

Collaborative filtering is 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. The similarity in taste of two users is calculated based on the similarity in the rating history of the users. This is the reason why refers to collaborative filtering as "people-to-people correlation." Collaborative filtering systems are usually categorized into two subgroups: **memory-based** and **model-based** methods (Agrawal, 2016), (Ekstrand, Riedl & Konstan ,2011).

Memory-based collaborative filtering utilizes the entire user-item data to generate predictions. The system uses statistical methods to search for a set of users who have similar transactions history to the active user. This method is also called nearest-neighbor or user-based collaborative filtering (Sarwar, Karypsis, Konstan & Riedl, 2001).

Model-based collaborative filtering provides recommendations by developing a model from user ratings (Sarwar, Karypsis, Konstan & Riedl, 2001). In addition to using explicit data such as ratings, collaborative filtering can also use implicit information by observing the habits of users, such as music played, applications downloaded, websites visited, or books read (Bobadilla, Ortega, Hernando, & Gutiérrez, 2013).

The **user-item rating matrix** (e.g. Table 1.1) in E-commerce usually contains part information of the historical transaction records. In Table 1.1, each row records a purchase (a collection of item names) happened in a session, and there may be multiple products for each purchase.

Problem 1.2.1: Given a user- item matrix as in Table 1.1, which has ratings between 1- 5 given by each user for items they have interacted with, while for other items their ratings are unknown. The task here is to predict the unknown ratings for user "7" on item "e" for purposes of recommendation using Collaborative filtering algorithm.

Solution 1 for Problem 1.2.1:

The solution for problem 1.2.1 using user-based collaborative filtering algorithm is as follows:

userId/Products	a	b	c	d	e
1	2		3		5
3	3	1	5		
4	1			3	4
7	2	4	1	1	?

Table 1.1 User-item matrix for rating.

Input: A user-item rating matrix (Table 1.1) where ratings are specified in the range from 1-5 such as {1, 2, 3, 4, 5} which indicates users likeliness for the items {a, b, c, d, e}.

Output: Predictions of User "7" ratings on item "e".

Step 1: Compute the mean ratings ru ($r1$, $r2$, . . . , rm) for each user where rui is the observed rating of user u for item i, only consider the specified ratings.

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

Equation 1. 1 Equation to compute mean rating

The average ratings for user 1, 3, 4 and 7 are 10/3, 3, 8/3, 2 respectively.

Step 2: Compute similarity between the user "7" and other users using Cosine (u, v) which is known as Cosine similarity.

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

Equation 1. 2 Formula to Compute Cosine similarity

For example, the cosine similarity between user "1"and user "7" is (1, 7) = $\frac{2\times2+3\times1}{\sqrt{2^2+3^2}\times\sqrt{2^2+1^2}} = 0.868$ following Equation 1.2. Similarly, sim (3, 7) = 0.553, sim (4, 7) = 0.707; the correlation similarity between user "1" and "7" is sim (1, 7) = $\frac{(2-\frac{10}{3})\times(2-2)+(3-\frac{10}{3})\times(1-2)}{\sqrt{(2-\frac{10}{3})^2+(3-\frac{10}{3})^2}\times\sqrt{(2-2)^2+(1-2)^2}} = 0.245$. Similarly, sim (3, 7) = −0.878, sim (4, 7) = −0.196.

Step 3: Select the Top –N neighbors of User "7" which has highest similarity to it, in our case where N is set to 2, user "1" and user "4" has the highest similarity to user "7".

Step 4: Compute the raw ratings using the Top-N users (User 1 and User 4). Compute the rating value of rating for user 7 on item e $(r_{7,e})$ using user "1" and user "4" with the cosine similarity formula in Equation 1.2,

$$\frac{sim(1,7)\times(r_{1,e}-\overline{r_1})+sim(4,7)\times(r_{4,e}-\overline{r_4})}{|sim(1,7)|+|sim(4,7)|}=\frac{0.868\times(5-\frac{10}{3})+0.707\times(4-\frac{8}{3})}{0.868+0.707}=1.517$$

Step 5: Compute prediction value using sum of mean rating and rating value for the item.
For example, mean centric rating of User "7" is 2 as calculated in step 1 and value of raw ratings for user 7 on item e $(r_{7,e})$ gives, 2+1.5 = 3.5.

CF recommendation systems have some limitations such as:

(1) **Cold Start problem**: When new items or new users appear in the database, these items are not rated yet by any users, and the users' preferences are unknown;

(2) **Sparsity problem**: When the known rating data (Sarwar, Karypis, Konstan& Riedl, 2000) takes only a very small proportion in the user-item rating matrix.

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

1.2.3 Hybrid filtering

Both CF and CBF have their benefits and demerits; therefore, if we combine both together, then the benefits of both can be used to overcome the demerits of others (Kumar & Fan, 2015). For example, according (Fan, Pan, & Jiang, 2014), 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. To resolve the rating problem, implicit rating system like ChoiRec12 (Choi, Yoo, Kim, & Suh, 2012) derived from user behaviors (for example, purchases, clicks) across E-commerce and clickstream data analysis techniques like KimRec05 (Kim, Yum, Song, & Kim, 2005) and LiuRec09 (Liu, Lai &Lee, 2009) are used. However, users purchase behavior is always dynamic in nature and purchase of items may be different in each purchase (Moe & Fader, 2004). 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. 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) (Mooney & Roy, (1999)). 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) for information retrieval (Musto, 2010), or Probabilistic models such as Naïve Bayes Classifier, Decision Trees or Neural Networks (Xhemali, Hinde & Stone, 2009) extracted from those contents. Hybrid approach allows recommendation with 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 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.

1.3 Need for 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 like 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 user because user choice depend altogether on product quantity, price and overall rating of the purchased product.

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.

1.4 Data Mining

Data mining targets to explore information or patterns hidden in the large data set, but not people finding easily recognizable patterns during their daily life. According to (Chen, Han, & Yu, 1996) " Data mining, which is also referred to as knowledge discovery in databases, means a process of non - trivial extraction of implicit, previously unknown and potentially useful information (such as knowledge rules, constraints, regularities) from data in databases". Some of the unsupervised learning data mining techniques include Clustering, Association Rule Mining and supervised learning data mining technique like Classification.

1.4.1 Clustering

Clustering is a process of grouping several similar objects together (Jain & Dubes, 1998), clustering is unsupervised data mining technique, which does not need to be labeled manually and can automatically divide the data into set or group of clusters of similar objects using techniques such as distance measures to decide the closeness of data objects. The K-means clustering (Hartigan & Wong, 1979) is one of the used clustering approaches in the field of data mining (Steinbach, Karypis, & Kumar, 2000). K-means clustering is used, when we have unlabeled data which cannot be defined into categories or groups. The K-means algorithm works iteratively to assign each data point to one of K groups based on the features that are given.

Problem 1.4.1: Consider the Table 1.2 as Input data with Height and Weight, the two important attributes. Using the K-means algorithm for clustering, we aim to find the possible clusters using the Table 1.2.

Height (cm)	**Weight (kg)**
185	72
170	56
168	60
179	68
182	72
188	77

Table 1.2 Input data to clustering algorithm

Solution for problem 1.4.1: The K-means clustering algorithm consists of five major steps:

Input: a set of objects O = {I1, I2, I3, … In}and each object has n-dimensional attributes O_i such as Height and Weight, 1<= i <=n.

Output: subsets of objects such as [{O1, O4}; {O2, O6, O3} …].

Step 1: Randomly pick centroid from available objects. 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 1.3.** and name then H1 and W1.

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

Table 1.3 Maximum and minimum cluster centroids

Step 2: Calculate the distance between the centroid and other objects. The distance can be calculated using the Euclidean distance formula (**Equation 1.3**).

$$\textbf{E. D}=\sqrt{(A_H - H_1)^2 + (A_w - W_1)^2}$$

Equation 1. 3 Euclidean distance formula

Where, **XH=** Observation value of height, **H1=** Centroid value of cluster 1 for height, **Xw =** Observation value of height, **W1=** Centroid value of cluster 1 for weight. 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 1.4 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 object in a cluster.

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

Table 1.5 Table showing update of centroid in new cluster in K-means method

Step 4: Repeat step 1, 2 and step 3 until the centroids stop changing. The output created in our example is present in **Table 1.6**.

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

Table 1.6 Cluster created by K-means method

Step 5: Return the k clusters. In this case, clusters returned are {(185,72), (179,68), (182,72), (188,77)} and {(170,56), (168,60)}.

1.4.2 Association Rule Mining

In data mining, association is useful for analyzing and predicting customer behavior. (Agarwal & Srikant, 1994) play an important part in shopping basket data analysis. Association rule mining is primarily focused on finding frequent co-occurring associations among a collection of items. It is sometimes referred to as "Market Basket Analysis", since that was the original application area of association mining. Apriori (Agrawal & Srikant, 1996), is an algorithm for frequent item set mining and association rule learning over transactional databases. A general survey of association rule mining techniques is given by (Hipp, Güntzer & Nakhaeizadeh, 2000). An association rule expression is of the form X ⇒ Y, where "⇒" is intended to give a direction to the nature of correlation between the set of items X and Y.

Support(s): The support of an itemset X ⊆ I is the fraction of transactions in (T) that contain both X and Y. The support count of an Itemset in a transaction database can be calculated as the number of transactions of the database that contain the itemset.

$$Support\ (itemset)\ =\ \frac{number\ of\ tuples\ in\ the\ itemset}{total\ number\ of\ tuples\ in\ the\ database}$$

Equation 1. 4 Equation to compute support of an itemset

X. The confidence of the rule X ⇒ Y is the conditional probability that a transaction in T contains Y, given that it also contains X.

$$\text{Confidence (X -> Y)} = \frac{Support\ (X\ U\ Y)}{Support\ (A)}$$

Problem 1.4.2: Consider for the given transactions, let say T= {T1, T2, T3, T4} given in Table 1.7, some items are bought in all these transactions, where candidate set (C_1) = {A, B, C, D} using association rule mining (Apriori algorithm), we can find the set of frequent patterns from large itemsets (L_i) iteratively by computing the support of each itemset in the candidate set C_i.

Transaction Id (TID)	**Items**
T1	A,B,C,D
T2	A,B,D
T3	A,B
T4	B,C,D
T5	B,C
T6	C,D
T7	B,D

Table 1.7 Transaction database for Apriori Algorithm

Solution for Problem 1.4.2:

Input: Transaction database with transaction id and items purchased as given in Table 1.7 and minimum support =2.

Output: Frequent pattern items

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 (L2) from candidate set (C2).

Frequent item (L2) is obtained by following the same procedure as in step 1. We can count the occurrence of each item in candidate set (C2), and infrequent items are removed to create frequent itemset (L2) = {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 U L_2= {A, B, C, D, AB, BC, BD, CD}.

1.4.3 Classification

Classification is a data mining function that assigns items in a collection to target categories or classes. The objective of classification is to accurately predict the target class for each record in the data. For example, a classification model used to identify loan applicants as low, medium, or high credit risks (Kotsiantis, Zaharakis & Pintelas, 2007). The classification using decision tree induction (Apté & Weiss, 1997) is one of the most widely used classification technique. The decision tree has two types of nodes, decision node (internal nodes) and a leaf node. A decision node specifies test (asks a question) on a single attribute. A leaf node indicates a class.

Example of Classification by decision tree.

Consider the client data which comprises of attributes (Age, Job Status, House Owned, Credit Score, Credit offer), using the decision tree classification we must determine whether the person is eligible for credit card offer.

TID	Age	Job Status	House Owned	Credit Score	Credit Offer
1	Young	Jobless	No	Fair	No
2	Young	Jobless	No	Good	No
3	Young	Employed	Yes	Fair	Yes
4	Middle	Employed	Yes	Good	Yes
5	Middle	Jobless	Yes	Excellent	Yes

Table 1.8 Dataset to be classified by the decision tree

Therefore, the decision tree is used to determine the credit card eligibility based on their Credit Score and house ownership.

1.5 Sequential Pattern

A sequence occurring in an ordered list of events with respect to time are called the Sequential Pattern (Agrawal & Srikant, 1995). A sequential Pattern is generally enclosed within the angular

brackets (< >), and each itemset contains sets of items separated by commas (,). For example, a sequential pattern in an e-commerce system such as < (Bread, Milk), (Bread, Milk, Sugar), (Milk), (Tea, Sugar)> means customer has bought(Bread, Milk)together in his first purchase transaction, (Bread, Milk and Sugar) in the second purchase, Milk alone in the third purchase and (Tea and Sugar) together in the fourth purchase. An item can occur at most once in an event of a sequence but can occur multiple times in different events of a sequence. The number of instances of items in a sequence is called the length of the sequence. A sequence with length l is called an l-sequence (Han, Pei & Kamber, 2011).

1.6 Sequential Database

Sequence database is composed of a collection of sequences {**s1, s2, …, sn**} 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 (). Let us consider a very common example of a grocery store as shown in Table 1.9, which contains <CustomerID, PurchasedItem, Timestamp>.

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

Table 1.9 Item-Purchase Database

The above is a historical database from which we can generate the sequential database, which could be interpreted as in Table 1.10 where SID represents the Sequence Identifier.

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

Table 1.10 Sequential Database

The above Table 1.10 is the sequential database created from the historic data, the SID (01) has the purchase sequences for the customer (01) such as < (Bread, Milk), (Bread, Milk, Sugar), (Milk)>. In the first purchase, he bought (Bread, Milk), then (Bread, Milk, Sugar) and finally (Milk).

*1.7 **Sequential Pattern Mining***

Sequential Pattern Mining is a process of discovering all frequent sequential subsequences from transactions of items with minimum threshold support (Agrawal & Srikant, 1996). Sequential Pattern Mining discovers 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 discussed in (Mabroukeh & Ezeife, 2010) survey. 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 = {i1, i2, . . . , ik}, 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 SPM can be divided into four main categories of SPM algorithms, namely, **apriori-based:** AprioriAll (Agrawal & Srikant, 1995), GSP (Srikant & Agrawal, 1996), PSP (Masseglia, Poncelet & Cicchetti, 1999), SPAM (Ayres, Flannick, Gehrke & Yiu, 2002); **pattern-growth:** FreeSpan (Han et al., 2000), PrefixSpan (Pei. et al, 2001), WAP-mine (Pei, Han, Mortazavi & Zhu, 2000); **early-pruning:** LAPIN (Yang, Wang & Kitsuregawa, 2007), HVSM (Song, Hu & Jin, 2005) and **hybrid** algorithms: SPADE (Zaki, 2001).

1.8 Utility Framework

Utility is a quantitative representation of user preference and can be defined as “A measure of how ‘useful’ (i.e. profitable) an itemset is” (Yao & Hamilton, 2006). In simpler words, the utility of an item in a transaction is an importance value of the item in that particular transaction computed from the product of the per unit price value and quantity of item bought. Utility is introduced into frequent pattern mining to mine for patterns of high utility measured over various kinds of objective criterias such as cost, profit, or other measures of user preference of itemsets. This has led to high utility pattern mining, which selects interesting patterns based on minimum utility rather than minimum support (Yin, Zheng & Cao, 2012). A high utility itemset is a set of values

that appears in a database and has a high importance to the user, as measured by a utility function. High utility itemset mining generalizes the problem of frequent itemset mining by considering item quantities and weights. The goal of utility mining is to discover all the itemsets whose utility values are higher than the user specified threshold in a transaction database. We start with the definition of a set of terms that leads to the formal definition of utility mining problem considering. Assume $I = \{i_1, i_2, ..., i_n\}$ is a set of items and $D = \{TS_1, TS_2, \ldots, TS_n\}$ be a transaction database D where each transaction sequences (TS), $TS_i \in D$ is a subset of I given in Table 1.11 for $P = \{p1, p2, \ldots, pl\}$ be a pattern (itemset), where $P \subseteq I$ and $l \in [1, n]$. Table 1.12 contains the price or quality of each item in the transaction database D.

TID	Transactions	Transaction Utility (TU)
T1	(a,2) (d,4) (e,1)	15
T2	(e,2) (f,2)	4
T3	(a,1)(b,1)(c,4)(d,5)	34
T4	(b,2)(d,5)(e,3)	23
T5	(a,1)(c,2)(d,5)(e,3)	24

Table 1.11 Transaction Database

ITEM	a	b	c	d	e	f
Weight/Quality ($)	3	5	4	2	1	1

Table 1.12 Quality Table

1.8.1 High Utility Itemset Mining (HUIM)

Mining high utility itemsets from databases refers to finding the itemsets which can bear high profits. Here, the meaning of itemset utility is the interestingness, importance, or profitability of an item to users (Tseng, Wu, Fournier & Yu, 2016). The utility of items in a transaction database consists of two aspects, namely **external utility,** and **internal utility**. Every item in the itemsets is associated with an additional value, called internal utility which is the quantity (i.e. count) of the item. An external utility is attached to an item, showing its quality (e.g. price) (Yin, Zheng & Cao, 2012). Utility of an itemset is defined as the sum of all the product of its external utility and its internal utility of all items. An itemset is called a high utility itemset if its utility is no less than

a user-specified minimum utility threshold; otherwise, it is called a low-utility itemset. The authors first defined the problem of mining high utility itemsets, and a theoretical model of utility mining was proposed.

Problem 1.8.1: For a Transaction Table 1.11 (input database D) where each transaction has items and its internal utility (count of item bought) . The quality table in the Table 1.12, which contains the external utilities of all the items, namely I = {a, b, c, d, e, f} and a user specified minimum utility threshold ξ. itemset = {a, b, c, d, e, f}, find the high utility itemsets.

Solution for problem 1.8.1:

Input: Transaction table (Table 1.10.1) and Quality table (Table 1.10.2), **Output**: The High Utility Itemset Patterns.

