ISSN:2153-182X, E-ISSN: 2153-1838
Vol. 17 No. 02 (2023)

MINING SEQUENTIAL RULES FROM UNCERTAIN SEQUENCES DATABASE

Imane Seddiki[1]*, Farid Nouioua[1,2] and Abdelbasset Barkat[3]
[1]Computer science Department, Mouhamed El Bachir El Ibrahimi University, Bordj Bou Arreridj 34000, Algeria .
[2]LIS UMR-CNRS 7020, Aix-Marseille University, Marseille, France.
[3]Laboratory of Informatics and its Applications, Faculty of Mathematics and Computer Ccience, University of M'sila, M'sila 28000, Algeriaabdelbasset.barkat@univ-msila.dz.
*Corresponding author(s). E-mail(s): imane.seddiki@univ-bba.dz;
Contributing authors: farid.nouioua@gmail.com; abdelbasset.barkat@univ-msila.dz;

Abstract
Since the amount of data has become a vital fuel to powering many real-world applications, many issues arise in data mining regarding storage and real-time aspects. Indeed, mining association rules is one of the strategies used to find associations and relationship rules among large sets of data items. However, this data does not always have precise values due to the fact that it may be either incomplete or uncertain, which is a challenging problem when applying mining algorithms. To deal with these issues, we propose a new approach for mining sequential rules from uncertain sequence databases that consists of two main steps: The first one is to extract the set of probabilistic rules, and the second one is to filter this set using the sequential information in the data.
Keywords: Association rule, Sequential rule, uncertain data, probabilistic database, sequences database.

1 Introduction

Association rules mining is a powerful technique in the field of data mining [5]. This technique has already proved its strength in different contexts and especially in modeling the behavior of customers by analyzing a market-basket database representing the purchases of these customers. In this context, each association rule reflects how some purchases are associated with some others.

Association rules mining has been extended to sequential databases which contain a set of sequences. Unlike classical transactional databases where the order of events does not matter, events in a set of sequences should respect some partial order. A sequential rule is a pattern extracted from original sequence database to reflect relevant associations between subsequences of events. Such a rule is then employed to identify and predict the occurrences of some subsequences among set of items [1]. Sequence databases are used in many domains, such as bioinformatics to store the DNA (DeoxyriboNucleic Acid) or Protein sequences, and marketing to store the purchases sequence of customers.

In many real life applications, data could be stored and processed within missed or mislabeled information. For example, data collected from wireless sensors network may be uncertain due to external interfering factors or sensor aging, which makes the process of extracting association or sequential rules rather complicated and needs special treatment to deal with uncertain data. The uncertainty of this data is often modeled as an existential probability which is either associated with every sequence in the database or with every item in each sequence in the database. Hence, if the probability of a sequence (item) is equal to one, this means that the occurrence of this sequence (item) is certain.

This paper aims at solving the problem of extracting sequential rules from uncertain sequence data (uncertain sequence database). The remainder of this paper is organized as follows: In Section 2, the related work is discussed. In Section 3, we provide the definition as well as the formalization of the problem. The proposed approach is explained in section 4. Experimental results are presented in Section 5. Finally, Section 6 concludes the paper.

2 Related work

In this section, we present a review of the main related work to mining sequential rules from uncertain data. After a deep review of past literature, we noticed that the related work can be divided into two main categories: The first one is about finding frequent itemsets from probabilistic data, and the second one is about mining sequential patterns and rules from sequential data. Additionally, we can split the first category into three subcategories based on the method used to find the frequent itemsets: The first subcategory uses the concept of expected support. The second one consists of work that uses exact methods,i.e., developing methods to compute the exact probability of patterns frequencies while the third subcategory consists of approaches that rely on approximation models [22] in which the frequency probability of a pattern is approximated by a probability law . In the next few paragraphs, we are going to discuss a set of papers that cover all the categories. For more details, the reader can refer to two surveys that have successfully synthesized most of established research for mining uncertain data [9, 10]. Table 1 below represents a comparative summary for a set of key existing work related to our problem.

Within the first category, which uses the expected support as a main metric, we discuss the study entitled ”Mining interesting patterns from uncertain databases” [23]. In this paper, the authors claim that all the proposed algorithms for mining frequent itemsets from uncertain database used support based constraint to reduce the search space, which is not enough because the frequent itemsets may have weak affinity. Therefore, they propose a new approach called Weighted Uncertain Interesting Pattern Mining (WUIPM), where they use the concept of expected support to identify the set of itemsets that are considered frequent, and propose a new tree structure WUIP-tree where a weight value is given for each item, which represents the importance of this item. Then, they introduce a new measure called weighted expected support confidence (wUConf) to mine interesting patterns with importance. The authors claim that WUIPM is faster than the other algorithms.

In the paper entitled ”Mining Uncertain Data with Probabilistic Guarantees” [3], the authors aim to discover frequent patterns using an exact method, where they propose two algorithms to discover frequent patterns from probabilistic databases: The first one uses a bottom-up strategy while the second one uses a top-down strategy. Both algorithms are based on the possible world semantics. The authors explain how these algorithms can be extended to discover maximal frequent patterns to generate the set of association rules.

