2017

Mining High Utility Sequential Patterns from Uncertain Web Access Sequences using the PL-WAP

Sravya Vangala
University of Windsor

Mining High Utility Sequential Patterns from Uncertain Web Access Sequences using the PL-WAP

By

Sravya Vangala

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

2017

Mining High Utility Sequential Patterns from Uncertain Web Access Sequences using the PL-WAP

by

Sravya Vangala

APPROVED BY:

_____

S. Nkurunziza
Department of Mathematics and Statistics

_____

R. Gras
School of Computer Science

_____

C. I. Ezeife, Advisor
School of Computer Science

[May 04, 2017]

AUTHOR'S DECLARATION OF ORIGINALITY

I certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniqus, 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 submitted for a higher degree of any other University or Institution.

ABSTRACT

In general, the web access patterns are retrieved from the web access sequence databases using various sequential pattern algorithms such as GSP, WAP and PLWAP tree. However, these algorithms do not consider sequential data with quantity (internal utility) (e.g., the amount of the time spent by the user on a web page) and quality (external utility) (e.g., rating of a web page in a website) information. These algorithms also do not work on uncertain sequential items (e.g., purchased products) having probability (0, 1). Factoring in the utility and uncertainty of each sequence item provides more product information that can be beneficial in mining profitable patterns from company's websites. For example, a customer can purchase a bottle of ink more frequently than a printer but the purchase of a single printer can yield more profit to the business owner than purchase of multiple bottles of ink. Most existing traditional uncertain sequential pattern algorithms such as U-Apriori, UF-Growth, and U-PLWAP do not include the utility measures. In U-PLWAP, the web sequences are derived from weblog data without including the time spent by the user and the webpages are not associated with any rating. By considering these two utilities, sometimes the items with lower existential probability can be more profitable to the website owner. In utility based traditional algorithms, the only algorithm related to both uncertain and high utility is PHUI-UP algorithm which considers the probability and utility as different entities and the retrieved patterns are not dependent with both due to two different thresholds, and it does not mine uncertain web access database sequences.

This thesis proposes the algorithm HUU-PLWAP miner for mining uncertain sequential patterns with internal and external utility information using PLWAP tree approach that cut down on several database scans of level-wise approaches. HUU-PLWAP uses uncertain internal utility values (derived from sequence uncertainty model) and the constant external utility values (predefined) to retrieve the high utility sequential patterns from uncertain web access sequence databases with the help of U-PLWAP methodology. Experiments show that HUU-PLWAP is at least 95% faster than U-PLWAP, and 75% faster than PHUI-UP algorithm.

KEYWORDS: High Utility Mining, High Utility Sequential Mining, Uncertain Data Mining, Frequent Sequential Patterns, Existential Probability Generation, Tree-Based Mining, Probabilistic Data Mining, Tuple-Based Uncertainty.

DEDICATION

To my granny Late Rajeswari Devi and grandfather Late Krishnamurthy. Special thanks to my husband Sunil and daughter Siri.

ACKNOWLEDGEMENT

My sincere appreciation goes to my mentor Baba, grandparents MohanaRao & Sailaja, parents Sekhar & Bhargavi and in-laws Harinadhababu & Vasantha. Your perseverance and words of encouragement gave me the extra energy to see this work through.

I will be an ingrate without recognising the invaluable tutoring and supervision from Dr. Christie Ezeife. Your constructive criticism and advice always gave me the needed drive to complete this work. I like to thank Dr. C. I. Ezeife for continuous research assistantship positions.

Special thanks go to my external reader, Dr. Severien Nkurunziza, my internal reader, Dr. Robin Gras and the chair, Dr. Luis Rueda for accepting to be on my thesis committee. Your decision, despite your tight schedules, to help in reading the thesis and providing valuable input is highly appreciated.

And lastly to friends and colleagues at the University of Windsor, I say a very big thank you for your advice and support throughout the duration of this work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

1.1. DATA MINING

Data mining refers to the knowledge discovery from data (KDD), the KDD process include (a) data selection (which retrieves from databases those target data, i.e., data relevant to the analysis task), (b) data pre-processing (which integrates target data from various sources and cleans target data by removing noise and inconsistent data), (c) data transformation (which summarizes or aggregates the pre-processed data into appropriate forms for mining), as well as (d) pattern evaluation and knowledge interpretation (which identifies interesting ones from the mined patterns and represents or visualizes these interesting patterns or knowledge to users). As such, data mining refers to the systematic extraction of patterns from transformed data stored or captured in large databases, data streams, data warehouses, or information repositories (Han et al., 2009). Common data mining tasks include classification, clustering, association rule mining, frequent pattern mining and sequential pattern mining (Han and Kamber 2011).

Classification (Alpaydin 2011) referred to supervised learning, is a data mining task that predicts categorical class labels for unseen new data instances (i.e., test data) based on the previously known information (i.e., training data). Data are usually classified using algorithms as decision tree induction (Siciliano and Conversano 2009), k-nearest neighbor classification (Peterson 2009) and frequent pattern-based classification (Cheng et al., 2007). For example, in Credit evaluation method, the risk is the potential loss a bank would suffer if a bank borrower fails to meet its obligations or pay interest on the loan and repay the amount borrowed by agreed terms. The Bank calculates it by the information of the customer at the bank which is accessible and relevant in calculating the financial capacity of the customer. The attributes used in the data are such as Customer ID, Salary, Age, Profession, Credit history, and Bank Balance. The bank has a

record of pre-existing loans containing such customer data and whether the loan was paid back or not. From this data of bank customers, the aim is to infer a general rule coding the association between a customer's attributes and the credit risk for the further loan approval. That is the classification model fits a model to the customer historical data to be able to calculate the credit risk for a new customer application to decide whether to accept or refuse new customer application for loan approval (Bolton and Hand 2001).

Clustering is the process of organizing data instances into groups whose members are similar in some way. A cluster is a collection of data instances which are "similar" to each other and are "dissimilar" to data instances in other clusters (Liu 2007). One of the partitioning based clustering paradigm 'k-Means algorithm' explained in steps below:

Step 1: Select K points as the initial centroids and Repeat.

Step 2: From K- cluster by assigning all points to the closest centroids.

Step 3: Recomputed the centroid of each cluster.

The example explaining k-means Clustering algorithm on Student Database to define a model that makes a prediction about fail and pass ratio of a student based on performance in the exam. To get that, the students are grouped in three ways based on their final grades. 1. Assign similar possible labels to number of possible grades as in Table 1.

No of Students	Class	Marks	%
2	A	90-100	95
3	B	60-70	65
9	D	70-80	75
4	C	40-60	50

Table 1: Number and percentage of students regarding class obtained

2. Group the students in three classes “High,” “Medium,” “Low” as in Table 2.

Categories	Low	Medium	High
No of Students	4	12	2

Table 2: Number of Students Regarding Class

3. Categorized students with one of two class labels “Passed” & “Failed” in Table 3.

Class	Marks	No of Students	%
Passed	60< = Percentage	14	78
Fail	60> Percentage	4	50

Table 3: Percentage of students

Association rule mining was a fundamental data mining task. Its objective is to find all co-occurrence relationships, called associations, among the attribute values of tuples in a database table (Liu 2007). The classic application of association rule mining is the market basket analysis using the frequent pattern mining algorithm such as Apriori (Aggarwal and Srikanth 1995), which aims to discover how items purchased by customers in a supermarket are associated. The support count of Itemset (Set of items purchased in each transaction) in transaction database is the number of transactions in the database that contain the itemset. The **Support** of a set of items defined as the number of tuples (e.g.: Bread, Butter) or the percentage of the database tuples in the table that contains these set of items. Support (itemset) = number of tuples in the itemset/total number of tuples in the database. From the Table 4, Support (Bread U Butter) = 3/4 = 75%. The **Confidence** of a rule is defined as the percentage of transactions in database that contain the right -hand side of the rule set of items when they also contain the items on the left-hand side. Confidence (rule) = number of tuples with itemsets on the right-hand side of the rule/number of tuples with the itemsets on the left-hand side. From the Table 4, the Confidence (Bread -> Butter) = | Bread and Butter| / | Bread| = 3/3 = 100%.

Frequent Pattern Mining aims to discover how items purchased by customers in a supermarket with a frequency no less than a user-specified threshold. For Example, Apriori algorithm (Aggarwal and Srikanth 1995) finds the set of frequent patterns iteratively by computing the support of each itemset in candidate set.

Input: A Database D; C_1 refers to Candidate-1itemsets (the items present in the database); MinSupport 'm';

Output: Frequent patters FP; **Other Variables**: i for i-itemset; F_i =Frequent i-itemset;

Step 1: i=1; F_i= {Set of C_1 itemsets in the database with support count} >= m for each itemset in C_1.

Step 2: If $F_i \neq \Phi$ then i=i+1; else end i=1,

Join Step: $C_i = F_{i-1} \bowtie_{\text{apriori-gen}} F_{i-1}$. The apriori-gen join of F_i with F_i joins every itemset 'm' of first F_i with every itemset 'n' of second F_i where n > m and first (i-1) members of itemsets 'm' and 'n' are the same. **Prune Step:** If a candidate itemset C in C_i has a subset in F_{i-1} that is not frequent then C is pruned from C_i before the database scan for support. This is called Apriori or Downward Closure property.

Step 3: If $C_i \neq \Phi$, then find F_i else go to Step 3.

Step 4: The frequent patterns generated are FP = $\cup F_i$.

Example for Apriori Algorithm: **Input:** A Transaction Database in the Table 4, Candidate - 1 items in database C_1= {Bread, Butter, Milk, Sugar}; MinSupport 'm' = 2; F = Set of Frequent items. **Output:** Frequent patters FP

Transaction ID	Set of items purchased
T1	Bread, Butter, Milk
T2	Bread, Butter
T3	Bread, Butter, Milk, Sugar
T4	Milk, Sugar

Table 4: A Sample Transaction Database

Step 1: F_i= {Set of items in database with support count}>= m for each itemset in C_i. From Table 4, F_1= {Bread, Butter, Milk, Sugar}. **Step 2:** From Join and Prune steps, Candidate items generated from Table 4 are: C_2= $F_1 \bowtie_{\text{apriori-gen}} F_1$; C_2 = {(Bread, Butter: 3), (Bread, Milk: 2), (Bread, Sugar: 1), (Butter, Milk):2), (Butter, Sugar: 1), (Milk, Sugar: 2)}. F_2 = {(Bread, Butter), (Bread, Milk), (Butter, Milk), (Milk, Sugar)}.C_3= $F_2 \bowtie_{\text{apriori-gen}} F_2$ = {(Bread, Butter, Milk: 3), (Bread, Milk, Sugar:1), (Butter, Milk, Sugar: 1)}. F_3= {(Bread, Butter, Milk)}; **Step 3:** C_4= $F_3 \bowtie_{\text{apriori-gen}} F_3$ = Φ. **Step 4:** The frequent patterns generated are FP = $\cup F_i$, From Table 4, FP=$F_1 \cup F_2 \cup F_3$ = {Bread, Butter, Milk, Sugar, (Bread, Butter), (Bread, Milk), (Butter, Milk), (Milk, Sugar), (Bread, Butter, Milk)}

Limitation: The main problem with the Apriori Algorithm is that it produces a substantial number of candidate itemsets, which requires multiple database scans that increases the execution time through huge disk input/output operations.

1.2. SEQUENTIAL PATTERN MINING

Frequent Pattern mining, however, does not consider the sequence in which the set of items are purchased. Sequential pattern mining discovers frequent subsequences as patterns in a sequence database. A sequence database D store a number of records, where all records are sequences of ordered events, without any time order (for example, retail customer transactions or purchase sequences in a store showing, for each customer, the collection of store items they purchased every week for one month). An example of a sequential pattern is “5% of customers buy bed first, then mattress and then pillows” represented as the sequence < (bed), (mattress), (pillows)>. More than one item can be purchased at a time to represent a sequence of sets of items like (< (bed, sofa), (mattress), (pillows)>. Given a support threshold minsup, a sequence s_a is called a frequent sequential pattern on D if sup_D (s_a) ≥ minsup (Mabroukeh and Ezeife 2010).

As the proposed algorithm HUU-PLWAP is based on two factors, one is Utility which incorporates the input data with the quantity and quality (e.g. price) of the purchased items and second one was Uncertainty in web access sequences, so the sequential web access mining from a Database D, with a minimum support ξ, and set of candidate events E = {a, b, c, d} representing web page addresses, is restricted with the following characteristics (1) Patterns in a weblog consist of contiguous page views (items in the sequences). The same user can access no two pages at the same time, and so sequences contain only 1-itemsets (i.e., singleton itemsets). An example sequence is < bcabdac >, which is different from a general sequence like < (ba) (ab) d (ac)>. (2) In web usage mining, as the lexicographic order of page references in a time order transaction sequence is important. (Mabroukeh and Ezeife 2010). Several algorithms explained the mining of sequential patterns such as GSP (Pei et al., 2000), SPAM (Ayres et al., 2002), WAP (Pei et al., 2000), PL-WAP (Ezeife and Lu 2005). One of the **S**equential **Pa**ttern **M**ining Algorithms with vertical bitmap representation was introduced by (Ayres et al., 2002), abbreviated as **SPAM** algorithm. It is claimed to be the first strategy for mining sequential patterns to traverse the lexicographical sequence tree in a depth-first fashion. The SPAM algorithm, utilizes a depth-first traversal of the search space combined with a vertical bitmap representation to store each sequence.

Input: A Sequence Database D (in the Table 5), Candidate-1 itemset Sequence (Items present in the sequence database) C_1, Minimum Support `m`.

Output: L= Set of frequent sequential patterns.

Other Variables: Vertical bitmap B_i for each item sequence in candidate 1-sequence.

Example: D is a database in the Table 5 of web access sequences from customer database, sorted according to users CID (User IDs), and C_1 = {a, b, c, d,} denotes the set of candidate 1-sequences, then the task is to find the set L of frequent sequential patterns, given a minimum support threshold m =2.

CID	Web access Sequence
1	< {a, b, d}, {b, c, d}, {b, c, d}>
2	<{b}, {a, b, c}>
3	< {a, b}, {b, c, d}>

Table 5: Weblog Sequence Database

Step 1: **Data Representation (Converting database to vertical bitmap):** For efficient counting of support, SPAM algorithm uses a vertical bitmap representation of the data. A vertical bitmap is created for each item in the dataset, and each bitmap has a bit corresponding to each transaction in the dataset. The Database D in Table 5 is converted, into a vertical bitmap by setting (1) or (0) shown in Figure 1, where each candidate-1 item (such as a, b, c, d)) has a vertical column and each subsequence (e.g., CID1 = < {a, b, d}, {b, c, d}, {b, c, d}>) is mapped in the 1-event bitmaps. For example, the first transaction CID 1 the bitmap obtained after mapping to events a, b, c, d is: 1101, 0111, and 0111.

CID	{a}	{b}	{c}	{d}
1	1	1	0	1
1	0	1	1	1
1	0	1	1	1
-	0	0	0	0
2	0	1	0	0
2	1	1	1	0
-	0	0	0	0
-	0	0	0	0
3	1	1	0	0
3	0	1	1	1
-	0	0	0	0
-	0	0	0	0

Figure 1: Vertical Bitmap of items in database

Step 2: Candidate generation and Construction of the Lexicographic tree: The tree is constructed for CID 1 extending from the candidate 1-item sequences to candidate 2-item sequences which will start with the first item subsequence ‘a’ of a, b, d and perform for

each suffix sequence to be extended with such as (a, b, and d) a sequence-extended children sequences generated in the S(sequence-extension)-Step of the algorithm (for example, from CID 1, aa, ab, ad are obtained respectively) and itemset-extended children sequences generated by the I(item-extension)-Step of the algorithm at each node (For example, from CID 1,(a, a),(a, b), (a, d) are obtained respectively). Thus, for the first subsequence of CID=1, the lexicographical tree for items a, b and d obtained is shown in the Figure 2, assuming a maximum sequence size of 3.

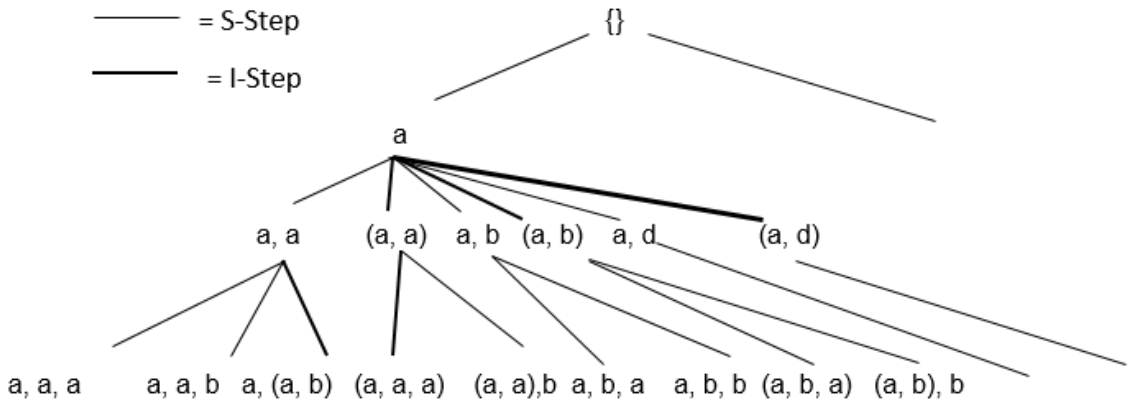

Figure 2: Lexicographic tree

Step 3: Depth first traversal and finding the Frequent Sequences from tree generated: Each element in the tree is generated only by either an S-step or an I-step as explained in step 2. SPAM traverses the tree generated in depth-first search manner and at each node it checks the support of each sequence-extended or itemset-extended child against min sup recursively. Sequences of the form “(ab)” is I step sequence and sequences of form “a, b” is S-step sequence, both have certain procedures to be followed for checking if they are frequent or not. The S-step and I-Step procedures of item {a} are in Figure 3. Let us consider the bitmap B ({a}), and to generate B ({a}, {b}). Since ({a}, {b}) is a sequence-extended sequence of ({a}), an S-step process on B ({a}) is performed. For S-step, bitmap of {a} need to transform into ‘{a} s’. The index of the first 1 bit in {a} should be transformed into 0. Then all the bits which are behind this bit should be set to "1".

When the transformed bitmap '{a} s' is obtained, the bit-AND operation is performed on '{a} s' and {b} to get the result of S-step of ({a}, {b}). As support count of ({a}, {b}) was 4, the pattern ({a}, {b}) was considered as frequent.

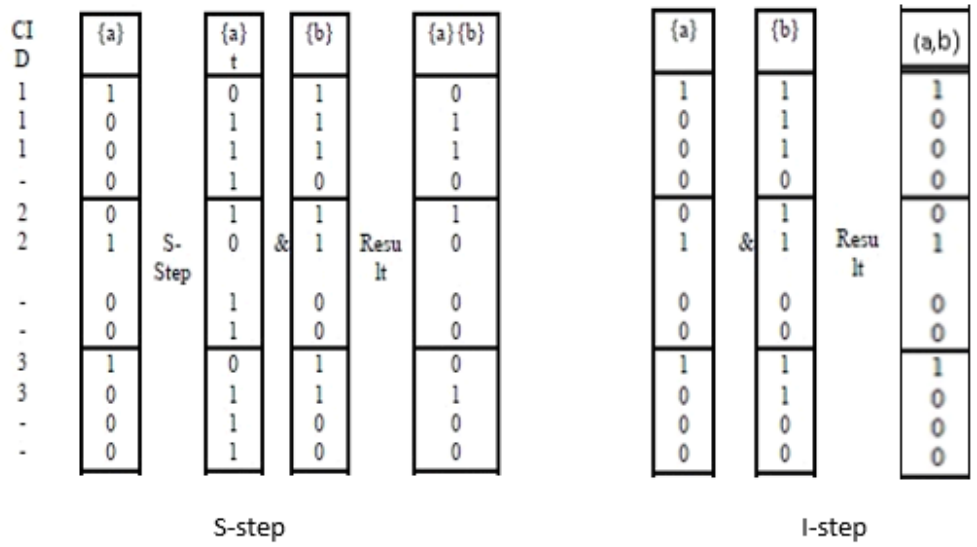

Figure 3: The S-step and I-Step procedures

For I-step, **bit-AND operation on the bitmaps is performed** e.g. of ({a} and {b} to get the result of {a, b}). After I-step or S-step is completed, the number of sequences that have more than one "true" bits in bitmap results are accumulated. As shown in Figure 3, the support count of {a} is 3; {b}, is 7 and {a, b} is 3. If min sup is set to 50% (i.e., 2 sequences), all three are viewed as frequent and will not be pruned, else will be pruned. SPAM can generate the complete set of frequent sequential patterns by traversing only through parent sequences which are frequent, in lexicographic tree through Apriori property.

Step 4: **Pruning Strategies**: As the search space is more for traversing the tree in depth first fashion, two pruning techniques(Apriori-based) such as S-step and I-step Pruning that prunes candidate s-extensions and i-extensions of a node n in the tree are implemented. They aimed at minimizing the size of Sequence S_n for each node n. At the same time, the techniques guarantee that all nodes corresponding to frequent sequences are visited. S-step prunes S-step children. For example, if node ({a}) in the tree is

considered and suppose that S ({a}) = {a, b, c, d}, I ({a}) = {b, c, d}. The possible sequence-extended sequences are ({a}, {a}), ({a}, {b}), ({a}, {c}) and ({a}, {d}). Suppose that ({a}, {c}) and ({a}, {d}) are not frequent. By the Apriori principle, none of the following sequences can be frequent either: ({a}, {a}, {c}), ({a}, {b}, {c}), ({a}, {a, c}), ({a}, {b, c}), ({a}, {a}, {d}), ({a}, {b}, {d}), ({a}, {a, d}), or ({a}, {b, d}). Hence, when we are at node ({a}, {a}) or ({a}, {b}), so I-step or S-step is not performed for items c and d, i.e. S({a},{a}) = S({a},{b}) = {a, b}, I({a},{a}) = {b}, and I({a},{b}) = ∅. I-step Pruning prunes I-step children. For example, if the same node ({a}) described in the S-step pruning section. The possible itemset-extended sequences are ({a, b}), ({a, c}), and ({a, d}). If ({a, c}) is not frequent, then ({a, b, c}) must also not be frequent by the Apriori principle. Hence, I ({a, b}) = {d}, S ({a, b}) = {a, b}, and S ({a, d}) = {a, b}.

Advantage: It is the first algorithm to utilize a depth-first traversal for the search space in the tree which is combined with a vertical bitmap representation to store each sequence.