The problem of mining high utility itemset is to discover all the itemsets whose utility is no less than ξ, from example, (a, 2) in T1 means the quantity of 'a' is 2. Therefore, the utility of (a, 2) in T1 is u (a, T1) = 3 × 2 = 6, which indicates the profit/price of 'a' is 6. Furthermore, the utility of T1 is u (T1) = u (a, T1) +u (d, T1) +u (e, T1) = 6+8+1 = 15. It is also called the transaction utility of T2. The utility of the whole database is the sum of all the transactional utilities of the whole database. Therefore, u (D) = u (T1) + u (T2) + ... + u (T5) = 15+4+... + 24 = 100. The utility of itemset {ad} in T1 is u ({ad}, T1) = 6 + 8 = 14, and the utility in the database is u ({ad}) = 14+13+13 = 40. Assume ξ = 35, then {ad} is a high utility itemset. Other high utility itemsets are {acd}, {bd}, {cd}, {d} and {de} with the utilities of 50, 35, 44, 38 and 35 respectively. According to the **downward closure property**, a pattern's support is no less than that of its super-pattern. The downward closure property does not hold in high utility pattern mining. However, when it comes to the utility framework as in the example above, the utility of {d} is 38, which is bigger than 35 (the utility of {de}) and smaller than 50 (the utility of {acd}). Both {acd} and {de} are the super-patterns of {d}, but the utilities could be either bigger or smaller. It obviously does not hold the downward closure property anymore.

Limitation: The problem 1.8.1 suffers from the large candidate generation process with more memory consumption and execution time and it fails to follow the downward closure property. Later a more efficient High utility itemset mining (HUIM) algorithm (Liu & Qu, 2012), EFIM

(Efficient high-utility Itemset Mining) (Zida, Fournier-Viger, Lin, Wu & Tseng, 2015) introduced a one-phased algorithm which decreases the time and memory required for EFIM.

Later sequential pattern mining was introduced in the High Utility Mining. A sequence is of high utility only if its utility is no less than a user specified minimum utility (Ahmed, Tanbeer, Jeong & Lee 2010). Following the high utility pattern mining approach, highly profitable sequential patterns are retrieved, which are more informative for retailers in determining their marketing strategy. Utility Span (US) and Utility level (UL) algorithms were also proposed to mine high utility sequential patterns (Ahmed, Tanbeer, Jeong & Lee, 2010) before understanding these algorithms, it is better to understand relevant definitions in the Utility Framework.

1.8.2 Preliminary Definitions for High Utility Sequential Pattern Mining

Let a sequence *S*, denoted by {*s*1, *s*2, …, *sr*}, be an ordered list of patterns, that is, each s_q ($1 \leq q \leq r$) is a pattern *P*, and each pattern appearing in a sequence is called an element of the sequence. A Sequence Database (SDB) contains several transaction sequences (TS), where TS_s: $\{TS_1, TS_2, \ldots, TS_m\}$. *TSk* ($1 \leq k \leq m$) contains a tuple $<SID_k, S_k>$, where SID_k is the sequence ID, and S_k is the sequence of the TS_k. TS_k is said to contain a sequence, *X*, if *X* is a subsequence of S_k.

Sequence ID	Sequence with internal utility	Sequence utility ($)
S1	a(3) {a(2) b(6) d(2)} f(1) a(5) d(1)	130
S2	e(3) {a(2) b(5)} d(1) c(4)	85
S3	{c(1) f(2)} b(3) {d(1) e(4)}	74
S4	a(2) {b(7) d(4) } {a(6) b(3)} e(5)	180
S5	{d(1) f(3)} c(5) g(2)	67
S6	d(2) e(1) {a(7) b(8)} d(3) b(6) e(3)	207

Item	Profit per unit ($)
a	5
b	7
c	3
d	10
e	6
f	8
g	9

Table 1.13 Sequence database with internal utility and external utility

Definition 1. Table 1.13 shows an example SDB with internal and external utility values. Here, the *internal utility* values represent the quantities of items in sequences, and the *external utility* value of each item represents profit ($) per unit of that item. For example, in *Table 1.13*, *iu*(*b*, *S*1)=6, and *eu*(*b*)=7. However, an item may appear multiple times in a TS. In that case, $iu(ij, S_k)$ is the addition of all the quantities of *ij* in sequence *Sk*. For example, in *Table 1.13*, $iu(a, S_1)$=10.

Definition 2. Sequence utility, *su*(*ij*, *Sk*), is the quantitative measure of utility for item *ij* in *TSk*, defined by

$$su(ij, S_k)=iu(ij, S_k)\times eu(ij).$$

Equation 1. 5 Sequence Utility formula for an item

For example, $su(b, S_1)= 6\times7=42$ in Table 1.13.

Definition 3. A sequence, for example, $X= \{x1, x2, \ldots, xm\}$, is called an *m*-sequence, where $X\subseteq S_k$, $xp\subseteq I$, and $1\leq p\leq m$. To calculate the internal utility of an item, *ij*, in a sequence *X* ($X\subseteq S_k$), we must take only the internal utility of *ij* in *X*. For example, *iu*(*d*, *de*(*ab*), *S*6)=2 (where *X*=*de*(*ab*)). Hence, as with an item, a sequence *X* may have multiple distinct occurrences in TS_k. Accordingly, for sequence utility of *X* in S_k, $su(X, S_k)$ is defined by

$$su\ (X, S_k) = \sum\ \sum \text{su (ij, X, S}_\text{k}\text{) for all X} \in Sk\ \forall\ X\text{ij } \text{Ɛ X}$$

Equation 1. 6 Sequence Utility formula for sequence

However, in the above equation, we refer to only all distinct occurrences of *X*. For example, sequence *de* has two distinct occurrences in *S*6. Hence, *su*(*de*, *S*6) = (2×10+1×6) + (3×10+3×6) = 26+48 =74 in *Table 1.13.*

Definition 4. The sequence utility of a transaction is the sum of products of internal (*iu*) and external (*eu*) utilities of each item in a transaction. The sequence utility of TS_k is sum of utility of all the items in the transaction defined by:

$$\text{Su (TS}_\text{k}\text{)} = \sum \text{su(ij, S}_\text{k}\text{) for ij Ɛ S}_\text{k}.$$

Equation 1. 7 Sequence Utility formula for a transaction (k)

For example, $su(TS_1)=su(a, S1)+su(b, S1)+su(d, S1)+su(f, S1)$ =50+42+30+8=130.

Definition 5. The sequence utility of a sequence say X in SDB is the sum of sequence utility of X in all the transactions of SDB. The sequence utility of a sequence *X* in an SDB is defined by

$$su\ (X, SDB) = \sum \sum \text{su(X, S}_\text{k}\text{) for TS}_\text{k}\text{ Ɛ SDB and X is subset of S}_\text{k}\text{ .}$$

Equation 1. 8 Sequence utility formula for sequence in SDB

For example, *su*(*a*(*bd*)*a*, *SDB*) =*su*(*a*(*bd*)*a*, *TS*1) +*su*(*a*(*bd*)*a*, *TS*4) =102+129=231 in *Table 1.13.*

Definition 6. The sequence utility of the whole Sequential Database is the summation of all the transaction utilities in the database. The sequence utility value of an SDB is defined as

$$su(SDB) = \sum \mathrm{Su}(\mathrm{TS_k}) \text{ for } \mathrm{TS_k} \in \mathrm{SDB}.$$

Equation 1. 9 Sequence utility value of SDB

For example, *su*(*SDB*)=743 in *Table 1.13.*

Definition 7. The minimum sequence utility threshold, δ, is given by the percentage of sequence utility value of the database. In *Table 1.13*, if δ is 30% or can be expressed as 0.3, then the minimum sequence utility value can be defined as

$$minSeqUtil = \delta \times su\ (SDB).$$

Equation 1. 10 Formula to compute Minimum Sequence Utility threshold

Hence, in this example, *minSeqUti l*=0.3×743=223 in *Table 1.13*.

Definition 8. A sequence *X* is a high-utility sequential pattern if $su(X) \geq minSeqUtil$. Mining high-utility sequential pattern means discovering all the sequences *X* having criteria $su(X) \geq minSeqUtil$. For *minSeqUtil*=223, *a*(*bd*)*a* is a high-utility sequential patten as *su*(*a*(*bd*)*a*) = 231. The sequential pattern mining does not satisfy the downward closure property. To maintain the downward closure property in high-utility sequential pattern mining, we use a new measure called *sequence-weighted utility* (*swu*). The *swu* value of a sequence *X* is defined by

$$swu\ (X) = \sum \mathrm{su}(\mathrm{TS_k}) \text{ for X is subset of } \mathrm{S_k} \text{ and } \mathrm{TS_k} \in \mathrm{SDB}$$

Equation 1. 11 Formula to compute Sequence weighted utility

Definition 9. *X* is a high-*swu* sequence if s*wu*(*X*) ≥ *minSeqUtil*.

1.9 Problem definition

The main aim of the thesis is to improve the HSPRec19 system (Bhatta, Ezeife & Butt, 2019) and its performance. The HSPRec19 system first mined frequent sequential patterns of user purchases from historical and click purchase data with the GSP miner algorithm (Agrawal & Srikant, 1995), then used these discovered sequential pattern rules to enrich the user-item rating matrix before running the collaborative filtering algorithm for product recommendations. The HSPRec19 system discovers sequential rules based on simple item minimum support counts. While these patterns are helpful for the consumers but they do not ensure that the seller company is also gaining from this approach.

Thus, in this thesis, proposed HUSRec uses item and sequence utility values such as (quantity and price) of the item to predict the sequential rules which will bear high profits. Mining high utility patterns means finding the patterns (frequent patterns or sequential patterns) with high utility values which will help seller to increase revenue generation as well as improve the accuracy of recommendations made to the customers. The high profits can be measured in the terms of importance or profitability of the items from user to user. The aim of this thesis is to make the user-item matrix more informative as using mined high utility sequential patterns from the purchase and click history sequential database based on high profit yielding sequential patterns.

1.10 Problem Statement

For an online E-commerce system having session id's related to the clickstream (C) and purchase behavior (P) dataset for users if given a threshold minimum utility, the problem of discovering the frequent high utility sequential patterns over a dataset is to get all frequent sequences whose utility is no less than threshold utility, can enhance the user-item matrix which is the input to the collaborative filtering which could predict about the possible purchases that will be profitable for the retailer and also increase accuracy of the recommendations.

1.11 Thesis Contributions

The thesis aims to improve the work done in recommending the items in E-commerce system using the clickstream and purchase datasets. (Kim & Yum, 2011) integrated association mining rules with clickstream data to recommend the items but no behavioral sequences were considered. Then, HPCRec18 (Xiao & Ezeife, 2018) introduced the consequential bond i.e. matrix between clickstream and purchase data for the users. Further, HSPRec19 system (Bhatta, Ezeife

& Butt, 2019) discovers the frequent historical sequential pattern from click and purchase, so that discovered frequent sequential patterns are used to improve the user-item frequency matrix to improve recommendation. In this thesis, we have introduced the HUSRec system which aims to enhance the user-item frequency matrix by discovering the high utility frequent sequential patterns from the purchased data we can yield high revenue generations for the sellers.

1.11.1 Thesis Feature Contribution

1. **Converting the periodic purchase data to a High Utility Sequential Purchase database using internal and external utilities from the historic purchase dataset.**
   Developing a High Utility Sequential Purchase Database (HUSPDB) for E-commerce recommendation dataset from the purchase data using the internal utility (e.g., quantity of item bought) and external utility (e.g., cost price or profit earned per each item), further using this information we can calculate the sequence utilities of each sequence transaction in the database.

2. **Using high utility sequential patterns to improve the user item matrix and decrease the sparsity of user-item matrix**.
   The HSPRec19 (Bhatta, Ezeife & Butt, 2019) system used the sequential pattern mining algorithms for both purchase and clickstream data to improve the user-item frequency matrix, whereas in the proposed HUSRec system, the high utility sequential pattern mining algorithm (USpan algorithm) is used for the purchase database and sequential pattern mining algorithm (PrefixSpan) for click stream database which can improve the user-item rating matrix.

3. **Using the high utility sequential patterns to improve the consequential bond of the clickstream and purchase data**.
   In E-commerce click and purchase are two different types of events and the consequential bond is formed between them to understand the relationship between click stream and purchase data. This means that the click and purchase sequences, even if they contain the same items, their itemset sequences may be different.

4. **Improving the recommendation accuracy and sales profits with high utility sequential patterns.**

   We are aiming to improve user-item matrix by enhancing it with high utility sequential patterns having high utility values from purchase database and frequent sequential patterns from click stream database. This enhanced matrix is passed to CF algorithm and sales profits were calculated of each transaction for which recommendation were made. Some experimental tests were done among the existing e-commerce systems to compare the performances of each system.

1.11.2 Thesis Procedural Contribution

To make the specified feature contributions, this thesis proposes High Utility Sequential Pattern Recommender (HUSRec) system (**section 3.3**).

1. In the HSPRec19 (Bhatta, Ezeife & Butt, 2019) system, the author proposed the SHOD (Sequential Historical Periodic Database) to convert the historic data purchase data into sequential purchase database whereas in the proposed thesis system, the HUSRec system, we have proposed High Utility Sequential Database Generator (**HUSDBG**) which converts Historical Purchase data into High Utility Sequential Purchase Database (**section 3.4)**. The proposed HUSDBG enhances the SHOD sequential database generator by including the high utility factors of internal (e.g., quantity of item bought) and external utilities (e.g., cost or price of item) which are used to compute the sequence utility of the generated sequences.
2. In the HSPRec19 (Bhatta, Ezeife & Butt, 2019) system, the author has used GSP sequential pattern mining algorithm for mining both the purchase and click stream data but in the proposed HUSRec system, we are using High Utility Sequential Pattern Miner (HUSPM) such as USpan algorithm to mine the purchase data with high utility sequential patterns above the threshold minimum sequence utility and enhance the user-item matrix with the high utility patterns and the PrefixSpan sequential pattern mining technique for the click stream data mines the frequent sequential patterns and enhances matrix with them is explained in **section 3.5**
3. Using the sequential purchase sequences with internal utilities (quantity of item bought in each transaction) and the transaction sequence utilities to improve the consequential bond between sequential clickstream data and sequential purchase data. For example, in the HSPRec19 system (Bhatta, Ezeife & Butt, 2019), the consequential bond can be stated as the join between

the sequential click stream and purchase databases having only sequences of items applied on their common session ids column, whereas in the proposed HUSRec system when the sequential clickstream and high utility sequential purchase data are joined, the consequential bond in HUSRec system gives clickstream <item ids>, purchase sequence <item id (quantity)> and the sequential utilities of each transaction. For example, <3, 5, 2, 3> is a click stream sequence on session id (S1) and < (3(1)), (5(3)), (2(1), 3(3))> is a purchase sequence of S1 with the internal utilities of each item and the sequence utility (45 in dollars) is the profit earned through this purchase.

4. The High Utility Sequential Rules (HUSR) mined using the USpan algorithm have utility values greater than the threshold minimum sequence utility from sequence purchase data and the frequent sequence patterns from the click stream data will improve the user-item matrix and the accuracy of the recommendations also, after the recommendations we have compared the sales profit generated.

1.12 Outline of Thesis

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

CHAPTER 3: Discusses the proposed E-commerce high utility sequence database and proposed high utility sequential pattern recommendation system (HUSRec). This also includes an example application of the proposed technique in comparison with HSPRec19 system.

CHAPTER 4: Discusses the experimental implementation for proposed high utility sequential pattern recommendation system (HUSRec), required tools and technologies. Most importantly, a comparative analysis with the HSPRec19 system.

CHAPTER 5: Discusses about the future work and conclusion.

CHAPTER 2: RELATED WORK

Over the years, based on the relevant techniques and research ((Agrawal & Srikant, 1994), (Agrawal & Srikant, 1995), (Schafer, Frankowski, Herlocker & Sen, 2007)) proposed during the 1990s, many companies and researchers have been improving recommender methods and systems. In addition to the user-item rating matrix, some other data sources such as clickstream data, metadata and transactions have been discovered and utilized to improve recommendations. Clickstream data has been used to predict a user's next request discover patterns to build profiles for customers, find the possibilities of purchasing items etc, such as HPCRec18 (Xiao & Ezeife, 2018), HSPRec19 (Bhatta, Ezeife & Butt, 2019). For improving recommendation accuracy and make better recommendations, clickstream data has been integrated into some recommendation systems such as LiuRec09 (Liu, Lai &Lee, 2009)**,** Chen13Rec (Chen & Su, 2013), Kim11Rec (Kim & Yum, 2011), HPCRec18 (Xiao & Ezeife, 2018), HSPRec19 (Bhatta, Ezeife & Butt, 2019).

This chapter is divided into four sections which comprehend the earlier research done in the field of Sequential Pattern Mining (SPM), E-commerce Recommendation Systems based on Clickstream analysis, High Utility Pattern Mining Algorithms and their comparisons.

2.1 Sequential Pattern mining Algorithms

Sequential pattern mining (SPM) discovers frequent subsequences as patterns (sequential patterns) in a sequence database. SPM is an important problem with broad applications, including the analysis of customer purchase behavior, web access patterns, scientific experiments, disease treatment, natural disasters, and protein formations. An SPM algorithm mines frequent sequential patterns from a sequential database as sequences with support greater than or equal to a given minimum support that can be used later by end users or management to find associations between the different items or events in their data for purposes such as marketing campaigns, business reorganization, prediction and planning. In this section, we will be discussing about General Sequential Pattern Mining (GSP) and Prefix Span Algorithm.

2.1.1 GSP (Generalized sequential pattern mining) algorithm by (Srikant & Agrawal, 1996).

GSP is an Apriori-based sequential pattern mining algorithm introduced by (Srikant & Agrawal, 1996). The main process in the GSP is candidate generation (C_k) and pruning (L_k) (Srikant & Agrawal, 1996). According to the algorithm, first sequence W1 and second sequence W2 can be merged, if subsequences obtained by removal of the first element of sequence W1 and last element of sequence W2 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.

Problem 2.1: Find the frequent sequential patterns from a database with sequence of items in each transaction with minimum support 2 using GSP algorithm.

Solution for problem 2.1: **Input**: sequence database (Table 2.1), minimum support=2 and candidate set (C1) = {A, B, C, D, E, F, G}. **Output**: Frequent sequential patterns.