The default way to deal with uncertainty data while mining for frequent itemsets is using possible world semantics. However, this is not efficient because When there is more uncertainty in the data, the number of possible worlds increases. The authors of the paper [21] propose a new

ISSN:2153-182X, E-ISSN: 2153-1838
Vol. 17 No. 02 (2023)

approach to solve this problem by using a probability distribution function with two models: the Poisson distribution and the normal distribution which can be classified under the first category that rely on approximation model. The objectives of this research are: 1) finding itemsets with frequentness probabilities that are greater than some threshold 2) mining itemsets with the k highest frequentness probabilities. The proposed approach consider both tuple and attribute uncertainty models.

Since there are more than one possible world in uncertain database, the authors use S(Wj,I) to describe the support of an itemset I in possible world Wj than they define the probability that S(I) has a value of i, denoted by PrI(i) = Pwj∈W,S(Wj=i) Pr(wj), and probability mass function (pmf) PrI(i)(i = 1...n) named as the support pmf. Finally they define the concept of the frequentness probability of I: Prfreq(I) = Pi≥minsup PrI(i). Because computing this function is too expensive, the authors use the expected support and the two probability distribution (Poisson and normal distribution) to approximate the distribution of s(I) in efficient way.

In a conclusion, they state that the models of expected support and Poisson approximation are both easy to implement and efficiently computed because pruning can be performed. However, with the model of Normal approximation pruning cannot be performed and its implementation require more effort than the previous two

In the work presented in [4] the authors present an algorithm called ProFPGrowth, based on the classical FP-Growth algorithm. The main task of this algorithm is to mine a probabilistic database to extract the set of frequent itemsets known in this paper as the probabilistic frequent itemsets. They also introduce the probabilistic Frequent Pattern Tree, or ProFP-Tree which represents a transformation of the probabilistic database into a tree format to facilitate the process of getting information and to minimize the number of access to the original database. The identification of the set of probabilistic frequent itemsets is based on the generating functions. The authors claim that these functions reduce the computation complexity to O(N).

Under the second category that deals with mining sequential patterns and rules from sequential data, we are going to review two papers, the first study presents the algorithm CMRules algorithm in a paper entitled ”Mining sequential rules common to several sequences” [1]. According to its authors, CMRules is widely used in applications such as: stock market analysis, weather observation, drought management and e-learning. To reduce the search space, the algorithm first transforms the sequence database into a transaction database. Then, it extracts the set of association rules that respect the minimum support and confidence thresholds in a classical way. After that, These association rules are examined to keep only those which respect the initial sequential ordering. In the end of this paper, the authors evaluate the CMRules algorithm in three ways: its time complexity, its performance on real datasets and its results in an intelligent agent in the international space station, where the agent became able to learn the sequential relationships between events common to its execution. The authors show that CMRules obtained interesting results in all these three contexts.

The second study presents a new idea for mining sequential rules in the paper entitled ”ERMiner: Sequential Rule Mining Using Equivalence Classes” [14], where the authors start to discuss the problem of sequential rules mining by showing the drawback of an existing algorithm named (Rule Growth), which performs a costly database projection operation repeatedly. To fix this drawback, they use a novel searching method based on the equivalence classes to mine the set of sequential rules, and they propose a data structure named SCM (Sparse Count Matrix) to reduce the search space.

Table 1 a comparative summary of rule mining related work

Works	Year	Type of data	Target	dataset	based Algo	Prog language
CMRules [1]	2012	Sequences	Sequential rules	Kosarak dataset	Apriori	Unknown
ProFP-Growth [4]	2012	Probabilisitc	Probabilistic frquent patterns	artificial datasets The accidents dataset	FP-Growth	Unknown
Liwen Sun et al[3]	2010	Probabilisitc	Probabilistic Association rules	IBM data generator The accidents dataset	Apriori with dynamic programing	C++
ERMiner[14]	2014	Sequences	Sequential Association rules	Sign, Snake, FIFA BMS and Kosarak10k	Equivalence Classes	Java
TP-rules[15]	2021	relational	Association rules	Boko Haram Bdataset	Equivalence Classes	uknown
CM-SPAM algorithm[16]	2020	Sequences	frequent sequential pattern	China pollution data PM2.5	Sequential pattern mining	uknown
CARM[16]	2021	conventional data	Association rules	online energy data portals	Context-based association rule mining	python
TUFP algorithm [19]	2020	Probabilistic	top-k frequent patterns	Chainstore, Foodmart, Retail and T10I4D100K,	TUFP algorithm	C
WFSPM framework[17]	2021	Sequences	frequent sequential pattern	WSpan and IUA	MaxPWS pruning	python
approximation model[21]	2013	Probabilistic	frequent itemsets	accidents T10I4D100k IBM	Poisson and normal distribution	C
WUIPM algorithm[23]	2016	Probabilistic	frequent itemsets	Frequent itemset mining implementations dataset repository	Expected support WUIP-tree structure	C/C++

3 Background and problem statement

In the next paragraphs, we present the definitions of some basic concepts that are necessary for our work according to the type of the data (transactional, sequential and uncertain).Then, we conclude this section by giving the theoretical basics and the definitions of the exact problem considered in this paper.