Limitation: As SPAM uses a depth-first traversal of the search space, it is quite space-inefficient for smaller datasets.

1.3. WEB MINING

Web mining categorized into three different classes based on which part of the Web is mined. These three categories are (i) Web content mining, (ii) Web structure mining and (iii) Web usage mining (Vijayalakshmi et al., 2010)

(i) Web content mining: It is the task of discovering useful information available on-line. There are various kinds of Web content which can provide useful information to users, for example, multimedia data, structured (i.e. XML documents), semi-structured (i.e. HTML documents) and unstructured data (i.e. plain text). The aim of Web content mining is to provide an efficient mechanism to help the users to find the information they seek. Web content mining includes the task of organizing and clustering the documents and providing search engines for accessing different documents by keywords, contents, etc.

(ii) Web structure mining: It is the process of discovering the structure of hyperlinks on the Web. Authoritative pages contain useful information and supported by several links pointing to it, which means that these pages are highly referenced. Web structure mining can exploit the graph structure of the web to improve the performance of the information retrieval and to improve classification of the documents.

(iii) Web usage mining: There are three types of log files that can be used for Web usage mining. Log files are stored on the server side, on the client side and the proxy servers. By having more than one place for storing the information of navigation patterns of the users makes the mining process more difficult. Some commonly used data mining algorithms for Web usage mining are association rule mining, sequence mining and clustering (Vijayalakshmi et al., 2010).

1.4. HIGH UTILITY ITEMSET MINING

The utility is introduced into pattern mining to mine for patterns of high utility by considering the quality (such as profit) and quantity (such as a number of items purchased) of itemsets. This has led to high utility pattern mining (Yao et al., 2004), which selects interesting patterns based on minimum utility rather than minimum support. High utility itemset refers to those set of items which has a high utility such as profit in a database. Web usage mining is to discover user traversal patterns of Web pages from Weblog records. For an example, a popular Website may register the Weblog records in the order of hundreds of megabytes every day from various users, which provide rich information about the Webpage importance. The utility is a measure of how “interesting” or “useful” a Web page is. As a result, it allows Web service providers to select user preferences of different traversal paths.

Step 1: By introducing the concept of utility into web path traversal pattern mining problem, each web page associated with the internal(quantity) utility value (which could

be the browsing time a user spent on a given page) and also the external(quality) utility value (which could be the end user's preference).

Step 2: The utility of a web page is the product of internal and external utility in which a web page refers to an item, a traversal sequence refers to an itemset, the time a user spent on a given page X in a browsing sequence T is defined as utility denoted as u(X, T) (Zhou et al., 2007). The more time a user spent on a Web page, the more interesting it is to the user. Thus, utility-based web path traversal pattern mining is to find all the Web traversal sequences that have high utility beyond a minimum threshold.

Step 3: The challenge of utility mining is that it does not follow "downward closure property," that is, a high utility itemset may consist of some low utility sub-itemsets. The two-phase algorithm proposed by (Yao et al., 2006) aimed at solving this difficulty which was described in the two phases. In Phase I, transaction-level utility is proposed and defined as sum of the utilities of all transactions containing X. High transaction-level utility sequences are identified in Phase I. A Transaction-level Downward Closure Property is that any subset of a high transaction-level weighted utility (TWU) itemset must also be high in transaction-level weighted utility.

Step 4: In Phase II, only one database scan is performed to filter out the high transaction-level weighted utility itemsets that are low utility itemsets. The size of candidate set is reduced by only considering the supersets of high TWU sequences.

An example, the Utility based Web Traversal Pattern(UWTP) Mining Algorithm (Zhou et al., 2007), which extends the Two-Phase algorithm to traversal path mining problem.

Input: A Traversal path database and a Profit table is given in Table 6, Candidate -1 Sequences C1= {A, B, C, D}; Minimum Utility Threshold = 12;

Output: High Utility Sequential Patterns

From **Step 1**, the internal utility of the web page in each sequence represents the time spent on it. The external utilities of the web pages represents the ratings given by the different users of the website. Both are given given in the Table 6.

TID	Traversal Path
T1	A (2) B (1)
T2	B (6) D (1) C (1)
T3	A (1) C (1) D (3)
T4	A (1) D (4) B (5)

WEB PAGE	A	B	C	D
PROFIT(EU)	1	1	1	1

Table 6: A Traversal Path Database and Profit Table

From **Step 2,** The utilities of A was (2*1+1*1+1*1) = 4; B = (1*1+6*1+5*1) = 12; C = (1*1+1*1) = 2; D = (1*1+3*1+4*1) = 8; AB = (3+6) = 9; AC = 2; AD = 9; BC = 7; BD = 16; CD = 6; ABD = 10; ACD = 5; BCD =8; the transaction-level utility are T1 :3; T2:8; T3=5; T4=10.From the database in Table 6, From **Step 3**, the TWU sequences are A= 18; B= 21; C= 13; D= 23; AB= 13; AC= 13; BC= 8; BD= 18; CD= 13; ABD= 10; ACD= 5; BCD= 8. From **Step 4**, If the Minimum Utility Threshold = 12, then the High Utility Sequential Patterns generated are A, B, C, D, AB, AC, BD, and CD.

Advantage: The high utility traversal paths may assist Web service providers to design better web link structures, thus cater to the user's interests in the websites.

Limitation: Not confined to larger databases.

1.5. REAL LIFE SCENARIOS OF THE HIGH UTILITY MINING

Frequent itemsets may only contribute a small portion of the overall profit, whereas non-frequent itemsets may contribute a substantial portion of the profit. So, they are mined separately.

1. RETAIL BUSINESS: It may be interested in identifying its most valuable customers (customers who contribute a major fraction of the profits to the company). Those are the customers, who may buy full priced items, high margin items, or gourmet items (high-quality premium foods), which may be absent from a large number of transactions but purchase high profitable items. For example, Customer buying ink will be more frequent

than the printer, but the value if the printer is high which can be profitable to the business owner. The utility is based on quality and quantity of an item in the database.

2. WEB LOG DATA: A sequence of web pages visited by a user can be defined as a transaction. Since the number of visits to a web page and the time spent on a webpage is different between different users, the total time spent on a page by a user can be viewed as internal utility and the webpage rating provided by the website can be measured as the external utility. The website designers and web service providers can catch the interests or behavior patterns of the customers by looking at the utilities of the page combinations and then consider re-organizing the link structure of their website to cater to the preference of users. Frequency is not sufficient to answer questions, such as whether a webpage is highly profitable, or whether a web traversal has a strong impact.

1.6. HIGH UTILITY SEQUENTIAL PATTERN MINING

Sequential pattern mining has emerged as an important topic in data mining. It has proven to be essential for handling order-based critical business problems, such as behavior analysis, gene analysis in bioinformatics and weblog mining (Parmar et al., 2015). The selection of interesting sequences is generally based on the support framework, and sequences of high frequency are treated as significant. In the traditional sequential pattern mining algorithms, the downward closure property (also known as Apriori property) (Agrawal et al., 1995) plays a fundamental role for varieties of algorithms designed to search for frequent sequential patterns. The utility is introduced into pattern mining to mine for patterns of high utility by considering the quality (such as profit) and quantity (such as a number of items purchased) of itemsets. This has led to high utility pattern mining (Yao et al., 2004), which selects interesting patterns based on minimum utility rather than minimum support. Later sequential pattern mining is 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. Following the high utility pattern mining approach, highly

profitable sequential patterns are retrieved, which are more informative for retailers in determining their marketing strategy. First, as with high utility itemset mining, the downward closure property does not hold in utility-based sequence mining. This means that most of the existing algorithms cannot be directly transferred, from frequent sequential pattern mining to high utility sequential pattern mining. Later with the advent of the sequence weighted utility, the Apriori property issue is resolved as the normal sequence utility does not hold the property but the weighted sequence utilities follows the Apriori property from which the high utility sequential patterns are generated. To effectively prune the search space, the concept of Sequence- Weighted Utility (SWU) (Liu et al., 2005) was proposed to serve as an over-estimate of the true utility of a sequence, which has the downward closure property. This SWU model is incorporated into the proposed framework, and a new model called Transaction based Sequence-Weighted Utility (TSWU) is introduced to effectively prune the search space. A sequence is regarded as a high utility sequence (HUS) if its SWU is no less than the minimum utility threshold. An SWU-based algorithm finds HUS's and then identifies the high utility sequences from HUS and spends much time to filter out low utility sequences from HUSs, thus prolonging the execution time.

For example, USpan (Yin et al., 2012) is one of the high utility sequential pattern mining algorithms composed of lexicographic q-sequence tree, 2 concatenation mechanisms and 2 pruning strategies.

Input: A sequence database, Profit table, Minimum Utility threshold.

Output: High Utility Sequential Patterns

Step 1: For utility-based sequences, the concept of the Lexicographic Sequence Tree is utilized 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.

Step 2: Suppose for a k-sequence t, the operation of appending a new item to the end of t is to form (k+1)-sequence concatenation. If the size of t does not change, the operation

I-Concatenation will occur. Otherwise, if the size increases by one, S-Concatenation is occurred. For example, <ea>'s I Concatenate and S-Concatenate with b result in <e(ab)> and <eab>, respectively. Assume two k-sequences t_a and t_b are concatenated from sequence t, then $t_a < t_b$ if

i) t_a is I-Concatenated from t, and t_b is S-Concatenated from t, or

ii) both t_a and t_b are I-Concatenated or S-Concatenated from t, but the concatenated item in t_a is alphabetically smaller than that of t_b.

For example, <(ab)>, < ((ab)b)>, <(abc)> < (ab)b>, <(ab)c> < <(ab)d> .

Step 3: A lexicographic q-sequence tree (LQS-Tree) T is a tree structure satisfying the following rules: Rule1: Each node in T is a sequence along with the utility of sequence, while the root is empty and Rule 2: Any node's child is either an I-Concatenated or S-Concatenated sequence node of the node itself. Rule 3: All the children of any node in T are listed in an incremental and alphabetical order.

Step 4: Additionally, if minimum utility threshold = 0, then the complete set of the identified high utility sequential patterns forms a complete LQS-Tree, with complete search space. USpan uses a depth-first search strategy to traverse tree to search for high utility patterns.

Step 5: The I-Concatenation and the S-Concatenations are applied to the LQS Tree.

Step 6: The depth and width pruning techniques are further applied to remove the unpromising candidates from the tree. An example explaining the USpan algorithm is mentioned in the section 2.4.2.

Advantage: It follows the bitmap representation like in SPAM algorithm which is suitable for larger datasets.

Limitation: It follows the lexicographic tree construction for generating high utility sequential patterns which are more time consuming and it does not handle the uncertain sequences.

1.7. UNCERTAIN DATA

A common cause of uncertainty comes from measurement errors: the locations of users obtained through RFID (Radio Frequency Identification) (Sistla et al., 1998) or GPS (Global Positioning System) of moving objects are not precise (Khoussainova et al., 2006) data collected from sensors in habitat monitoring systems (such as temperature and humidity) and noisy (Deshpande et al., 2004). As an example, Table 7 shows the records of the ids of the speeding vehicles (Vehicle in the table) and the speed (Speed) captured by sensor nodes (SID) at a certain location (Loc.) and time (Time). Each record is given an occurrence probability (P) representing its confidence to be true. In this example, records R1 and R2 cannot appear in the same possible world; that is, R1 and R2 are exclusive as they occur on the same day and same time. R3 is independent with them; this means the generation rule containing R3 only is independent with generation rule {R1, R2}. There are six possible worlds for this uncertain database, as shown in Table 8, along with corresponding possible world's occurrence probabilities.

RID	SID	TIME	LOC	VEHICLE	SPEED	PROB
R1	S1	2: 00PM	L1	HB1235	120	0.7
R2	S2	2: 00PM	L1	HB1238	150	0.2
R3	S3	3: 45PM	L1	HB2568	170	0.9

Table 7: Speeding Vehicle Records

Possible World	Occurrence	Probability
W1	{φ}	0.01
W2	{R1}	0.07
W3	{R2}	0.02
W4	{R3}	0.09
W5	{R1, R3}	0.063
W6	{R2, R3}	0.18

Table 8: Number of possible worlds from the Speeding Vehicle records

It is assumed that the items occurring in a transaction are known for certain. However, this is not always the case. For instance, in many applications, the data is inherently noisy, such as data collected by sensors or in satellite images. In real-life applications, utility and probability are two different measures for an object (e.g., a useful pattern). The utility is a semantic measure which is based on the user's prior knowledge and goals, while probability is an objective measure in which the object or pattern has existential probability. Up to now, most algorithms of High Utility Mining such as U-Mining (Yao et al., 2004), Two-Phase algorithm (Yao et al., 2006), HUI-Miner (Liu and Qu 2012), PHUI-UP (Lin et al., 2016) have been extensively developed to handle precise data, which are not suitable to mine the data with uncertainty. To the best of my knowledge, the proposed framework will be the first work to address the issue of Mining High Utility Itemsets from the uncertain web access sequence database.

Example 1: Measured values in sensor data applications are notoriously imprecise. An example is an ongoing project at Purdue University (Aggarwal et al., 2009) that tracks the movement of nurses to study their behavior. Nurses carry RFID tags (Chen et al., 2005) as they move around the hospital. Numerous readers located around the building report the presence of tags in their vicinity. The collected data is stored centrally in the form "Nurse2 in room6 at 10:10 am". Each nurse carries multiple tags. Difficulties arise due to the variability in the detection range of readers; multiple readers detecting the same tag; or a single tag being repeatedly detected between two readers (e.g., between room6 and the hallway – is the nurse in room6 all the time, just that the hallway sensor is detecting his tag or is she actually moving in and out?). Thus, the application may not be able to choose a specific location for the nurse always with 100% certainty.

Example 2: Data collected from sensors (e.g., temperature sensors for weather, or GPS-based location data from cell phones); there is almost always some amount of inherently associated uncertainty. Also, due to resource limitations such as battery power of sensor and network bandwidth, sensors only transmit data intermittently. Consequently, it is

infeasible for a sensor database to contain the exact value of each sensor at any given point in time. Thus, the traditional model of a single value for a sensor reading is not a natural fit with this data (Agrawal 2009). Instead, a more appropriate model is one where the sensor attribute can be represented as a probability distribution reflecting the inherent uncertainties and interpolation between measurements. Overall, these kinds of emerging database applications require models which can handle uncertainty and semantics to define useful queries on such data. A major choice for each model is whether to incorporate probability values at the tuple or attribute level. This leads to two slightly different approaches in modeling and representing uncertain data (Aggarwal and Philip 2009). There are two main approaches for modeling uncertain relational data. One approach (Tuple uncertainty) is to attach a probability value with each tuple – the probability captures the likelihood of the given tuple being present in the given relation (Aggarwal 2009). The probability values for different tuples are assumed to be independent of each other unless some dependency is explicitly given. These dependencies across tuples can be used to express mutually exclusive alternatives. Such tuples are called x-tuples. The Table 9 shows uncertainty information expressed using tuple uncertainty. The tuples for Carid = Car1 are grouped together in an x-tuple, so they are mutually exclusive. Thus, Car1 has problems with either Brakes or Transmission with probability 0.1 and 0.2 respectively.

Car Id	Problem	Probability
Car1	Brakes	0.1
Car2	Tires	0.9
Car 1	Transmission	0.2
Car 2	Suspension	0.8

Table 9: Example of a Relation with x-tuples

The second approach (Attribute uncertainty) allows for probability values at the attribute level. Table 10 is expressed using Attribute Uncertainty. It should be noted that both models are similar, as they use possible world's (existence of an item or not) semantics for probabilistic calculations and for verifying the correctness of operations.

Car Id	Problem
Car1	{(Brakes,0.1), (Tires,0.9)}
Car2	{(Trans,0.2), (Suspension,0.9)}

Table 10: Example of a relation with Attribute Uncertainty

It is assumed that the items occurring in a transaction are known for certain data. However, it is not always the same case. For example, in many applications, the data seems to be uncertain particularly when it was from GPS or RFID, Sensors (Bernecker et al., 2009). Up to now most algorithms of High utility itemset mining are developed to read precise data, and are not useful for uncertain data. The proposed framework will be the first solution of retrieving the high utility sequential patterns from uncertain weblog sequences. The differences between certain and uncertain data were given in Table 11.

<table>
<tr><th colspan="2">Certain Web access data</th><th colspan="2">Uncertain Web Access Data</th></tr>
<tr><td colspan="2">The data is clean and can be used directly for the process of the mining</td><td colspan="2">The data is inaccurate and needs to be preprocessed further for the mining.</td></tr>
<tr><td colspan="2">Example:</td><td colspan="2">Example:</td></tr>
<tr><td>Car Id</td><td>Problem</td><td>Car Id</td><td>Problem</td></tr>
<tr><td>Car1</td><td>{(Brakes), (Tires)}</td><td>Car1</td><td>{(Brakes,0.1), (Tires,0.9)}</td></tr>
<tr><td>Car2</td><td>{(Trans), (Suspension)}</td><td>Car2</td><td>{(Trans,0.2), (Suspension,0.9)}</td></tr>
</table>

Table 11: Certain vs. Uncertain Data

Uncertainty in Sequential Pattern Mining:

In sequential pattern mining algorithms, one of the algorithms deals with the uncertain data was the U-PLWAP algorithm (Uncertain Pre-linked position coded web access pattern algorithm) (Kadri and Ezeife 2011). It is noted here that unlike in traditional, precise sequences where occurrence count of an item automatically contributes towards the support count of an item, the sum of the product of the existential probability values is used in arriving at support counts. An equally important observation with the uncertain data items is that items with the same label can have different existential probability values in single sequence (for example, < (a:0.1, b:0.3, c:0.4, a:0.5) > the a has multiple existential probability values in the same transaction). It is, therefore, important to record item's label, occurrence count and the existential probability values to determine frequent sequences accurately. The steps of UPLWAP algorithm are (Kadri and Ezeife 2011):

Step1: The U-PLWAP scans the sequence database to discover the frequent 1-sequences. This is done by adding up all existential probability values for each item whenever they occur in the database. Whenever an item is repeated in a sequence, it's the higher existential probability is only added. The frequent 1-sequences are those items with counts greater than or equal to the minimum support threshold.

Step2: Each of the frequent 1-sequences is used to create entries in header table.

Step3: A second scan is used to create U-PLWAP tree after the non-frequent data items have been removed from the sequence. The U-PLWAP tree is created starting from a null root. Sequences are read from the database and nodes are created for each item in the sequence. Each node contains the item's label, multiple occurrence counts and their corresponding existential probability and position code, denoted as label: count: position code. Since similar labels with different existential probabilities in a path are merged into one node, each sequence read is identified by its sequence ID and recorded against the existential probabilities of its items in every node. A left node is created if no node already

exists, otherwise, a right node is created and the count initialised to 1. If node already exists, its count is incremented by 1. The item label and its existential probabilities are read from the sequence database. Entries created in the header table are then used to link their corresponding nodes by traversing the U-PLWAP tree in a pre-ordered fashion (from root to left node first before right node).

Step4: The U-PLWAP tree created is then mined recursively using prefix conditional search until no more items found. The algorithm then backtracks to the null root to start mining for sequences starting with a fresh item from the header table. The example explaining for U-PLWAP algorithm is mentioned in the section 2.2.2.

Advantage: The PLWAP tree algorithm is implemented in the uncertain data with the same methodology.

Limitation: The algorithm does not apply on utility. The support count, which is the sum of the all existential probability values for each item, sometimes involve in low existential probabilities with high profitable value.

Uncertainty in High Utility Itemset Mining:

The only algorithm which has worked on the uncertain data in high utility itemset mining was PHUI-UP algorithm (Lin et al., 2016). The potential high utility itemset mining (PHUI-UP) in uncertain databases, is proposed to efficiently discover not only the itemsets with high utilities but also the itemsets with high existential probabilities in an uncertain database based on the tuple uncertainty model (in which a probabilistic database represents a set of possible “certain” database instances (worlds), where a database instance corresponds to a subset of uncertain tuples, and each tuple is associated with a probability denoting the likelihood that it exists in the relation). Each instance (of possible world) is associated with the probability that the world is “true”. The probabilities reflect the probability distribution of all possible database instances. The PHUI-UP algorithm (potential high utility itemsets upper-bound-based mining algorithm) is first presented to

mine potential high utility itemsets (PHUIs) using a level-wise search. As it adopts a generate-and-test approach to mine PHUIs, it suffers multiple database scans. The "utility" can be viewed as the user-specified importance, i.e., weight, cost, risk, unit profit or value; but numerous discovered HUIs may not be the patterns required by a retailer to take efficient decisions, since traditional HUIM algorithms do not consider existence probabilities. Discovered patterns may be misleading if they have low existential probabilities. The PHUI-UP algorithm has two phases: In first phase, the HTWPUIs are found until no candidate is generated, and in the second phase, PHUIs are derived with an additional database scan.

Step 1: The proposed PHUI-UP algorithm takes input as Uncertain Transaction database and scans the database to find the TWU (Transaction Weighted Utilities) and the probabilities (PRO) values using the tuple based uncertainty model for all the 1-itemsets. The utility and transaction utility and the weighted transaction utilities are calculated.

Step2: Once all 1-itemsets satisfies Transaction Weighted Utilities TWU of an itemset ≥ min utility threshold (MUT) and Probability (PRO) of an itemset ≥ min probability threshold (MPT), they are all put into the set of potential high transaction-weighted utilization itemsets (HTWPUIs). Then k is set to 2, and the candidates C2 are generated by applying Apriori-gen (HTWPUI1) using the lexicographic order of items.

Step 3: The uncertain database is then re-scanned to calculate the TWU and Pro values of each itemset in C2. The results should satisfy TWU(X) ≥ MUT and Pro(X) ≥ MPT.

Step 4: These itemsets are thus added to the set of HTWPUIs. Then k=3, and the procedure is applied in the same way. The first phase terminates when no candidate is generated. An additional database rescans required in the second phase to find the final PHUIs from the candidate HTWPUIs. Based on the designed TWPUDC property, (which ensures that no supersets of small transaction-weighted probabilistic and utilization itemsets are in the preliminary candidate set (correctness)) it can extract the complete

PHUIs from the candidate HTWPUIs (completeness). An example is mentioned in the section 2.3.3.

Advantages: It is the first algorithm to mine high-utility itemsets from an uncertain transaction database.

Limitations: Because of Apriori-based approach it suffers from a high space and time complexities. Also, it considers the probability as a separate entity not appended with the utility measure. It deals with transaction database, not with the sequence database.