SID	Sequences
1	<(A), (B), (F,G), (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 2.1 Sequence Database

Step 1: Find 1- frequent sequence (L1) satisfying minimum support. For example, (L1) = {< (A):4>, <(B):4>, <(C):3>, <(D):4>, <(F):4>, <(G):4>}.

Step 2: Generate candidate sequence (C_k=2) using L1 $GSPjoin$ L1. To generate larger candidate set 2, use 1-frequent sequence (L1) found in step 1 to join itself using GSP join way, which can be written as L (k-1) $GSPjoin$ L (k-1) and it requires every sequence (W1) found in first L (k-1) joins with other sequence (W2) in the second if subsequences obtained by removal of the first element of W1 and last element of W2 are same. In our case, we are generating sequences with candidate

2, (Ck=2), which can generate 51 types of 2-length candidate set using Apriori algorithm as present in Table 2.2.

<(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 2.2 Candidate Generation Table

Step 3: Find 2- frequent sequences (L2) by counting the occurrence of 2-sequences in candidate sequence (C2) to keep the only sequence with occurrence or support count in the database greater than or equal to the minimum support. For example, L2= {<(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 (Ck) and prune (Lk) for finding frequent sequence is an empty set.

Output: Finally, the output frequent sequences as union of L1 U L2 U L3 U L4 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 2.3 Frequent Sequences Table

2.1.2 PrefixSpan (Prefix-projected sequential pattern mining) algorithm by (Pei. et al, 2001).

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

Problem 2.1.2: Find the frequent sequential patterns from a database with sequence of items in each transaction with minimum support 2 using PrefixSpan algorithm.

Solution for problem 2.1.2:

Input: sequence database (Table 2.4), **Min. support**=2**, Candidate sets**= {A, B, C, D, E, F},

Output: Frequent sequential patterns.

SID	**Sequences**
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 2.4 Sequence Database

Step 1: Count the support of single unique item and keep only sequences with support count greater than or equal to the minimum support count of 2 as in Table 2.5.

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

Table 2.5 Support of Singleton Sequence

Step 2: Prune singleton sequences which has specified minimum threshold. 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. For example, for each sequence of the sequence database (Table 2.4), 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 2.6 gives the projected database for all the items with support 1.

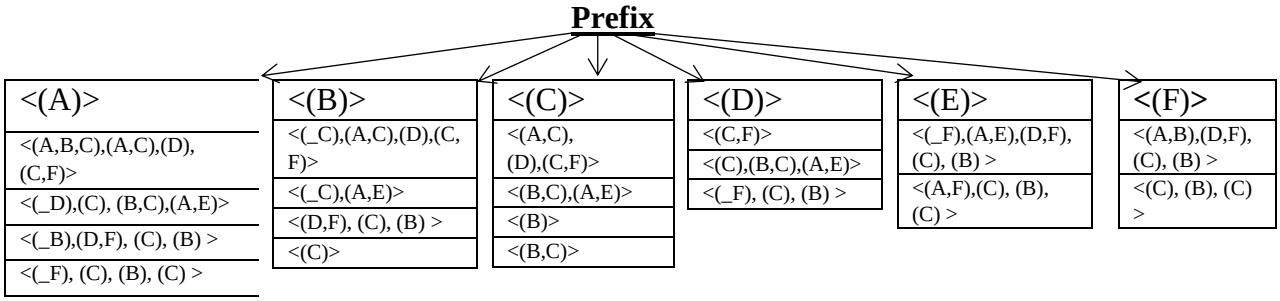

Table 2.6 Project Database

Step 4: Find frequent sequences from the projected databases and check 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 2.7.

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

Table 2.7 Project Database for (D)

2. The projected database is scanned to find the frequent items in it. In our example, only <(B)> and <(C)> are frequent.

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

Table 2.8 Frequent Items in Project Database

1. Now, the projected database for sequence < (D), (B)> and < (D), (C)> are constructed using step 4. Furthermore, their respective projected databases are scanned to get the frequent items in their projected databases.

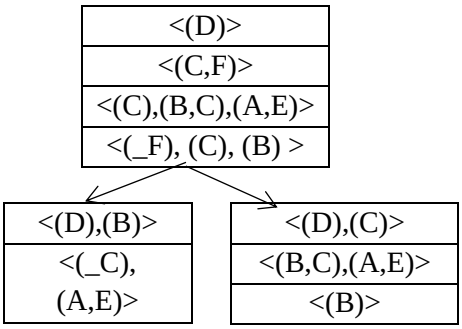

Table 2.9 Project Database of Sequence <(D), (B)> and <(D), (C)>

Step 5: Since item present in the projected database <(D), (B)> is infrequent. So, compute frequency of item present in the projected database <(D), (C)>.

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

Table 2.10 Frequency of Item in project database of Sequence <(D), (C)>

Step 6: 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 2.11 Project Database of Sequence <(D), (C), (B)>

PrefixSpan **(**Pei. et al, 2001) is an efficient pattern growth method because it outperforms apriori based and pattern growth algorithms and explores prefix-projection which reduces the size of projected database and leads to efficient processing in sequential pattern mining and more efficient with respect to running time, space utilization and scalability then Apriori based algorithms and FreeSpan (Han et al., 2000) algorithm, and PrefixSpan consumes a much smaller memory space in comparison with GSP and SPADE (Zaki, 2001).

2.2 E-commerce Recommendation Systems on click stream data

Many recommendation systems only use the purchase data of users for e-commerce recommendation, while some are based on navigational and behavioral pattern data (Moe, 2003).

Recommendation systems on clickstream data can be categorized to two groups, one uses different variables such as visiting path, visiting frequency to predict the purchase interests and preferences and the other one is based on stages which calculate the consequential or conditional relationship between different stages to calculate the purchase possibility. Specifically, we will discuss about Kim11Rec (Kim & Yum, 2011), which uses a stage-based approach, HPCRec18 (Xiao & Ezeife, 2018), which introduced the consequential bond between the clicks and purchase database and HSPRec19 (Bhatta, Ezeife & Butt, 2019), which integrated the sequential patterns to the HPCRec18 and improved the user-item frequency matrix for collaborative filtering.

2.2.1 Recommender system based on click stream data using association rule mining (Kim11Rec) by (Kim, & Yum, 2011).

Kim05Rec (Kim, Yum, Song, & Kim, 2005) system used collaborative filtering technique based on navigational and behavioral patterns of customers. Kim11Rec (Kim & Yum, 2011) system was proposed to integrate association rules in Kim05Rec and 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. 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, and bookmarking as given in Table 2.12.

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 2.12 E-commerce click stream data

Step 2: *Association rule mining*. Firstly, 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 CDA and CDB becomes the candidate of association rule. For each pair (CDi and CDj, $i \neq j$) the corresponding support is calculated using

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

Equation 2. 1 Equation to count support

For example, support (CD_A, CD_B) =10/50 where product CD_A in case 1 and product CD_B in case 2 as given in Table 2.1. 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. 2 Equation to compute lift value

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

For each pair, whose lift is greater than a specified threshold (1 in our case) is selected for generating more elaborate association rules.

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.

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

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 (Herlocker, Konstan, Terveen & Riedl, 2004). But fail to integrate some valuable information from the historical purchases and the 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 between clicks and purchases to generate potential ratings to improve the rating quantity.

Problem 2.2.2: Consider frequency and the consequential table containing clicks and purchases as shown in Table 2.13 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, enhance the user-item frequency matrix.

Solution for problem 2.2.2: **Input:** Consequential bond and purchase frequency matrix.

Output: enhanced user-item matrix.

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 2.13 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 into numbers between 0 and 1 using the unit vector formula as given in Equation 2.3.

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

Equation 2. 3 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√12+22+02+32⁄=0.53. In the same way, we can get normalize frequency matrix as shown in Table 2.14.

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 2.14 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|))

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

Equation 2. 4 Longest common subsequence rate

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 as calculated in Table 2.15.

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

Table 2.15 Weighted transactional table of purchase set created from consequential bond

Step 4: Using TWFI (Transaction-based Weighted Frequent Item), which takes a weighted transaction table, where weights are assigned to each transaction as input and returns items with weighted support in each transaction (Yun & Leggett, (2005)). For example, let's consider minimum weighted support=0.1, then, we will have frequent weighted transaction table as shown in Table 2.16.

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. 16 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. 17 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. 18 Weight for item present in purchase pattern

2.2.3 Mining the sequential pattern based on daily purchase history for the collaborative filtering, (HSPRec19) by (Bhatta, Ezeife &Butt, 2019).

The major goal of the proposed Historical Sequential Recommendation (HSPRec19) is to mine frequent sequential pattern from E-commerce historical data to enhance a user-item rating matrix from discovered patterns. The authors (Bhatta, Ezeife & Butt, 2019) has proposed two algorithms such as HSPRec19 (Historical sequential recommendation) and SHOD (Sequential historical periodic database) System and created a daily purchase sequence database of customer database based on consequential bond between the click and the purchase database.

Problem 2.2.3: For user-item purchase frequency matrix (Table 2.19) which has historic purchase information, find the sequential patterns from the consequential bond of click and purchase dataset to enrich the user-item matrix using HSPRec19 and SHOD algorithms.

Solution for problem 2.2.3:

Input: minimum support, historical click and purchase database containing consequential bond.

Output: rich user-item matrix

Step 1: Create a user-item frequency matrix from historical purchase. In our case, the user-item frequency matrix created from historical purchase is present in Table 2.19.

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 2. 19 User-item frequency matrix created from historical purchase

Step 2: Convert historical purchase to the sequential database using SHOD on daily purchases. Here in our case, let's construct purchase sequential database from historical purchase information as present in Table 2.20.

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

Table 2. 20 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 2.21 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 2. 22 Rich user-item frequency matrix created with help of sequential rule

Step 5: As we can see in that there is no purchase information of user 4. So, we must find the suggested items for user 4 using the following steps:

1. Form click sequential database from the consequential bond. Here, we are creating a daily sequential database for given click and purchase data.

SID	Click Sequence
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 2. 23 Sequential database created from consequential table

2. Create n-frequent click sequential pattern from click sequential database using the GSP algorithm as follows: 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 rules from frequent click sequential pattern. Here in our case possible sequential rule from n-frequent sequences from click sequences are

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

Table 2. 24 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, for click sequence < (Cheese, Milk)> thus item < (Cream)> is recommended from the sequential rules.

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 2. 25 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. 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)**>}.

1. Calculate LCSR (X, Y) = |common(X,Y)| / max(|X|,|Y|) = 59 = 0.55
2. 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
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 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 2. 26 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 2.27.

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 2. 27 Weighted purchase patterns

Step 8: Assign weighted purchase patterns (Nakkuma & Abe, 1998) to Weighted Frequent Purchase Pattern Miner (WFPPM) module and compute a weight for item present in weighted purchase pattern using formula: $Ritemi=\frac{\Sigma \text{CPS containing}}{\text{itemini=1Support (itemi)}}$

i. Count support of item: Milk(3) ,Cream (3), Cheese(3), Honey(4), Butter(3).

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 2.28.

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 2. 28 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 2. 29 Quantitatively rich purchase user-item purchase frequency matrix

2.3. High Utility Sequential Pattern mining algorithms

Although sequential pattern mining treat all items as having the same importance/utility and assumes that an item appears at most once at a time point, which do not reflect the characteristics in the scenario of several real-life applications and thus the useful information of sequences with high utilities (high profits) are lost. High utility sequential pattern considers the external utility (e.g., unit profits) and internal utility (e.g., quantity) of items such that it can provide users with patterns having a high utility (e.g., profit).

2.3.1 Utility Span Algorithm (US) by (Ahmed, Tanbeer, Jeong & Lee 2010).

Utility Span extends the traditional sequential pattern mining approach. Utility Span is a high-utility sequential pattern mining with a pattern growth approach. To avoid the several database scans in the Utility Level (Ahmed, Tanbeer, Jeong & Lee 2010), the US algorithm is proposed in a manner that it needs maximum of three database scans. Therefore, it significantly reduces the overall runtime for mining high-utility sequential patterns.

Sequence ID	Sequence with internal utility	Sequence utility ($)
S1	a(3) {a(2) b(6) d(2)} f(1) a(5) d(1)	130
S2	e(3) {a(2) b(5)} d(1) c(4)	85
S3	{c(1) f(2)} b(3) {d(1) e(4)}	74
S4	a(2) {b(7) d(4) } {a(6) b(3)} e(5)	180
S5	{d(1) f(3)} c(5) g(2)	67
S6	d(2) e(1) {a(7) b(8)} d(3) b(6) e(3)	207

Item	Profit per unit ($)
a	5
b	7
c	3
d	10
e	6
f	8
g	9

Table 2. 30 Sequence database with internal utility

Problem 2.3.1: Find the high utility sequential patterns based on their minimum sequence utility from a sequence database which has internal utility, sequence utility and has profits gained per item in Table 2.30.

Solution for problem 2.3.1: **Input:** minimum sequence threshold utility and a sequence dataset with internal and external utilities. **Output:** high utility sequential patterns

1. Calculate the *minSeqUtil* using formula in **definition 7** in Chapter 1.

2. Scan the SDB once to detect length-1 *swu* sequences. Subsequently, it generates projected databases by considering length-1 *swu* sequences as prefixes with a second database scan. Then, using a pattern growth approach, it divides the search spaces (projected databases) recursively and applies the same technique into them. The US algorithm only generates the high-*swu* sequences without generating many intermediate candidates.

Prefix	**Projected SDB**	**Candidate high swu sequences with swu values**
a	(abd)fad:130, (_b)dc:85, (bd)(ab)e:180,(_b)dbe:207	**17 high-swu sequences** a:602, aa:310, ab:517, (ab):602, ad:602, ae: 387, a(ab):310, aba:310, a(bd):310, abe:387, (ab)d:422, (ab)e:387, a(bd)a:310, ada:410, abd:387, ade:387, abde:387
b	(_d)fad:130, dc:85, (de):74, (_d)(ab)e:180, dbe:207	**8 high-swu sequences** b:676, ba:310, bb:387, bd:496, (bd):310, be:461,bbe:387, (bd)a:310
c	(_f)b(de):74	**1 high-swu sequence** c:226
d	fad:130, c:85, (_e):74, (ab)e:180, (_f)c:67, e(ab)dbe:207	**10 high-swu sequences** d:743, da:517, db:387, dd:337, dad:337, d(ab):387, dae:387, d(ab)e:387, dbe:387
e	(ab)dc:85, (ab)dbe:207	**8 high-swu sequence** e:546, ea:292, eb: 292, e(ab):292,ead:292,e(ab)d:292, ebd:292
f	ad:130, b(de):74, c:67	**1 high-swu sequence** f:271

Table 2. 31 Candidate generation for US algorithm

3. For prefix *a*, (*_b*) means that the last item in the prefix, which is *a*, forms one element (*ab*). However, in the a-projected database, we get <*a*: 310, *b*: 517, *_b*: 602, *d*: 602, *e*: 387, *f*: 130, and *c*: 85>. Items *c* and *f* cannot form a candidate sequence with item *a* as they have low-*swu* values (130 and 85, respectively) in the *a*-projected database with respect to the *minSeqUtil*. Now, according to the divide-and-conquer rule, we apply the same technique on the projected databases of *aa*, *ab*, (*ab*), *ad*, and *ae*. The *aa*-projected database contains {(*_bd*) *fad*: 130, (*_b*)*e*: 180}, and we get <*a*: 130, *_b*: 310, *d*: 130, *f*: 130, *e*: 180>. So, only one

candidate sequence, *a*(*ab*): 310, is generated. The *ab*-projected database contains {(_*d*)*fad*: 130, (_*d*)(*ab*)*e*: 180, *e*: 207}, and we get <*a*: 310, *b*: 180, *d*: 130, _*d*: 310, *e*: 387, *f*: 130>. Candidate sequences *aba*: 310, *a*(*bd*): 310, and *abe*: 387 are generated here.

4. Similarly, the other candidate sequences are generated. Table 2.31 shows that a total of 17 candidate sequences are generated by prefix-*a*. A third database scan is needed to discover the high-utility sequential patterns from the high-*swu* sequences. The US algorithm discovers the high-utility sequential patterns <*b*: 266, (*ab*): 239, (*ab*)*d*: 238, *adbe*: 226, *a*(*bd*)*a*: 231, *d*(*ab*)*e*: 250>.

2.3.2 USpan Algorithm (Yin, Zheng & Cao, 2012)

(Yin, Zheng & Cao, 2012) proposed a new definition for high utility sequential pattern mining, which aims at finding sequences having a maximum utility. USpan algorithm satisfied Downward Closure Property, USpan is composed of a lexicographic q-sequence tree, two concatenation mechanisms, and two pruning strategies.

item	a	b	c	d	e	f
Weight/quality	2	5	4	3	1	1

Table 2. 32 Quality table for each item

SID	q-sequence
1	<(e, 5) [(c, 2) (f, 1)] (b,2)>
2	<[(a, 2) (e, 6)] [(a, 1) (b, 1) (c, 2)] [(a, 2) (d, 3) (e, 3)]>
3	<(c, 1) [(a, 6) (d, 3) (e, 2)]>
4	<[(b, 2) (e, 2)] [(a, 7) (d, 3)] [(a, 4) (b, 1) (e, 2)]>
5	<[(b, 2) (e, 3)] [(a, 6) (e, 3)] [(a, 2) (b, 1)]>

Table 2. 33 Sequence table with quantity

USpan is composed of a lexicographic quantitative-sequence tree (LQS-tree) which is the search space, two concatenation mechanisms I-Concatenation and S-Concatenation to generate newly concatenated utility-based sequences, and two pruning strategies. Based on the LQS-tree structure, USpan (Yin, Zheng, & Cao, 2012) adopts the sequence-weighted utilization (SWU) measure and the Sequence Weighted Downward Closure (SWDC) property to prune unpromising sequences and to improve the mining performance.