3.1 Transactional data

ISSN:2153-182X, E-ISSN: 2153-1838
Vol. 17 No. 02 (2023)

Association rule mining is one of the most important and well researched techniques of data mining. It aims at extracting interesting correlations, frequent patterns, associations or casual structures among sets of items in the transaction databases (TD) or other data repositories [6].
A Transaction database is defined as a set of transactions, where each transaction is a set of items, formally : $TD = \{t_1, t_2, t_3 ... t_n\}$ where $t_x \subseteq I$ is a transaction and $I = \{i_1, i_2, i_3 .... i_m\}$ is the set of all items.
An association rule between X and Y such that $X, Y \subseteq I$ and $X \cap Y = \emptyset$ is denoted by $X \rightarrow Y$.
Recall that the support of a subset of items $X \subseteq I$ is defined as the number of transactions that contain this subset: $Supp(X) = Card\{t \in TD \ s.t. \ X \subseteq t\}$, and the support of a rule $X \rightarrow Y$ is defined as the support of the union of the left side and the right side of the rule: $supp(X \rightarrow Y) = supp(X \cup Y)$. The confidence of a rule is defined as the ratio between its support and the support of the left side of the rule : $conf(X \rightarrow Y) = supp(X \cup Y)/supp(X)$.
An association rule $X \rightarrow Y$ is valid if and only if : $Supp(X \rightarrow Y) \geq minSup$ and $Conf(X \rightarrow Y) \geq minConf$ where $minSup$ (resp. $minConf$) is a minimum support (resp. confidence) threshold.

3.2 Sequential data

Sequential database. A sequence database is a database that contains a set of sequences where each sequence is constituted from n ordered set of transactions.
Formally, let $I = \{i_1, i_2, ..., i_n\}$ be a set of items. A sequence database (SD) is defined as a set of sequences $S = \{s_1, s_2, ..., s_n\}$ where each sequence s_x is an ordered list of transactions (itemsets): $s_x = \{X_1, X_2, ..., X_m\}$ such that
$X_1, X_2, ..., X_n \subseteq I$ [1].
Table 2 shows a sample of a sequence database containing five sequences.

Table 2 A sequence database

ID Seq	Sequences
Sequence 1	$[\{a\},\{b\},\{d\},\{e,f\}]$
Sequence 2	$[\{a,b\},\{c\},\{d\},\{e\}]$
Sequence 3	$[\{c,d\},\{f\}]$
Sequence 4	$[\{a,e\},\{c,d\},\{f\}]$
Sequence 5	$[\{a\},\{b,c\},\{d\},\{e\}]$

Sequential rules. The property that characterizes sequence databases leads to the emergence of a new type of rule called sequential rule, which is defined in [1] and [7] as follows: A sequential rule $X \rightarrow Y$ is a relationship between two itemsets X, Y such that $X \cap Y = \emptyset$. The interpretation of the rule $X \rightarrow Y$ is that if the items in X occur in some transactions of a sequence, the items in Y will occur in some transactions of the same sequence but afterward all the items of X. Note that items in X and in Y are unordered (they are not required to occur in the same transaction in a sequence).
Sequential support and sequential confidence. To find the set of relevant sequential rules, the researchers used an adaptation of the two measures used to find association rules. The first one called sequential support, is given by: $Sup(X \bullet Y)/Card(S)$ while the second one called sequential confidence is given by: $Sup(X \bullet Y)/Sup(X)$. The notation $sup(X \bullet Y)$ denotes the number of sequences where all the items of X appear before all the items of Y [8].

Uncertain data

According to [2], there are two types of uncertainty in probabilistic databases:

ISSN:2153-182X, E-ISSN: 2153-1838
Vol. 17 No. 02 (2023)

- **Tuple level uncertainty.** Each tuple in the database has an existential probability attribute, this model is characterized by its simplicity in terms of design and query. Table 3 represent an example of this type of database[3].
- **Attribute level uncertainty.** The uncertainty is about the value of an attribute of a transactions. I.e. each attribute has its own value of existential probability, which indicate the degree of certitude of this value. Table 3 represent an example of this type of database[4].

Table 3 A probabilistic database example(Tuple level uncertainty vs Attribute level uncertainty).