1.8. THESIS PROBLEM AND CONTRIBUTIONS

This main aim of this thesis is to provide a High utility sequential web-access mining solution HUU-PLWAP (High Utility Uncertain Pre-Linked Position Code Web Access Pattern) miner for mining uncertain High utility web access sequences using the U-PLWAP tree mining approach. The existing approaches of USpan (Yin et al., 2012) and HUS-Span (Zihayat et al., 2015) are all based on precise high utility data sequences with 100% certainty. U-PLWAP proposed by (Kadri and Ezeife 2011), an uncertain data implementation of PLWAP algorithm which considers only find frequent sequential patterns but there can be the pattern which is more profitable for the owner and least frequent, then these kinds of patterns (which consider the internal (quantity) and external (quality) of the events in the sequence) are retrieved using the proposed miner. Also The items or web pages with low existential probability are removed based on the frequency count, but sometimes they can be more profitable (for example buying a printer will be more profitable than purchasing ten ink bottles for the business owner). So, to overcome this problem the proposed HUU-PLWAP miner considers input data with the internal existential probability quantity and external quality to derive much more useful patterns for the better website design. For web traversal sequences with internal utility, i.e., the time spent by the user on the web page and the external utility, i.e., rating given by the website owner. The U-PLWAP algorithm generates the sequence using the access history probabilities which is different from the proposed framework as it generates the sequence database based on the sequence uncertainty model (which is based on the possible world model mentioned in the section 3.2.) along with the utility based mining.
The PHUI-UP (Potential High Utility Itemset-Utility Pattern) algorithm is proposed by (Liu et al., 2016) for mining the uncertain high utility itemsets from the transaction database which considers the probability and utility measures as different entities. To resolve all these mentioned issues, the thesis has proposed a solution called HUU-PLWAP.

Existing Systems	Research Goal	Technique to Obtain Relevant Aspects	Limitations
U-PLWAP (Kadri and Ezeife 2011)	To retrieve the sequential patterns from uncertain weblog sequences.	Used PLWAP Technique to mine the uncertain weblog sequences which are derived from the user's access historical probabilities.	Not confined to internal utility (time spent on a webpage by an user) and external utility (rating of the webpage given by the website owner)
U-Span (Yin et al., 2012)	To obtain the high utility sequential patterns from the traditional databases	Adopted the lexicographic tree in the SPAM algorithm in the traditional sequential algorithm to the Utility criteria. Two pruning techniques are added to reduce the number of candidates generated.	Not confined to uncertain web access sequence databases. Follows the level wise approach which generates more number of candidates.
PHUI-UP (Lin et al., 2016)	To extract the high utility itemsets from the uncertain databases.	It adopts a level-wise candidate generation approach for mining high-utility sequential patterns	As Apriori-based approach is used, it may suffer from a high space and time complexities. Not confined to sequence databases.
HUU-PLWAP Miner (Proposed Framework)	To obtain the high utility sequential patterns from the uncertain sequence databases	The proposed miner uses the compact tree low database scan approach similar to U-PLWAP considering the probabilistic internal utility and constant external utility for mining high-utility uncertain sequential patterns.	Works only on the single set sequences (for example, < {a}, {b}, {c}>). The probabilistic value for the external utility is not defined.

Table 12: Summary of the existing systems with their limitations

The contributions of this thesis are therefore as follows:

1) Feature Contributions:

i. **Developing a Sequential Pattern Miner for Uncertain Data with Internal and external Utility**: HUU-PLWAP considers the items with lower existential probability whereas the U-PLWAP considers only find frequent sequential patterns. But there can be the pattern which is more profitable for the owner and least frequent, then these kind of patterns (which considers the internal (quantity) and external (quality) of the events in the sequence) are retrieved using the proposed miner

ii. **Using Possible World Sequence of an Item model to derive internal utility for more profitable patterns:** PHUI-UP (Lin et al., 2016) does not consider the internal probabilistic value where HUU-PLWAP calculates the uncertainty associated with the count of webpage retrieval along with the utility using on the sequence uncertainty model.

iii. **More time and space efficient approach based on PLWAP compact tree structure that reudces multiple scan of the database:** Existing approaches of High utility sequential mining such as USpan (Yin et al., 2012) have been extensively developed to handle precise data, which are not suitable to mine the sequence database with uncertainty and utility. Also, USpan (Yin et al., 2012) and PHUI-UP uses level wise approach for mining which can be more time to consume and a lot of memory consumption which can be improved by compact tree structure .

iv. **Probabilistic Internal Utility and Constant External Utility:** The PHUI-UP considers the uncertainty and utility as separate, so the patterns emerged can be different than the desired. The proposed HUU-UPLWAP considers the probabilities values in the internal utilities to retrieve the relevant patterns which can give the profit to the website owners in accessing the uncertain web access sequence database.

2). Procedural Contributions:

i. **High utility sequential mining method for uncertainty sequence database**: The proposed approach is the High utility sequential mining method for uncertainty sequence database. It follows the U-PLWAP methodology, but It considers both internal (time spent by the user on each web page which will vary in number) and external (the website rating given by the end users which is constant value) utilities of a web page which are not involved in U-PLWAP. It derives the internal utility value with uncertainty from the sequence uncertainty based model calculation.

ii. **Conditional Sequential Search**: The HUU-PLWAP algorithm recursively mines the HUU-PLWAP tree using the prefix conditional sequential search in the U-PLWAP along with uncertain internal utility and avoids the generation of the repetitive candidates, and multiple databases scans in the USpan. It also avoids the bitmap representation.

iii. **HUU-PLWAP uses the compact tree low database scan approach**: USpan is a vertical database bitmap transformation approach of the SPAM Algorithm (Ayres et al., 2002) in the sequential pattern approach in the frequent itemset mining and it uses the lexicographic tree approach for mining the patterns which can ne be more time consuming. The proposed miner applies the compact tree low database scan approach which is similar to the U-PLWAP tree (Kadri and Ezeife 2011) along with consideration of uncertain internal utility values and constant external utility values.

iv. **Sequence Based Uncertainty Model**: The proposed HUU-PLWAP algorithm derives the sequence database from the sequence based uncertainty model (which is based on the possible world model based on the item existence or not). These uncertain values are multiplied with the count of the webpage retrieval

(represents the internal utility) to get the uncertain internal utilities values used for mining the high utility sequential patterns.

1.9. THESIS OUTLINE

In Chapter 2, a detailed related work on Sequential Pattern Algorithms in Certain Data, and Uncertain Data in the sections 2.1 and 2.2. High Utility Mining and High Utility Sequential Mining are provided in sections 2.3 and 2.4. In Chapter 3, a proposed solution framework HUU-PLWAP is provided in section 3.1 with running example in section 3.2. In Chapter 4, experimental results including comparisons between the existing and proposed approach and the time complexity are provided. Finally, Chapter 5 provides some concluding remarks and future work.

2. RELATED WORK

Sequential pattern mining refers to the identification of frequent sub sequences in sequence databases as patterns. It provides an effective way to analyze the sequential data. The selection of interesting sequences is generally based on the frequency/support framework: sequences of high frequency are treated as significant. In the last two decades, researchers have proposed many techniques and algorithms such as APRIORI (Aggarwal and Srikanth 1995), FP-GROWTH ALGORITHM (Han et al., 2000), GSP (Srikant and Agrawal 1996), WAP (Pei et al., 2000), PL-WAP (Ezeife and Lu 2005) and so on, based on Apriori and Pattern growth methods for extracting the frequent sequential patterns, in which the downward closure property (also known as Apriori property) plays a fundamental role.

Some of the algorithms such as U-Apriori (Chui et al., 2007), UF-Growth (Leung et al., 2008), U-PLWAP (Kadri and Ezeife 2011) are also implemented in uncertain databases at the same time, the relative importance of each item has been introduced in frequent pattern mining, by which the “high utility itemset mining” has been proposed. Instead of selecting high frequency patterns, the utility-based methods extract itemsets with high utilities, and many algorithms and strategies such as U-Mining (Yao et al., 2004), Two Phase algorithm (Liu et al., 2005), PHUI-UP (Lin et al., 2016) ...have been proposed. These methods can only process the itemsets in the utility framework. Even, if there is an algorithm that considers the business impact (namely utility); it is very difficult to obtain the most valuable patterns from the high utility sequences based on a given minimum utility threshold. High utility sequential pattern mining has been considered as an important research problem and several relevant algorithms such as Novel Approach (Ahmed et al., 2010), USpan (Yin et al., 2012), HUSP (Zihayat et al., 2015), HUS-Span (Wang et al., 2014) have been proposed. The discussion in this chapter is categorized into the Sequential Pattern Mining approaches in certain data and in Uncertain data in

(section 2.1; section 2.2), High Utility Mining approaches (section 2.3) and High Utility Sequential Pattern Mining approaches (section 2.4) respectively.

2.1. SEQUENTIAL PATTERN ALGORITHMS IN CERTAIN DATA

Frequent Pattern Mining aims to discover how items purchased by customers in a supermarket with frequency no less than a user-specified threshold (m). For Example, Apriori algorithm (Aggarwal and Srikanth 1995) finds the set of frequent patterns iteratively by computing the support of each itemset in the candidate-1 itemset. An example is explained in Section 1.1. Another approach for Frequent Pattern Mining algorithm was Frequent Pattern tree (FP-Growth), which was explained in Section 2.1.1.

2.1.1. FP-GROWTH ALGORITHM (Han et al., 2000)

The problem of candidate set generation during frequent pattern mining process found that candidate set generation can be costly especially when many patterns are present and when such patterns are long. (Han et al., 2000) proposed a frequent mining algorithm, FP-growth, based on frequent pattern tree (FP-tree) data structure. The large database is compressed into FP-tree therefore removing repetitive database scan. This divide and conquer, conditional mining approach also remove candidate set generation.

Step 1: The FP-tree is built on the intuition that if frequent items are used to re-order items in the database, multiple transactions sharing same itemset can be represented by the same path in FP-tree by registering their counts.

Step 2: FP-tree is constructed after a first scan of the database is carried out where frequent items are found and ordered. The order is then used to enter items into the FP-tree during second scan.

Step 3: The mining process is done by starting from the bottom of the header table. The process is then repeated for all items on the header table.

Step 4: The conditional FP-tree is repeatedly mined when more than one frequent item is found.

Example: **Input:** The Transaction database in Table 13; minimum support is 3, Candidate 1-itemset = {a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p}.

Output: The Frequent Patterns.

Step 1: The items are re-ordered according to descending order of frequent items (f:4, c:4, a:3, b:3, m:3, p:3). Items h, l, j, k, l has been removed from the database since they are not frequent.

TID	ITEMS BOUGHT	(ORDERED) FREQUENT ITEMS
100	f, a, c, d, g, i, m, p	f, c, a, m, p
200	a, b, c, f, l, m, o	f, c, a, b, m
300	b, f, h, j, o	f, b
400	b, c, k, s, p	c, b, p
500	a, f, c, e, l, p, m, n	f, c, a, m, p

Table 13: Transaction database (Han et al., 2000)

The items are then inserted into the FP-tree in the ordered fashion as in **Step 2** and the FP-tree generated is as given in Figure 4.

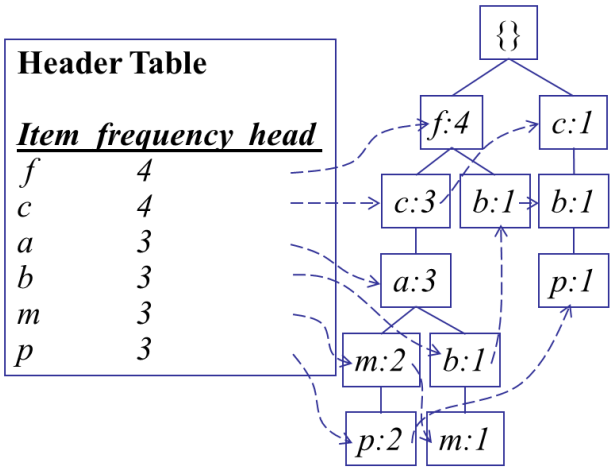

Figure 4: Constructed FP-Tree (Han et al., 2000)

As per **Step 3,** Item p has 2 tree paths f:4, c:3, a:3, m:2, p:2 and c:1, b:1, p:1. Removing p and ensuring only counts for item p are present gives p's conditional base tree: f:2, c:2, a:2, m:2, and c:1, b:1. Only item c make the minimum count of 3 (sum), therefore forms p's conditional FP-tree (c: 3). Pattern 'cp: 3' is therefore frequent. The process is then repeated for all items on the header table. The conditional FP-tree is repeatedly mined as in **Step 4** and the final output the frequent patterns generated are given in the Table 14.

ITEM	CONDITIONAL PATTERN BASE	CONDITIONAL FP-TREE
p	{(f:2, c:2, a:2, m:2), (c:1, b:1)}	{(c:3)} | p
m	{(f:4, c:3, a:3, m:2), (f:4, c:3, a:3, b:1, m:1)}	{(f:3, c:3, a:3)} | m
b	{(f:4, c:3, a:3, b:1), (f:4, b:1), (c:1, b:1)}	Ø
a	{(f:3, c:3)}	{(f:3, c:3)} | a
c	{(f:3))}	{(f:3)} | c
f	Ø	Ø

Table 14: The FP-growth mining process (Han et al., 2000)

Advantage: It resolved the problem of candidate set generation during frequent pattern mining process which was costly especially when many patterns are present and when such patterns are long.

Limitations: The execution time is more for the larger databases and it is not suitable for the non-sequential patterns.

2.1.2. GSP ALGORITHM (Srikanth and Aggarwal 1996)

(**Generalized Sequential Pattern** algorithm) is used for sequence mining. The algorithms for solving sequence mining problems are mostly based on the a priori (level-wise) algorithm. To discover all the frequent items in a level-wise fashion. It simply means counting the occurrences of all singleton elements in the database. Then,

the transactions are filtered by removing the non-frequent items. At the end of this step, each transaction consists of only the frequent elements it originally contained.

Step 1: This modified database becomes an input to the GSP algorithm. This process requires one pass over the whole database. It makes multiple database passes. In the first pass, all single items (1-sequences) are counted. From the frequent items, a set of candidate 2-sequences are formed, and another pass is made to identify their frequency.

Step 2 The frequent 2-sequences are used to generate the candidate 3-sequences, and this process is repeated until no more frequent sequences are found. The two main parts of the algorithm are **Candidate-Gen Joining phase,** where the candidates for the next pass are generated by joining $F_{(k-1)}$ with itself from a given set of frequent (k-1)-frequent sequences $F_{(k-1)}$, and the **Pruning phase** which eliminates any sequence, at least one of whose subsequences is not frequent. Finally, non-frequent sequences are removed. Example: **Input:** Web Access Sequence Database in the Table15; minimum support is 3, Candidate 1-itemset = {a, b, c, d, e, f}.

Output: The Sequential Patterns.

USERID	TID	WEB ACCESS SEQUENCES
100	T1	< abdac >
200	T2	< eaebcac >
300	T3	< babfaec >
400	T4	< afbacfc >
500	T5	< abad >

Table 15: Web Access Sequence Database

The first pass on the database determines the support for each item in finding frequent 1-sequences as in **Step 1** based on the minimum support of 3 records out of the 5 (i.e., 60%). The first (k = 1) scan over the Table 15 generates the set of candidates 1-sequences C_1 = {a:5, b:5, c:2, d:2, e:4}, giving the seed set of frequent 1-sequences L_1 =

{a, b, e}. The generate-and-test feature of candidate sequences in GSP has two phases explained in **Step 2**. During the join phase, candidate sequences are generated by joining L_{k-1} with itself using GSP-join. From the seed set of three 1-sequences L1 = {a, b, e}, a candidate set C_2 of twelve 2-sequences (3×3+ 3×2 2 = 12) is generated, giving an exploding number of candidate sequences, C_2 = {aa, ab, ae, ba, bb, be, ea, eb, ee, (ab), (ae), (be) }, where parenthesis denote contiguous sequences (i.e., no time gaps), the algorithm then scans the sequence database for each candidate sequence to count its support, after the pruning phase, to get L_2 = {ab:3, ae:4, ba:3, bb:3, be:4, (ab):3}, now GSP-join L_2 with L_2 (i.e., L_2 GSP L_2) to get C_3 = {aba, abb, abe, bab, bae, b(ab), bba, bbe, bbb, (ab)a, (ab)b, (ab)e}, and so on, until C_k = {} or L_{k-1} = {}. The set of mined frequent sequences is eventually fs = U kL_k. To show an example of pruning the contiguous subsequence L_3 = {(ab) c, (ab) d, a (cd), (ac) e, b (cd), bce}, then C4 = {(ab)(cd), (ab)ce}, and (ab)ce will be pruned because its contiguous subsequence ace is not in L3. The algorithm terminates when no new sequential pattern is found in a pass, or no candidate sequence can be generated.

Advantages: GSP employs a hash-tree to reduce the number of candidates that are checked for sequences. It is 2 to 20 times faster than Apriori-All.

Limitations: The GSP algorithm scans the original database and the problem of generating explosive candidate sets as in Apriori-like algorithms.

2.1.3. WAP TREE ALGORITHM (Pei et al., 2000)

WAP (web access pattern) tree is devised to register access sequences and corresponding counts compactly, so that the tedious support counting can be avoided. It also maintains linkages for traversing prefixes with respect to the same suffix pattern efficiently. The original access sequence database is not needed any more, because the size of WAP-tree is usually much smaller than that and the construction of WAP-tree is quite efficient as it scans the access sequence database only twice (Pei et al.,

2000).Instead of searching patterns level-wise as Apriori, WAP tree uses conditional search (a partition-based divide-and-conquer method), which narrows the search space by looking for patterns with the same suffix, and count frequent events in the set of prefixes with respect to condition as suffix. The main steps involved in this technique are summarized:

Step 1: The WAP-tree stores the web log data in a prefix tree format as the frequent pattern tree (FP-tree) for non-sequential data. The algorithm first scans the web log once to find all frequent individual events.

Step 2: Secondly, it scans the web log again to construct a WAP-tree over the set of frequent individual events of each transaction and finds the conditional suffix patterns.

Step 3: Constructs the intermediate conditional WAP-tree using the pattern in **Step 2**.

Step 4: Finally, it goes back to repeat Steps 3 and 4 until the constructed conditional.

Example: **Input:** Web Access Sequence Database in the Table16; minimum support is 3, Candidate 1-itemset = {a, b, c, d, e, f}.

Output: The Sequential Patterns.

USERID	WEB ACCESS SEQUENCES	FREQUNET SUBSEQUENCES
100	abdac	abac
200	eaebcac	abcac
300	babfaec	babac
400	afbacfc	abacc

Table 16: A Sample database of web access sequences (Pei et al., 2000)

From **the Steps** explained above, the sequence ‘abac’ is inserted into the initial tree with only one virtual root. It creates a new node (a: 1) (i.e., labeled as a, with count set to 1) as the child of the root, and then derives the branch a \ (a: 1)! (b: 1)! (a: 1)! (c: 1)", in which arrows point from parent nodes to children ones. Second, the second sequence abcac is inserted. It starts at root. Since the root has a child labeled a, a's count is increased by 1, i.e., (a: 2) now. Similarly, for (b: 2), the next event, c, does not match the existing node a,

and a new child node c: 1 is created and inserted. The remaining sequence insertion process can be derived accordingly and shown in the Figure 5.

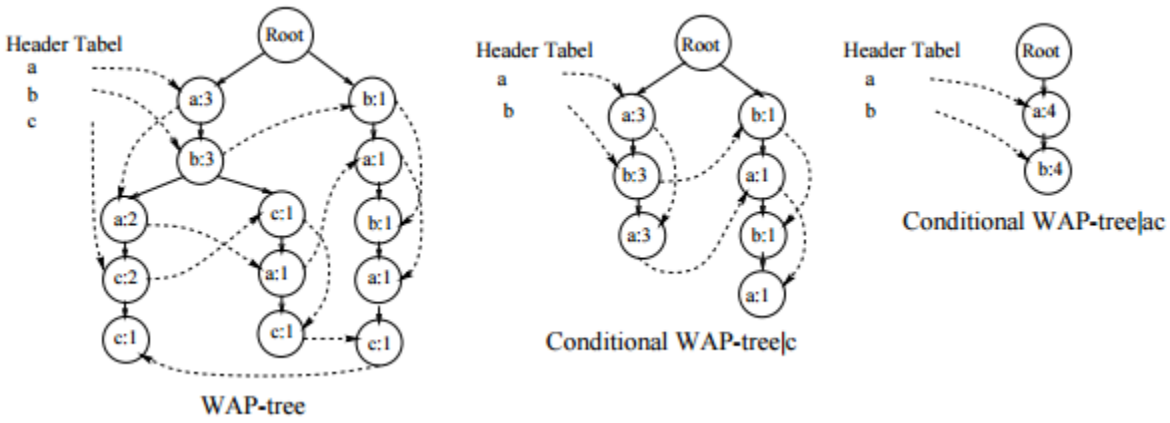

Figure 5: Example of web access pattern tree (Pei et al., 2000)

2.1.4. PL-WAP ALGORITHM (Ezeife and Lu 2005)

To eliminate the need of recursively construction intermediate trees in WAP tree, **PLWAP algorithm** was proposed, which uses position codes generated for each node such that antecedent/descendant relationships between nodes can be discovered from the position code (Ezeife and Lu 2005). The concept of binary tree is used in generating the position codes. The root has a position code of null. Starting from the root, all tree nodes are assigned a position code using the following rule. The position code of the leftmost child node is the position code of its parent concatenated with '1' at the end; the position code of any other node is the same as appending '0' to the position code of its closest left sibling (Ezeife and Lu 2005). The steps explaining the algorithm are:

Step 1: The algorithm first scans the web access sequence database (WASD) to obtain support count for all events in it. Events with support greater than or equal to a specified threshold are said to be frequent.

Step 2: A second scan is then used to eliminate non-frequent events from the original sequences. These new sequences are then used to construct a PLWAP-tree with each node representing label, count and position code of the event along a particular path.

Step 3: The PLWAP-tree is then traversed in a pre-ordered fashion starting from the root to the left sub-tree followed by the right sub-tree to build to build the header node linkages. Each event has a queue linking all its nodes in the order in which they are inserted. The head of each queue is registered in a header table for the PLWAP tree.

Step 4: The PLWAP algorithm then recursively mines the PLWAP tree using prefix conditional sequence search and generates the frequent sequential patterns.

Example: **Input:** Web Access Sequence Database in the Table17; minimum support is 75%, Candidate 1-itemset = {a, b, c, d, e, f}.

Output: The Sequential Patterns.