Input: sequence, sequence utility, Sequence Database and minimum utility threshold

Output: high utility patterns

Step 1: Lexicographic Q-sequence Tree

For utility-based sequences, the authors adapted the concept of the Lexicographic Sequence Tree in (Ayres, Gehrke, Yiu & Flannick, 2002) to the characteristics of q-sequences, and come up with the Lexicographic Q sequence Tree (LQS-Tree) to construct and organize utility based q-sequences. Suppose a sequence of length-k (k-sequence) *t* appends every new item at the end of t to form (k+1) sequences concatenation. The type of concatenation depends on the two conditions that if the size of *t* does not change it is called as I-concatenation, or if the size of *t* increases by one then it is called S-concatenation. A lexicographic q-sequence tree (LQS-Tree) T is a tree structure satisfying the following rules:

1. Each node in T is a sequence along with the utility of the sequence, while the root is empty.
2. Any node's child is either an I-Concatenated or S-Concatenated sequence node of the node itself.
3. All the children of any node in T are listed in an incremental and alphabetical order.

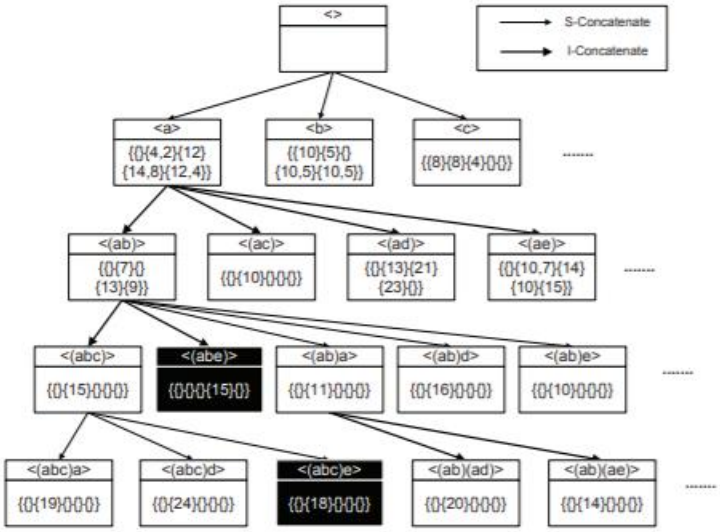

Figure 2. 1 The Complete-LQS-Tree for the Example of Q-Sequence database

The LQS-Tree has an empty q-sequence as root, while the nodes in the black boxes such as (abe) are leaves in the LQS-Tree. The bold lines and the light lines represent I-Concatenation and S-Concatenation, respectively. Nodes within the same parent are arranged in increasing order. The utilities of the sequences are in the bottom of the respective boxes. For example, for a given sequence t and database S, if we want to calculate v (ea), we simply find all the q-subsequences in

each q-sequence that match (ea) and calculate and aggregate the utilities of those q-subsequences. We obtain v((ea)) = {{8, 10}, {16, 10}, {15, 7}} and umax((ea)) = 41.

USpan (Yin, Zheng & Cao, 2012) consequently uses a depth-first search strategy to traverse the LQS-Tree to search for high utility patterns. As shown in *Figure 2.1*, USpan (Yin, Zheng & Cao, 2012) first generates the children of the root, then takes (a) as the current node, checks whether (a) is a high utility pattern, and scans for possible children. If (a)'s first children, i.e. ((ab)), are not taken as the current node, the same operations will apply to ((ab)). This will be happening recursively until there is no other node in the LQS-Tree to visit.

Step 2: Scanning subroutine: Scan the projected database S($\vartheta(t)$) to concatenate the items into lists i.e. put I-Concatenation items into *ilist* or/and S-Concatenation items into *slist*.

Concatenations with q-sequence is illustrated below with an example:

Input*:* sequence t: [(b, 2) (e, 2)] [(a, 7) (d, 3)] [(a, 4) (b, 1) (e, 2)]

Each element in the matrix in Table 2.34 is a tuple; the first value shows the utility of the q-item, and the second is the utility of the remaining items in the q-sequence; we call it remaining utility.

items	q-itemset 1	q-itemset 2	q-itemset 3
a	(0, 50)	(14, 24)	(8, 7)
b	(10, 40)	(0, 24)	(5, 2)
d	(0, 40)	(9, 15)	(0, 2)
e	(2, 38)	(0, 15)	(2, 0)

Table 2. 34 utility matrix of Q-sequence for s4 in Table 2.22

I-Concatenation: In the above Table 2.34, only items larger than b can be I-Concatenated, i.e. entries in the rectangle from d1 to e3 are possible items, so items corresponding to e1 = (2, 38) to e3 = (2, 0) can be used to form the q-sub sequences that match the sequence <(be)>. The utilities of <(be)> are the utilities of ϑ (<b>,t) plus the newly added q-items utilities e1 = (2, 38), e3 = (2, 0), i.e. ϑ(<(be)>,t) = {10 + 2, 5 + 2} = {12, 7}. Similarly, we have ϑ(<a>,t) = {14, 8}, and utilities for its I-Concatenated sequences ϑ(<(ab)>,t) = {13}, ϑ(<(ad)>,t) = {23}, ϑ(<(ae)>,t) = {10}, etc.

S-Concatenation: Let's continue with <(be)>. As we can see from the utility matrix, there is no other literal that can be I-Concatenated to <(be)>. Q-items that can be S-Concatenated to the q-sub

sequences are in the rectangle region from a1 to e3. Thus, sequences such as <(be)a>, <(be)b>, <(be)d> and <(be)e> are the candidates. <[(b, 2)(e, 2)](a, 7)> and [(b, 2)(e, 2)](a, 4)> match sequence <(be)a>, whose utilities are ϑ(<(be)a>,t) = {12 + 14, 12 + 8} = {26, 20}. We also have ϑ((be)b>, t) = {17}, ϑ(<(be)d>, s4) = {21}, ϑ(<(be)e>, t) = {14}.

Step 3: The tree is constructed in this way and then the high utility sequences are outputted if qualified, and recursively is carried out to go deeper in the LQS-Tree as shown in Figure 2.2.

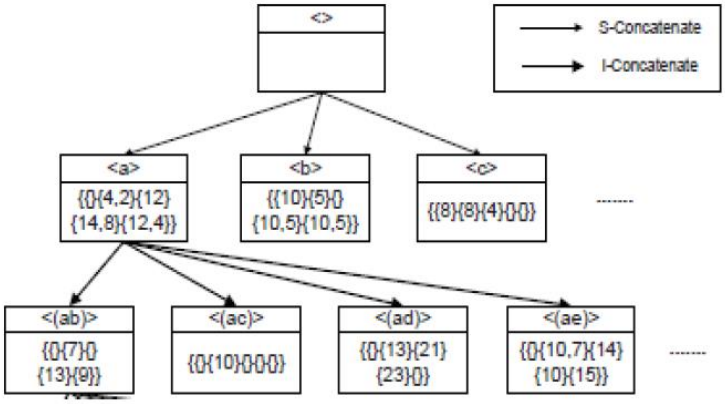

Figure 2. 2 LQS tree

2.4 Comparison of the existing models

2.4.1 Comparison of existing Sequential Pattern Mining algorithms.

Name	Description	Limitations
GSP (Srikant &Agrawal, 1996)	GSP is an apriori algorithm based on candidate generation and pruning strategy.	To decide whether one sequence is frequent or not, it is necessary to scan the entire database, to verify whether pattern is present in each sequence in the database.
PrefixSpan (Pei. et al, 2001)	PrefixSpan is a pattern-growth mining method which find the sequential pattern by avoiding generation of candidates based on creation of a prefix projected database.	PrefixSpan algoritm creates huge prefix projected databases for scanning which could occupy lots of memory space but less than candidate generation.

Table 2. 35 Comparison of existing Sequential Pattern Mining algorithms

2.4.2 Comparison of existing systems based on clickstream.

Name	Description	Limitations
Kim11Rec (Kim & Yum, 2011)	Recommended product for the customers based on clickstream data using collaborative filtering and Association rule mining.	The model is producing recommendation based on the clickstream data only based on support and confidence.
HPCRec18 (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.	Unable to integrate sequential pattern during qualitative and quantitative analysis of user-item matrix.
HSPRec19 (Bhatta, Ezeife & Butt, 2019)	HSPRec19 first generates an E-Commerce sequential database from historical purchase data using SHOD and mines frequent sequential purchase patterns 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.	Unable to utilize the information such as price and quantity of items to yield high profit sales.

Table 2. 36 Comparison of existing recommendation systems based on Clickstream data

2.4.2 Comparison of existing high utility sequential pattern mining algorithms.

Name	Description	Limitations
US (Ahmed, Tanbeer & Jeong, 2010)	US algorithm is the first algorithm to mine sequential patterns with high utility values using minimum scans of the database.	The algorithm was specific and focused on simple situations.
USpan (Yin, Zheng & Cao, 2012)	Uspan targets to satisfy the downward closure property using the lexicographic tree structures of the sequence databases. Two pruning techniques are added to reduce the number of candidates generated.	A shortcoming of USpan is that the data representation with respect to the utility matrix is quite complex and memory costly.

Table 2. 37 Comparison of existing high sequential pattern mining algorithms.

CHAPTER 3: The Proposed High Utility Sequential Pattern Recommendation (HUSRec) System

The High Utility Sequential Pattern Recommendation (HUSRec) System recommends the products based on the clickstream and purchase data. But sometimes such approaches may not be sufficiently practical for industrial needs. This is since many patterns returned by sequential pattern mining are not particularly related to a business need so that businesspeople do not know which patterns are truly profitable for their business to generate more revenue. The existing sequential pattern algorithms consider only binary frequency values of items and equal importance/ significance values of distinct items. Moreover, they use support measures to detect whether the sequence is frequent or not. However, this assumption cannot truly represent many real-life scenarios. For example, in a retail market, each item has a different price/profit value, and a user may buy multiple copies of the same item.

The proposed HUSRec system will use the extra information given in the dataset such as the quantity of the item bought which directly tell us about demand of the item and price of item to calculate utility value of each item in the transaction which can help us to mine high utility value items that can contribute in making profitable business decisions. Motivated by the above real-life scenarios, in this thesis, we propose a novel framework HUSRec for mining high-utility sequential patterns using the High Utility Sequential Database Generator (HUSDBG). Our framework considers both internal (e.g., the quantity of item bought in the sequence) and external (e.g., cost price of each item) utilities of a sequence and uses a measure, sequence utility, to calculate the utility value of the sequence. It mines the high-utility sequential patterns using the USpan algorithm (Yin, Zheng & Cao, 2012) which enhances the user-item matrix with high utility items and sequential rules are mined using the PrefixSpan algorithm, in HSPRec19 (Bhatta, Ezeife & Butt, 2019) system, GSP sequential pattern mining algorithm is used to find the sequential patterns based on the support count and candidate generation strategy, the main disadvantage of using GSP algorithm is when applied to large e-commerce datasets, the execution time of GSP algorithm is more for these large datasets and the memory storage for each candidate generation is needed which can slow down the recommendation system. Therefore, PrefixSpan algorithm is more preferred than the GSP to find sequential patterns. In Figure 3.1, we can understand how proposed

HUSRec system is different from HSPRec19 and Figure 3.2 shows the flowchart of the proposed High utility Sequential Pattern Recommendation (HUSRec) system.

3.1 Figure to show how the HSPRec19 system is different from the proposed High Utility Sequential Pattern Recommendation (proposed HUSRec) System.

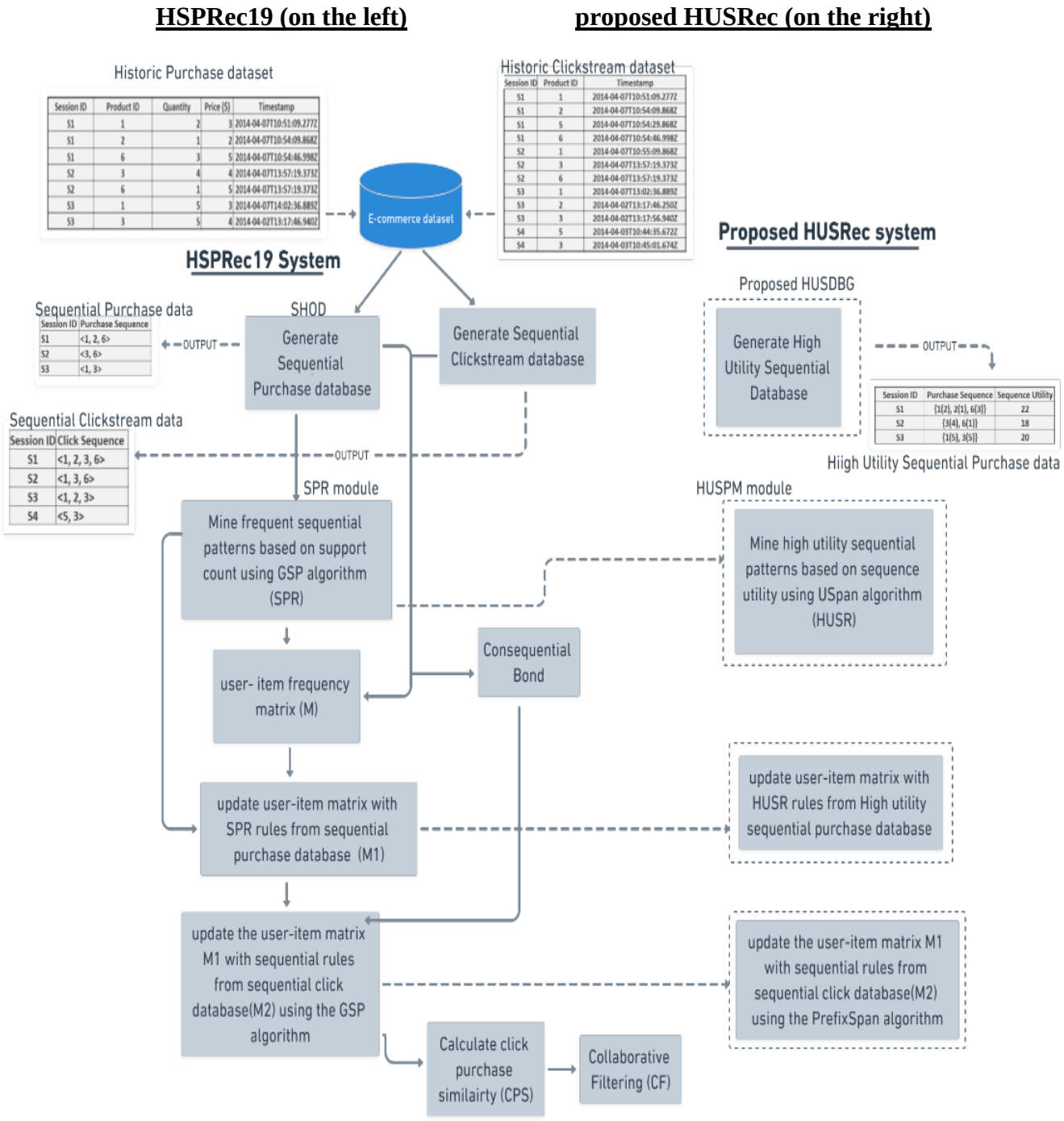

Figure 3. 1 The major differences between the HSPRec19 and proposed HUSRec systems.

3.2 Flowchart of the proposed High Utility Sequential Pattern Recommendation (proposed HUSRec) System.

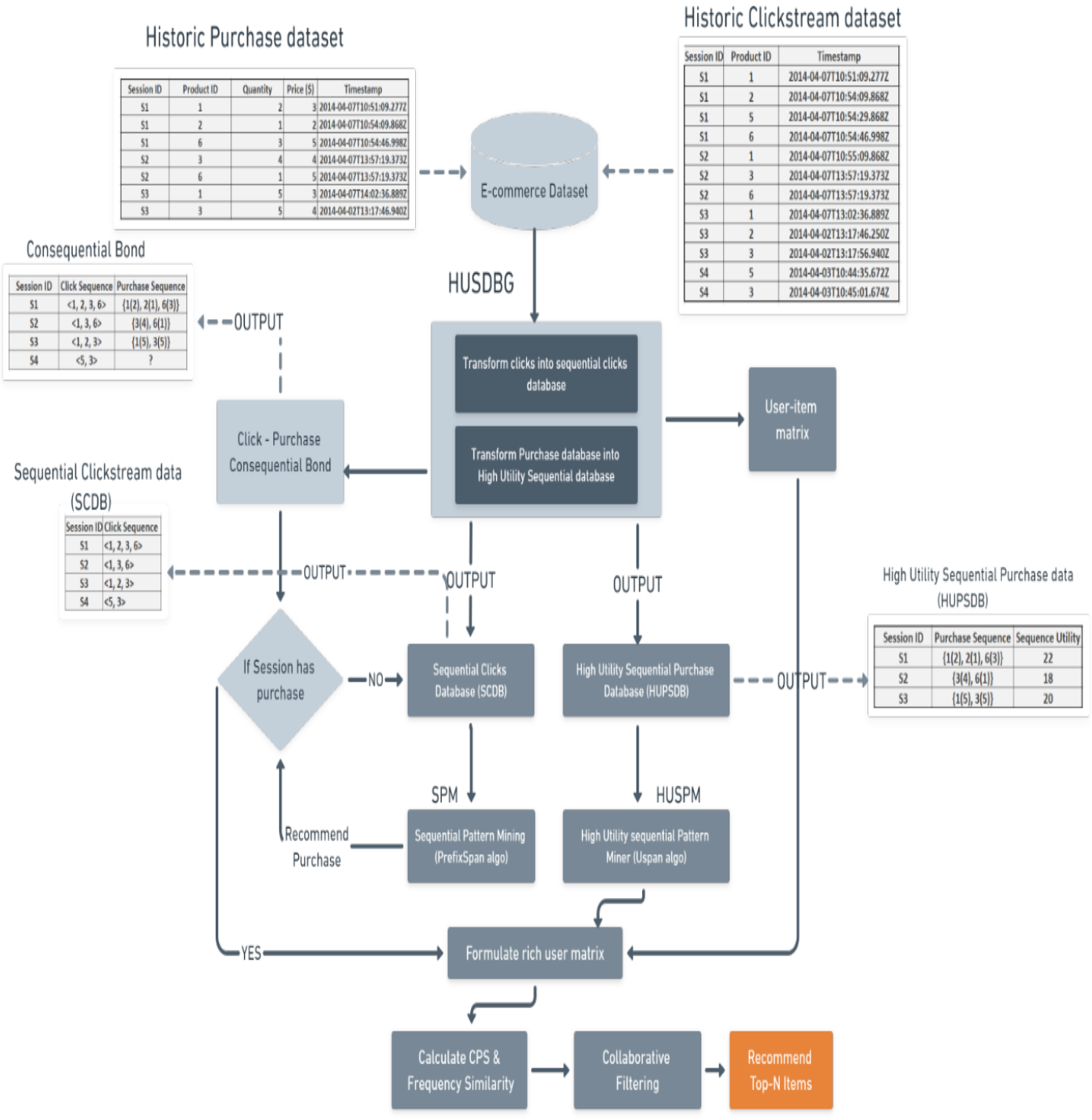

Figure 3. 2 Flowchart of proposed HUSRec System

3.3 (High Utility Sequential Pattern Recommending System) HUSRec Algorithm

Algorithm 1: High Utility Sequential Pattern Recommendation System (HUSRec)

Input: historical click database, historical purchase database, user-item purchase frequency matrix (M).