ID	loc	time	speed	trac	weather	prob
t1	x	8-9pm	30-40	high	Rain	0.1
t2	x	7-8am	80-90	low	null	1.0
t3	x	8-9pm	80-90	low	Foggy	0.5
t4	x	8-9pm	30-40	high	Rain	0.2
t5	y	2-3pm	50-60	low	Sunny	1.0

TID	Transaction
1	(A, 1.0) (B, 0.2), (C, 0.5)
2	(A, 0.1), (D, 1.0))
3	(A, 1.0), (B, 1.0), (C, 1.0), (D, 0.4)
4	(A, 1.0), (B, 1.0), (D, 0.5)
5	(B, 0.1), (C, 1.0)

Problem definition

In this paper we are interested in finding sequential rules from sequence databases that contain some uncertain data. The uncertainty is considered at the attribute level and it is modeled by the existential probability values associated to items.

Definition 1 Given a set of items $I = i_1,i_2,...,i_n$, a probabilistic sequence database (PSDB) is defined as a set of sequences $S = s_1,s_2,...,s_n$, where each sequence s_x is an ordered list of transactions (itemsets), $s_x = X_1,X_2,...,X_m$ such that $X_1,X_2,...,X_n \subset I$. These transactions may contain items that are not certain, and the presence of an uncertain item in a transaction is defined by an existential probability ($0 < p(item) < 1$). The item is considered to be certain if the existential probability is equal to 1.

In the following table (table 3.4) we consider the probabilistic sequence database which contains eight sequences. Each one containing certain and uncertain items. This PSDB is used as a running example in this paper.

Table 4 Example of PSDB

Sequences	Items

ISSN:2153-182X, E-ISSN: 2153-1838
Vol. 17 No. 02 (2023)

1	(Z, 0.2),(W, 0.6)
2	(X, 1.0), (Y, 1.0),(Z, 1.0), (W, 0.4)
3	(X, 1.0), (Y, 0.3), (Z, 0.8)
4	(X, 0.4), (Y, 0.7)
5	(Y, 0.4), (Z, 1.0))
6	(X, 0.1), (Y, 1.0),(Z, 1.0)
7	(X, 1.0), (Y, 1.0), (W, 1.0)
8	(Y, 0.3), (Z, 1.0)

The objective here is to extract the set of sequential rules that are considered as useful and strong enough. In other words, we look for rules that are ”sufficiently probable” and that respect the event order imposed in the sequences.There are established methods for extracting relevant rules from certain sequential data, known as sequential rules, as well as from uncertain and classical data, called probabilistic and association rules respectively. However, the focus of this paper is to combine these approaches in order to extract relevant rules from uncertain and sequential data, which we call ”probabilistic sequential rules”. The next section explains the method that we propose to extract the probabilistic sequential rules from uncertain sequence databases.

The proposed Approach

Before describing our proposal, let us present some assumptions that help to clearly and precisely specify our objectives. In this work, we assume that :

- The data are stored in probabilistic sequence database (PSDB).
- Each item is a singleton item (there is no nested sequences).
- Each item in every sequence has an existential probability in the interval $]0,1]$ (items with probability equal to 0 do not occur at all and hence they are not presented in the database.)

The key ideas on our proposal are as follows :

- To deal with uncertainty of data, we use the expected support measure to evaluate the probabilistic frequency of subsequences and sequential rules.
- To take into account the ordering constraints present in sequences, our approach operates in two phases: In the first phase the temporal information are ignored and a set of (classical) probabilistic rules is generated. Then, in the second phase, the rules generated in the first phase are filtered to keep only those that respect the temporal information.
- Our proposition is an FP-Growth-style algorithm to mine sequential rules.

Based on the problem definition and the set of assumptions presented above, we propose in this study an approach that takes place in six steps :

1. Transforming the probabilistic sequence database (PSDB) into a probabilistic database (PDB).
2. Extracting the set of singleton frequent items.
3. Generating a structure called the probabilistic frequent pattern tree (PFPTree).
4. Extracting the set of probabilistic frequent items by using the PFP-Tree.

ISSN:2153-182X, E-ISSN: 2153-1838
Vol. 17 No. 02 (2023)

5. Generating all probabilistic rules.
6. Filtering this set of rules using the temporal information present in the sequence database.

In the next sections we will give a detailed explication of each of the six steps stated above:

Transforming PSDB into PDB

We will apply a simple principle used in [1] to transform a probabilistic sequence database into a probabilistic database, which consists of ignoring the order of items imposed by the sequential nature of data to obtain a normal probabilistic database. it means that sequences are treated just like sets of items.

Extracting the set of singleton frequent patterns

In order to identify the set of singleton frequent patterns from the PDB, we have to consider all possible worlds generated from this PDB. Each possible world is obtained by taking a decision for each uncertain item whether it actually occurs or not. Hence a possible world w_i (the i^{th} possible world) represents a possible (certain) realization of the PDB with a probability denoted by $P(w_i)$. Using possible worlds, the expected support of an item A in the PDB by the following equation:

$$ExpSup(A) = \sum_{i=1}^{W} P(w_i)A_{w_i} \qquad (1)$$

The expected support of A is equal to the sum of multiplications of the probability of each possible world and the number of occurrences of A in this world (denoted in the equation by A_{wi}). However, the use of this equation is very expensive in terms of execution time, because the number of possible worlds generated from a PDB increases exponentially according to the number of uncertain items ($Card(W) = 2(number\ of\ uncertain\ items)$). Fortunately, we can prove that (the proof is available in the paper [12]) the expected support of an item A may be computed without computing all possible worlds as follows:

$$ExpSup(A) = \sum_{j=1}^{PDB} \prod_{x \in A} P(x) \qquad (2)$$

By applying Equation 2 on the running example (table 3.4) to calculate the expected support for each singleton item, we found that: $ExpSup(X) = 3.5, ExpSup(Y) = 4.7, ExpSup(Z) = 5, ExpSup(W) = 2$.