TID	Web Access Sequences	Frequent Subsequences
100	abdac	abac
200	eaebcac	abcac
300	babfaec	babac
400	afbacfc	abacc

Table 17: The Web Access Sequence Database (Ezeife and Lu 2005)

From **Step 1**, since events ‘d’ ,’ e‘ and ‘f ‘ are 50% frequent, they are removed in frequent sub sequences and the frequent 1-sequnces generated are a,b,c. **Step 2:** The resulting reduced sequences are then used to build PLWAP tree as shown in Figure 6. As in **Step 3**, the mining starts by mining sequences with the same prefix. Starting from frequent 1-sequence, the PLWAP algorithm mines starting from the root (for the first time). Using the header linkage, it traverses the tree to identify frequent 1-sequences by searching for first occurrence of an event say ‘a’. The position code then helps in preventing duplicate count of support of an event in the same suffix sub tree. The addition of these counts is then used to compare with the specified minimum support. The event ‘a’ is counted with nodes

a: 3:1 and a: 1:101. The total count for ‘a’ is 4, making it a frequent 1-sequence. To find the next frequent sequence with ‘a’ as prefix, the PLWAP tree is rooted at b: 3:11 and b: 1:1011. The first occurrence of say ‘a’ in these sub-trees is then noted with the counts. For ‘a’ event, the following nodes are identified using b: 3:11 and b: 1:1011 as roots: a: 2:111, a: 1:11101 and a: 1:10111. Again ‘a’ occurs 4 times, making it frequent. Therefore, ‘aa’ is frequent sequence.

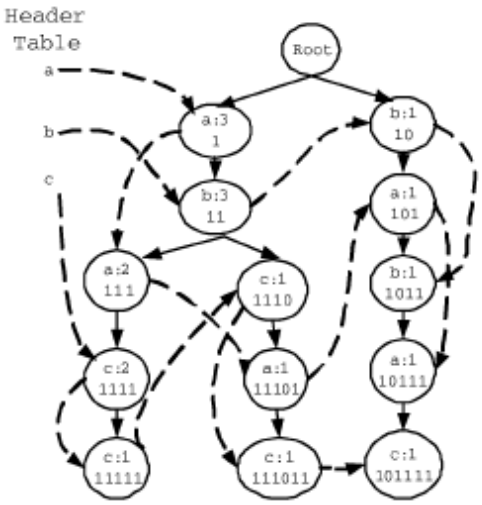

Figure 6: The PLWAP tree with the header linkages (Ezeife and Lu 2005)

The next 3-sequence with ‘aa’ as its prefix can be found by setting the root of the PLWAP tree to c: 2:1111, c: 1:111011 and c1:101111. There are no other events except ‘c’ with counts of c: 2:1111, c: 1:111011 and c1:101111, making it a total of 3, which makes ‘c’ frequent. Therefore, ‘aac’ is frequent. Shifting the root to c: 1:11111 gives no other frequent event. Step 4: The algorithm after recursive mining then backtracks to find other possible frequent sequence combinations at each level. The same procedure is followed and frequent sequences found are (a,aa, aac, ac, ab, aba, abac, abc, b, ba, bac, bc, c).

Advantages: This algorithm includes efficiency in terms of I/O and memory utilisation since it eliminates the need to store intermediate WAP tree. The use of pre-order linking of header nodes of the same suffix tree also makes searching of nodes easier.

Limitations: There is a necessity for exploring the new methods to reduce or eliminate false positive results from mined results for applications needing highly precise results.

2.2. SEQUENTIAL PATTERN ALGORITHMS IN UNCERTAIN DATA

In the real-life applications, uncertainty may be introduced when data is collected from noisy data sources such as RFID, GPS, wireless sensors, and Wi-Fi systems (Aggarwal 2010) When applied to incomplete or in accurate data, traditional pattern mining algorithms (e.g., FIM, ARM) cannot be applied to discover the required information (Chun et al., 2007). Many algorithms have been developed to discover useful information in uncertain databases (De et al., 2013). The U-Apriori algorithm was first proposed to mine frequent itemsets in uncertain databases using a generate-and-test and breadth-first search approach (Chui et al., 2007). The Algorithm U-PLWAP tree (Kadri and Ezeife 2011) is constructed to mine the sequential patterns in the uncertain sequences without generating the candidate sequences, recursive construction of the intermediate tree and the necessity of the scan the sequence database repeatedly. All the mentioned algorithms are explained in the related work section 2.2.1 and 2.2.2.

2.2.1. U-APRIORI ALGORITHM (Chui et al., 2007)

The algorithm explained for an uncertain dataset D consists of d transactions [t_1,,. t_d]. A transaction t_i contains a number of items. Each item x in t_i is associated with a non-zero probability Pt_i (x), which indicates the likelihood that item x is present in transaction t_i. There are thus two possibilities of the world. In one case, item x is present in transaction t_i; in another case, item x is not in t_i. These two possibilities represent the two possible worlds, W_1 and W_2, respectively. As the real world is unknown, the probability P (Wi) be the probability that world W_i being the true world, then P (W_1) = Pt_i (x) and P(W_2) =1 − Pt_i (x). For example, let item q be another item in ti with probability Pt_i (q). If the observations of item p and item q are independent, then there are four possible worlds. The probability of the world in which t_i contains both items p, q for example, is Pt_i (p) •Pt_i (q) (Chui et al., 2007). It works similar to the process of Apriori explained in the section 1.1. But the only difference was it derives the patterns from uncertain data.

Advantages: U-Apriori works like Apriori (downward closure property) and proves that it is also applicable in uncertain databases in which very subset of a frequent item set is frequent and vice versa.

Limitations: Generates enormous number of candidates and does not fit for large datasets.

2.2.2. U-PLWAP TREE ALGORITHM (Kadri and Ezeife 2011)

Uncertain Position Coded Pre-Order Linked Web Access Pattern (U-PLWAP) algorithm is proposed to provide solution in mining data that are associated with uncertainty. The uncertainty can be introduced, for example, in web logs, based on web log histories of different users. It is noted here that unlike in traditional precise sequences where occurrence count of an item automatically contributes towards the support count of an item, the sum of the product of the existential probability values are used in arriving at support counts. An equally important observation with uncertain data item is that items with the same label can have different existential probability values but higher value was considered. It is therefore important to record item's label, occurrence count and existential probability values to accurately determine frequent sequences. The steps of algorithm are:

Step1: The U-PLWAP scans the sequence database to discover the frequent 1-sequences. This is done by adding up all existential probability values for each item whenever they occur in the database. Whenever an item is repeated in a particular sequence, higher existential probability is considered. The frequent 1-sequences are those items with counts greater than or equal to the minimum support threshold.

Step2: Each of the frequent 1-sequences is used to create entries in the header table.

Step3: A second scan is used to create U-PLWAP tree after the non-frequent data items have been removed from the sequence. The U-PLWAP tree is created starting from a null root. Sequences are read from the database and nodes are created for each item in the

sequence. Each node contains the item's label, multiple occurrence counts and their corresponding existential probability and position code, denoted as label: count: position code. Since similar labels with different existential probabilities in a path are merged into one node, each sequence read is identified by its sequence ID and recorded against the existential probabilities of its items in every node. A left node is created if no node already exists, otherwise, a right node is created and the count initialised to 1. If node already exists, its count is incremented by 1. The item label and its existential probabilities are read from the sequence database. Entries created in the header table are then used to link their corresponding nodes by traversing the U-PLWAP tree in a pre-ordered fashion (from root to left node first before right node).

Step4: The U-PLWAP tree created is then mined recursively using prefix conditional search. Starting from the root with a particular frequent item 'a' (in order to find all frequent items starting with a) on the header table, all sub-trees are traversed to search for the first occurrence of item 'a'. Its expected support count is calculated by summing all existential probabilities recorded for each sequence ID for all nodes found. Item 'a' is frequent if its expected support count is greater or equal to the specified minimum support count. The sub-trees are rooted at this point. A sequence of cumulative product of existential probabilities of items contained in each sequence ID from the root of U-PLWAP tree to all the new roots is then generated. Since this is the only item found so far, the entries in the sequence of the cumulative product of the existential probabilities are the same as the values of the existential probabilities for each sequence ID present in the new root found. The search continues down the tree to find another frequent sequence say 'aa'. The same process is repeated once another first occurrence of 'a' is found on all sub-trees. The sequence of cumulative product of the existential probability of items found for each sequence ID is then updated with the product of the existential probability of the newly found 'a' and the last cumulative product of existential probability having the same sequence ID. The expected support count of 'aa' is found by summing all entries in the

updated sequence of cumulative products of existential probabilities. If no such item 'a' is found, the algorithm backtracks to the last root and search for another item 'b'. The process continues recursively until no more items are found. The algorithm then backtracks to the null root to start mining for sequences starting with a new item from header table.

Example: **Input:** Web Access Sequence Database in the Table18; minimum support is 25%, Candidate 1-itemset = {a, b, c, d, e, f}. **Output:** The Sequential Patterns.

From the **Step1**, Assuming the minimum support value is 1 out of 4 tuples that is 25% in Table 18. The expected support count values for all the items are calculated by adding up the existential probabilities of the item in each tuple. For example, the existential probability of item 'a' in user ID 10 is 1. The same value (1) is present in user IDs 20, 30 and 40. The expected support count of item 'a' is therefore given as 1 + 1 + 1 + 1 = 4. In the same vain, item 'b' has existential probabilities 0.5, 0.25, 0.5 and 1 in user IDs 10, 20, 30 and 40 respectively. The support count of item 'b' is therefore calculated as 0.5 + 0.25 + 0.5 + 1 = 2.25. The same process is used to find expected support counts for items 'c', 'd', 'e' and 'f' as: Item c = 0.75 + 0.25 + +0.75 + 0.5 = 2.25; Item d = 0.5 + 0.5 + 0.5 +0.25 = 1.75; Item e = 0.2 + 0.5 = 0.7 Item f = 0.2; Items 'e' and 'f' are non-frequent, so pruned.

User ID	Web logs	Frequent sub-sequences
10	(a:1, b:0.5, c:0.75, d:0.5)	(a:1, b:0.5, c:0.75, d:0.5)
20	(a:1, b:0.25, d:0.5, c:0.25, d:0.5, e:0.2)	(a:1, b:0.25, d:0.5, c:0.25, d:0.5)
30	(a:1, b:0.5, c:0.75, d:0.25, e:0.5)	(a:1, b:0.5, c:0.75, d:0.25)
40	(b:1, c:0.5, a:1, d:0.5, c:0.5, f:0.2)	(b:1, c:0.5, a:1, d:0.5, c:0.5)

Table 18 : Sample uncertain sequence (Kadri and Ezeife 2011)

The frequent items found as in **Step 2** were a, b, c and were the entries in the header table which serves as the input to the U-PLWAP tree. From the **Step 3** in the algorithm, the U-PLWAP tree is then built by first creating the root which is null. The first sequence

a: 1, b: 0.5, c: 0.75, d: 0.5 is entered into the tree as the leftmost sub-tree since there no previously existing nodes. The count for each node initialised to 1 and their corresponding existential probability value are recorded. Each of the existential probabilities is also recorded against the sequence ID. The position codes for the nodes are given using the rule specified in step 2 of section 3.3 will explain the process. The second sequence a: 1, b: 0.25, d: 0.5, c: 0.5, d: 0.5 is read.

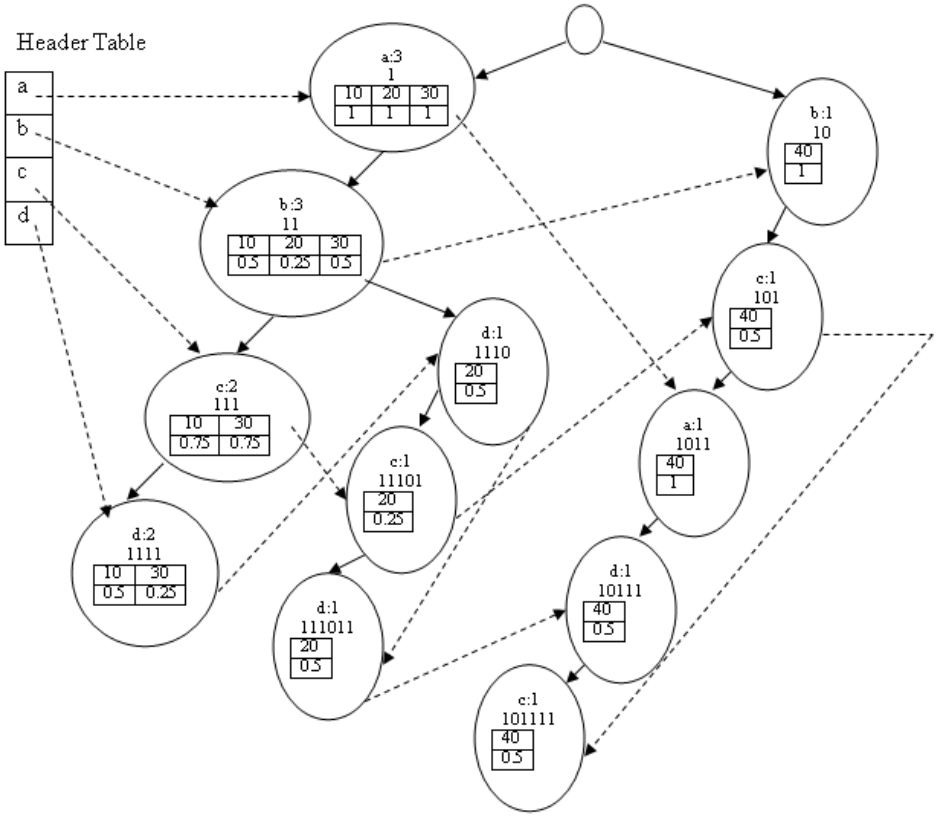

Figure 7: U-PLWAP Tree Constructed from the Example in the Table 18

Starting from the root, since a node with label ‘a’ already exist with the same existential probability, the count of this node is incremented by 1. Its sequence ID is recorded against its existential probability. Item ‘b’ is read. The existential probability 0.25 is recorded against sequence ID 20 and the count of label ‘b’ set to 2. Item‘d’ is read. Since no node exists with label‘d’ along this path, a new child of node b: 2:11 is created. New nodes are also created for items ‘c’ and‘d’. The other two sequences are read following the same. Header table linking the corresponding nodes in a pre-ordered fashion (by visiting the

root, leftmost sub-tree first and then right sub-tree) is also created. The completed linked U-PLWAP tree is constructed as in Figure 7. As per **Step 4,** further mining starts with frequent sequences starting with ‘a’. From the root, the U-tree is traversed for the first occurrence of a in the suffix trees of the root node. In this case: a: 3:1 and a: 1:1011. The expected support of ‘a’ is then calculated as (1 +1+1) + (1) = 4. This confirms ‘a’ is frequent since 4 are greater than the minimum support 1. A sequence of cumulative products of existential probability of all items found so far is created at this point. Since these are the first set of items, the sequence is (1, 1, 1, 1) representing each of the 4 paths 10, 20, 30 and 40 respectively. The tree is then rooted at points a: 3:1 and a: 1:1011 and their suffix trees are traversed for another occurrence of 'a' to check if ‘aa’ is frequent. No occurrence of ‘a’ is found. The algorithm then backtracks to the roots a: 3:1 and a: 1:1011. A new sequence ‘ab’ is then checked by traversing the suffix trees to search for the first occurrence of ‘b’. Item ‘b’ is found at b: 2:11. The suffix trees are then rooted at this point. The expected support for ‘ab’ is then calculated by summing the product of entries representing each path in the cumulative product sequence and the existential probability values of items in the new found ‘b’. The expected support for ‘ab’ is then found as: (1 x 0.5) + (0.25 x 1) + (1 x 0.5) = 1.25. Since 1.25 is greater than 1, ‘ab’ is frequent. The entries of the new sequence of cumulative product of existential probability is then generated as (0.5, 0.25, 0.5) representing the paths 10, 20 and 30 respectively. Using the same technique, ‘aba’, and ‘abb’ were not found. Item ‘c’ is found at nodes c: 2:111 and c: 1:11101. The suffix trees are rooted at these nodes. The expected support of ‘abc’ is 0.8125, so it was not frequent. The process is recursively repeated following the same logic and the frequent sequences found are: a, ab, ac, ad, b, bc, ba, bd, c, d.

Advantage: The PLWAP tree algorithm is successfully implemented in the uncertain data with the same methodology.

Limitation: Limitation: The algorithm does not apply on utility. The support count, which is the sum of the all existential probability values for each item, sometimes involve in low existential probabilities with high profitable value.

2.3. HIGH UTILITY ITEMSET MINING

The term “mining high utility itemsets” first appeared in (Chan et al., 2003), but the concept and definition of high utility data mining was quite different. It is widely believed that utility based itemset mining, sequence mining and web mining originated in (Yao et al. 2004).

2.3.1. FOUNDATIONAL APPROACH (Yao et al., 2004)

It is widely believed that this was the first and foundational paper of high utility pattern mining. The authors first defined the problem of mining high utility itemsets, and a theoretical model of utility mining was proposed. Specifically, two types of utilities for items, namely internal utility and external utility were first proposed (Tseng et al., 2013).

Example: **Input:** Table 19 is the transaction table (input database D) where the items in each transaction are associated with an internal utility. The quality table in the Table 20, 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}. **Output:** The High Utility Itemset Patterns.

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 19: Transaction Database

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

Table 20: Quality Table

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 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. The downward closure property does not hold in high utility pattern mining. The property states that a pattern's support is no less than that of its super-pattern. However, when it comes to the utility framework as in the examples 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 any more.

Advantages: A utility upper bound called Expected Utility for the itemset is introduced, which can be used to prune unpromising candidates.

Limitations: It suffers from the large candidate generation process with more memory consumption and execution time. It fails to follow the downward closure property.

2.3.2. THE TWO-PHASE ALGORITHM (Liu et al., 2005)

It is the most important high utility based algorithm which introduced the downward closure property in the utility mining framework. The Two-Phase algorithm has two phases. In the **first phase**, it generates a set of high utility candidates (Manike and Om 2015) and all the high utility itemsets are in this set. In the **second phase**, an extra database scan is performed to filter the high utility itemsets from the candidates. The key contribution of Two-Phase is the proposal of the Transaction-Weighted Downward

Closure Property. The Transaction-weighted Downward Closure Property (TDCP) is quite likely to Downward Closure Property) in traditional frequent pattern mining. TWU is short form for Transaction-Weighted Utilization which was defined as the sum of the transactions' utilities which contain the itemset (Liu et al. 2005). The equation of it is not very difficult to understand as it alike to the calculation in the Apriori property. Therefore, suppose HTWU to be the set of itemsets whose TWU is no less than the minimum utility threshold ξ, and HU to be the high utility itemsets. Then HU ⊆ HTWU. The Two-Phase algorithm extracts the HTWU in the first phase with the help of TDCP (Transaction-weighted Downward Closure Property), then it scans for HU in the second phase. Example: **Input:** Transaction Database and Profit Table in the Figure 8; Minimum Utility Threshold, (MUT) = 30; High Utility Candidate 1-itemset = {p1, p2, p3, p4, p5};

Output: The High Utility Itemset Patterns.

T_{id}	Transactional path	TU	Items(pages)	TWU
T_1	$(p_3,1),(p_1,1),(p_4,1)$	8	p_1	65
T_2	$(p_3,6),(p_5,2),(p_1,2)$	27	p_2	61
T_3	$(p_3,1),(p5,1),(p1,1),(p_2,2),(p_4,6)$	30	p_3	96
T_4	$(p_3,3),(p_5,1),(p_2,4),(p_4,3)$	20	p_4	58
T_5	$(p_3,2),(p_5,1),(p_2,2)$	11	p_5	88

Item	p_1	p_2	p_3	p_4	p_5	p_6	p_7
Unit profit	5	2	1	2	3	1	1

Figure 8: Transaction Database and Profit Table (Bakariya and Thakur 2015)

During the **first phase**, the utility of an item in transaction depends on quantity and profit value. For example, the utility of item {p1} in the transaction T1 is 5. The utility of an itemset (i.e. more than one item) X is a summation of a utility for all the items contained in X. For example, a utility of itemsets {p1p4} in the transaction T1 is 7, and utility of an itemset refers the total profits of the itemset in the database; for example, an itemset {p1p4} appears in transactions T1 and T3. Utilities in T1 and T3 are 7 and 17, respectively; therefore, a utility of the itemsets {p1p4} is 24. An itemset is called as a high utility itemset

if and only if its utility is no less than a user-specified threshold otherwise that itemset is called a low utility itemset. For example, {p2p4}:30 are a high utility itemset but {p2} is a low utility itemset; our goal is to find all the high utility itemsets from a database under a user-specified minimum utility threshold (MUT).

During **second phase**, an itemset X is called high utility itemset if U (X) > = MUT otherwise those itemsets are discarded. TU (Transaction Utility) and TWU (Transaction Weighted Utility) can be calculated using support and profit value of an item as shown in Figure 9. For example, if Minimum Utility Threshold, (MUT) = 30. There are following high utility itemsets, having its minimum utility 30 or more. The values of an itemsets calculated are: {p2p4}: 30, {p2p3p5}: 31, {p1p3p5}: 31, {p2p3p4}: 34, {p2p4p5}: 36, {p2p3p4p5}: 40 and {p1p2p3p4p5p6}: 30.

T_{id}	p_1	p_2	p_3	p_4	p_5	p_6	p_7	TU
T_1	1	0	1	1	0	0	0	8
T_2	2	0	6	0	2	0	5	27
T_3	1	2	1	6	1	5	0	30
T_4	0	4	3	3	1	0	0	20
T_5	0	2	2	0	1	0	2	11
Profit	5	2	1	2	3	1	1	
TWU	65	61	96	58	88	30	38	

Figure 9: Transaction Table

The main challenge in utility mining is that the downward closure property (superset of a low utility itemset may be a high utility itemset) cannot be applied. For example, {p2} is a low utility itemset but its superset {p2p4} is a high utility itemset because item utility of {p2} is 16 (low utility itemset because 16 <= 30) but the superset of {p2} itemset is {p2p4} and utility of {p2p4} is 30 (high utility itemset because 30>=30). Therefore, pruning search space is difficult in utility mining. Transactional Utility (TU) in every transaction has been calculated. The transaction utility for each transaction TU (T_d) =U (T_d, T_d), for Transactional utility for T1 was TU (T1) =U (T1, T1) = ({p1p3p4}, T1) =8. Similarly, all

transaction utility for all transactions is calculated. HUIM algorithm computes a utility for each transaction. For example, the utility of the transaction T1 is 8. Then it finds all the items and their TWUs. The TWU of an item is the summation of the utilities of all the transactions containing the item. For example, the TWU of item {p1} is 65, since item {p1} appears in T1, T2, and T3 (U (T1, T1) + U (T2, T2) + U (T3, T3) =. 8+ 27+ 30=65), similarly the TWU of item {p2} is 61 (U (T3, T3) + U (T4, T4) + U (T5, T5) = 30+ 20+ 11= 61). The high utility itemsets which follows the rule (TWU>=MUT), are accurate and profitable.