Intermediate Inputs: Consequential Bond (CB), High Utility Purchase Sequential Database (HUPSDB), sequential click database (SCDB), High Utility Sequential Rules (HUSR), Sequential Pattern Rule (SPR), user-item matrix after being updated with HUSR (M1), user-item matrix after being updated with SPR (M2), Weighted High Utility Occupancy Matrix (WHUOM), each user *u's* rating of item *i* in the matrices is referred to as r_{ui}.

Output: user-item enriched frequency matrix after normalization (normalized M2)

Procedure:

BEGIN:

1. Generate High Utility Purchase Sequential database (HUPSDB) using the High Utility

Sequential Database Generator (HUSDBG) **Algorithm 2** in **section 3.4.** from the historic purchase dataset.

2. Generate Sequential Click stream Database (SCDB) from the historic clicks dataset,

explained in Part 2. of **section3.4.**

3. Create a user- item purchase frequency matrix (M) from **section3.3.**
4. **for** each session id *sid* do:

i. Mine High Utility Sequential Rules (HUSR) using the High Utility Sequential Pattern Miner (HUSPM) which is based on USpan algorithm from the High Utility Purchase Sequential Database (HUPSDB) as explained in **section 3.5**.

ii. Update the initial user-item frequency matrix (M) based on HUSR to get updated matrix (M1).

end for.

5. Set the initial value of Weighted High Utility Occupancy Matrix (WHUOM) as '**null**'

explained in **section 3.7**.

6. **for** each session id *sid* do:
7. **if** *sid* has both click and purchase sequences:

i. Calculate Click-Purchase Similarity (CPS) between historic click Sequential database and historic purchase Sequential database explained in **section 3.6**.

if CPS is not null:

Update Weighted High Utility Occupancy Matrix (WHUOM) with CPS_s values

else:

i. Mine the sequential rules (SPR) from the Sequential Clickstream database (SCDB) using the PrefixSpan algorithm based the minimum support.

ii. Update user-item matrix (M1) with SPR from CSDB if CB has empty session.

iii. Weighted High Utility Occupancy Matrix ($WHUOM_{s)}$ updated with CPS_s calculated between purchase and historic clicks.

end if.

end for:

8. Rating of item i by user for session *sid* has value of item as in Weighted High Utility Occupancy Matrix ($WHUOM_{s)}$.
9. Modified matrix is M2.
10. Normalize M2 to get the normalized M explained in **section 3.8**.
11. Passed to User based collaborative Filtering based on user similarity.

END.

Problem 3.1: Given a transaction database D, clicks database C (Table 3.1) from an online E-commerce system having session id's related to the clickstream and P for purchase dataset for the user (Table 3.2) with prices of the items (Table 3.3). The problem of enhancing the user-item matrix which is the input to the collaborative filtering qualitatively by mining the high utility purchase sequential patterns (which are both frequent and have high utility occupancy) so as to

discover patterns with high transaction utility and aggregate the frequent clickstream patterns which could possibly lead to a purchase.

Solution for Problem 3.1: ***Input:*** The purchase sequence dataset with sequence utility and threshold Sequence Utility and clickstream sequences. ***Output:*** High utility sequential rules which have utility value more than threshold utility.

Session Id	Click Sequence	Timestamp
S1	a	2014-04-07T10:51:09.277Z
S1	b	2014-04-07T10:54:09.868Z
S1	a	2014-04-07T10:54:46.998Z
S1	d	2014-04-07T10:57:06.808Z
S1	c	2014-04-07T10:58:00.306Z
S1	e	2014-04-07T10:59:56.112Z
S2	b	2014-04-07T13:56:37.614Z
S2	f	2014-04-07T13:57:19.373Z
S2	a	2014-04-07T13:58:37.446Z
S2	c	2014-04-07T13:59:50.710Z
S2	d	2014-04-07T13:00:38.247Z
S3	a	2014-04-07T14:02:36.889Z
S3	b	2014-04-02T13:17:46.940Z
S3	a	2014-04-02T13:26:02.515Z
S3	e	2014-04-02T13:30:12.318Z
S3	c	2014-04-02T13:31:43.567Z
S3	d	2014-04-02T13:33:12.871Z
S4	e	2014-04-03T10:44:35.672Z
S4	a	2014-04-03T10:45:01.674Z
S4	b	2014-04-03T10:45:29.873Z
S4	e	2014-04-03T10:46:12.162Z
S4	d	2014-04-03T10:46:57.355Z
S4	a	2014-04-03T10:53:22.572Z
S5	f	2014-04-06T14:50:13.638Z
S5	d	2014-04-06T14:52:54.363Z
S5	a	2014-04-06T14:53:18.268Z

S6	a	2014-04-07T09:01:28.552Z
S6	b	2014-04-07T09:03:39.903Z
S6	d	2014-04-07T09:04:00.598Z
S6	e	2014-04-07T09:06:09.115Z
S6	a	2014-04-07T09:07:42.212Z
S6	b	2014-04-07T09:21:10.012Z

Table3. 1 Clickstream Historic Dataset

SID	Transaction	Quantity	Timestamp
S1	a	3	2014-04-07T10:51:09.277Z
S1	b	6	2014-04-07T10:54:39.868Z
S1	d	2	2014-04-07T10:55:34.998Z
S2	e	3	2014-04-07T13:07:06.808Z
S2	a	2	2014-04-07T13:18:00.306Z
S2	b	5	2014-04-07T13:29:56.112Z
S2	c	4	2014-04-07T13:36:37.614Z
S2	d	1	2014-04-07T13:57:19.373Z
S3	a	1	2014-04-07T14:08:37.446Z
S3	b	3	2014-04-07T14:29:50.710Z
S3	d	1	2014-04-07T14:30:38.247Z
S3	e	4	2014-04-07T14:42:36.889Z
S3	a	2	2014-04-02T15:17:46.940Z
S4	a	2	2014-04-02T13:26:02.515Z
S4	b	7	2014-04-02T13:30:12.318Z
S4	a	6	2014-04-02T13:31:43.567Z
S4	e	5	2014-04-02T13:33:12.871Z
S4	a	3	2014-04-03T10:44:35.672Z
S5	d	1	2014-04-03T10:45:01.674Z
S5	f	2	2014-04-03T10:45:29.873Z
S5	a	5	2014-04-03T10:46:12.162Z

Table3. 2 Historic Purchase Database

Item	Price per unit
a	5
b	7
c	3
d	10
e	6
f	8

Table3. 3 Table of items with their prices

Step 1: Historical data generation.

Convert the original click and purchase data (Table 3.1 and 3.2) into the sequential click database (SCDB) shown in Table 3.4 and High Utility purchase sequential database (HUPSDB) in Table 3.6 which has transaction sequence data with internal utility and sequence utility on daily basis timestamp.

User id	Click Sequence
S1	{a b a d c e }
S2	{b f a c d}
S3	{a b a e c d}
S4	{e a b e d a}
S5	{f d a}
S6	{ a b d e a b}

Table3. 4 Sequential Click database created from the historic clicks dataset.

SID	Transaction sequence data with internal utility
S1	{ a(3) b(6) d(2) }
S2	{e(2) a(2) b(5) c(4) d(1)}
S3	{ a(1) (b(3) d(1) e(4) a(2)}
S4	{ a (2)b(7) (a(6)e (5)) a(3)}
S5	{d(1)f(2)a(5) }

Table3. 5 Transaction sequence table of purchase data with the internal utility

SID	Transaction sequence	Sequence Utility ($)
S1	{ a(3) b(6) d(2) }	77
S2	{e(2) a(2) b(5) c(4) d(1)}	79
S3	{ a(1) (b(3) d(1) e(4) a(2)}	70
S4	{ a (2)b(7) (a(6)e (5)) a(3)}	134
S5	{d(1)f(2)a(5) }	36

Table3. 6 High utility Sequential Purchase data with sequence utility.

Step 2: Create consequential between the sequential clicks database (SCDB) and the High Utility Purchase sequential database (HUPSDB) based on session ids as mentioned in Table 3.7.

User id	Click Sequence	Purchase Sequence
S1	{a b a d c e}	{a(3) b(6) d(2) }
S2	{b f a c d}	{e(2) a(2) b(5) c(4) d(1)}
S3	{a b a e c d}	{a(1) (b(3) d(1) e(4) a(2)}
S4	{e a b e d a}	{a (2)b(7) (a(6)e (5)) a(3)}
S5	{f d a}	{d(1)f(2)a(5) }
S6	{a b d e a b}	?

Table3. 7 Consequential Bond b/w sequential clicks and high utility purchase sequential data

Step 3: Convert historical purchase information (present in **Table 3.2)** to user-item purchase frequency (present in **Table 3.8**) by counting the number of each purchased by a user. For example, User 2 purchased item1 twice and purchased item3 4 times.

Item/ User ID	a	b	c	d	e	f
S1	3	6	?	2	?	?
S2	2	5	4	1	2	?
S3	3	3	?	1	4	?
S4	11	7	?	?	5	?
S5	5	?	?	1	?	2
S6	?	?	?	?	?	?

Table3. 8 user-item frequency matrix based on quantities bought in a transaction

Step 4: Input HUPSDB purchase sequential database (**Table 3.6**) to High Utility Sequential Pattern Miner (HUSPM) module present in **section 3.5** to generate high utility sequential rules from sequential purchases.

I. Create the Lexicographic Q-Sequence tree for the utility - based sequences, where root is empty < >. Any node's child is either an I-Concatenated or S- Concatenated sequence node of the node itself.

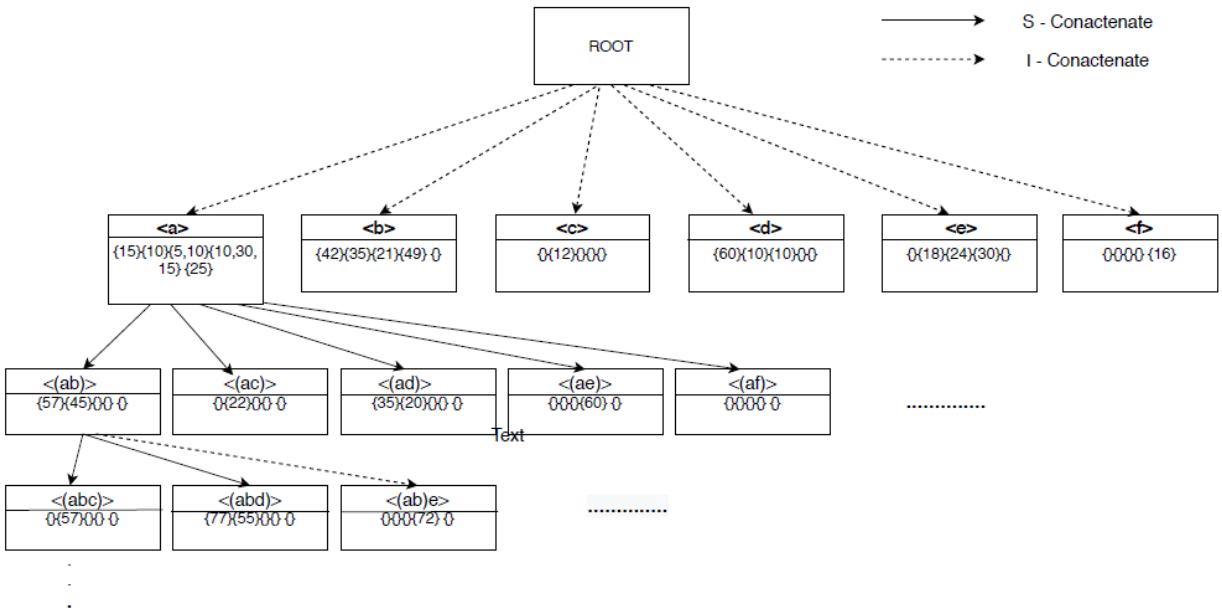

Figure 3. 3 The Complete-LQS-Tree for the Example of Q-Sequence database

II. Create a utility matrix for every q-sequence, here let's say s4, each element in the matrix is a tuple; the first value shows the utility of the q-item, and the second is the utility of the remaining items in the q-sequence; we call it remaining utility. The items that do not appear in the q-sequence are given zero utility value.

Items	*q- itemset 1*	*q- itemset 2*	*q- itemset 3*
a	(10, 124)	(30, 45)	(15, 0)
b	(49, 75)	(0, 45)	(0, 0)
e	(0, 75)	(30, 15)	(0, 0)

Table3. 9 utility matrix of Q-sequence for s4 in Table 3.6.

The I – concatenations for a sequence <a> in s4, u(<(ab)>, s4) = {10+49} = {59} and S-Concatenations for sequence <(ab)> in s4, u(<ab>e), s4) = {59+30} = US{89}.

III. Width Pruning - The item will be considered promising if after concatenating an item (i) to sequence (q) forms a new sequence (q`) explained in **section 3.5.**

SWU (q') ≥ threshold minimum utility

SWU (< (ab)e>) = {134+70} = {204}

IV. Depth Pruning - The Depth Pruning stops the USpan algorithm from going deeper in the LQS- Tree. When the gap is so large that even if all the utilities of the remaining q-items are counted into the utility of the sequence, the cumulative utility still cannot satisfy minimum threshold utility required. The high utility sequential patterns found are as follows:

ID	Sequential rule	Sequence Utility
R1	{b} → {d}	226
R2	{a} → {b}	360
R3	{a, b} → {d}	226
R4	{a, b} → {e}	204
R5	{d} → {a}	106
R6	{e} → {a}	283
R7	{e} → {b}	79

Table3. 10 High Utility sequential rules based on Confidence and Sequence Utility

Step 5: Reconstruct user-item purchase frequency matrix by using High Utility purchase sequential pattern rules. Thus, enhanced user-item purchase frequency matrix is present in **Table 3.8**. As we can see from enhanced user-item purchase frequency matrix (**Table 3.11**), there is no purchase information for User6. 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 3.7.**

Item/ User ID	a	b	c	d	e	f
S1	3	6	?	2	**1**	?
S2	2	5	4	1	2	?
S3	3	3	?	1	4	?
S4	11	7	?	**1**	5	?
S5	5	**1**	?	1	?	2
S6	?	**?**	?	?	?	?

Table3. 11 enriching the user-item frequency matrix.

Step 6: As we know that there is no information about the user 7. To determine more purchase information of user 7, the click and purchase information needs to be studied.

I. Form a click sequential database from the click and purchase consequential bond for the daily purchase and click transactions.

Session id	Click Sequence
S1	{a b a d c e}
S2	{b f a c d}
S3	{a b a e c d}
S4	{e a b e d a}
S5	{f d a}
S6	{a b d e a b}

Table3. 12 Table containing the sequential click Sequence and Session Id

II. Using the Prefix Span algorithm, mine the sequential patterns in the click sequence database.

Rule no	Sequential Rule
1	a → b
2	a, b → d
3	a, b → c
4	b, a → e
5	d → a
6	b → d

Table3. 13 Sequential rules found after the mining of click sequence dataset using PrefixSpan algorithm.

III. 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 of User 7 using part 2 in **section 3.5**.

User Id	Click Sequence	Purchase Sequence	Recommend item
S1	{a b a d c e}	{a b d}	
S2	{b f a c d}	{e a b c d}	
S3	{a b a e c d}	{a (b d e) a}	
S4	{e a b e d a}	{ab (ae) a}	
S5	{f d a}	{d f a}	
S6	{a b d e a b}	?	**a b d**

Table3. 14 Recommending item from the click sequences when purchase did not happen

Step 7: 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 **section 3.6.**

User Id	Click Sequence	Purchase Sequence	Recommend item	CPS Similarity
S1	{a b a d c e}	{a b d}		0.562
S2	{b f a c d}	{e a b c d}		0.64

S3	{a b a e c d}	{a (b d e) a}		0.58
S4	{e a b e d a}	{ab (ae) a}		0.57
S5	{f d a}	{d f a}		1
S6	{a b d e a b}	?	a b d	?

Table3. 15 CPS calculated between each click and purchase sequence for which purchase happened

Item/ User ID	**a**	**b**	**c**	**d**	**e**	**f**
S1	3	6	?	2	**1**	?
S2	2	5	4	1	2	?
S3	3	3	?	1	4	?
S4	11	7	?	**1**	5	?
S5	5	**1**	?	1	?	2
S6	**1**	**1**	?	**1**	?	?

Table 3.14 enriching the user-item frequency matrix

Step 8: Supply Click Purchase Similarity (CPS) value to purchase pattern including a recommended item from Sequential Pattern Rule (SPR) (**Table 3.16**).

User ID	**Purchase Sequence**	**CPS Similarity**
1	{a b d}	0.562
2	{e a b c d}	0.64
3	{a (b d e) a}	0.58
4	{ab (ae) a}	0.57
5	{d f a}	1
6	?	?

Table3. 16 Assigning CPS values to the purchase sequences

Step 9: Assigning weights on basis of utility occupancy.