4.3 Constructing the probabilistic frequent pattern tree

The method for constructing the probabilistic frequent pattern tree (PFPtree) is inspired from [4], where we adapt the algorithm used by the authors to construct our version of the PFP-tree. In our work the construction of PFP-Tree needs a single scan of the PDB. The proposed algorithm takes the transactions of the PDB one by one and inserts them into the PFP-Tree, such that each transaction is represented by a path in the PFP-Tree. The transaction starts from the root of the tree (en empty node), and each node represents a singleton item in a transaction. The tree is structured as follows (let us consider a node n in the PFP-Tree) :

- n.item: a label used to denote the node. The label of the item is used here because each node represents one item.
- n.list: a list of transactions where the item of this node is present. This list is composed of

ISSN:2153-182X, E-ISSN: 2153-1838
Vol. 17 No. 02 (2023)

a set of pairs: Each pair contains the transaction label and the existential probability of the item in this transaction (if the item is certain we put the existential probability equals 1).

- n.next: a pointer to the next node in the tree. This field should be null if the current node is a leaf.

label	list	next
A	$t_1 = 1$ $t_2 =$ 0.6	t_j

The algorithm starts with a tree that contains a single element; which is the root of the tree (null element). Then, it scans the transactions one by one, starting with transaction t_1 that will be inserted directly as a path in the tree ($path = root \rightarrow t_1.item_1 \rightarrow t_1.item_2 ... \rightarrow t_1.item_n$) see figure 1.

The following transactions have two possibilities: either the path corresponding to the current transaction already exists in the tree. In this case an update is made by adding the label of this transaction and the existential probability of the current item in the corresponding node in the transactions list. Or the path corresponding to the currant transaction does not exist in the tree, thus, we completely create it.

Figure 2 represents the whole PFP-Tree generated after executing the algorithm on the running example.

Extracting the set of probabilistic frequent itemsets

In this step, we extract the set of probabilistic frequent itemsets based on the singleton frequent items identified in the previous step. Because the data are not all certain, we employ the concept of expected support used for the singleton frequent item defined by equation 2. For instance, by applying the equation 2 on the running example (table 3.4) to calculate the expected support for the itemset $\{Z, W\}$, we find that $ExpSup(\{Z, W\}) = 1.9$.

Generating probabilistic rules

The probabilistic rules is generated from the set of probabilistic frequent itemsets, where for each frequent itemset I^f, all its nonempty subsets are generated. For example, if s is a subset of I^f, we consider the rule $R : s \rightarrow I^f - s$ if its confidence (see equation 2) is greater or equal to $minConf$, where $minConf$ is the minimum confidence thresholds. Of course, when we want to compute the confidence of a generated rule we simply use the PFP-tree generated in

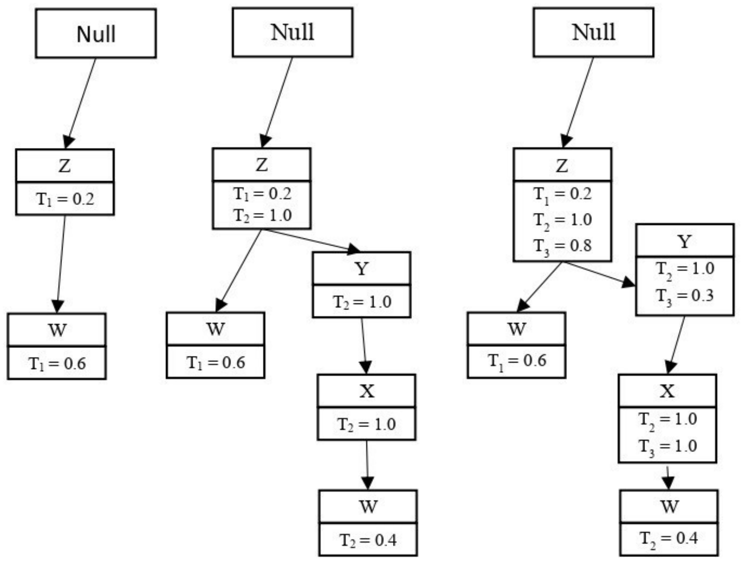


Fig. 1 Evolution (left to right) of the PFP-Tree after inserting the first three transactions step 3

rather than accessing the database.

$$Conf(R) = probSupp(I^f)/probSupp(S) \qquad (3)$$

Filtering probabilistic rules

After extracting the set of probabilistic rules, we move to the step of sequential filtering, such that for each rule found in the previous step we will calculate the sequential support and confidence as they are defined in this paper in section 3, to finally select only rules that have sequential confidence greater than or equal to a predefined value named minimum sequential confidence and denoted *SeqConf*.