Advantages: Upper bound TWU is tighter than the Expected Utility. Because of the simplicity, clean nature and low complexity.

Limitations: It involves in multiple scans of the database which can increase the runtime and the memory consumption.

2.3.3. PHUI-UP ALGORITHM (Lin et al., 2016)

The potential high-utility itemsets upper-bound-based mining algorithm (PHUI-UP) in uncertain databases is proposed to efficiently discover not only the itemsets with high utilities but also the itemsets with high existence probabilities in an uncertain database. The PHUI-UP algorithm is first presented to mine (PHUIs) using a level-wise search. Since PHUI-UP adopts a generate-and-test approach to mine PHUIs, it suffers from the problem of repeatedly scanning the database. The “utility” can be viewed as the user-specified importance, i.e., weight, cost, risk, unit profit or value; Traditional HUIM algorithms do not consider existence probabilities. It preserves the downward closure property from Two-phase model, thus reducing search space for finding PHUIs.

Step 1: The proposed PHUI-UP algorithm takes input as Uncertain Transaction database and scans the database to find the TWU (Transaction Weighted Utility) and the probabilities values (Pro) of all 1-itemsets. The utility and transaction utility and the weighted transaction utilities are calculated.

Step2: Once all 1-itemsets satisfies TWU(X) ≥ min utility threshold (MUT) and Pro(X) ≥ min probability threshold (MPT), they are all put into the set of potential high transaction-weighted utilization itemsets (HTWPUIs). Then k is set to 2, and the candidates C2 are generated by applying Apriori-gen (HTWPUI1) using the lexicographic order of items.

Step 3: The uncertain database is then res-canned to calculate the TWU and Pro values of each itemset in C2. The results should satisfy TWU(X) ≥ MUT and Pro(X) ≥ MPT.

Step 4: These itemsets are thus added to the set of HTWPUIs. Then k=3, and the procedure is applied in the same way. The first phase terminates when no candidate is generated. An additional database rescans required in the second phase to find the final PHUIs from the candidate HTWPUIs. Based on the designed TWPUDC property, (which ensures that no supersets of small transaction-weighted probabilistic and utilization itemsets are in the preliminary candidate set (correctness)) it can extract the complete PHUIs from the candidate HTWPUIs (completeness).

Example: **Input:** uncertain transaction database in the Table 21 and Profit Table with the external utilities {A: 4, B: 1, C: 12, D: 6, E: 15} ; Minimum Utility Threshold, (MUT) = 110 and the Minimum Potential Probability Threshold (PRO) = 1.5; High Utility Candidate 1-itemset = {A, B, C, D, E} ; **Output:** The High Utility Itemset Patterns.

TID	TRANSACTIONS	PROBABILITY
1	(A,2), (C,3), (E,2)	0.9
2	(B,1), (D,2)	0.7
3	(A,1), (B,2), (C,1), (E,3)	0.85
4	(C,2)	0.5
5	(B,3), (D,2), (E,1)	0.75
6	(A,2), (C,2), (D,5)	0.7
7	(A,1), (B,1), (D,4), (E,1)	0.45
8	(B,4), (E,1)	0.36
9	(A,3), (C,3), (D,2)	0.81
10	(B,2), (C,3), (E,1)	0.6

Table 21: Uncertain Transaction database

Step 1: The utility and transaction utility and the weighted transaction utilities are calculated. The minimum utility threshold and the minimum potential probability threshold are respectively provided. In the example in Table 21, the values are {A:303,3.71; B:222,3.71; C:336,4.36; D:209,3.41; E:283,3.91}. The minimum utility threshold is set to 25% and that the minimum potential probability threshold is set to15%.

Step 2: Once all 1-itemsets satisfies TWU(X) ≥110.5 and Pro(X) ≥1.5, they are all put into the set of HTWPUI's. Then k = 2, then the candidates C2 by applying Apriori_gen (HTWPUI1) using lexicographic order of items. The set C2 is {AB; AC; AD; AE; BC; BD; BE; CD; CE; DE}.

Step 3: The uncertain database is then rescanned to calculate the TWU and Probability values of each itemset in C2. The results are {AB:107, 1.3; AC:259,3.26; AD:122,1.51; AE:181,2.2; BC:116,1.45; BD:87,1.9; BE:190,2.05; CD:122,1.51; CE:190,2.35; DE:74,1.2}. Among these, only the item-sets {AC:259,3.26; AD:122,1.51; AE:181,2.2; BE:190,2.05; CD:122,1.51; CE:190,2.35} satisfy TWU(X)≥110.5 and Pro(X)≥1.5.

Step 4: These itemsets are thus added to the set of HTWPUIs. Then k=3 and the procedure is applied in same way. The first phase terminates when no candidate is generated. Then an additional database rescans are performed to find the final PHUIs from the candidate HTWPUIs. The results of the PHUI-UP algorithm are {C:168,4.36; E:135,3.91; AC:140,3.26; BE:117,3.01; CE:174,2.35; ACD:122,1.51; ACE:135,1.75}.

Advantages: It is the first algorithm to mine high-utility itemsets from an uncertain transaction database.

Limitations: As Apriori-based approach is used for generating candidates at each level, it may suffer from a high space and time complexities. Also, it considers the probability as separate entity not appended with the utility measure. It deals with transaction database not with the sequence database.

2.4. HIGH UTILITY SEQUENTIAL PATTERN MINING

While high utility itemset mining has been extensively studied, the incorporation of the high utility concept into sequential pattern mining has also begun (Yin 2015). The UL algorithm proposed by (Ahmed et al., 2010) is directly dependent on the maximum length of the candidate sequences as it adopts the level-wise Apriori mechanism. On the other hand, the number of database scans required for the US algorithm proposed by (Ahmed et al., 2010) is totally independent of the maximum length of candidate sequences. USpan (Yin et al., 2012) is one of the high utility sequential pattern mining algorithms which were explained in brief in the section 2.4.2. An improved projection-based algorithm is proposed with an effective pruning strategy (IPA) to discover high sequential utility patterns in a quantitative sequence database in (Lan et al., 2012).

2.4.1. A NOVEL APPROACH FOR MINING HIGH-UTILITY SEQUENTIAL PATTERNS IN SEQUENCE DATABASES (Ahmed et al., 2010)

Existing approaches are not capable of mining high-utility sequences. So, a novel approach is proposed two algorithms such as **UL (Utility Level)** and **US (Utility Span)** algorithms

1. UL (Utility Level) algorithm: UL is simple and straightforward, used for generating high-utility sequential patterns.

Input: A sequence database SDB with utility values, minSeqUtil, k – a maximum sequence

Output: The complete set of high-utility sequential patterns.

Step 1: At first, it generates the candidates for high-swu sequences, and subsequently generates actual candidates for high utility sequential patterns. Finally, it detects the high-utility sequential patterns from high-swu sequences. In level-1, all the distinct items are candidate length-1 high-swu sequences. The UL algorithm scans the SDB once to generate all the length-1 high-swu sequences.

Step 2: In level-2, UL generates length-2 candidate sequences by joining it with other length-1 high-swu sequences which include it. On the other hand, a length-1 candidate sequence joins with others to form length-2 single element candidate sequences. The original SDB must be scanned once again with all these candidate sequences to detect the high-swu sequences. A sequence X joins with another sequence Y if the subsequence obtained by dropping the first item of X is the same as the subsequence obtained by dropping the last item of Y. The new candidate sequence is formed by joining sequence X with the last item of Y. The added item becomes a separate element if it was a separate element of Y, and otherwise, part of the last element of X.

Step 3: From level-3 to the last level, the UL algorithm generates high-swu candidate sequences for k^{th} level by joining L_{k-1} (high-swu sequences for $(k–1)^{th}$ level) with L_{k-1}.

Step 4: After generating all the candidate high-swu sequences by joining, UL algorithm prunes those candidate sequences having at least one subsequence which is not a length-(k–1) high-swu sequence.

An Example explaining the **UL algorithm:**

Input: A sequence database SDB in the Table 22 with internal and external utility values, minimum sequence utility threshold, δ=30%; minSeqUtil = δ× sequence utility (SDB), k – a maximum sequence.

Output: The complete set of high-utility sequential patterns.

The Table 22 shows an example SDB with internal and external utility values. Here, the internal utility values represent the quantities of items in sequences (e.g., iu (b, S_1) =6), and the external utility value of each item represents profit ($) per unit of that item (e.g., eu (b) =7). However, an item may appear multiple times in TS. In that case, iu (i_j, S_k) is the sum of all quantities of i_j in sequence S_k. For example, in Table 22, iu (a, S_1) =10.

Sequence ID	Sequence with internal utility	Sequence utility ($)
S_1	$a(3)$ $\{a(2)$ $b(6)$ $d(2)\}$ $f(1)$ $a(5)$ $d(1)$	130
S_2	$e(3)$ $\{a(2)$ $b(5)\}$ $d(1)$ $c(4)$	85
S_3	$\{c(1)$ $f(2)\}$ $b(3)$ $\{d(1)$ $e(4)\}$	74
S_4	$a(2)$ $\{b(7)$ $d(4)\}$ $\{a(6)$ $b(3)\}e(5)$	180
S_5	$\{d(1)$ $f(3)\}$ $c(5)$ $g(2)$	67
S_6	$d(2)$ $e(1)$ $\{a(7)$ $b(8)\}$ $d(3)$ $b(6)$ $e(3)$	207

(a)

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

(b)

Table 22: Examples of (a) sequence database (b) external utility (Ahmed et al., 2010)

Sequence Utility is calculated. For example: su (b, S_1) =6×7=42; su (de, S6) = (2×10+1×6) + (3×10 +3×6) =26+48=74. Sequence utility of Sequence TS_k is defined. For example, su (TS_1) =su (a, S_1) +su (b, S_1) +su (d, S_1) +su (f, S_1) = 50+42+30+8 =130. Sequence utility of a sequence X in an SDB is defined. For example, su (a (bd) a, SDB) =su (a (bd) a, TS_1) + su (a (bd) a, TS_4) = 102+129 = 231. Sequence utility value of an SDB is defined as, for example su (SDB) =743. The minimum sequence utility threshold, δ, is given by the percentage of sequence utility value of the database. In Table 22, if δ is 30% or 0.3, then the minimum sequence utility value can be defined as minSeqUtil = δ× su (SDB). Hence, in this example, minSeqUtil =0.3×743=223. A sequence X is a high-utility sequential pattern if su(X) ≥ minSeqUtil. Mining high-utility sequential pattern (husp) means discovering all sequences X having criteria su(X) ≥ minSeqUtil. For minSeqUtil=223, a (bd) a is a husp as su (a (bd) a) =231. The problem addressed that for high-utility sequential pattern mining is that sequence utilities do not have the downward closure property. Consider minSeqUtil=223, in which "a" is a low-utility sequential pattern as su (a) =135, but its super-sequence a (bd) a is a high-utility sequential pattern as su (a (bd) a) =231. Therefore, the downward closure property is not satisfied. To maintain the downward closure property in high-utility sequential pattern mining, a new measure called sequence-weighted utility (swu) is derived. For example, swu (g) =su (TS5) =67 in Table 22, for minSeqUtil=223, as swu (g) < minSeqUtil, any super-sequence of g cannot

be a high-swu sequence and obviously cannot be a high-utility sequential pattern. All the high-utility sequential patterns are generated from the high-swu sequences.

Level	Candidate high-*swu* sequences	Candidate high-utility seq. pattern (high-*swu* sequence) with *swu* values
1	**7 candidates** *a*, *b*, *c*, *d*, *e*, *f*, *g*	**6 high-*swu* sequences** *a*: 602, *b*: 676, *c*: 226, *d*: 743, *e*: 546, *f*: 271
2	**51 candidates (14 candidates not appear in SDB at all)** *aa*, *ab*, *ac*, *ad*, *ae*, *af* *ba*, *bb*, *bc*, *bd*, *be*, *bf* …… (*ab*), (*ac*), (*ad*), (*ae*), (*af*) (*bc*), (*bd*), (*be*), (*bf*) ……	**17 high-*swu* sequences** *aa*: 310, *ab*: 517, *ad*: 602, *ae*: 387, *ba*: 310, *bb*: 387, *bd*: 496, *be*: 461, *da*: 517, *db*: 387, *dd*: 337, *de*: 387, *ea*: 292, *eb*: 292, *ed*: 292, (*ab*): 602, (*bd*): 310
3	**70 candidates (25 candidates not appear in SDB at all)** *aaa*, *aab*, *aad*, *aae*, *a*(*ab*), *aba*, *abb* …… (*ab*)*a*, (*ab*)*b*, (*ab*)*d*, (*ab*)*e* ……	**18 high-*swu* sequences** *aba*: 310, *abe*: 387, *ada*: 310, *adb*: 387, *ade*: 387, *a*(*ab*): 310, *a*(*bd*): 310, *bbe*: 387, *dad*: 337, *dae*: 387, *dbe*: 387, *d*(*ab*): 387, *ead*: 292, *ebd*: 292, *e*(*ab*): 292, (*ab*)*d*: 422, (*ab*)*e*: 387, (*bd*)*a*: 310
4	**4 candidates** *adbe*, *a*(*bd*)*a*, *d*(*ab*)*e*, *e*(*ab*)*d*	**4 high-*swu* sequences** *adbe*: 387, *a*(*bd*)*a*: 310, *d*(*ab*)*e*: 387, *e*(*ab*)*d*: 292

Table 23: Candidate generation process for UL algorithm (Ahmed et al., 2010)

Step 1: The level-by-level candidate generation-and-testing mechanism is shown in Table 23 for the SDB and minSeqUtil=30%. In level-1, all the distinct items are candidate length-1 high-swu sequences. The UL algorithm scans the SDB once to generate all the length-1 high-swu sequences. **Step 2:** In level-2, 51 candidate high-swu sequences are generated as shown in Table 23. For one length-1 high-swu sequence, UL generates length-2 candidate sequences by joining it with other length-1 high-swu sequences which include it. For example, for length-1 candidate sequence a, length-2 candidate sequences aa, ab, ac, ad, ae, and af are generated. As there are 6 length-1 candidate sequences, a total of 6×6=36 length-2 candidate sequences are generated. On the other hand, a length-1 candidate sequence joins with others to form length-2 single element candidate sequences, such as when sequence a joins b, c, d, e, and f to form (ab), (ac), (ad), (ae), and (af). Similarly, sequence b joins with c, d, e, and f to form (bc), (bd), (be), and (bf). In this way, 5 candidate sequences for a, 4 candidate sequences for d, and 1 candidate sequence for e are generated. Accordingly, a total of 5+4+3+2+1=15 ((6×5)/2)) candidate sequences are generated, that is, for N distinct items, (N× (N–1))/2 length-2 single-element candidate sequences are generated. Hence, a total of 36+15=51 length-2

candidate sequences are generated. **Step 3:** The original SDB must be scanned once again with all these candidate sequences to detect the high-swu sequences. It is noticeable that among the 51 candidates, 14 candidates, for example, ca, cc, ef, and (ce), do not appear in the SDB at all. However, Table 23 also shows that 17 high-swu sequences (candidate high-utility sequential patterns) are generated from these 51 candidates. **Step 4:** After generating all the candidate high-swu sequences by joining, UL algorithm prunes those candidate sequences having at least one subsequence which is not a length-(k–1) high-swu sequence. For example, after generating length-3 candidate high-swu sequence eae from length-2 high-swu sequences ea and ae, it prunes eae as its subsequence ee is not a high-swu sequence. However, for level-3, this algorithm generates a total of 70 candidate sequences after joining and pruning and scans SDB again to detect 18 high-swu sequences as shown in Table 23. Similarly, it generates length-4 candidate high-swu sequences and calculates 4 high-swu sequences. No candidate high-swu sequence is generated for level-5. Finally, it scans the SDB with these high-utility-swu sequences to discover the high utility sequential patterns. Consequently, the UL algorithm discovers a total of 6 high-utility sequential patterns <b: 266, (ab): 239, (ab) d: 238, adbe: 226, a (bd) a: 231, d (ab) e: 250> by generating a total of 132 candidates and with five database scans.

2. US (Utility Span) algorithm: UL can discover the final resultant high-utility sequential patterns successfully, it suffers from the level-wise candidate generation-and-testing problem and hence generates a substantial number candidate sequences. Moreover, its number of database scans is directly dependent on the maximum length of candidate sequences. To solve the problem of UL approach, US algorithm is proposed.

Input: An SDB with utility values, minSeqUtil

Output: The complete set of high-utility sequential patterns

Step 1: US algorithm always needs a maximum of three database scans. Therefore, it significantly reduces the overall runtime for mining high-utility sequential patterns. First, the US algorithm scans 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. By utilizing this divide-and-conquer method, it generates very few candidates compared to the UL algorithm. Note that the US algorithm only generates the high-swu sequences without generating a large number of intermediate candidates.

Step 2: UL algorithm generates a projected database means that last item in the prefix.

Step 3: Now, according to the divide-and-conquer rule, the same technique is applied on the projected databases. It also shows the other candidate sequences generated by other prefix items. However, as mentioned earlier, the US algorithm only generates the high-swu sequences as candidates.

Consider the example presented in the Table 24. It needs a total of 5 database scans as the maximum candidate length is 4 (last scan is needed for detecting high-utility sequential patterns from the high-swu sequences). From **Step 1**, the US algorithm scans the SDB once to detect length-1 swu sequences. From **Step 2**, For prefix a, the UL algorithm generates a projected database (_b) means that last item in prefix, which is a, forms one element (ab). However, in the a-projected database, <a: 310, b: 517, _b: 602, d: 602, e: 387, f: 130, and c: 85> are obtained. Items c and f cannot form 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. Consequently, 6 candidates sequence a: 602, aa: 602, ab: 517, (ab): 602, ad: 602, and ae: 387 are generated. The aa-projected database contains {(_bd) fad: 130, (_b)e: 180}, and <a: 130, _b: 310, d: 130, f: 130, e: 180> are obtained. So, only one candidate sequence, a(ab): 310, is generated.

Prefix	Projected SDB	Candidate high-utility seq. pattern (high-*swu* sequence) 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, *a*(*bd*)*a*: 310, (*ab*)*d*: 422, (*ab*)*e*: 387, *ada*: 410, *adb*: 387, *ade*: 387, *adbe*: 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, *de*: 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* sequences** *e*: 546, *ea*: 292, *eb*: 292, *ed*: 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 24: Candidate generation process for US algorithm (Ahmed et al., 2010)

The ab-projected database contains {(_d) fad: 130, (_d)(ab)e: 180, e: 207}, and <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. Similarly, the other candidate sequences are generated. The Table 24 shows that a total of 17 candidate sequences are generated by prefix-a. As in **Step 3**, according to the divide-and-conquer rule, the same technique is applied on the projected databases of aa, ab, (ab), ad, and ae. It a third database scan is needed to discover the high-utility sequential patterns from the high-swu sequences. The US algorithm discovers the same set of resultant high-utility sequential patterns <b: 266, (ab): 239, (ab)d: 238, adbe: 226, a(bd)a: 231, d(ab)e: 250> as discovered by the UL algorithm. However, in this example, it generates only 45 candidates and scans the database three times in contrast to the 132 candidates and five database scans of the UL algorithm.

Advantages: The number of database scans needed by the US algorithm is independent on the maximum length of the candidate sequences.

Limitations: The number of database scans needed by the UL algorithm is directly dependent on the maximum length of the candidate sequences as it adopts the level-wise Apriori mechanism.

2.4.2. U-SPAN ALGORITHM (Yin et al., 2012)

USpan is the algorithm used for mining high utility sequential patterns. USpan is composed of a lexicographic q-sequence tree, 2 concatenation mechanisms and 2 pruning strategies.The algorithm is explained in the steps below:

Step1: The concept of sequence utility is considered by the quality and quantity associated with each item in a sequence.

Step 2: A complete lexicographic quantitative sequence tree (LQS-tree) is used to construct and organize utility based q-sequences, two concatenation mechanisms I-Concatenation and S-Concatenation generate newly concatenated sequences; Suppose for a k-sequence t, the operation of appending a new item to the end of t is to form (k+1)-sequence concatenation. If the size of t does not change, the operation I-Concatenation will occur. Otherwise, if the size increases by one, S-Concatenation is occurred (Yin et al., 2012). For example, <ea>'s I Concatenate and S-Concatenate with b result in <e(ab)> and <eab>, respectively. Assume two k-sequences t_a and t_b are concatenated from sequence t, then $t_a < t_b$ if t_a is I-Concatenated from t, and t_b is S-Concatenated from t, or both t_a and t_b are I-Concatenated or S-Concatenated from t, but the concatenated item in t_a is alphabetically smaller than that of t_b.

Step 3: A lexicographic q-sequence tree (LQS-Tree) T is a tree structure satisfying the following rules: Rule1: Each node in T is a sequence along with the utility of sequence, while the root is empty and Rule 2: Any node's child is either an I-Concatenated or S-Concatenated sequence node of the node itself and 3. All the children of any node in T are listed in an incremental and alphabetical order.

Step 4: USpan consequently uses a depth-first search strategy to traverse the LQS-Tree to search for high utility patterns. Two pruning methods, width and depth pruning, substantially reduce the search space in the LQS-tree; A sequence is of high utility only if its utility is no less than a user-specified minimum utility.

Step 5: The complete set of the identified high utility sequential patterns forms a complete LQS-Tree, which covers the complete search space.