The utility occupancy can be defined as the ratio of the utility of item or itemset, say X, in that transaction divided by the total utility of that transaction (Gan, Lin, Fournier-Viger, Chao, & Yu, 2020). The utility occupancies of (a) in **Table 3.16** are calculated using **Equation 3.3** as: uo(a) = (uo(a; T1) + uo(a; T2)+ uo(a; T3) + uo(a; T4) + uo(a; T5))/5 = (0.562 + 0.64 + 0.58 + 0.57 + 1)/5 = 0.67. Similarly, calculate for b, c, d, e and f.

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 3.17**).

Item/User ID	**a**	**b**	**c**	**d**	**e**	**f**
S1	3	6	?	2	**1**	?
S2	2	5	4	1	2	?
S3	3	3	?	1	4	?

S4	11	7	?	**1**	5	?
S5	5	**1**	?	1	?	2
S6	**0.67**	**0.588**	0.64	**0.69**	0.596	1

Table3. 17 user-item matrix with utility occupancy weights for each item

Step 11: Normalize quantitatively rich user-item purchase frequency matrix (**Table 3.17**) using unit normalization formula present in **section 3.7** to provide the level of user's interest on item between 0 and 1 as shown in **Table 3.18**. We can see, that normalized quantitatively rich user-item matrix (**Table 3.18**) is less sparse compared to initial user-item purchase frequency matrix (**Table 3.8**).

Item/ User ID	**a**	**b**	**c**	**d**	**e**	**f**
U1	0.25	0.5	?	0.166	**0.083**	?
U2	0.14	0.35	0.285	0.071	0.142	?
U3	0.50	0.50	?	0.16	0.67	?
U4	0.78	0.5	?	0.07	0.35	?
U5	0.89	0.17	?	0.17	?	0.35
U6	0.204	0.179	0.194	0.210	0.181	0.030

Table3. 18 Normalization of the user-item matrix

3.4 High Utility Sequential Database Generator (HUSDBG)

Algorithm 2: High Utility Sequential Database Generator (HUSDBG)

Input: Purchase data (*SessionId, timestamp, item id, price, quantity*), click data (*SessionId, timestamp, item id, category*)

Output: High Utility Purchase Sequential database (HUPSDB), sequential clickstream database (SCDB)

Part 1: for the historic purchase dataset

BEGIN:

1. Read first line of the purchase data file and store *SessionId* and timestamp in temp variable *temp_sessionid*.
2. **for** each line in the purchase data file:
   if (*SessionId* == temp_*sessionid*):

      Calculate Timeduration between *timestamp* and

      temp_timestamp.

if (*Timeduration* > 24):

Add item to a daily high utility transaction

with their quantity bought

else:

Add –I after the item to indicate end of the itemset

end if:

else:

Add –S after the item to indicate the end of the

sequence and calculate sequence utility for the

high utility transactions.

end if:

end for:

3. Save the high utility purchase sequential database.

END

Part 2: for historic clickstream dataset (SCDB)

The conversion of historic clickstream dataset to sequential click database is same as SHOD algorithm in HSPRec19 (Bhatta, Ezeife & Butt, 2019) system.

BEGIN:

1. Read first line of the clickstream data file and store session id and timestamp in temp variable.
2. **for** each line in the clicks data file:

if (*SessionId* == *temp_sessionid*):

Calculate Timeduration between *timestamp* and *temp_timestamp*.

if (Timeduration > 24):

Add item to daily sequence purchase database

else

Add –I after the item to indicate end of the itemset

else

Add –S after the item to indicate the end of the sequence

end if

end for

END

Working of HUSDBG with an example

The first step in the algorithm is to get the historic purchase data and click data from e-commerce dataset. As we aim to mine data in order to find the pattern which can get high sales on the daily basis, the high utility purchase sequential database (HUPSDB) should compromise of items, their respective quantity and the sequence utility of each unique transaction.

To explain the HUSDBG algorithm step by step, let us consider historical purchase data (**Table 3.19**) as input, the table has same information as Table 3.2 where *Sid* represents Session identity, Productid represents product identity, quantity represents quantity bought of the product and *Purchastime* represents timestamp when purchase occurred.

SID	Transaction	Quantity	Timestamp
S1	a	3	2014-04-07T10:51:09.277Z
S1	b	6	2014-04-07T10:54:39.868Z
S1	d	2	2014-04-07T10:55:34.998Z
S2	e	3	2014-04-07T13:07:06.808Z
S2	a	2	2014-04-07T13:18:00.306Z
S2	b	5	2014-04-07T13:29:56.112Z
S2	c	4	2014-04-07T13:36:37.614Z
S2	d	1	2014-04-07T13:57:19.373Z
S3	a	1	2014-04-07T14:08:37.446Z
S3	b	3	2014-04-07T14:29:50.710Z
S3	d	1	2014-04-07T14:30:38.247Z
S3	e	4	2014-04-07T14:42:36.889Z
S3	a	2	2014-04-02T15:17:46.940Z
S4	a	2	2014-04-02T13:26:02.515Z
S4	b	7	2014-04-02T13:30:12.318Z
S4	a	6	2014-04-02T13:31:43.567Z
S4	e	5	2014-04-02T13:33:12.871Z
S4	a	3	2014-04-03T10:44:35.672Z
S5	d	1	2014-04-03T10:45:01.674Z
S5	f	2	2014-04-03T10:45:29.873Z
S5	a	5	2014-04-03T10:46:12.162Z

Table3. 19 user-item purchase frequency matrix created from historical data

Item	Price per unit
a	5
b	7
c	3
d	10
e	6
f	8

Table3. 20 Table containing prices per unit for each item

Step 1: Read the first line of record from historical purchase data (buys.txt in our case) and store *sessionid*, *timestamp* into a temporary variable. For example, let's store first line from Table 3.19 into variable as:

TSessionId= S1, Ttimestamp= 2014-04-07T10:51:09.277Z.

Step 2: Read another line from the historical database and check recently read *sessionid* with *sessionid* stored in a temporary variable (*TSessionId*). If *SessionId* 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* and the *Quantity to* itemset in daily.txt file. In our case, the purchased time difference between two products {a**,** b} purchased by user for {*TSessionId*= S1} is less than 24 hrs. So, add two items to itemset in daily.txt

   a – 3, b -6

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,

   a -3, b -6 -I

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.

a -3, b -6 –I -S

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 3.21.

SID	Purchase Sequence with internal utilities
1	a-3, b-6, d-2 -I-S
2	e-3, a-2, b-5, c-4, d-1 -I-S

3	a-1-I, b-3, d-1, e-4 -I, a-2, -I-S
4	a-2, b-7 -I, a-6, e-5 -I, a 3–I-S
5	d- 1, f-2, a-5 -I-S

Table3. 21 Sequential database created from historical transactional database

Which is alternatively represented as shown in Table 3.22, where angular bracket < > indicates sequence, [ ] contains item set purchased together within the same session and () gives the quantity of the item bought.

SID	Purchase sequence with internal utilities.
S1	< a(3) , b(6), d(2) >
S2	< e(3), a(2), b(5), c(4), d(1) >
S3	< a(1), [b(3),d(10) ,e(4)], a(2) >
S4	< a(2),b(7), [a(6), e(5)], a (3) >
S5	< d(1), f(2), a(5) >

Table3. 22 Transaction sequence table of purchase data with the internal utility.

Step 6: Calculate the utility of a sequence of Transactions to determine the *minSeqUtil.*
The *transaction utility* of an item is directly obtained from the information stored in the transaction dataset. For example, the quantity of an item in Table 3.19 is internal utility of a transaction. The *external utility* of an item is given by the user, discounts or profits, in our case its item cost as given in Table 3.20. By combining a transaction dataset and a utility table (or utility function) together, the discovered patterns will better match a user's expectations than by only considering the transaction dataset itself on the everyday sales data. This part can be divided into further two parts:

i. Calculate the sequence utility of a transaction using the **definition 4**. For example, $su(TS_1) = su(a, S_1) + su(b, S_1) + su(d, S_1) + su(f, S_1) = 15+42+20= 77$. Similarly, calculate transaction utilities of every sequence, $su(TS_2) = 79$, $Su(TS_3) = 70$, $Su(TS_4) = 134$, $Su(TS_5) = 36$.

ii. According to **definition 6**, calculate the sequence utility of SDB. For example, $su(SDB)= 77+ 79+ 70+ 134+ 36 = 396$ in Table 3.23. For the minSeqUtil, we need to consider some threshold such as 30% of sequence utility of SDB, 0.3 * 396 = 119 approx.

SID	Transaction sequence	Sequence Utility
S1	{ a(3) b(6) d(2) }	77
S2	{e(2) a(2) b(5) c(4) d(1)}	79
S3	{ a(1) [b(3) d(1) e(4)] a(2)}	70

S4	{ a (2)b(7) (a(6)e (5)) a(3)}	134
S5	{d(1)f(2)a(5) }	36

Table3. 23 Sequence Utility of each transaction

Using the step 1 and 2, we can create the sequential clickstream database but as the database contains only the clicks, for sequential clickstream database the step 3 get modified and does not append the quantities, else work as Step 3 and 4.

3.5 High Utility Sequential Pattern Miner (HUSPM) and Sequential Pattern Miner

This module can be divided into two components as sequential rules have to mined for purchase and click sequence databases.

1. *High Utility Sequential Rule mining for the Purchase database.*

The output of HUPSDB module for the purchase database was the sequence database according on daily purchases with the quantities of the items called internal utility and a dataset of items with their unit prices called external utility.

Input: The purchase sequence dataset with sequence utility and threshold sequence utility.

Output: High utility sequential rules which have utility value more than threshold utility.

Step 1: Forming the Lexicographic Q-Sequence (LQS) Tree.

For the utility-based sequences, the concept of Lexicographic q - Sequence Tree is used to construct and organize utility based q-sequences. Each node in the tree is the sequence with the utility of the sequence. These nodes are either I – concatenated or S-Concatenated sequence node to itself.

Step 2: Concatenations

Using concatenations, we generate the node's children's utilities by concatenating the corresponding items. Suppose we have a k-sequence q, we call the operation of appending a new item to the end of q to form (k+1)-sequence concatenation. If the size of q does not change; we call the operation I-Concatenation. Otherwise, if the size increases by one, we call it S-Concatenation. For example, <ea>'s I-Concatenated and S-Concatenated with b result in <e(ab)> and <eab>, respectively.

Step 3: Width Pruning

The width pruning strategy helps to avoid constructing unpromising items/ children into the LQS- tree using the Sequence Weighted Utility (SWU) of the sequence. The item will be considered promising if after concatenating an item (say i) to sequence (say q) forms a new sequence (say q`) has:

$$SWU(q') \geq \text{threshold minimum utility}$$

Step 4: Depth Pruning

The Depth Pruning stops the USpan algorithm from going deeper in the LQS- Tree. When the gap is so large that even if all the utilities of the remaining q-items are counted into the utility of the sequence, the cumulative utility still cannot satisfy minimum threshold utility required. Here, by using the depth pruning strategy we can backtrack USpan instead of waiting to go deeper and returning with nothing.

3. Sequential Pattern mining (SPM) for the Clickstream database.

The output of SCDB for the clickstream database contains the historical sequence dataset of clicks. We can mine the clickstream dataset using the PrefixSpan algorithm explained in **section 2.1.2** as the number database scans in this algorithm are less compared to Apriori based algorithms.

Input: historic sequence database of clicks and minimum support.

Output: frequent sequential pattern rules.

3.6 Click Purchase Similarity (CPS) Module

Inspired by the idea of Chen13Rec (Chen & Su, 2013), we introduce CPS (Click purchase similarity) which takes the frequency and position of items in click and purchase sequences of the user into consideration to calculate the similarity. To compute the CP similarity between click and purchase sequences of each session, 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) (Paterson, & Vlado, 1994) (Bergroth, Hakonen & Raita, 2000).

(**LCSR**) (X, Y) = $\boldsymbol{LCS}$ ($\boldsymbol{X}$, $\boldsymbol{Y}$) $\boldsymbol{max}$(|$\boldsymbol{X}$| ,|$\boldsymbol{Y}$|).

In our case, **X** represents click sequence and Y represents purchase sequence and LCS is defined in **Equation 3.1**.

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

Equation 3. 1 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 3.2** to click and purchase vectors (Sidorov, Gelbukh, Gómez & Pinto, 2014).

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

Equation 3. 2 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 sequences similarity and frequency similarity. We can explain these two steps using the both clickstream sequence s X = <a, b, a, d, c, e> and Y = < a, b, d> using following steps:

i. Compute the longest common subsequences, LCS (X, Y) between click sequences. For example, LCS (<a, b, a, d, c, e>, <a, b, d>) is 3 because of common subsequence (a, b, d).
ii. Find the maximum number of items occurring in click sequences as Max (X, Y). In our case, Max (X, Y) is 6.
iii. Compute sequences similarity of click (X) and purchase (Y) sequence as LCS (X, Y)/Max (X, Y) =3/6=0.5.
iv. 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 sequence (X) is: [(a):2, (b):1, (c): 1, (d):1, (e): 1]. Similarly, frequency count of click sequence (Y) is: [(a):1, (b):1, (d):1].
v. Then, use the Cosine similarity function in **Equation 3.2** to get the frequency similarity between click sequence (X) and (Y) as Cosine (X, Y). In our case, Cosine (<2,1,1,1,1,0>, <1,1,0,1,0,0>) =0.81.

vi. The Clickstream Similarity of user click sequence CSSM (X, Y) = 0.8*0.5 + 0.2*0.81 = 0.562, where α =0.8 and β=0.2.

vii. This CPS (X, Y) can be used as weight or probability that user *u* will purchase the entire sequence as shown in Table 3.24.

3.7 Weighted High Utility Occupancy Matrix (WHUOM)

Weighted High Utility Occupancy Matrix (WHUOM) module takes weightage assigned by the CPS module as input and generate frequent items with weight (u's rating of item i in the matrices referred to as r_{ui}) present in purchased patterns based on the utility occupancy. So major steps of WHUPPM are:

User ID	Purchase Sequence	CPS Similarity
1	{ a b d }	0.562
2	{e a b c d}	0.64
3	{ a (b d e) a}	0.58
4	{ ab (ae) a}	0.57
5	{d f a }	1
6	?	?

Table3. 24 Weighted purchase pattern

The utility occupancy (Gan, Lin, Fournier-Viger, Chao, & Yu, 2020) of an itemset say, X in a supporting transaction say, Tq is denoted as uo(X, Tq), and defined as the ratio of the utility of X in that transaction divided by the total utility of that transaction:

$$U_o\,(X, T_q) = \frac{u\,(X,Tq)}{tu\,(Tq)}$$

Equation 3. 3 Utility Occupancy

The utility occupancies (**Equation 3.3**) of (a) in Table 3.24 are calculated as: uo(a) = (uo(a; T1) + uo(a; T2)+ uo(a; T3) + uo(a; T4) + uo(a; T5))/5 = (0.562 + 0.64 + 0.58 + 0.57 + 1)/5 = 0.67. Similarly, calculate for b, c, d, e and f.

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.

i. Count support of item: Count the occurrence of items presented in weighted purchase pattern. For example, {support (a): 5, support (b): 4, support (c): 1, support (d): 4, support (e): 3, support (f): 1}

ii. Calculate the weight of individual item: Compute weight of individual item from weighted purchase pattern (Table 3.1.4) using **Equation 3.4**.

$$\text{Ritem} = \frac{\sum_{i=1}^{n} CPS \in itemi}{count\ of\ transactions\ with\ itemi}$$

Equation 3. 4: Formula to compute weight in WHUOM

For example, $Ritem\ 1 = 0.562 + 0.64 + 0.58 + 0.57 + 1/\ 5 = 0.68$.

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

3.8 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**).

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

Equation 3. 5 Unit normalization function

3.9 Comparison between HSPRec19 and proposed HUSRec Systems

Let's compare the HSPRec19 and proposed HUSRec systems using the same input click stream dataset and purchase dataset.

Input historical clickstream dataset: Let's consider **Table 3.25** as input for both the systems, some samples have been taken from the clicks dataset which has attributes such as:

Session Id – The id of the session is represented as an integer number, which may have one or many clicks.

Timestamp- The time when the click occurred. Format: "***YYYY-MM-DDThh:mm:ss.SSSZ****".Item Id/ Product* - The unique identifier for the item that has been clicked, represented as an integer number.

Session Id	Click Sequence	Timestamp
S1	apples	2014-04-07T10:51:09.277Z
S1	bread	2014-04-07T10:54:09.868Z
S1	apples	2014-04-07T10:54:46.998Z
S1	Milk	2014-04-07T10:57:06.808Z
S1	Cheese	2014-04-07T10:58:00.306Z

S1	Eggs	2014-04-07T10:59:56.112Z
S2	bread	2014-04-07T13:56:37.614Z
S2	Honey	2014-04-07T13:57:19.373Z
S2	apples	2014-04-07T13:58:37.446Z
S2	Cheese	2014-04-07T13:59:50.710Z
S2	Milk	2014-04-07T13:00:38.247Z
S3	apples	2014-04-07T14:02:36.889Z
S3	bread	2014-04-02T13:17:46.940Z
S3	apples	2014-04-02T13:26:02.515Z
S3	Eggs	2014-04-02T13:30:12.318Z
S3	Cheese	2014-04-02T13:31:43.567Z
S3	Milk	2014-04-02T13:33:12.871Z
S4	Eggs	2014-04-03T10:44:35.672Z
S4	apples	2014-04-03T10:45:01.674Z
S4	bread	2014-04-03T10:45:29.873Z
S4	Eggs	2014-04-03T10:46:12.162Z
S4	Milk	2014-04-03T10:46:57.355Z
S4	apples	2014-04-03T10:53:22.572Z
S5	Honey	2014-04-06T14:50:13.638Z
S5	Milk	2014-04-06T14:52:54.363Z
S5	apples	2014-04-06T14:53:18.268Z
S6	apples	2014-04-07T09:01:28.552Z
S6	bread	2014-04-07T09:03:39.903Z
S6	Milk	2014-04-07T09:04:00.598Z
S6	Eggs	2014-04-07T09:06:09.115Z
S6	apples	2014-04-07T09:07:42.212Z
S6	bread	2014-04-07T09:21:10.012Z

Table3. 25 Historical Clicks dataset

Input historical purchase dataset: Let's consider **Table 3.26** as input for both the systems, some samples have been taken from the purchase dataset which has attributes such as

Session Id- The id of the session is represented as an integer number, which may have one or many clicks.