If $x \rightarrow y$ is a rule in the set of rules found in the previous step, we define :

$$SeqSupp(x \rightarrow y) = Sup(x \bullet y)/Card(S) \qquad (4)$$

The notation $sup(x \bullet y)$ represents the number of sequences where x appears before y in the database and $Card(S)$ denotes the number of sequences in the database.
The sequential confidence of the rule is calculated by dividing the number of sequences where the left side appears before the right side of the rule, by the number of sequences that contain the left side of the rule :

$$SeqConf(x \rightarrow y) = Sup(x \bullet y)/Supp(x) \qquad (5)$$

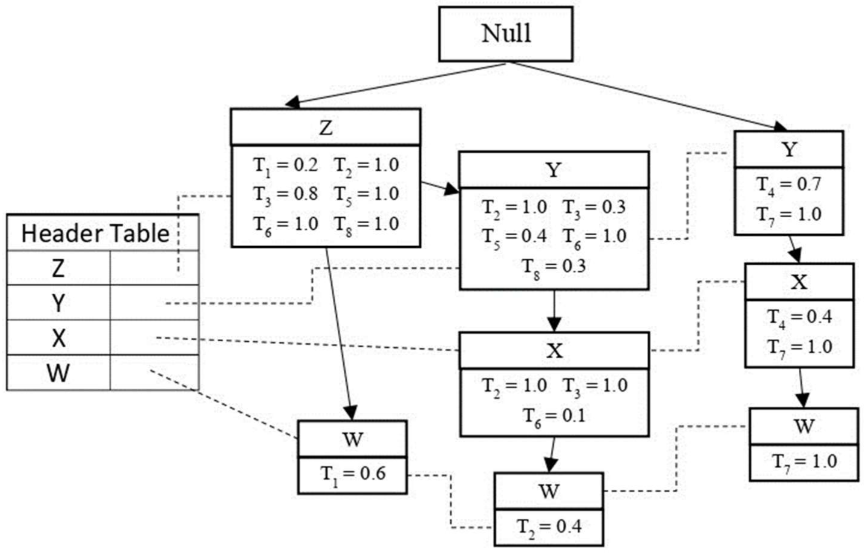


Fig. 2 The PFP-Tree after scanning the whole PDB

At the end of this step, only the rules with sequential confidence greater than or equal to a predefined threshold are selected.

5 Experimental results

To implement the proposed algorithm, we have used Python as a programming language installed on a Windows machine powered by Intel Core i5 processor with 16 GB of memory. For testing our algorithm, we have used both real dataset (Mushroom [20]) and synthetic datasets:

5.1 Testing the algorithm on synthetic data

Before discussing our results, we would like to give the reader an overview of our synthetic data. Our data are generated randomly. We simulated the definition of a probabilistic sequence database (PSDB) and generated data that contained a set of sequences. Each sequence is composed of a set of items, where each item has its own existential probability between zero and one. Then we create a set of files (10 files) that contain a number of these sequences. The first file has 100 sequences and the last one has 8k sequences.

Figure 3-(a) shows the time taken by the proposed algorithm to generate the set of sequential rules measured in seconds. The time depends on the number of sequences in data. We can see three plots in the figure 3-(a), one for the time taken by the whole algorithm, while the second for the time taken to generate the set of probabilistic rules and the last one for the time

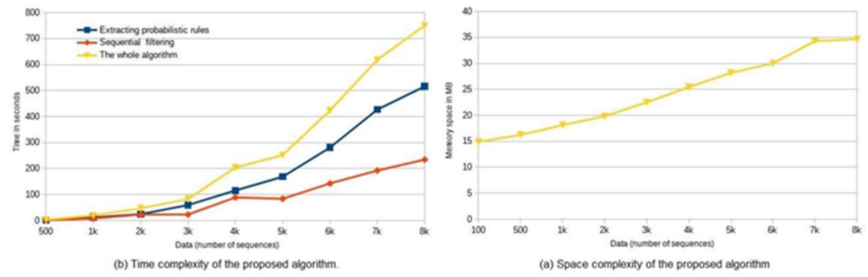


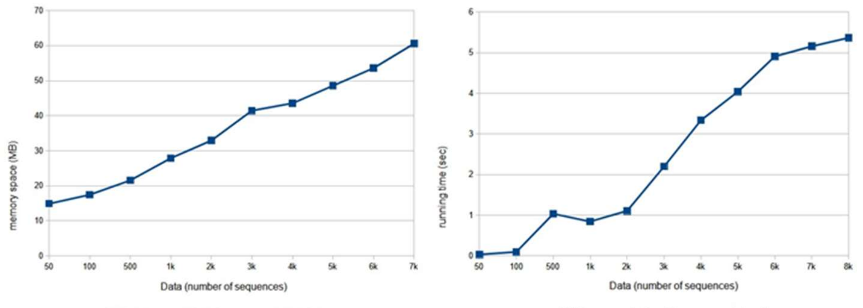
Fig. 3 Temporal and Spatial Complexity of the proposed algorithm tested on synthetic data

taken for the sequential filtering, where the value of the first one is equal to the sum of the two remaining times.