Example : **Input :** A sequence database shown in the Table 25, shows five sequences listed with the itemsets associated with the quantity, i.e., a number of items purchased in each sequence (in SID = 1 is e=5). In the Profit table from the Table 26, each item's price is given, which represents the quality (Price) of the item in a transaction. The minimum utility threshold ξ = 0; **Output:** High Utility Sequential Patterns.

sid	q-sequence
1	$\langle(e,5)[(c,2)(f,1)](b,2)\rangle$
2	$\langle[(a,2)(e,6)][(a,1)(b,1)(c,2)][(a,2)(d,3)(e,3)]\rangle$
3	$\langle(c,1)[(a,6)(d,3)(e,2)]\rangle$
4	$\langle[(b,2)(e,2)][(a,7)(d,3)][(a,4)(b,1)(e,2)]\rangle$
5	$\langle[(b,2)(e,3)][(a,6)(e,3)][(a,2)(b,1)]\rangle$

Table 25: A Single Sequence Database with Five Transactions

Item	a	b	c	d	e	f
Price	2	5	4	3	1	1

Table 26: Profit Table

Step 1: The utility of a single item can be defined as its purchased quantity times its profit. The utility of an itemset is the sum of the utilities of all its items. For example, for s1, the utility of q-item (e, 5) is u (e, 5) = 5 × 1 = 5, which is also the utility of the first itemsets utility. Similarly, the utility of s1 and S are u(s1) = u (e, 5) + u (c, 2) + u (f, 1) + u (b, 2) = 5 × 1 + 2 × 4 + 1 × 1 + 2 × 5 = 24 and u(S) = u(s1) + u(s2) + u(s3) + u(s4) + u(s5) = 24 + 41+27+50+37 = 179 respectively. The utility of sequence ea is umax (<ea>) = 10 + 16 + 15 = 41. If the minimum utility is ξ = 40, then sequence <ea> is a high utility sequential pattern since umax(s) = 41 ≥ ξ. In frequent sequential pattern mining, the downward closure property serves as the foundation of pattern mining algorithms. However, this property does not hold in the high utility pattern mining problem. umax (<ea>) = 41, but umax(<e>) = 5 + 6 + 2 + 2 + 3 = 18, which is lower than its super-pattern. The utilities of

the sequential patterns <(ae)>, <(ae)a>, <(ae)(ab)>, <(ae)(abc)> and <(ae)(abc)a> are 49, 33, 41, 25 and 29 respectively. There is no such property as anti-monotonicity in the maximum utilities. Therefore, it is not surprising that given ξ > 0, the high utility sequences may not form a complete-LQS-Tree. For example, for ξ = 60, the high utility sequential patterns are {(be)a(ab)}, {ba(ab)}, {(be)aa} and {(be)ab}. Obviously, these four patterns cannot form a complete-LQS-Tree.

Step 2: USpan consequently uses a depth-first search strategy to traverse the LQS-Tree to search for high utility patterns. As shown in Figure 10, USpan first generates the children of the root. It then takes <a> as the current node, checks whether '< a >' is a high utility pattern, and scans for <a>'s possible children. If '< a >'s first children, i.e. < (ab) >, are not taken as the current node, the same operations will apply to < (ab) >.

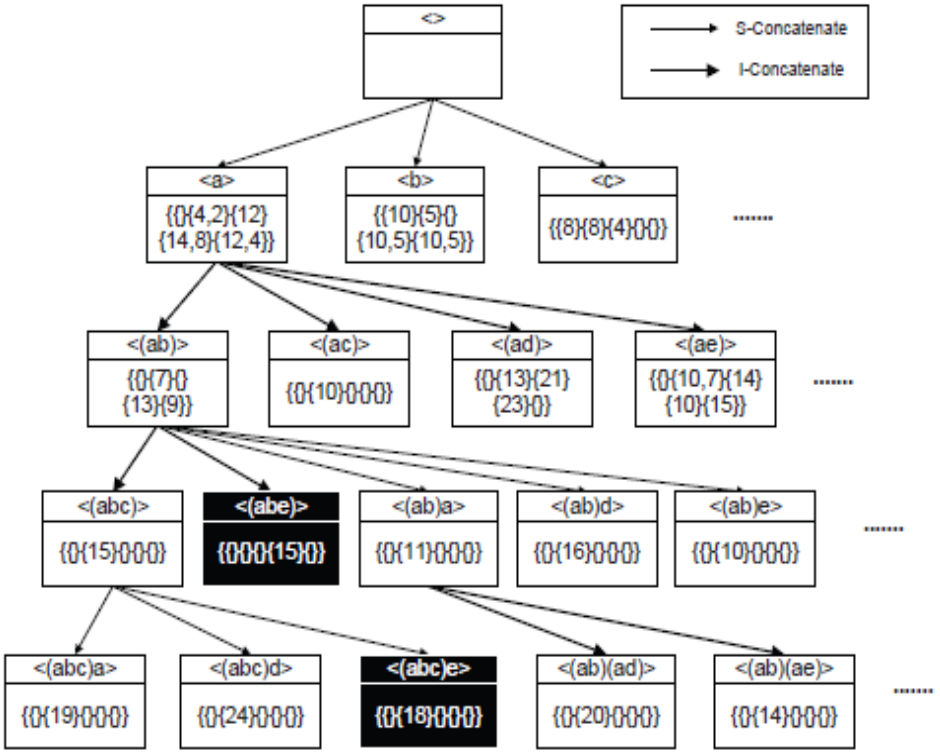

Figure 10: Sample LQS Tree (Yin et al., 2012)

This procedure will be recursively invoked until there is no other node in the LQS-Tree to visit. It then takes <a> as the current node, checks whether <a> is a high utility pattern, and scan for <a>'s 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 procedure will be recursively invoked until there is no other node in the LQS-Tree to visit. Sample LQS tree was given in Figure 10. **Step 3:** The I-Concatenation and the S-Concatenations are applied to the LQS Tree. **Step 4:** The depth and width pruning techniques are further applied to remove the unpromising candidates from the tree.

Advantages: It is similar to SPAM (Ayres et al., 2002) algorithm in the sequential pattern mining of the frequent patterns.

Limitations: The execution time will be more for the tree traversal and it is often costly to perform the concatenations and the pruning strategies. Also, it was unsuitable for the bigger datasets.

3. PROPOSED HUU-PLWAP (HIGH UTILTY UNCERTAIN-PRELINKED POSITION CODE WEB ACCESS PATTERN) MINER

The motivation of the proposed solution came from the disadvantages that are associated with PHUI algorithm and the U-PLWAP algorithm discussed in the section 2.3.3 and section 2.2.2. However, since U-PLWAP only mines the uncertain sequences without the measure utility and proposed algorithm which is based on U-PLWAP extended to cater for utility measure with both internal (quality) and external (quantity) utilities values. In the PHUI-UP (Lin et al., 2016) algorithm the patterns generated are from the level wise candidate approach which can generate numbers of candidates that causes space and time complexities. In PHUI algorithm the utility and the probability measures are taken as separate entities and further the existential probability is same for all the individual items in the transaction database. Also it does not deal with uncertain sequence databases. The following extensions are made to U-PLWAP and PHUI to mine uncertain data to generate high utility uncertain sequential patterns. The input data is the database UWASDB with probabilistic internal utilities and external utility derived from the e-commerce website and the output was the high utility uncertain sequential web access patterns (HUUP).

Preparing data for the HUU-PLWAP algorithm: Uncertain We Access Sequence Database Retrieval: A database generated by the sequence based uncertain model is called an uncertain web access sequence database. To interpret an uncertain Sequence database 'D' the possible world model is applied in which visited page exists in sequence is one possible world and other possible world is where it does not exists in sequence. The uncertain transactions are mutually independent. The probability of a Possible world w is calculated by the following formulae from Equation (1) (Bernecker et al., 2009),

$$\mathrm{P}\ (w) = \prod_{s \epsilon I}\left(\prod_{x \in s} P(x \in s) * \prod_{x \notin s}(1 - P(x \in s))\right) \rightarrow Equation\ (1)$$

Where ‘x’ represents the webpage or an event present in sequence‘s’ for every s ∈ Item, P (w) = Probability of Possible World; Possible world: w; P (x ∈ s) = Probability of Webpage ‘x’ present in the web sequence‘s’; (1- P (x ∈ s)) = Probability of Webpage ‘x’ not present in the web sequence‘s’. Thus, the number of possible worlds of a database increases exponentially in both the number of Sequences .The process is explained with an example in the section 3.2.1.

Definitions used for the algorithm: The utilities are assigned to the webpages are:- The internal utility is the time spent by the user on each web page and the external utility is the profit for each webpage which can be processed based on the rating by the website (Bakariya and Thakur 2015).

- The internal utility value of a web page is represented by iu (e, S_i). ‘e' is the event or webpage and ‘S’ is the sequence.
- External utility eu (e) is the significance value of web page or event e.
- Utility of an event or webpage ‘e’ was defined as the product of the internal utility and the external utility, i.e., U (e, S_i) = iu (e, S_i) * eu (e, S_i). → Formula 1
- Web Access Sequence utility (wasu) (e, S_i), is the quantitative measure of utility for web page e in sequence S_i defined by wasu (e, S_i) = ∑ (iu (e, S_i) * eu (e, S_1)) → Formula 2 for every e in S_i.

3.1. THE HUU-PLWAP ALGORITHM (Algorithm 1)

The HUU-PLWAP Algorithm is explained through below steps in the Figure 11. The example related to the algorithm is mentioned in the section 3.2.

Input: The database UWASDB with probabilistic internal utilities and external utility (the profit table). The minimum utility threshold = d; minimum sequential threshold minSeqUtil μ; Other Variables: wasu =web access sequence utility; waswu =web access sequence weighted utility

Output: Complete set of high utility uncertain sequential web access patterns (HUUP)

Algorithm 1: The HUU-PLWAP algorithm

Input: Uncertain Web Access Sequence Database (UWASDB) with internal probabilistic utility (iu_i^j = internal utility of item j of sequence i and the external utility eu $_k$ =external utility of kth candidate item C1 in the profit table,C1= Candiadate-1-Itemsets, Minimum Utility Threshold d; minSeqUtil μ;e = event or webpage.
Other Variables: wasu =web access sequence utility; waswu =web access sequence weighted utility
Output: Complete set of high utility uncertain sequential web access patterns (HUUP)

Begin:

1. Find the Frequent-1 waswu sequence values by calling the The Frequent-1 Waswu algorithm (from the figure 12)
2. Insert all Frequent-1 waswu sequence values in the Link header table
3. Build HUU-PLWAP tree from the UDB by calling HUU-PLWAP_tree algorithm (from the figure 13)
4. Recursively mine the HUU-PLWAP tree by calling Mine algorithm (from the figure 14)

End

Figure 11: The HUU-PLWAP algorithm

3.1.1. THE FREQUENT-1 WASWU ALGORITHM (Algorithm 2)

Step 1: Find the Frequent-1 waswu sequence values (single-element Frequent-1 waswu sequences):

The utilities and sequence utilities of all the Candidatet-1 items are calculated based on the formulae 1and 2 mentioned above. The most challenging problem for high utility Sequential mining is that web access sequence utilities do not follow the downward closure property. Consider a minimum utility threshold, then web page X is a low utility WAS as wasu (x) = p, but its super-sequence {x, y} is a high utility WAS. Therefore, the downward closure property is not satisfied. To maintain the downward closure property in high utility WAS mining, a new measure called web access sequence weighted utility (WASWU) is defined using the two-phase approach discussed in the section 2.3.2. The web access sequence weighted utility value (waswu) of a sequence X is defined by the equation **waswu (X) = ∑ wasu (X present in each Sequence) →Equation (2).**

If d is the minimum utility threshold then the minSeqUtil 'μ' was derived from the waswu (UWASDB)* d = μ. The frequent-1 waswu sequence values are said to be frequent only

if the value of waswu (e) for event or webpage ‘e ‘should be greater than or equal to μ. The process is explained in The **Frequent_1_Waswu algorithm** algorithm in the Figure 12. The output was the set of waswu values which are above the minimum sequential threshold μ.

Algorithm 2: The Frequent-1_Waswu algorithm

Input: Uncertain Web Access Sequence Database (UWASDB) with internal probabilistic utility (iu_i^j= internal utility of item j of sequence i and the external utility eu k =external utility of kth candidate item C1 in the profit table,C1= Candiadate-1-Itemsets, Minimum Utility Threshold d; minSeqUtil μ;

Output: Complete set of frequent 1-waswu sequence values W1

Intermediate variables: Iterator k, Check set Ch (records events found already in a sequence)

Other Variables: wasu =web access sequence utility; waswu =web access sequence weighted utility Utility U;

Begin
Initialize W1 to empty set
For each sequence S in UWASDB
Begin
Initialize Check set Ch to empty set
For each event e in sequence S
Begin
Calculate U(e)=iu(e)*eu(e)
Calculate wasu(e)= ∑(iu(e)*eu(e))
Calculate waswu = ∑ (wasu(e) present for each event e in sequence S_i
μ = d * waswu(UWASDB)

If waswu(e)> μ
If event e ∈ W1
If e ∉ Ch
Add e to Check set Ch
Else
Add event e to W_1
Set count of e in W_1
End
End
Return W_1
End

Figure 12: The Frequent-1 Waswu algorithm

Step 2: The second step in the HUU-PLWAP algorithm leads to the insertion of Frequent-1 waswu values (which are obtained from Step **1** through algorithm in the Figure 12) into the Link Header table 'H'.

3.1.2. THE HUU-PLWAP TREE ALGORITHM (Algorithm 3)

Step 3: The third step of the HUU-PLWAP algorithm leads to the construction of the HUU-PLWAP tree as in the Figure 13 with the input of single-element Frequent-1 waswu sequence values generated in the Link header table as mentioned in the **Step 2**.

Algorithm 3: The HUU-PLWAP_tree algorithm

Input: Uncertain Web Access Sequence Database (UWASDB) , with internal probabilistic utility (iu_i^j= internal utility of item j of sequence i and the external utility e_k =external utility of kth candidate item C1 in the profit table, Link header table L

Output: HUU-PLWAP Tree T, Linked Link header table

Intermediate variables: minSeqUtil μ , Boolean Variable NodeFound

Begin:
Create root node of T with event label set to NULL, position code set to NULL and count set to 0
For each access sequence S in UWASDB
Begin
Extract frequent-1 waswu values S′ from S by removing all events in S but not in L
Let S′ = $e_1e_2e_3 \ldots e_n$ where e_i are events in S′ and n is the length of S′
Let current_node point to the root node of T
For i = 1 to n do
Begin
If left child of current_node is NULL, then create a new child node (e_i: 1) and its position code
is formed by appending "1" at the end of the position code of the current_node
Register waswu value of e_i against the sequence ID of S′
Make the current_node point to the newly created node

Else If current_node is labeled e_i , then set Nodefound to true
Increase support count by 1 and register waswu of e_i against the sequence ID of S′

Else
Let the current_node point to current_node.rightsibling, and keep checking whether current node is labeled e_i , until there is no more right sibling or e_i is found

If NodeFound
Then increase the count of e_i by 1 and register waswu value of e_i against the sequence ID of S′

Make the current_node point to the node of e_i
Else
Create a new child node (e_i: 1) and its position code is formed
by appending "0" at the end of the position code of the current_node
Register waswu value of e_i against the sequence ID of S′
Make the current_node point to the newly created node
End
End

1. Entries in the Link header table are then linked to corresponding nodes in the HUU-PLWAP tree in a pre-ordered fashion (from root to left child and then right child)
2. Return T with the linked Link header table

End

Figure 13: HUU-PLWAP -tree construction algorithm.

Step 2: Construction of HUU-PLWAP tree (Algorithm 3)

The **HUU-PLWAP Tree Construction** was explained through steps mentioned in the Figure 13 and the terms used in the algorithm are defined as follows:

Step i: The HUU-PLWAP tree algorithm works by first creating the root nodes and initialising its label and position code to null while its count set to 0. It first detects the candidate web pages (single-element Frequent-1 waswu values) in first database scan.

Step ii: Insert all waswu values obtained from the **Step 1** as per **Step 2** in the Link header table.

Step iii: A sequence 'X' is a high utility web access sequence if wasu(X) is greater than or equal to μ. Mining high utility WASs means discovering all the sequences X having criteria wasu(X) ≥ μ. Those values are above μ are inserted into H.

Step iv: The HUU-PLWAP tree is created starting from a null root. Sequences are read from the database and nodes are created for each item in the sequence. Each node contains the sequence label, waswu values and position code, denoted as label: waswu values: position code. A left node is created if no node already exists, otherwise, a right node is created and the count initialised to 1. If node already exists, its count is incremented by 1. The position code is found by appending '1' to the end of position code

of current node. The sequence label and its waswu values are read from the sequence database. Entries created in the header table are then used to link their corresponding nodes by traversing the HUU-PLWAP tree in a pre-ordered fashion (from root to left node first before right node). This makes tree traversal faster and more efficient since similar nodes in the same suffix tree are brought closer. When a left child already exists, and has same label as the item read, its count is incremented by 1. However, if the left child has a different label from the item read; the HUU-PLWAP algorithm continues to search all the right siblings for a match. When a match is found, its count is incremented by 1. To facilitate the tree traversals, adjacent links are maintained in our tree structure. If no match is found in all the right siblings, a fresh right sibling is created. The label of the newly created right sibling is set to the item while count is set to 1. The current node is then set to the newly created node. The next item is then read while keeping track of the point reached in the tree through the current node pointer.

Step v: The complete process is repeated until the end of the sequence fetched is reached. A new sequence is then fetched, this time the current node pointer is set to the root of the HUU-PLWAP tree. A third database scan is needed for finding the high utility web access sequences from the candidate sequences using the waswu values.

Step vi: The HUU-PLWAP tree created is then mined recursively using prefix conditional search. Starting from the root with a frequent item to find all the high waswu values which are above the minSeqUtil on the header table, all sub-trees are traversed.

3.1.3. THE HUU-PLWAP MINER ALGORITHM (Algorithm 4)

Step 3: Recursive mining using the mining algorithm is performed on the conditional tree of web pages. This was explained in the Miner algorithm in the Figure 14.

Algorithm 4: The High Utility Uncertain PL-WAP miner Algorithm

Input: HUU-PLWAP tree T, Root_set R, Link header table L, Frequent n-waswu sequence W, MinSeqUtil μ(R contains root, W is empty when the algorithm is called the first time)

Output: High utility Uncertain n-sequence W′

Intermediate variables: S stores the information of node whether it is the ancestor of the following node in the queue of similar nodes, C stores the total count of event e_i in different suffix tree

Begin:

1. If R is empty, return
2. For each event e_i in L, find the suffix tree of e_i in T (e_i |suffix tree)
   Save first event in e_i –queue to S
   Following the e_i –queue
   If event e_i is the descendant of any event in R, and is not the descendant of S
   Insert node of e_i into new root set R′
   Replace S with e_i
   If W is empty
   Enter all Frequent-1 entries of waswu sequnces in node e_i into W′, each identified by sequence ID and waswu ($event_n$) < μ

else Delete $event_n$

Append e_i to end of W to form W′ and output W′
Call algorithm 3 passing R′, W′

End

Figure 14: The Mine Algorithm

Step I: The Mine algorithm works by accepting the root of the HUU-PLWAP tree (The Root_set R is initialised with the root of the U-PLWAP tree), set of frequent-1 waswu sequences 'W' as input (W is set to empty set). The link header table L is also an input. Mining starts by finding frequent sequence that begins with entries in the header table with the same prefix. For each entry in L, the algorithm finds its set of suffix tree(s). Each node found is entered into new set of R (R′) if it is the first occurrence of event e_i along each suffix tree path. The queue linking all nodes of type e_i is followed in order to discover the first nodes in each suffix tree path. The value of count C is then compared to the minimum sequential utility value μ. If count is greater or equal to μ, e_i is appended to the end of F and printed as output. The mine algorithm is then called again with new sets R′, W′. If count is less than μ, then this indicates that all frequent sequences starting with

event e_i are found and no more left. The next event is selected from the header table L and the whole process repeated.

Step II: However, this mining procedure can be used for mining operation with a HUU-PLWAP-tree. The search continues down the tree to find all the high utility web access sequences and the candidates are generated. Candidate sequences prefixing other web pages can be generated in a same fashion. The process continues recursively until no more items is found. The algorithm then backtracks to the null root to start mining for sequences starting with a fresh item from the header table. Finally the high utility uncertain web access sequential patterns will be generated.

3.2. THE HUU-PLWAP MINER EXAMPLE

Preparing data for the HUU-PLWAP algorithm: Uncertain We Access Sequence Database Retrieval: Consider an ecommerce website Windsor Star Website. To maximize sales, customer interactions can be analysed to find sequence of webpages that are highly visited and profitable for the website company. This information can be used for advertising directed at this group. For example, by providing special orders that include all new productive webpages, the store can encourage new purchases inside the website. The Windsor Star website uses a sequence database, shown in Table 27 in which the customer 'A' checks News Feed Updates in Windsor Star website, every time he visits the website and music column in the webpage 20% of the time. Customer B dedicates 40% to the movie videos in the webpage of her visits and 70% time in Game column in the webpage.

Sequence ID	Sequence Probability
S_A	{News Feed,1; Music,0.2}
S_B	{Video,0.4; Game,0.7}

Table 27: Uncertain Sequence Database from Windsor Star Website

The uncertain data model is based on the possible world's semantics with existential uncertain items. An uncertain web access sequence is a sequence with the uncertain data with the probability of the user visiting the webpages. A database generated by the sequence based uncertain model is called an uncertain web access sequence database. To interpret an uncertain Sequence database 'D' the possible world model is applied. An uncertain sequence database (single set sequences) generates possible worlds, where each world is defined by a fixed set of (certain) sequences (Bernecker et al., 2009). A possible world is instantiated by generating each sequence $S_i \in D$ according to the occurrence probabilities P ($x \in S_i$) where x is a webpage or an event. Consequently, each probability P ($x \in S_i$) derives two possible worlds per transaction: One possible world in which x exists in S_i, and one possible world where x does not exist in S_i. Thus, the number of possible worlds of a database increases exponentially in both the number of transactions and the number of uncertain items contained in it. Each possible world w is associated with a probability that that world exists, P (w). One possible world in which the visited page exists in the Sequence, and another possible world where visited page does not exist in the Sequence. In the proposed HUU-PLWAP, sequence uncertainty model is used which is similar to the tuple uncertainty model. The uncertain transactions are mutually independent. Thus, the decision by customer A has no influence on customer B. This assumption is reasonable in real world applications. The probability of a world w is calculated by the following formulae from Equation (1),

$$P(w) = \prod_{s \epsilon I}\left(\prod_{x \in s} P(x \in s) * \prod_{x \notin s}(1 - P(x \in s))\right)(\text{From}, \text{Eq}(1))$$

Where 'x' represents the webpage or an event present in sequence's' for every s ∈ Item

P (w) = Probability of Possible World

P (x ∈ s) = Probability of Webpage 'x' present in the web sequence's'

(1- P (x ∈ s)) = Probability of Webpage 'x' not present in the web sequence's'

Example: The Possible world 3 is obtained from the Table 28: =P (News Feed $\in S_A$) * (1 -P (Music $\in S_A$)) * (1-P (Game $\in S_B$) *P (Video $\in S_B$)) = (News Feeds belongs to Sequence A) (Music does not belong to Sequence A) (Game does not belong to Sequence B) (Video belongs to Sequence B) =1*0.8*0.3*0.4=0.096

Thus, the number of possible worlds of a database increases exponentially in both the number of Sequences and the number of uncertain web access data (Visited pages data) contained in it. Each possible world w in the example mentioned in section 3.2 is associated with probabilities that that world exists which are shown in Table 28.