Timestamp - the time when the buy occurred. Format: "***YYYY-MM-DDThh:mm:ss.SSSZ***".

Item ID – The unique identifier of item that has been bought represented as an integer number.

Price – The price of the item could be represented as an integer number as shown in Table 3.27.

Quantity – The quantity of item in this buying represented as an integer number.

SID	Product/ Item ID	Quantity	Timestamp	Price
S1	apples	3	2014-04-07T10:51:09.277Z	5
S1	bread	6	2014-04-07T10:54:39.868Z	7
S1	Milk	2	2014-04-07T10:55:34.998Z	10
S2	Eggs	3	2014-04-07T13:07:06.808Z	6
S2	apples	2	2014-04-07T13:18:00.306Z	5
S2	bread	5	2014-04-07T13:29:56.112Z	7
S2	Cheese	4	2014-04-07T13:36:37.614Z	3
S2	Milk	1	2014-04-07T13:57:19.373Z	10
S3	apples	1	2014-04-07T14:08:37.446Z	5
S3	bread	3	2014-04-07T14:29:50.710Z	7
S3	Milk	1	2014-04-07T14:30:38.247Z	10
S3	Eggs	4	2014-04-07T14:42:36.889Z	6
S3	apples	2	2014-04-02T15:17:46.940Z	5
S4	apples	2	2014-04-02T13:26:02.515Z	5
S4	bread	7	2014-04-02T13:30:12.318Z	7
S4	apples	6	2014-04-02T13:31:43.567Z	5
S4	Eggs	5	2014-04-02T13:33:12.871Z	6
S4	apples	3	2014-04-03T10:44:35.672Z	5
S5	Milk	1	2014-04-03T10:45:01.674Z	10
S5	Honey	2	2014-04-03T10:45:29.873Z	8
S5	apples	5	2014-04-03T10:46:12.162Z	5

Table3. 26 Historical Purchase Dataset

Item	Price per unit
Apple	5
Bread	7
Cheese	3
Milk	10
Eggs	6
Honey	8

Table3. 27 Table containing prices per unit for each item.

Steps of HSPRec19 System:

Step 1: Convert the historical purchase into the sequential database using the SHOD algorithm.

SID	Purchase sequences
S1	< Apples, Bread, Milk>
S2	< Eggs, Apples, Bread, Cheese, Milk >
S3	< Apples, [Bread, Milk , Eggs], Apples >
S4	< Apples, Bread, [Apples, Eggs], Apples >
S5	< Milk, Honey, Apples >

Table3. 28 Daily Purchase Sequential Database created from the historical transaction data

Step 2: Create the consequential bond between sequential clicks and purchases data.

SID	Click Sequence	Purchase Sequence
S1	{Apples, Bread, Apples, Milk, Cheese, Eggs}	{ Apples, Bread, Milk }
S2	{ Bread, Honey, Apples, Cheese, Milk }	{ Eggs, Apples, Bread, Cheese, Milk }
S3	{ Apples, Bread, Apples, Eggs, Cheese, Milk }	{ Apples ,(Bread, Milk, Eggs) Apples }
S4	{ Eggs, Apples, Bread, Eggs, Milk, Apples ,}	{Apples, Bread, (Apples , Eggs,) Apples }
S5	{Honey, Milk, Apples }	{ Milk , Honey, Apples }
S6	{ Apples, Bread, Milk, Eggs, Apples, Bread}	?

Table3. 30 Consequential Bond for HSPRec19 system.

Steps of proposed HUSRec System:

Step 1: Convert the historical purchase into the sequential database using the HUSDBG algorithm.

SID	Transaction sequence	Sequence Utility
S1	{Apples (3) Bread (6) Milk (2)}	77
S2	{Eggs (2) Apples (2) Bread (5) Cheese (4) Milk (1)}	79
S3	{Apples (1) [Bread (3) Milk (1) Eggs (4)] Apples (2)}	70
S4	{Apples (2) Bread (7) [Apples (6) Eggs (5)] Apples (3)}	134
S5	{Milk (1) Honey (2) Apples (5)}	36
S6	?	?

Table3. 29 High utility Sequential purchase database with sequence utility

Step 2: Create the consequential bond between the high utility sequential purchase data and sequential click data.

Sid	Click Sequence	Purchase Sequence
S1	{Apples, Bread, Apples, Milk, Cheese, Eggs}	{Apples (3), Bread (6), Milk (2)}
S2	{Bread, Honey, Apples, Cheese, Milk}	{Eggs (2), Apples (2), Bread (5), Cheese (4), Milk (1)}
S3	{Apples, Bread, Apples, Eggs, Cheese, Milk}	{Apples (1), [Bread (3), Milk (1) Eggs (4)], Apples (2)}
S4	{Eggs, Apples, Bread, Eggs, Milk, Apples}	{Apples (2), [Bread (7), Apples (6)], Eggs (5), Apples (3)}
S5	{Honey, Milk, Apples}	{Milk (1), Honey (2), Apples (5)}
S6	{Apples, Bread, Milk, Eggs, Apples, Bread}	?

Table3. 31 Consequential Bond for proposed HUSRec system.

Step 3: Create user-item frequency matrix from the purchase information.

Item/SID	Apples	Bread	Cheese	Milk	Eggs	Honey
S1	1	1	?	1	?	?
S2	1	1	1	1	1	?
S3	2	1	?	1	1	?
S4	3	1	?	?	1	?
S5	1	?	?	1	?	1
S6	?	?	?	?	?	?

Table3. 32 user-item frequency matrix

Step 4: Pass the sequential purchase data to sequential pattern mining (SPM).

Rule No	Sequential Rules
1	Apples → Milk
2	Apple, Bread → Milk
3	Apple, Bread → Eggs
4	Apples → Bread

Table3. 35 Sequential rules from SPM.

Step 5: update user-item matrix with sequential rules.

Item/SID	Apples	Bread	Cheese	Milk	Eggs	Honey
S1	1	1	?	1	1	?
S2	1	1	1	1	1	?
S3	2	1	?	1	1	?
S4	3	1	?	1	1	?
S5	1	1	?	1	?	1
S6	?	?	?	?	?	?

Table3. 37 Updated user-item matrix with SPM rules

Step 3: Create user-item frequency matrix from the purchase information.

Item/SID	Apples	Bread	Cheese	Milk	Eggs	Honey
S1	3	6	?	2	?	?
S2	2	5	4	1	2	?
S3	3	3	?	1	4	?
S4	11	7	?	?	5	?
S5	5	?	?	1	?	2
S6	?	?	?	?	?	?

Table3. 34 User-item frequency matrix

Step 4: Pass the HUPSDB to High utility sequential pattern mining (HUSPM).

Rule ID	Sequential rule	Utility
R1	{Bread} → {Milk}	226
R2	{Apples} → {Bread}	360
R3	{Apples, Bread} → {Milk}	226
R4	{Bread, Apples} → {Eggs}	204
R5	{Eggs} → {Apples}	283

Table3. 36 High Utility Sequential rules mined using USpan algorithm.

Step 5: update user-item matrix with High utility sequential rules.

Item/SID	Apples	Bread	Cheese	Milk	Eggs	Honey
S1	3	6	?	2	1	?
S2	2	5	4	1	2	?
S3	3	3	?	1	4	?
S4	11	7	?	1	5	?
S5	5	1	?	1	1	2
S6	?	?	?	?	?	?

Table3. 38 Updated user-item matrix with HUSR.

Step 6: Recommend items for the sessions where no purchase has happened with sequential rules mined from sequence clickstream data.

Item/SID	Apples	Bread	Cheese	Milk	Eggs	Honey
S1	1	1	?	1	1	?
S2	1	1	1	1	1	?
S3	2	1	?	1	1	?
S4	3	1	?	1	1	?
S5	1	1	?	1	?	1
S6	1	1	?	1	?	?

Table3. 39 matrix updated with recommended items in Session 6.

Step7: Calculate CPS for the updated user-item matrices from HSPRec19 systems.

SID	Purchase sequence	CPS Similarity
S1	< Apples, Bread, Milk>	0.562
S2	< Eggs, Apples, Bread, Cheese, Milk >	0.64
S3	< Apples, [Bread, Milk , Eggs], Apples >	0.58
S4	< Apples, Bread, [Apples, Eggs], Apples >	0.57
S5	< Milk, Honey, Apples >	.72
S6	?	?

Table3. 41 CPS values assigned to purchase sequences in HSPRec19.

Step 8: Assign weights for each item using Weighted Frequent pattern module.

Item/SID	Apples	Bread	Cheese	Milk	Eggs	Honey
S1	1	1	?	1	1	?
S2	1	1	1	1	1	?
S3	2	1	?	1	1	?
S4	3	1	?	1	1	?
S5	1	1	?	1	?	1
S6	0.60	0.588	0.62	0.69	0.596	0.72

Table3. 43 weights assigned to each item in the database using the CPS values from Table 3.41.

Step 6: Recommend items for the sessions where no purchase has happened with sequential rules mined from sequence clickstream data.

Item/SID	Apples	Bread	Cheese	Milk	Eggs	Honey
S1	3	6	?	2	1	?
S2	2	5	4	1	2	?
S3	3	3	?	1	4	?
S4	11	7	?	1	5	?
S5	5	1	?	1	1	2
S6	1	1	?	1	?	?

Table3. 40 matrix updated with recommended items in Session 6.

Step7: Calculate CPS for the updated user-item matrices from HUSRec systems.

SID	Purchase Sequence	CPS Similarity
S1	{Apples (3), Bread (6), Milk (2)}	0.54
S2	{Eggs (2), Apples (2), Bread (5), Cheese (4), Milk (1)}	0.63
S3	{Apples (1), [Bread (3), Milk (1) Eggs (4)], Apples (2)}	0.56
S4	{Apples (2), [Bread (7), Apples (6)], Eggs (5), Apples (3)}	0.70
S5	{Milk (1), Honey (2), Apples (5)}	0.69
S6	?	?

Table3. 42 CPS values assigned to purchase sequences in HUSRec system.

Step 8: Assign weights for each item using Weighted High Utility Occupancy Matrix (WHUOM) module.

Item/SID	Apples	Bread	Cheese	Milk	Eggs	Honey
S1	3	6	?	2	1	?
S2	2	5	4	1	2	?
S3	3	3	?	1	4	?
S4	11	7	?	1	5	?
S5	5	1	?	1	1	2
S6	0.62	0.60	0.63	0.60	0.63	0.69

Table3. 44 weights assigned to each item in the database using the CPS values from Table 3.42.

Step 9: Normalize the user-item matrix.

Item/SID	Apples	Bread	Cheese	Milk	Eggs	Honey
S1	0.16	0.16	?	0.16	0.16	?
S2	0.16	0.16	0.16	0.16	0.16	?
S3	0.32	0.16	?	0.16	0.16	?
S4	0.49	0.16	?	0.16	0.16	?
S5	0.16	0.16	?	0.16	?	0.32
S6	0.09	0.09	0.10	0.11	0.09	0.11

Table3. 45 Normalized matrix from HSPRec19 system.

Step 10: For collaborative filtering, similar users to S6 are found using cosine similarity. The top two similar users are S2 and S5 with similarity 0.88 and 0.79.

Step 11: Calculate the prediction rating of each item in Session 6, the item Apples and Bread has greater prediction rating. Therefore, they will be suggested to the customer.

Step 12: Considering the total items bought in the Session 6 are Apples and Bread, we calculate the profit earned by selling them per unit.
For example, cost price of Apples is $5 and of Bread is $7, the money earned will be $12.

Step 13: Compare the accuracy of the recommended products for Session 6, the HSPRec19 system recommended Apples and Bread but the actual purchase has Apples, Bread and Eggs. For this single session, HSPRec19 scored 75% accuracy.

Step 9: Normalize the user-item matrix.

Item/SID	Apples	Bread	Cheese	Milk	Eggs	Honey
S1	0.15	0.31	?	0.10	0.052	?
S2	0.10	0.26	0.20	0.052	0.10	?
S3	0.15	0.15	?	0.052	0.20	?
S4	0.57	0.36	?	0.052	0.26	?
S5	0.26	0.052	?	0.052	0.052	2
S6	0.032	0.031	0.03	0.031	0.032	0.036

Table3. 46 Normalized matrix from HUSRec system.

Step 10: For collaborative filtering, similar users to S6 are found using cosine similarity. The top two similar users are S2 and S3 with similarity 0.78 and 0.75.

Step 11: Calculate the prediction rating of each item in Session 6, the item Apples, Bread and Eggs has greater prediction rating. Therefore, they will be suggested to the customer.

Step 12: Considering the total items bought in the Session 6 are Apples, Bread and Eggs, we calculate the profit earned by selling them per unit.
For example, cost price of Apples ($5), Bread ($7) and Eggs ($6), the money earned will be $18.

Step 13: Compare the accuracy of the recommended products for Session 6, the HUSRec system recommended Apples, Bread and Eggs and the actual purchase has Apples, Bread and Eggs. For this single session, proposed HUSRec system scored 100% accuracy.

CHAPTER 4 EXPERIMENT AND EVALUATION METRICS

The normalized and enhanced user-item frequency matrix is passed to the collaborative filtering as input. We have used user-based collaborative filtering to generate the recommendations, firstly, the historical data is converted into the user-item frequency matrix using the three systems (HPCRec18, HSPRec19 and HUSRec) and then apply the user cosine similarity technique to find the top – 10 similar users and recommend the items from the most similar user. After applying the cosine similarity for 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 are using three different evaluation parameters (a) mean absolute error (MAE) (b) precision and (c) performance analysis of the proposed HUSRec system.

4.1 Dataset Selection

We use the dataset provided by YOOCHOOSE GmbH for ACM RecSys 2015 (Ben-Shimon, et al., 2015), which is from an online retailer in Europe. There are two files recording 33,040,175 clicks and 1,177,769 purchase events respectively; all the events happened in 9,512,786 unique sessions, the total amount of product is 52,739 belonging to 339 categories.

Data Format of Purchase data: {***SessionId***, ***Timestamp***, ***ItemID***, ***Price***, ***Quantity***}

{“SessionID”: “140806”, “TimeStamp”: “2014-04-07T09:22:28.280Z”, “ItemID”: “214578823”, “Price”: “1046”, “Quantity”: “1”}

{“SessionID”: “11”, “TimeStamp”: “2014-04-03T11:04:11.417Z”, “ItemID”: “214821371”, “Price”: “1046”, “Quantity”: “1”}

Where, ***SessionId*** is the unique id of the session is represented as an integer number, which may have one or many clicks; ***Timestamp*** is the time when the buy occurred of format: “***YYYY-MM-DDThh:mm:ss.SSSZ****”.;* ***ItemID*** is the unique identifier of item that has been bought represented as an integer number; ***Price*** is the price of the item, represented as an integer number as shown in Table 3.27 and *Quantity* is the quantity of item in this buying represented as an integer number.

Data Format of Clicks data: {***SessionId***, ***Timestamp***, ***ItemID***, ***Price***, ***Quantity***}

{“SessionID”: “1”, “TimeStamp”: “2014-04-07T10:57:00.306Z”, “ItemID”: “214577561”, “Category”: ”0”}

{“SessionID”: “2”, “TimeStamp”: “2014-04-07T13:56:37.614Z”, “ItemID”: “214662742”, “Category”: ”0”}

Where, *Category* represents the context of the click. The value "S" indicates a special offer, "0" indicates a missing value, a number between 1 to 12 indicates a real category identifier,any other number indicates a brand. E.g. if an item has been clicked in the context of a promotion or special offer then the value will be "S", if the context was a brand i.e BOSCH, then the value will be an 8-10 digits number. If the item has been clicked under regular category, i.e. sport, then the value will be a number between 1 to 12. To implement the proposed HUSRec system, we have used the following tools and infrastructure:

i. System Configuration: Windows 10, with 16 GB RAM and 64-bit Operating System, x64 based processor.
ii. Integrated Development Environment, such as Eclipse Java EE IDE for Web Developers and PyCharm.
iii. Programming Languages: Java SE Development Kit (1.8 version) and Python (3.7.0)
iv. Project manage tool: Apache Maven (3.5.3 version)

4.2 Dataset Evaluation Metrics

We used our historical dataset in user-based collaborative filtering to evaluate its performance with respect to MAE, precision, and graphs of comparisons of number of recommendations by each system and execution time taken by each system. 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. Let's understand the evaluation metrics in terms of recommendation systems.

	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. 1 Confusion matrix in terms of recommendation system.

In the Table 4.1, the relevant items purchased are considered recommended, the items are denoted by:

TP (True positive) if item was both recommended by system and purchased by customer.

FP (False Positive) if item was recommended to the customer but he didn't purchase it.

TN (True Negative) if item was purchased by the customer but not recommended to him in the suggestions.

FN (False Negative) if item was neither recommended or purchased by the customer.

Mean absolute error (MAE):

Mean absolute error is a metric used to compute the average of all the absolute value differences between the true and the predicted rating (Herlocker, Konstan, Terveen & Riedl, 2004).

$$MAE=\frac{\Sigma|\text{actual\_rating}-\text{predicted\_rating}|}{n}$$

Equation 4. 1 Mean absolute error formula

Thus, higher mean absolute errors mean, less efficient for accurate rating predication and lower mean absolute errors means highly efficient for accurate rating prediction.

Precision:

Precision is a description of a level of measurement that yields consistent results when repeated (Menditto, Patriarca & Magnusson, 2006). 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 in but recommended to him, then precision is defined as:

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

Equation 4. 2 Precision formula

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.