It is obvious that the execution time increases when the number of sequences increases. However, in terms of time complexity, the figure shows that the time complexity of the proposed algorithm is very acceptable and falls under O(nlog(n)) or O(n2) in the worst case using the big O notations to measure time complexity of an algorithm. The second graph (figure 3 (b)) represents the space complexity of the proposed algorithm, where the x-axis represents the number of sequences in our synthetic data, and the y-axis is the amount of memory needed to execute our proposed algorithm in megabytes. This test reveals that the memory space consumed by the algorithm is increasing according to the volume of input data. However, the algorithm shows its best performance when the input data becomes larger where the memory space used in that phase stays stable and continue to increase but with negligible rate.

5.2 Testing the algorithm on real data

The mushroom data set [20] used in this paper includes descriptions of academic samples corresponding to 23 species of gilled mushrooms in the Agaricus and Lepiota Family. Each species was determined as to be absolutely edible, absolutely poisonous, or edible of unknown and not recommended. The latter category is combined with toxic substances. The Guide states that there’s no simple rule for determining the edibility of a mushroom. In this phase we didn’t find a probabilistic sequence database used as dataset or a benchmark for evaluating these kind of researches. However a sequence database may be obtained from a transactional database by just considering order of items inside each transaction, and a probabilistic database is just an ordinary database where each item has its own existential probability. Therefore, we used the mushroom database and considered each transaction as a sequence and we add for each item a random value between zero and one represents an existential probability.



(a) Space complexity of the proposed algorithm. (b) Time complexity of the proposed algorithm.

Fig. 4 Temporal and Spatial Complexity of the proposed algorithm tested on real data

Figure 4 shows the spatial and temporal complexity of the proposed algorithm applied to the mushroom dataset. This set of experiments are performed with the value of (minimum support / number of sequences = 0.5). This figure contains two graphs, the first one 4-(a) shows the space complexity of our algorithm, where the x-axis shows the size of the dataset measured in terms of the number of sequences, while the amount of memory space used in each experiment appears on the y-axis. The graph confirms that the behavior of the proposed algorithm can be classified as quadratic space complexity to the size of data.

The second graph (4-b) represents the running time of our algorithm relative to the number of sequences in the dataset. In this case, the algorithm also shows a quadratic behavior where the running time increases while the number of sequences increases too. We can observe that in some cases, the execution time decreases between two experiments; Despite that, the data in the first experiment is larger than in the second one, this is because the number of rules found in the second experiment is bigger than the number in the first one, and this makes the time required to process this number of rules decreasing in the graph.

6 Conclusion

As long as only a few researches have been tackling the uncertainty of databases for improving the data mining performance, our proposed method contributed at discovering a new attribute of pattern of sequential rules from uncertain databases. Since our employed databases are noisy or mislabels, each item belongs every sequence is associated with an existential probability. The outcome proposed approach shown a remarkable performance in terms of complexity spatial and temporal.

To our knowledge and until writing these lines, there is no previous work which has considered the same problem. Therefore we couldn’t compare our results to similar ones in the same field. Our main contribution is to propose a new approach for mining sequential probabilistic rules from a probabilistic sequence database, where we explain in details the six steps in our proposition illustrated with a running example.

In the future, we aim to build a probabilistic sequence database which will be used for testing the new proposed mining algorithms, and we want to extend our work to cover the case of tuple uncertain model explained in this paper.

References

[1] P. Fournier-Viger, U. Faghihi, R. Nkambou, and E. M. Nguifo, ”CMRules: Mining sequential rules common to several sequences”, Knowledge-Based Systems, vol. 25, no. 1, pp. 63–76, 2012, Elsevier.