Possible World	Web Access Sequence Database	Sequence Probability
1	{News Feed}; {}	0.144
2	{News Feed, Music}; {}	0.036
3	{News Feed}; {Video}	0.096
4	{News Feed, Music}; {Video}	0.024
5	{News Feed}; {Game}	0.336
6	{News Feed, Music}; {Game}	0.084
7	{News Feed}; {Video, Game}	0.224
8	{News Feed, Music}; {Video, Game}	0.056

Table 28: Corresponding possible worlds

For example, in Table 28,

1. The Possible world 1 is obtained from

P (News Feed ϵ S_A) * (1 -P (Music ϵ S_A)) * (1-P (Game ϵ S_B) *(1 - P (Video ϵ S_B)) =1*0.8*0.6*0.3=0.144.

2. The Possible world 2 is obtained from

P (News Feed ϵ S_A) * P (Music ϵ S_A)) * (1-P (Game ϵ S_B) *(1 - P (Video ϵ S_B)) =1*0.8*0.4*0.3=0.036

3. The Possible world 3 is obtained from

P (News Feed ϵ S_A) * (1 -P (Music ϵ S_A)) * (1-P (Game ϵ S_B) *P (Video ϵ S_B)) =1*0.8*0.3*0.4=0.096.

4. The Possible world 4 is obtained from P (News Feed ϵ S_A) * (P (Music ϵ S_A)) * (1-P (Game ϵ S_B)) *P (Video ϵ S_B) =1*0.2*0.4*0.3=0.024

5. The Possible world 5 is obtained from

P (News Feed ϵ S_A) * (1 -P (Music ϵ S_A)) * P (Game ϵ S_B) *(1 - P (Video ϵ S_B)) = 1:0 *0:8 * 0:7 * 0:6 = 0:336.

6. The Possible world 6 is obtained from

P (News Feed ϵ S_A) * P (Music ϵ S_A)) * P (Game ϵ S_B) *(1 - P (Video ϵ S_B)) =1*0.2*0.7*0.6=0.084

7. The Possible world 7 is obtained from

P (News Feed ϵ S_A) * (1 -P (Music ϵ S_A)) * P (Game ϵ S_B) * P (Video ϵ S_B) =1*0.8*0.6*0.3=0.224

8. The Possible world 8 is obtained from

P (News Feed ϵ S_A) * (1 -P (Music ϵ S_A)) * (1-P (Game ϵ S_B) *(1 - P (Video ϵ S_B)) =1*0.2*0.4*0.7=0.056.

Now the utilities are assigned to the webpages in the example in the section 3.1. The internal utility is the time spent by the user on each web page and the external utility is the profit for each webpage which can be processed based on the rating by the website. The internal utility value of a web page is represented by iu (e, Si) . 'e' is the webpage and S is the sequence. For example, in Table 30, iu (Music, S2) = 0.036. In the same way, all the internal utilities are mentioned in the Table 29. In HUU-PLWAP approach, the internal utility is extracted by the product of quantity (visited time associated with each webpage by user) and the existential probability value generated based on the sequence based uncertainty model.

By considering the internal utilities with uncertainty, the Uncertain Web Access Sequence Database (UWASDB) has generated as in the Table 29.

SEQUENCE ID	UNCERTAIN SEQUENCE WITH INTERNAL UTILITY	PROB
S1	{News Feed:2}; {}	0.144
S2	{News Feed:1, Music:3}; {}	0.036
S3	{News Feed:4}; {Video:1}	0.096
S4	{News Feed:2, Music:5}; {Video:3}	0.024
S5	{News Feed:1}; {Game:1}	0.336
S6	{News Feed:1, Music:3}; {Game:2}	0.084
S7	{News Feed:1}; {Video:1, Game:3}	0.224
S8	{News Feed:6, Music:1}; {Video:3, Game:2}	0.056

Table 29 : UWASDB with Internal Utilities & Existential Probabilities

- For further calculation, the internal utility is multiplied with the probability to generate the uncertain probabilistic internal utility The Uncertain Web Access Sequence Database (UWASDB) with Internal Utilities is given in Table 30.

SEQUENCE ID	UNCERTAIN SEQUENCE WITH INTERNAL UTILITY(probability)
S1	{News Feed:0.288}
S2	{News Feed:0.036, Music:0.108}
S3	{News Feed:0.384}; {Video:0.096}
S4	{News Feed:0.048, Music:0.12}; {Video:0.0720}
S5	{News Feed:0.336}; {Game:0.336}
S6	{News Feed:0.084, Music:0.252}; {Game:0.168}
S7	{News Feed:0.224}; {Video:0.224, Game:0.672}
S8	{News Feed:0.336, Music:0.056}; {Video:0.168, Game:0.112}

Table 30: Uncertain Web Access Sequence Database (UWASDB) with Probabilistic Internal Utilities

The profit table for the Uncertain WASD is given in Table 31. External utility eu (w) is the significance value of web page w. For example, in Table 31, eu (News Feed) = 1. Similarly, the external utilities of Music, Video, and Game are 4, 2 and 3.

ITEM	News Feed	Music	Video	Game
PROFIT(EU)	1	4	2	3

Table 31: Profit Table

THE HUU-PLWAP ALGORITHM in Section 3.1 is explained through below example.

Input: The database UWASDB with Probabilistic internal utilities (given in Table 30) and external utility (the profit table in the Table 31). The minimum sequential threshold μ is calculated with 25% = 0.25 × 8.612= 2.153

Output: Complete set of high utility uncertain sequential web access patterns (HUUP)

3.2.1. THE FREQUENT-1 WASWU ALGORITHM (From Figure 12)

Step 1: Find the Frequent-1 waswu sequence values (single-element Frequent-1 waswu sequences):

- The utilities and sequence utilities of all the Candidatet-1 items are calculated based on the formulae 1and 2 mentioned above. Web Access Sequence utility (WAS), wasu (e, S_i), is the quantitative measure of utility for web page e in sequence S_i defined by wasu (e, S_i) = iu (e, Si) × eu (e, S_1). For example, wasu (News Feed, S_1) = 0.036 x1 = 0.036.
- Web Access Sequence utility (wasu) (e, S_i), is the quantitative measure of utility for web page e in sequence S_i defined by wasu (e, S_i) = ∑ (iu (e, S_i) * eu (e, S_1)) → Formula 2 for every e in S_i. For example, wasu (S_1) = 0.036; wasu (S_2) = 0.468; wasu (S_3) =0.576; wasu (S_4) =0.672; wasu (S_5) =1.344; wasu (S_6) =1.596; wasu (S_7) =2.688; wasu (S_8) =1.232;

- The web access sequence utility of a sequence X in a web access sequence database WASDB is defined by Example: wasu (News Feed, Music) = wasu (News Feed, Music, S_2) + wasu (News Feed, Music, S_4) + wasu (News Feed, Music, S_6) + wasu (News Feed, Music, S_8) = 0.468+0.528+1.092+0.56=2.646
- The web access sequence utility value of a web access sequence database is defined by example: wasu (WASDB) = 0.036 + 0.468 + 0.576 +0.672 +1.344 +1.596 + 2.688+1.232 = 8.612
- The most challenging problem for high utility Sequential mining is that web access sequence utilities do not have the downward closure property. Consider μ = 2.153, then web page News Feed is a low utility WAS as wasu (News Feed) = 0.036, but its super-sequence {News Feed, game} is a high utility WAS. Therefore, the downward closure property is not satisfied. To maintain the downward closure property in high utility WAS mining, a new measure called web access sequence weighted utility (WASWU) is defined using the two-phase approach discussed in the section 2.3.2.

3.2.2. CONSTRUCTION OF HUU-PLWAP TREE EXAMPLE (From Figure 13)

The **HUU-PLWAP Tree Construction** in the figure 13 is explained through the example:

Step i: The HUU-PLWAP tree algorithm works by first creating the root nodes and initialising its label and position code to null while its count set to 0. It detects candidate web pages (single-element Frequent-1 waswu sequences) in the first database scan.

Step ii: The web access sequence weighted utility value of a sequence X is defined by

$$\text{waswu (X)} = \sum \text{wasu (X in the Sequence)(from Equation 2)}$$

The X is a high waswu sequence only if waswu(X) is greater than or equal to μ.The downward closure property can be maintained by using this value. The waswu values are calculated as in Equation (2).

1) waswu (News Feed) = was(S1) + was(S2) + was(S3) + was(S4) + was(S5) + was(S6) + was(S7) + was(S8) =0.036 + 0.468 + 0.576 +0.672 +1.344 +1.596 + 2.688+1.232 = 8.612

2) waswu (Music) = was (S2) + was (S4) + was (S6) + was (S8)

= 0.468 + 0.672 ++1.596 + 1.232 =3.968

3) waswu (Video) = was (S3) + was (S4) + was (S7) + was (S8)

= 0.576 +0.672 + 2.688+1.232 = 5.168

4) waswu (Game) = was (S5) + was (S6) + was (S7) + was (S8)

= 1.344 +1.596 + 2.688+1.232 = 6.86

5) waswu (News Feed, Game) = was (S5) + was (S6) + was (S7) + was (S8)

= 1.344 +1.596 + 2.688+1.232 = 6.86

6) waswu (News Feed, Video) = was(S3) + was(S4) + was(S7) + was(S8)

= 0.576 +0.672 + 2.688+1.232 = 5.168

7) waswu (News Feed, Music) = was (S2) + was (S4) + was (S6) + was (S8)

= 0.468 + 0.672 +1.596 +1.232 = 3.968

8) waswu (Game, Video) = was(S7) + was(S8) = 2.688+1.232 = 3.92

9) waswu (Game, Music) = was(S6) + was(S8) = 1.344 +1.232 = 2.576

10) waswu (Video, Music) = was(S4) + was(S8) = 0.672 + 1.232 = 1.902

11) waswu (News Feed, Game, Video) = was(S7) + was(S8) = 2.688+1.232 = 3.92

12) waswu (News Feed, Game, Music) = was(S6) + was(S8) =1.596 + 1.232 = 2.828

13)waswu (News Feed, Video, Music) = was(S4) + was(S8) = 0.672 +1.232 = 1.904

14)waswu (Game, Video, Music) = was(S8) = 1.232

15) waswu (News Feed, Game, Video, Music) = was (S8) = 1.232.

Step iii: Insert all waswu values which are obtained and tabulated in the Table 32 with corresponding webpages in the Link header table.

NO OF PATTERNS	WASWU PATTERNS	WASWU VALUES OBTAINED
1	waswu (News Feed)	8.612
2	waswu (Music)	3.968
3	waswu (Video)	5.168
4	waswu (Game)	6.86
5	waswu (News Feed, Game)	6.86
6	waswu (News Feed, Video)	5.168
7	waswu (News Feed, Music)	3.968
8	waswu (Game, Video)	3.92
9	waswu (Game, Music)	2.576
10	waswu (Video, Music)	1.902
11	waswu (News Feed, Game, Video)	3.92
12	waswu (News Feed, Game, Music)	2.828
13	waswu (News Feed, Video, Music)	1.904
14	waswu (Game, Video, Music)	1.232
15	waswu (News Feed, Game, Video, Music)	1.232

Table 32: waswu values

Step iv: A sequence X is a high utility web access sequence if wasu(X) is greater than or equal to μ. Mining high utility WASs means discovering all the sequences X having criteria wasu(X) ≥ μ. For example, {News Feed, Game} is one of the high utility WAS. For μ= 2.153, waswu (Video, Music) is less than μ, any super-sequence of waswu (Video, Music) cannot be a high waswu sequence and obviously cannot be a high utility WAS. Accordingly, μ = 2.153 and waswu values of the web pages are: waswu (News Feed) = 8.612; waswu (Music) = 3.968; waswu (Video) = 5.168; waswu (Game) = 6.86 and the remaining are in the Table 32. Those values are above μ are inserted into H.

Step v: The HUU-PLWAP tree is created starting from a null root as in the Figure 15.

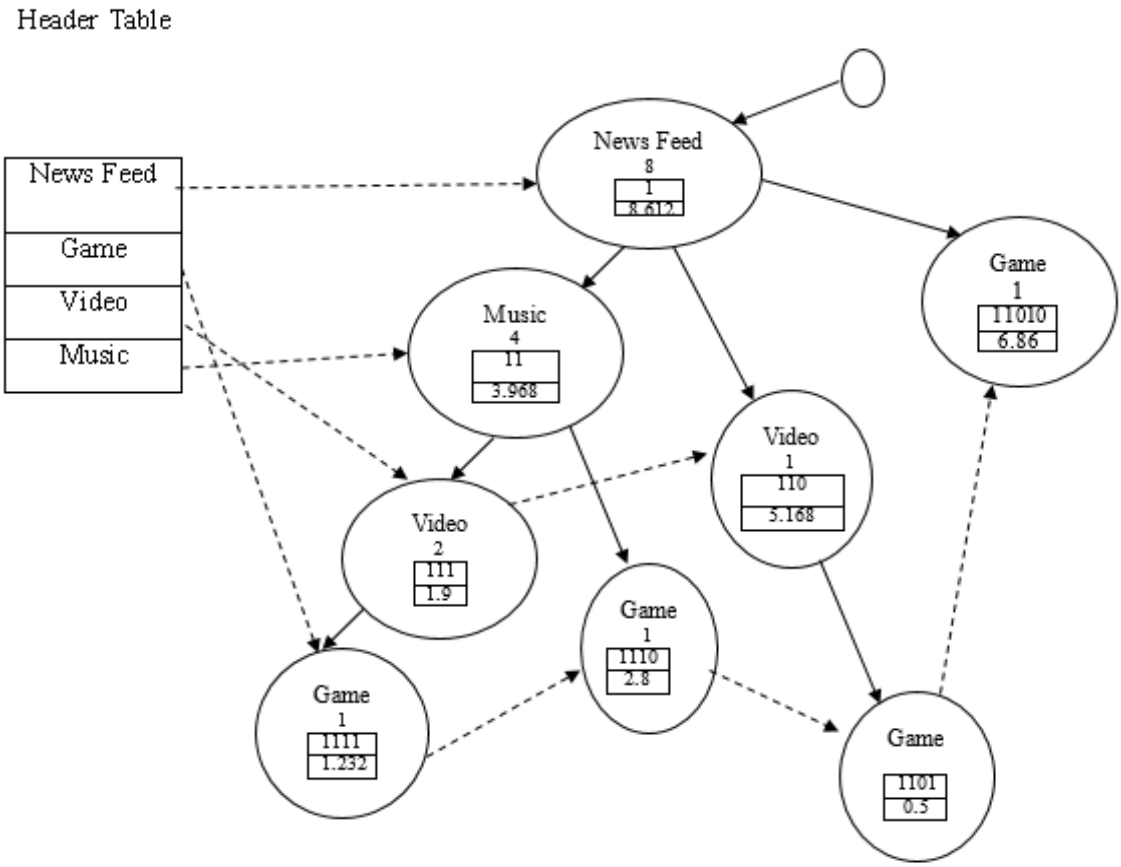

Figure 15 : The complete linked HUU-PLWAP tree

Sequences are read from the database and nodes are created for each item in the sequence. Each node contains the sequence label, waswu values and position code, denoted as label: waswu values: position code. The first sequence webpage (News Feed) is entered into the tree as the leftmost sub-tree since there no previously existing nodes. Otherwise, a right node is created and the count initialised to 1. If node already exists, its count is incremented by 1. The second sequence (News Feed) (Music) is read. Starting from the root, a node with label '(News Feed)' is inserted and the count of this node is incremented by 1. Its sequence ID is recorded against its waswu value. The position code is found by appending '1' to the end of position code of current node. The sequence label and its waswu values are read from the sequence database. The third sequence (News Feed) (Video) is read. Starting from the root, since a node with label '(News Feed)' already exist the count of this node is incremented by 1. Entries created in the header table are then used to link their corresponding nodes by traversing the HUU-PLWAP tree in a pre-ordered fashion (from root to left node first before right node). This makes tree traversal faster and more efficient since similar nodes in the same suffix tree are brought closer. When a left child already exists, and has same label as the item read, its count is incremented by 1. However, the left child has a different label from the item read; the U-PLWAP algorithm continues to search all the right siblings for a match. When a match is found, its count is incremented by 1. The fourth sequence (News Feed) (Music) (Video) is read. Starting from the root, since a node with label '(News Feed)' already exist the count of this node is incremented by 1. Its sequence ID is recorded against its waswu value. The fifth sequence (News Feed) (Game) is read. Starting from the root, since a node with label '(News Feed)' already exist the count of this node is incremented by 1. Its sequence ID is recorded against its waswu value. The sixth sequence (News Feed) (Music) (Game) is read. Starting from the root, since a node with label '(News Feed)' already exist the count of this node is incremented by 1.

Step vi: The complete process is repeated until the end of the sequence fetched is reached. A new sequence is then fetched, this time the current node pointer is set to the root of the HUU-PLWAP tree. A third database scan is needed for finding the high utility web access sequences from the candidate sequences using the waswu values. The prefixing paths of (Game) are: (News Feed, Music), (News Feed, Video), News Feed. The waswu values of web pages associated with web page Game are (News Feed, Music):2.8, (News Feed, Video):3.92, (News Feed): 6.86.

Step vii: The HUU-PLWAP tree created is then mined recursively using prefix conditional search. Starting from the root with a frequent item 'News Feed' (in order to find all the high waswu values which are above the minSeqUtil) on the header table, all sub-trees are traversed.

3.2.3. THE HUU-PLWAP MINER ALGORITHM (From Figure 14)

Step 3: Recursive mining using the mining algorithm is performed on the conditional tree of web pages. This was explained in the Miner algorithm in the Figure 14.

Step I: From the root, the HUU-PLWAP tree is traversed for the first sequence of 'News Feed' in the suffix trees of the root node. In this case: News Feed 1: 1: 8.612. The waswu value of 'News Feed' is above the threshold. This confirms 'News Feed' is a high utility uncertain sequential pattern. Likewise, all the sequences in the uncertain sequence database are traversed with the corresponding waswu values. Subsequently, for each web page (Game) in H, the "for loop" creates a new candidate sequence (News Feed) (Game) and inserts into candidate sequence list L.

Step II: Finally, second "for loop" creates the conditional suffix tree of sequence (News Feed) (Game) from the header table H and makes a recursive call to the mining procedure.

Step III: However, mining procedure can be used for mining operation with a HUU-PLWAP-tree. The search continues down the tree to find all the high utility web access

sequences and the candidates (Game, Video, News Feed), (Game, News Feed, Music), (News Feed, Game) are generated. Candidate sequences prefixing other web pages can be generated in a same fashion. The process continues recursively until no more items is found. The algorithm then backtracks to the null root to start mining for sequences starting with a new item from the header table. In total the high utility uncertain sequential patterns generated from uncertain sequence database are: (News Feed); (Music); (Video); (Game); (Game, Video); (News Feed, Game); (Video, News Feed); (Game, Music); (Game, Video, News Feed); (Game, News Feed, Music).

Step IV: The existing utility-based algorithm, PHUI needs several database scans and generates enormous candidates. The proposed approach significantly reduces both of them compared to the existing method. HUU-PLWAP is also richer than PHUI in output since HUU-PLWAP discovers frequent sequential pattern while PHUI does not.

4. COMPARATIVE ANALYSIS

This chapter shows analysis of how the proposed solution is more efficient than previously existing PHUI-UP methodology. But the PHUI-UP algorithm works on the uncertain transaction databases, not with the uncertain web access sequential databases. Also, it considers the probability measure as a separate entity not combined with the semantic measure utility. So, it quite difficult to compare with the proposed framework because in the proposed system the internal utility is combined with existential probability, unlike the PHUI-UP algorithm in which the internal utility is not combined with any existential probability. Also, the PHUI-UP generates the non-sequential high utility uncertain patterns. The second algorithm to compare with the proposed framework is U-PLWAP. It is an efficient algorithm, which is designed to mine uncertain data sequences in traditional sequential datamining algorithms which is not combined with the Utility measure. It also supports the expected support model which is not applicable in the proposed method. The probabilities defined in the U-PLWAP are from the access history probabilities, but in the proposed method the probabilities are generated based on the sequence based uncertainty model.

4.1. COMPARING HUU-PLWAP WITH PHUI-UP AND U-PLWAP

Comparing HUU-PLWAP with U-PLWAP and PHUI-UP might not be suitable since they produce different results. PHUI-UP is based on Apriori principle that generates candidate patterns (i.e., high utility uncertain itemsets). The Apriori technique inherits the problem of repeatedly scanning database and handling long sequences. To evaluate the performance of the proposed approach various experiments performed on synthetic datasets generated. The programs were written in Microsoft Visual C++ 6.0 and run with the Windows 10 operating system on an Intel® Core i5-2400 CPU 3.10 GHz with 4 GB main memory. In this section, experiments are conducted to see the effect of varying different parameters on HUU-PLWAP, U-PLWAP and PHUI-UP. The implementation is

done with C++ with the interface Code Blocks. The datasets are retrieved from Synthetic data is used from the SPMF: a data mining source library. However, these datasets do not provide the internal and external utility values of the sequences. As for the performance evaluation of the previous utility-based pattern mining (Yao et al., 2006), random numbers for the external and internal utility of each item are generated in each WAS, ranging from 1.0 to 10.0 and 1 to 10, respectively.

The following notations are used for the description of the datasets:
|D|: Number of sequences; |C|: Average length of the web access sequences; |N|: Number of events (web pages)

4.1.1. EFFECT OF MINIMUM (WAS) THRESHOLD ON EXECUTION TIME

The following parameters describe the dataset used for this experiment: |D| = 20k; |C| = 4; |N| = 1568. The Table 33 shows the results obtained from the algorithms HUU-PLWAP and PHUI-UP using different MINIMUM (WAS) Threshold.

MINIMUM (WAS) THRESHOLD (%)	HUU-PLWAP	PHUI-UP
10	1000	3310
20	900	3130
30	860	2720

Table 33: The performance of HUU-PLWAP and PHUI-UP with different min WAS Threshold

The experiment demonstrates that HUU-PLWAP is faster than PHUI-UP in all cases examined. The execution time increases with lower minimum WAS Threshold values as more high utility uncertain was sequences are found with lower minimum WAS Threshold. The graph in the Figure 16 shows the results obtained from the algorithms HUU-PLWAP and PHUI-UP using different Minimum (WAS) Threshold.