4.2.2 Result evaluation and Analysis

Since, we are working on the implicit information gathered from the historical e-commerce system. Firstly, we implemented the HPCRec18 system (Xiao & Ezeife, 2018), which forms the consequential bond between the click and purchase historic data and the forms the user-item matrix

from the purchased frequency of items is passed for collaborative filtering, then we have implemented HSPRec19 system (Bhatta, Ezeife & Butt, 2019), which found the sequential patterns from purchase which could lead to purchase of the item and improves the user-item frequency matrix before passing to the collaborative filtering. Then, we implemented HUSRec which recommends items based on the utility value of the item (product of quantity and price of the item) and sequential rules from the click history which diversifies the recommendations. The proposed HUSRec system provides with more recommendations as compared to the HPCRec18 (Xiao & Ezeife, 2018) and HSPRec19 (Bhatta, Ezeife & Butt, 2019).

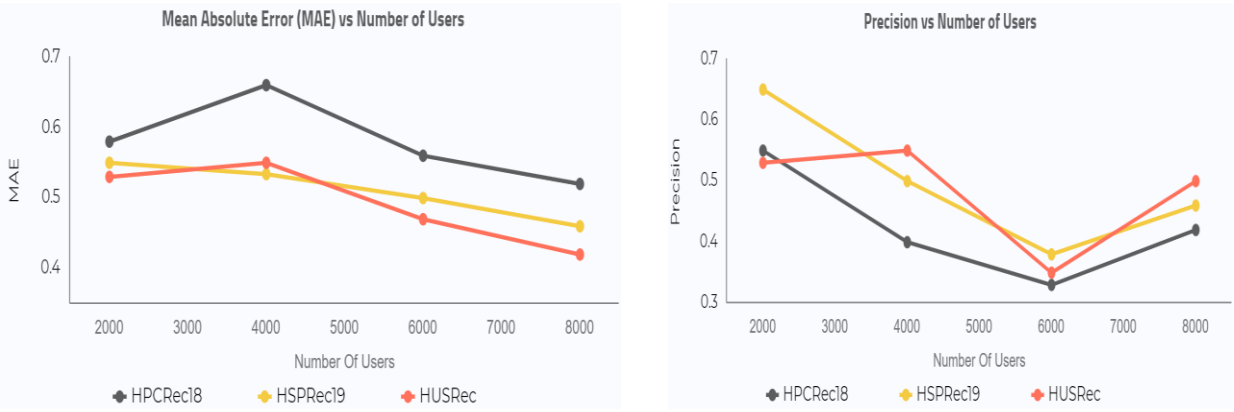

Figure 4. 1 Mean absolute error and Precision in user-based Collaborative filtering.

The Figure 4.1 contains graph of Mean absolute error and Precision for the HPCRec18, HSPRec19 and proposed HUSRec system, from these graphs we can study how the behavior of each system changes with increase in number of users. We can clearly see that Mean absolute error (MAE) decreases with increase in number of Users and in Precision graph we can see, however there is a slight difference in all the three systems but proposed HUSRec has performed well among all the systems in terms of MAE and precision. The least mean absolute error and consistent precision indicates that the accuracy of proposed HUSRec is more than the HSPRec19 and HPCRec18 systems.

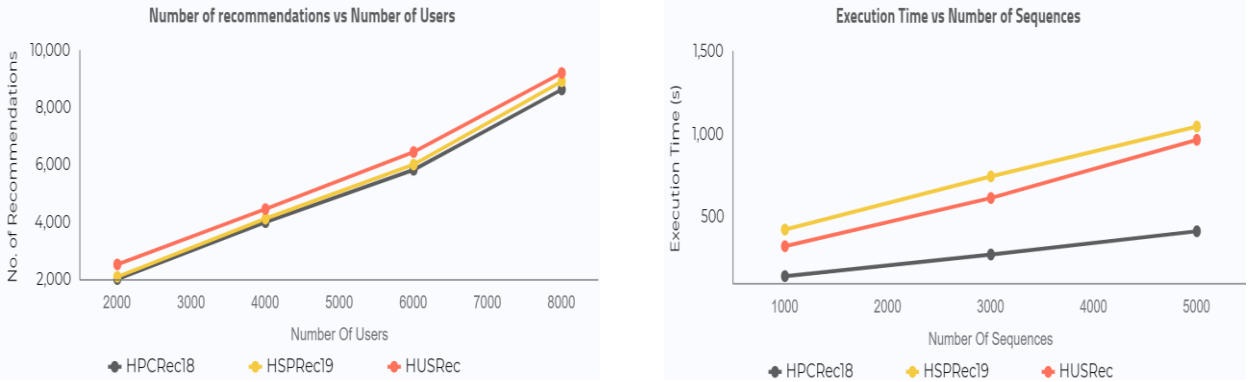

Figure 4. 2 Comparison of no. of recommendation and behavior of processing time for each system.

In Figure 4.2, we can see that there is a linear relationship of number of recommendations with number of users and execution time with number of sequences. The execution time is taken most by the HSPRec19 because of GSP mining algorithm which is based on candidate generation and HPCRec18 takes the least as there is no association rule mining performed among the click and purchase data. But when we talk about the memory usage, it's the HUSRec which occupies the most memory storage because of USpan high utility sequential pattern mining module stores the LQS-tree for the whole dataset.

4.2.3 Accuracy evaluation using precision

Recommendation system	Top-N	Neighbors	Number of users	Recommendation No	Precision	Relevant item
HPCRec18 (Xiao & Ezeife, 2018)	10	10	2000 4000 6000 8000	2050 4032 5857 8655	0.62 0.39 0.35 0.38	127145789 157267813 204998726 328886422
HSPRec19 (Bhatta, Ezeife & Butt, 2019)	10	10	2000 4000 6000 8000	2130 4156 6039 8938	0.65 0.49 0.38 0.45	139487262 20366267 22946728 40223456

Proposed HUSRec	10	10	2000	2560	0.742	214821290
			4000	4589	0.584	214716975
			6000	6477	0.470	214826803
			8000	9231	0.462	214546102

Table 4. 2 Precision evaluation with respect to different number of users

4.3 Complexity Analysis

4.3.1 Time complexity analysis of proposed HUSRec algorithm

The proposed HUSRec is composed of several modules HUSDBG (High Utility Sequential Database Genenrator), SPR (Sequential Pattern Rule), CPS (Click Purchase Similarity), WHUOM (Weighted High Utility Occupancy Miner), and Matrix normalization; thus, we are going to compare the time complexity of HSPRec19 and proposed HUSRec with respect to specified modules. Assume we have a dataset (say *D*) with the number of items in it (N) and has *S* number of sequences generated from database where the length of the longest sequence is *n*.

HSPRec19 system	**Proposed HUSrec system**
1. Time complexity analysis of SHOD algorithm. The time complexity of SHOD algorithm is O (N log N), where N is the number of total items in the database.	**Time complexity analysis of HUSDBG algorithm** The time complexity of HUSDBG module is O (N log N).
2.Time Complexity for Pattern Rule Mining. HSPRec19 system uses GSP algorithm for sequential pattern mining for both sequential purchase and click stream database. The I/O complexity of the GSP algorithm is O (n. N), where n is the length of the longest sequence in the database and N is the total number of items/ records in the sequential database.	**Time Complexity for Pattern Rule Mining.** HUSRec system has a high utility sequential pattern mining module (USpan algorithm) for sequential purchase database and the PrefixSpan algorithm for sequential click stream data. According to our system PrefixSpan has O ($S.n^2$) time and space complexity, where S is the number of sequences in database, and n is the length of longest sequence. The time complexity of

USpan is O(S Log N), as it traverses each sequence and create LQS-tree for them, where S is the number of sequences and N is the total number of unique items in the database.

CHAPTER 5 CONCLUSION AND FUTURE WORK

In this thesis, we proposed a High Utility Sequential Recommendation Systems (HUSRec) which converts the historic purchase dataset to the High Utility Purchase Sequential Database (HUPSDB) in order to mine the high utility sequential pattern rules that will recommend items which are not only relevant for the customers but will also bring sales profits to the sellers and increase in revenue generations. To conclude, the proposed HUSRec system has increased the number of recommendations offered to the customers as compared to the HSPRec19 system even there is improvement in the accuracy of these recommendations. The recommended products generated by the proposed HUSRec system have utility values more than the minimum threshold sequential utility which ensures that the proposed HUSRec system suggests products that could help product sellers to increase their revenue generation by making profit sales. We have compared the HPCRec18 and HSPRec19 systems on the same dataset with proposed HUSRec system. Furthermore, we have evaluated all three systems on Precision and Mean Absolute Error (MAE) with user-based collaborative filtering. The Precision and MAE shows that proposed HUSRec is more improved system than the HPCRec18 and HSPRec19 systems. Even the number of recommendations suggested are more in the proposed HUSRec system. Therefore, the proposed HUSRec gives better results with a high utility sequential pattern mining based e-commerce recommendations.

Below are some interesting extensions of this study and some avenues to explore for future works:

1. High Utility mining is still an active area of research, we have used USpan algorithm for mining high utility sequential pattern mining module which works efficiently for large- scale datasets with even low minimum utility value also but the pruning strategies used to find the patterns can be improved.
2. Multiple large data sources can be incorporated based on the high utility sequential pattern mining algorithms which have different data schemas and also make recommendations based on the overall dataset.
3. The e-commerce datasets are not limited to the purchase and clickstream information, there are other attributes available in the dataset such as discount can be used to analyze how they can impact the user-item frequency matrix which can impact the business decision making. For example, during the festive season sales, there are big discounts on the expensive gadgets but normally throughout the year they can’t be considered as the frequently bought items, so

discounts can increase the probability of item being purchased in future and can help to make business decisions such as how to control the inventory on festive seasons.

REFERENCES

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

Agrawal, R. & Srikant, R., (1994). Fast algorithms for mining association rules. *Proc. 20th int. conf. very large data bases, VLDB, 1215*, pp. 487-499.

Agrawal, R., & Srikant, R. (1995). Mining sequential patterns: Generalizations and Performance Improvements. *Proceedings of the Eleventh International Conference on Data Engineering*.

Ahmed, C. F. (2010). A Novel Approach for Mining High-Utility Sequential Patterns in Sequence Databases. *ETRI Journal*, *32*(5), 676–686.

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

Ayres, J., Flannick, J., Gehrke, J., & Yiu, T. (2002). Sequential PAttern mining using a bitmap representation. *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD 02*.

Ayres, J., Gehrke, J., Yiu, T. and Flannick, J. (2002) Sequential Pattern Mining Using a Bitmap Representation. In: Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, Edmonton.

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).

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.

Bhatta, R., Ezeife, C. I., Butt, M. (2019). Mining Sequential Patterns of Historical Purchases for E-commerce Recommendation. *Proceedings of the 21st International Conference on Big Data Analytics and Knowledge Discovery - DaWaK 2019*, Austria, Aug. 26 – 29, pages 57-72.

Bobadilla, J., Ortega, F., Hernando, A., & Gutiérrez, A. (2013). Recommender systems survey. *Knowledge-Based Systems, 46*, 109-132.

Chen, L., & Su, Q. (2013). Discovering user's interest at E-commerce site using clickstream data. *Service systems and service management (ICSSSM), 2013 10th international conference on*, (pp. 124-129).

Chen, M.-S., Han, J., & Yu, P. (1996). Data mining: an overview from a database perspective. *IEEE Transactions on Knowledge and Data Engineering*, *8*(6), 866–883.

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.

Fan, J., Pan, W., & Jiang, L. (2014). An improved collaborative filtering algorithm combining content-based algorithm and user activity. *Big Data and Smart Computing (BIGCOMP), 2014 International Conference on*, (pp. 88-91).

Gan, W., Lin, J. C., Fournier-Viger, P., Chao, H., & Yu, P. S. (2020). HUOPM: High-Utility Occupancy Pattern Mining. *IEEE Transactions on Cybernetics*, *50*(3), 1195-1208.

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

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

Han, J., Pei, J., Mortazavi-Asl, B., Chen, Q., Dayal, U., & Hsu, M.-C. (2000). FreeSpan. *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining – KDD.*

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.

Hipp, J., Güntzer, U., & Nakhaeizadeh, G. (2000). Algorithms for association rule mining—a general survey and comparison. *ACM sigkdd explorations newsletter*, *2*, 58-64.

Jain, A. K., & Dubes, R. C. (1988, January). Algorithms for clustering data. *Prentice Hall Advance Reference Series*.

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.

Kotsiantis, S. B., Zaharakis, I. D., & Pintelas, P. E. (2006). Machine learning: A review of classification and combining techniques. *Artificial Intelligence Review, 26*(3), 159-190.

Kumar, N. P., & Fan, Z. (2015). Hybrid User-Item Based Collaborative Filtering. *Procedia Computer Science, 60*, 1453-1461.

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.

Liu, M., & Qu, J. (2012). Mining high utility itemsets without candidate generation. *Proceedings of the 21st ACM International Conference on Information and Knowledge Management – CIKM '12*.

Mabroukeh, N. R., & Ezeife, C. I. (2010). A Taxonomy of Sequential Pattern Mining Algorithms. *ACM Computing Surveys*, *43*(1), 1–41.

Melville, P., Mooney, R. J. & Nagarajan, R. (2002). Content-boosted Collaborative Filtering for Improved Recommendations. *Eighteenth National Conference on Artificial Intelligence*, 187-192.

Menditto, A., Patriarca, M., & Magnusson, B. (2006). Understanding the meaning of accuracy, trueness and precision. *Accreditation and Quality Assurance, 12*(1), 45-47.

Michael D. Ekstrand, John T. Riedl and Joseph A. Konstan (2011), "Collaborative Filtering Recommender Systems", Foundations and Trends® in Human–Computer Interaction: Vol. 4: No. 2, pp 81-173.

Moe, W. W. (2003). Buying, searching, or browsing: Differentiating between online shoppers using in-store navigational clickstream. *Journal of consumer psychology, 13*, 29-39.

Moe, W. W., & Fader, P. S. (2004). Dynamic conversion behavior at e-commerce sites. *Management Science, 50*, 326-335.

Mooney, R. J., and Roy, L. (1999) Content-based book recommending using learning for text

categorization. In ACM SIGIR '99 Workshop on Recommender Systems: Algorithms and Evaluation, Berkeley, CA, 1999.

Musto, C. (2010). Enhanced vector space models for content-based recommender systems. *Proceedings of the Fourth ACM Conference on Recommender Systems - RecSys '10*.

Paterson, M., & Vlado, D. (1994). Longest common subsequences. *International Symposium on Mathematical Foundations of Computer Science*, (pp. 127-142).

Pazzani, M., Muramatsu, J., & Billsus, D. (1996, August). Syskill & Webert: Identifying Interesting Web Sites. *Proceedings of the Thirteenth National Conference on Artificial Intelligence, 1*, 54-61.

Pei, J., Han, J., Mortazavi-Asl, B., & Zhu, H. (2000). Mining Access Patterns Efficiently from Web Logs. *Knowledge Discovery and Data Mining. Current Issues and New Applications Lecture Notes in Computer Science*, 396-407.

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.

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

Sarwar, B., Karypis, G., Konstan, J., & Reidl, J. (2001). Item-based collaborative filtering recommendation algorithms. *Proceedings of the Tenth International Conference on World Wide Web - WWW '01*, (pp. 285-295).

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

Sidorov, G., Gelbukh, A., Gómez-Adorno, H., & Pinto, D. (2014). Soft Similarity and Soft Cosine Measure: Similarity of Features in Vector Space Model. *Computación y Sistemas*, *18*(3).

Song, S., Hu, H., & Jin, S. (2005). HVSM: A New Sequential Pattern Mining Algorithm Using Bitmap Representation. *Advanced Data Mining and Applications Lecture Notes in Computer Science*, 455-463.

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.

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).

Tseng, V., Wu, C., Fournier-Viger, P., Yu, P.S. (2016) Efficient Algorithms for Mining Top-K High Utility Itemsets. IEEE Trans. Knowl. Data Eng. 28(1), 54–67.

Xhemali, D., Hinde, C.J. And Stone, R.G. (2009). Naive Bayes vs. Decision Trees vs. Neural Networks in the classification of training web pages, Int. J. of Computer Science Issues, vol. 4, no. 1, (16-23).

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.

Yang, Z., Wang, Y., & Kitsuregawa, M. (2007, April). LAPIN: Effective Sequential Pattern Mining Algorithms by Last Position Induction for Dense Databases. *Advances in Databases: Concepts, Systems and Applications Lecture Notes in Computer Science,* 1020-1023.

Yao, H., Hamilton, H. J., & Butz, C. J. (2004). A Foundational Approach to Mining Itemset Utilities from Databases. *Proceedings of the 2004 SIAM International Conference on Data Mining*.

Yao, H., & Hamilton, H. J. (2006). Mining itemset utilities from transaction databases. *Data & Knowledge Engineering, 59*(3), 603-626.

Yin, J., Zheng, Z., & Cao, L. (2012). USpan: An Efficient Algorithm for Mining High Utility Sequential Patterns. *Proceedings of the 18th ACM SIGKDD International Conference on*

Knowledge Discovery and Data Mining - KDD 12.

Yun, U., & Leggett, J. J. (2005). WFIM: weighted frequent itemset mining with a weight range and a minimum weight. *Proceedings of the 2005 SIAM International Conference on Data Mining*, (pp. 636-640).

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

Zida, S., Fournier-Viger, P., Lin, J. C.-W., Wu, C.-W., & Tseng, V. S. (2015). EFIM: A Highly Efficient Algorithm for High-Utility Itemset Mining. *Lecture Notes in Computer Science Advances in Artificial Intelligence and Soft Computing*, 530–546.

VITA AUCTORIS

NAME	Komal Virk
PLACE OF BIRTH	Patiala, Punjab, India
YEAR OF BIRTH	1994
EDUCATION	St. Peter's Academy School, Patiala, Punjab, India (2009)
	Budha Dal Public School, Patiala, Punjab, India (2009 - 2011)
	R.I.M.T - IET College, Mandi Gobindgarh, Punjab, India (2011 - 2015)
	University of Windsor, Ontario, Canada (January, 2019 – December, 2020)