[2] P. Sen, A. Deshpande, and L. Getoor, “Representing tuple and attribute uncertainty in probabilistic databases,” Seventh IEEE International Conference on Data Mining Workshops (ICDMW 2007), pp. 507–512, 2007, IEEE.
[3] L. Sun, R. Cheng, D. W. Cheung, and J. Cheng, “Mining uncertain data with probabilistic guarantees,” Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 273–282, 2010.
[4] T. Bernecker, H-P. Kriegel, M. Renz, F. Verhein, and A. Zfle, “Probabilistic frequent pattern growth for itemset mining in uncertain databases,” International Conference on Scientific and Statistical Database Management, pp. 38–55, 2012, Springer.
[5] R. Agrawal, T. Imieliski, and A. Swami, “Mining association rules between sets of items in large databases,” Proceedings of the 1993 ACM SIGMOD international conference on Management of data, pp. 207–216, 1993.
[6] Q. Zhao and S. S. Bhowmick, “Association rule mining: A survey,” Nanyang Technological University, Singapore, pp. 135, 2003.
[7] Z. Zhao, D. Yan, and W. Ng, “Mining probabilistically frequent sequential patterns in large uncertain databases,” IEEE transactions on knowledge and data engineering, vol. 26, no. 5, pp. 1171–1184, 2013, IEEE.
[8] P. Fournier-Viger, R. Nkambou, and V. S. Tseng, “RuleGrowth: mining sequential rules common to several sequences by pattern-growth,” Proceedings of the 2011 ACM symposium on applied computing, pp. 956–961, 2011.
[9] C. K.-S. Leung, “Mining uncertain data,” Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, vol. 1, no. 4, pp. 316–329, 2011, Wiley Online Library.
[10] G. Modi, S. Bansal, and A. Patidar, “A survey on sequential rule mining techniques,” International Journal For Technological Research In Engineering, vol. 6, no. 3, 2018.
[11] W. Tong, C. K. Leung, D. Liu, and J. Yu, “Probabilistic frequent pattern mining by PUH-Mine,” Asia-Pacific Web Conference, pp. 768–780, 2015, Springer.
[12] C.-K. Chui, B. Kao, and E. Hung, “Mining frequent itemsets from uncertain data,” Pacific-Asia Conference on knowledge discovery and data mining, pp. 47–58, 2007, Springer.
[13] J. Li, B. Saha, and A. Deshpande, “A unified approach to ranking in probabilistic databases,” arXiv preprint arXiv:0904.1366, 2009.
[14] P. Fournier-Viger, T. Gueniche, S. Zida, and V. S. Tseng, “ERMiner: sequential rule mining using equivalence classes,” International Symposium on Intelligent Data Analysis, pp. 108–119, 2014, Springer.
[15] V. Subrahmanian, C. Pulice, J. F. Brown, and J. Bonen-Clark, “Temporal Probabilistic Rules and Policy Computation Algorithms,” A Machine Learning Based Model of Boko Haram, pp. 43–52, 2021, Springer.
[16] M. Shaheen and U. Abdullah, “CARM: Context Based Association Rule Mining for Conventional Data,” CMC-COMPUTERS MATERIALS & CONTINUA, vol. 68, no. 3, pp. 3305–3322, 2021.
[17] M. A. Islam, M. R. Rafi, A. Al-amin, and J. A. Ovi, “Weighted frequent sequential pattern mining,” Applied Intelligence, pp. 1–28, 2021, Springer.
[18] L. Zhang, G. Yang, and X. Li, ”Mining sequential patterns of PM2. 5 pollution between 338 cities in China”, Journal of environmental management, vol. 262, pp. 110341, 2020, Elsevier.

ISSN:2153-182X, E-ISSN: 2153-1838
Vol. 17 No. 02 (2023)

[19] T. Le, B. Vo, V.-N. Huynh, N. T. Nguyen, and S. W. Baik, ”Mining top-k frequent patterns from uncertain databases”, Applied Intelligence, vol. 50, no. 5, pp. 1487–1497, 2020, Springer.

[20] Mushroom Dataset, 1987, UCI Machine Learning Repository, http:// archive.ics.uci.edu/ml, 2013.

[21] T. Bernecker, R. Cheng, D. W. Cheung, H-P. Kriegel, S. D. Lee, M. Renz, F. Verhein, L. Wang, and A. Zfle, “Model-based probabilistic frequent itemset mining,” Knowledge and Information Systems, vol. 37, no. 1, pp. 181–217, 2013, Springer.

Article Title 17

[22] Y. Tong, L. Chen, Y. Cheng, and P. S. Yu, “Mining frequent itemsets over uncertain databases,” arXiv preprint arXiv:1208.0292, 2012.

[23] A. U. Ahmed, C. F. Ahmed, M. Samiullah, N. Adnan, and C. K.-S. Leung, “Mining interesting patterns from uncertain databases,” Information Sciences, vol. 354, pp. 60–85, 2016, Elsevier.

[24] Durga Bhavani, K., Ferni Ukrit, M. Design of inception with deep convolutional neural network based fall detection and classification model. Multimed Tools Appl (2023). https://doi.org/10.1007/s11042-023-16476-6

[25] K. Durga Bhavani, Dr. Radhika N. (2020). K-Means Clustering using Nature-Inspired Optimization Algorithms-A Comparative Survey. International Journal of Advanced Science and Technology, 29(6s), 2466-2472.

[26] K. D. Bhavani and M. F. Ukrit, "Human Fall Detection using Gaussian Mixture Model and Fall Motion Mixture Model," 2023 5th International Conference on Inventive Research in Computing Applications (ICIRCA), Coimbatore, India, 2023, pp. 1814-1818, doi: 10.1109/ICIRCA57980.2023.10220913