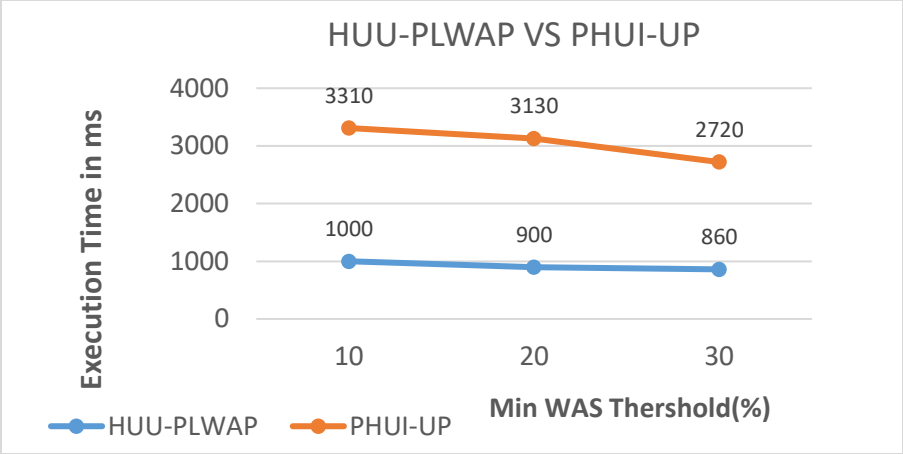

Figure 16: Comparing execution time of HUU-PLWAP & PHUI-UP with different min WAS Threshold

The performance of U-PLWAP and HUU-PLWAP with different minimum (WAS) Threshold is shown in the Table 34 and through the graph in Figure 17.

MIN WAS THERSHOLD (%)	HUU-PLWAP	U-PLWAP
0.005	156000	643000
0.05	102000	205000
0.5	60000	100000

Table 34: The performance of HUU-PLWAP and U-PLWAP with different min WAS Threshold

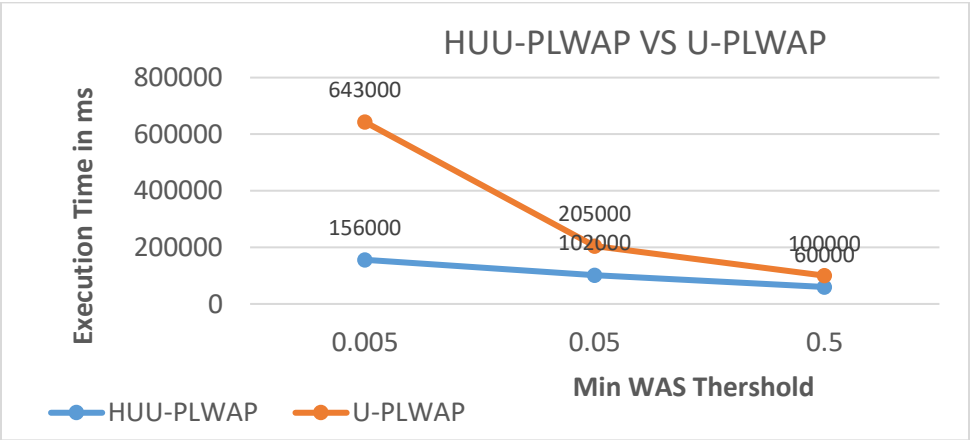

Figure 17: Comparing execution time of HUU-PLWAP & U-PLWAP with different min WAS Threshold

The experiment demonstrates that HUU-PLWAP is faster than U-PLWAP in all cases examined. The execution time increases with lower minimum WAS Threshold values as more high utility uncertain was sequences are found with lower minimum WAS Threshold.

4.1.2. EFFECT OF MINIMUM (WAS) THRESHOLD ON MEMORY USE

The following parameters are used for the dataset |D| = 20K; |C| = 4; |N| = 1125. The performance of HUU-PLWAP and PHU-UP with Minimum (WAS) threshold on memory use is shown in the Table 35 and through the graph in Figure 18.

Minimum WAS Threshold (%)	HUU-PLWAP	PHUI-UP
10	10000	4016
20	4624	2960
30	3080	2440

Table 35: Memory consumption of HUU-PLWAP, PHUI-UP with different min WAS Threshold

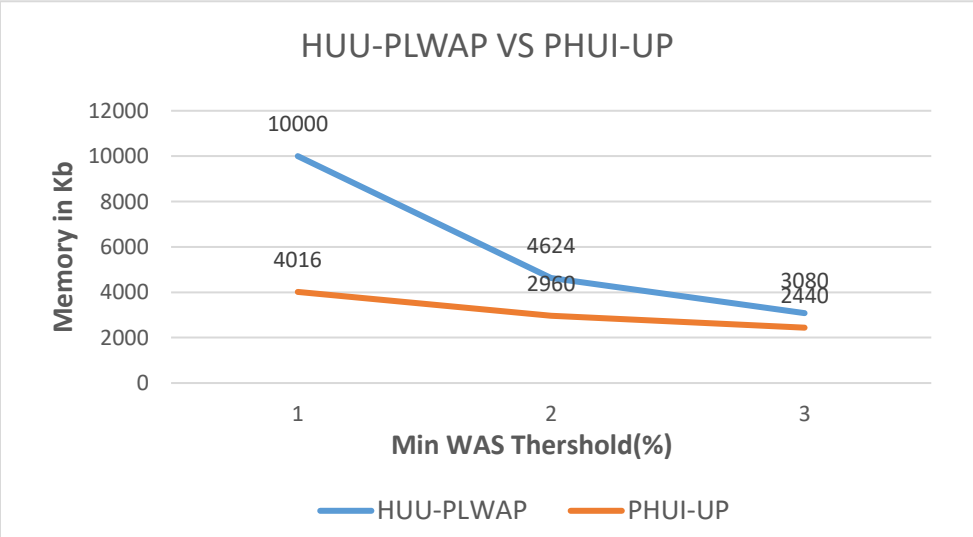

Figure 18: Memory utilised for the HUU-PLWAP, PHUI-UP with different min WAS Threshold

The memory requirement of HUU-PLWAP is lesser than PHUI-UP. The PHUI-UP algorithm also has the smallest memory requirement then U-PLWAP due to its limitation of generating only non-sequential frequent patterns. No additional memory is required to track the order of items.

The performance of HUU-PLWAP and U-PLWAP with Minimum (WAS) threshold on memory use is shown in Table 36 and through graph in Figure 19.

Minimum WAS Threshold (%)	HUU-PLWAP	U-PLWAP
0.005	10000	38000
0.05	4624	10000
0.5	3080	6800

Table 36: Memory consumption of HUU-PLWAP, U-PLWAP with different min WAS Threshold

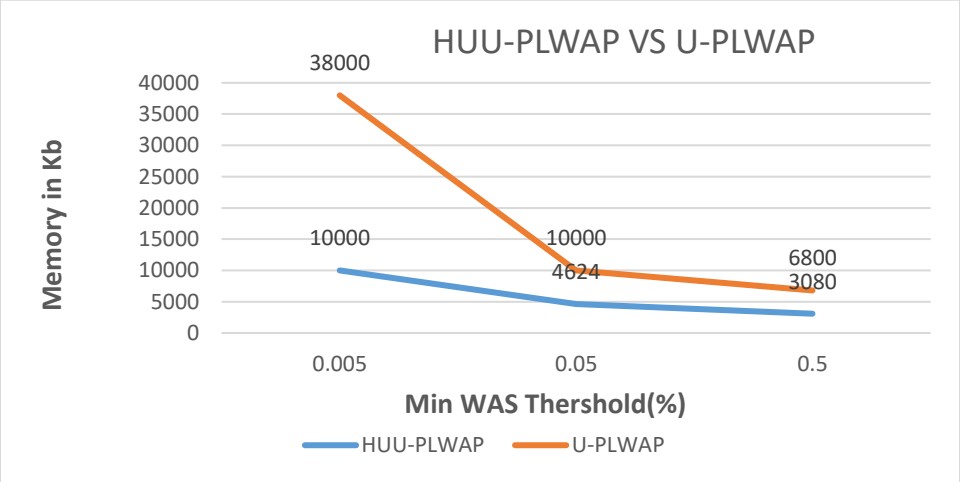

Figure 19: Memory utilised HUU-PLWAP, U-PLWAP with different min WAS Threshold

The memory requirement of HUU-PLWAP increases with increase in data size but the same cannot be said of U-PLWAP. This can be attributed to the fact that increase in data size does not automatically increase the amount of data generated during candidate sequence generation.

4.2. COMPLEXITY ANALYSIS

The time complexity for mining the tree in PLWAP is O (fp), where f is the number of frequent 1-sequences and p is the total number of resulting frequent patterns. The estimate of the time complexity of HUU-PLWAP is shown here. The worst-case scenario is considered in both cases. The following notations are used:

N – Number of sequences in the database

W – Number of 1-WASWUvalues

L – Length of the longest sequence

M – Length of the longest event queue for Mining

4.2.1. TIME COMPLEXITY OF HUU-PLWAP

The HUU-PLWAP algorithm also finds 1-WASWU values at the first scan of the database, builds the tree HUU-PLWAP tree with the second database scan. The 1-WASWU values found are then linked in HUU-PLWAP tree. The mining is then done recursively with the number of recursive calls for each 1-WASWU bounded by the longest possible sequence (Worst case).

The number of operations for computing the 1-WASWU values is: T1 = N x L.

The number of operations needed to construct the HUU-PLWAP tree is: T2 = N x W x L

Total time = (N x L) + (N x W x L) == (2(N x L) + 2 (N x W x L)).

For mining operation, time complexity was O (N)

Time complexity of HUU-PLWAP: O (2NWL + N) = O (2NWL)

5. CONCLUSION AND FUTURE WORK

5.1. CONCLUSION

Sequential pattern mining is a very important issue in data mining and machine learning. Most traditional sequence mining focuses on extracting patterns in the frequency/support framework which do not have business value and impact, and thus are not actionable for business decision-making. The introduction of “utility” brought not only valuable knowledge to sequence analysis but also recent problems and challenges. First, the absence of the Apriori Property in high utility sequence analysis makes the mining process fundamentally different with frequent sequence mining. Novel structures and algorithms need to be designed to improve performance and scalability and to conduct mining on platforms from industry level big data. Second, the measurements of a pattern utility in a sequence might be different. For example, in (Ahmed et al., 2010), the authors define the utility of a pattern as the sum of “all distinct occurrences” in a sequence. Different utility calculation definitions will lead to completely different utility bounds, properties, and pruning strategies. Third, the extracted patterns also suffer from a large amount of redundancy similar to frequent pattern mining. Many patterns look very alike. The challenge here is to explore approaches and algorithms which can efficiently and effectively summarize the patterns while losing only the smallest amount of information.

A generic framework is proposed to achieve some of the challenges which defined the calculations of the utility of a single item, an itemset, a sequence, a sequence database, an uncertain sequence. Also, instead of the maximum utility, it is always recommended to use the minimum utility threshold which is referred as minimum support in tradition sequential mining process to avoid the loss of valuable patterns. The utility-based approach has many limitations such as considering only precise databases instead of uncertain data. Also, the traditional uncertain sequential pattern algorithms such as U-Apriori, U-PLWAP do not include the utility measure. The only algorithm related to both uncertain and high utility PHUI-UP algorithm considers the probability and the utility as different entities and the retrieved patterns are not dependent with both due to two

different thresholds, and it does not deal with the web sequence databases. It also suffers from the level-wise candidate generation-and-test methodology, which needs several database scans and does not mine the web access sequences with different impacts/significances for different web pages. So, to overcome all the mentioned issues the proposed system is going to introduce an algorithm HUU-PLWAP which retrieves the high utility sequential patterns from the uncertain web access sequence databases with the help of U-PLWAP methodology, the uncertain sequence and their associated existential probabilities are retrieved based on the tuple based uncertainty model. Here the external utility values represent the quality (i.e., the rating of a web page) and the internal utility values are quantity based on the user activities (i.e., the number of times that page visited) which are uncertain and probabilistic. Experiments show that HUU-PLWAP is at least 95% faster than U-PLWAP, and 75% faster than a PHUI-UP algorithm.

5.2. FUTURE WORK

Extensive work can be done to finalize the framework, as well as extending to new areas, which include:

1. The limitation of the high utility sequential pattern mining framework is the utilities of the sequences are not allowed to be negative numbers, which is not applicable in some real-world applications

2. The proposed work works on the single set sequences (for example, < {a}, {b}, {c}>), the future work can be a part of multi-itemset sequences (for example, < {a, b}, {c, d}, {a, c, d} >.

3. The proposed system considers the uncertainty only in the internal utility value, but in future, there is the scope of considering a probabilistic value for the external utility too.

REFERENCES

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

Aggarwal, C. C., & Philip, S. Y. (2009). A survey of uncertain data algorithms and applications. IEEE Transactions on Knowledge and Data Engineering, 21(5), 609-623.

Aggarwal, C. C. (2009). Trio a system for data uncertainty and lineage. In Managing and Mining Uncertain Data (pp. 1-35). Springer US.

Aggarwal, C. C., Li, Y., Wang, J., & Wang, J. (2009, June). Frequent pattern mining with uncertain data. In Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 29-38). ACM.

Aggarwal, C. C. (Ed.). (2010). Managing and mining uncertain data (Vol. 35). Springer Science & Business Media.

Ahmed, C. F., Tanbeer, S. K., Jeong, B. S., & Lee, Y. K. (2009, April). An efficient candidate pruning technique for high utility pattern mining. In Pacific-Asia Conference on Knowledge Discovery and Data Mining (pp. 749-756). Springer Berlin Heidelberg.

Ahmed, C. F., Tanbeer, S. K., & Jeong, B. S. (2010). A novel approach for mining high-utility sequential patterns in sequence databases. ETRI journal, 32(5), 676-686.

Alpaydin, E. (2011). Introduction to Machine Learning (2nd Ed.). Cambridge, MA, USA: MIT Press.

Ayres, J., Flannick, J., Gehrke, J., & Yiu, T. (2002, July). Sequential pattern mining using a bitmap representation. In Proceedings of the eighth ACM SIGKDD

international conference on Knowledge discovery and data mining (pp. 429-435). ACM.

Bakariya, B., & Thakur, G. S. (2015). An Efficient Algorithm for Extracting High Utility Itemsets from Weblog Data. IETE Technical Review, 32(2), 151-160.

Bernecker, T., Kriegel, H. P., Renz, M., Verhein, F., & Zuefle, A. (2009, June). Probabilistic itemset mining in uncertain databases. In Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 119-128). ACM.

Bolton, R. J., & Hand, D. J. (2001). Unsupervised profiling methods for fraud detection. *Credit Scoring and Credit Control VII*, 235-255.

Chan, R. C., Yang, Q., & Shen, Y. D. (2003, November). Mining high utility itemsets. In Data Mining, 2003. ICDM 2003. Third IEEE International Conference on (pp. 19-26). IEEE.

Chen, L., Özsu, M. T., & Oria, V. (2005, June). Robust and fast similarity search for moving object trajectories. In Proceedings of the 2005 ACM SIGMOD international conference on Management of data (pp. 491-502). ACM.

Cheng, H., Yan, X., Han, J., & Hsu, C. W. (2007, April). Discriminative frequent pattern analysis for effective classification. In *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on* (pp. 716-725). IEEE.

Cheng, H., & Han, J. (2009). Frequent itemsets and association rules. In Encyclopedia of Database Systems (pp. 1184–1187). Springer.

Chui, C. K., Kao, B., & Hung, E. (2007, May). Mining frequent itemsets from uncertain data. In Pacific-Asia Conference on Knowledge Discovery and Data Mining (pp. 47-58). Springer Berlin Heidelberg.

Chun Kit Chui, Ben Kao, and Edward Hung. Mining frequent itemsets from uncertain data. In 11th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining, PAKDD 2007, Nanjing, China, pages 47 58, 2007.

Deshpande, A., Guestrin, C., Madden, S. R., Hellerstein, J. M., & Hong, W. (2004, August). Model-driven data acquisition in sensor networks. In Proceedings of the Thirtieth international conference on Very large data bases-Volume 30 (pp. 588-599). VLDB Endowment.

De Carvalho, J. V., & Ruiz, D. D. (2013, July). Discovering frequent itemsets on uncertain data: a systematic review. In International Workshop on Machine Learning and Data Mining in Pattern Recognition (pp. 390-404). Springer Berlin Heidelberg.

Ezeife, C. I., & Lu, Y. (2005). Mining web log sequential patterns with position coded pre-order linked wap tree. Data Mining and Knowledge Discovery, 10(1), 5-38.

Han, J., Cheng, H., Xin, D., & Yan, X. (2009). Frequent pattern mining: current status and future directions. Data Mining and Knowledge Discovery, 15 (1), 55–86.

Han, J., Pei, J., & Yin, Y. (2000, May). Mining frequent patterns without candidate generation. In ACM Sigmod Record (Vol. 29, No. 2, pp. 1-12). ACM.

Han, J., & Kamber, M. (2011). Data Mining, Concepts and Techniques (3rd Ed.). Waltham, MA, USA: Morgan Kaufmann.

Jain, A. K. (2010). Data clustering: 50 years beyond K-means. Pattern Recognition Letters, 31 (8), 651–666.

Kadri, O., & Ezeife, C. I. (2011, March). Mining uncertain web log sequences with access history probabilities. In Proceedings of the 2011 ACM Symposium on Applied Computing (pp. 1059-1060). ACM.

Khoussainova, N., Balazinska, M., & Suciu, D. (2006, June). Towards correcting input data errors probabilistically using integrity constraints. In Proceedings of the 5th ACM international workshop on Data engineering for wireless and mobile access (pp. 43-50). ACM.

Lan, G. C., Hong, T. P., Tseng, V. S., & Wang, S. L. (2012, August). An improved approach for sequential utility pattern mining. In Granular Computing (GrC), 2012 IEEE International Conference on (pp. 226-230). IEEE.

Leung, C., Mateo, M., & Brajczuk, D. (2008). A tree-based approach for frequent pattern mining from uncertain data. Advances in Knowledge Discovery and Data Mining, 653-661.

Lin, J. C. W., Gan, W., Fournier-Viger, P., Hong, T. P., & Tseng, V. S. (2016). Efficient algorithms for mining high-utility itemsets in uncertain databases. Knowledge-Based Systems, 96, 171-187.

Liu, Y., Liao, W. K., & Choudhary, A. (2005, August). A fast high utility itemsets mining algorithm. In Proceedings of the 1st international workshop on Utility-based data mining (pp. 90-99). ACM.

Liu, B. (2007). Web data mining: exploring hyperlinks, contents, and usage data. Springer Science & Business Media.

Liu, M. & Qu, J. (2012), Mining high utility itemsets without candidate generation, in 'Proceedings of the 21st ACM International Conference on Information and Knowledge Management', CIKM '12, ACM, New York, NY, USA, pp. 55–64.

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

Manike, C., & Om, H. (2015). Time-Efficient Tree-Based Algorithm for Mining High Utility Patterns. In Advances in Intelligent Informatics (pp. 409-418). Springer International Publishing.

Parmar, D. K., Rathod, Y. A., & Patel, M. M. (2013, December). Survey on high utility oriented sequential pattern mining. In Computational Intelligence and Computing Research (ICCIC), 2013 IEEE International Conference on (pp. 1-7). IEEE.

Pei, J., Han, J., Mortazavi-Asl, B., & Zhu, H. (2000). Mining access patterns efficiently from web logs. In Knowledge discovery and data mining. Current issues and new applications (pp. 396-407). Springer Berlin Heidelberg.

Peterson, L. (2009). K-nearest neighbor. Scholarpedia, 4(2), 1883. Retrieved April 18, 2017.

Sistla, A. P., Wolfson, O., Chamberlain, S., & Dao, S. (1998). Querying the uncertain position of moving objects. In Temporal databases: research and practice (pp. 310-337). Springer Berlin Heidelberg.

Siciliano, R., & Conversano, C. (2009). Decision Tree Induction. Encyclopedia of Data Warehousing and Mining, Second Edition, 624-630. Retrieved April 18, 2017.

Srikant.R and R. Agrawal (1996) Mining sequential patterns: Generalizations and performance improvements, Proc. of the 5th Int. Conf. Extending Database Technology, pp.3-17, 1996.

Tseng, V. S., Shie, B. E., Wu, C. W., & Philip, S. Y. (2013). Efficient algorithms for mining high utility itemsets from transactional databases. IEEE transactions on knowledge and data engineering, 25(8), 1772-1786.

Veeramuthu, P., Periyasamy, R., & Sugasini, V. Analysis of Student Result Using Clustering Techniques.

Vijayalakshmi, S., Mohan, V., & Raja, S. S. (2010). Mining of user's access behavior for frequent sequential pattern from web logs. International Journal of Database Management System (IJDM), 2.

Wang, J. Z., Yang, Z. H., & Huang, J. L. (2014). An efficient algorithm for high utility sequential pattern mining. In Frontier and Innovation in Future Computing and Communications (pp. 49-56). Springer Netherlands.

Yao, H., Hamilton, H. J., & Butz, C. J. (2004, April). A Foundational Approach to Mining Itemset Utilities from Databases. In SDM (Vol. 4, pp. 215-221).

Yin, J. (2015). Mining High Utility Sequential Patterns (Doctoral dissertation, University of Technology, Sydney).

Yin, J., Zheng, Z., & Cao, L. (2012, August). USpan: an efficient algorithm for mining high utility sequential patterns. In Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 660-668). ACM.

Zhou, L., Liu, Y., Wang, J., & Shi, Y. (2007, October). Utility-based web path traversal pattern mining. In Data Mining Workshops, 2007. ICDM Workshops 2007. Seventh IEEE International Conference on (pp. 373-380). IEEE.

Zihayat, M., Wu, C. W., An, A., & Tseng, V. S. (2015, October). Mining high utility sequential patterns from evolving data streams. In Proceedings of the ASE Big Data & Social Informatics 2015 (p. 52). ACM.

VITA AUCTORIS

Sravya was born in 1986 in Andhra Pradesh, India. She received her Bachelor's degree in Computer Science from Anna University, Chennai, India in 2008. She worked in multinational company Wipro Technologies in Bengaluru, India from 2009 to 2014. She completed her M.Sc. in Computer Science from the University of Windsor, Ontario Canada in May 2017. Her research interests include data mining, webpage development, 3d animation and pattern recognition and artificial intelligence.