ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

# A Brief Survey on Frequent Patterns Mining of Uncertain Data

**Purvi Y. Rana*, Prof. Pragna Makwana, Prof. Kishori Shekokar**
*Student, Department of Computer Engineering, SIE, Vadodara, Gujarat, India
Asst. Professor, Department of Computer Engineering, SIE, Vadodara, Gujarat, India
HOD, Department of Computer Engineering, SIET, Vadodara, Gujarat, India

## Abstract

Frequent pattern mining is the extraction of interested collection of items from dataset. Frequent Itemset mining plays an important role in the mining of various patterns and is in demand in many real life applications. When handling uncertain data U-Apriori, UF-growth, UFP-growth and PUF-growth are examples of well-known mining algorithms, which use the UF-tree, the UFP-tree and the PUF-tree respectively. However, these trees can be large, and thus degrade the mining performance. The researchers have proposed various algorithms like U-Apriori, UF-growth, UH-mine, PUF-growth etc. In this paper, we are presenting depth analysis of algorithms of mining frequent patterns from uncertain datasets and discuss some problems associated with these algorithms in transactional databases.

**Keywords**: frequent Itemsets, frequent patterns, uncertain data, existential probability.

## Introduction

Since the introduction, the problem of finding frequent patterns there have been numerous studies on mining and visualizing *frequent patterns* (i.e., *frequent itemsets*) from *precise* data such as databases of market basket transactions[1,2,4,7]. But There are situations in which users are uncertain about the presence or absence of some items or events[3,4,9]. In several applications, however, an item is not present or absent in a transaction, but rather the probability of it being in the transaction is given. This is the case for data collected from experimental measurements susceptible to noise. For example, in satellite picture data the presence of an object or feature can be expressed more faithfully by a probability score when it is obtained by subjective human interpretation or an image segmentation tool[10].

Such data is called uncertain data. Table I(right) presents a popular type of uncertain database. This example dataset consists of 4 transactions and 3 items. For every transaction, a score between 0 and 1 is given to reflect the probability that the item is present in the transaction. For example, the existence probability of 0.9 associated to item a in the first transaction represents that there is a 90% probability that a is present in transaction t1 and 10% probability that it is absent.

***Table I (left) actually represents an instantiation of the uncertain dataset depicted in Table I (right).***

| TID | a | b | c |
|---|---|---|---|
| t1 | 1 | 1 | 0 |
| t2 | 0 | 1 | 0 |
| t3 | 1 | 0 | 1 |
| t4 | 0 | 1 | 1 |

| TID | a | b | C |
|---|---|---|---|
| t1 | 0.9 | 0.8 | 0.2 |
| t2 | 0.7 | 0.3 | 0.8 |
| t3 | 0.4 | 0.5 | 0.6 |
| t4 | 0.9 | 0.8 | 0.4 |

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

## Uncertain frequent pattern mining algorithm

There are two approaches of uncertain frequent pattern mining algorithms:

***Table 2 Uncertain frequent pattern mining algorithms***

| Apriori based Algorithms | Tree based Algorithms |
|---|---|
| U-Apriori algorithm | UF-growth algorithm<br>UFP-growth algorithm<br>UH - mine algorithm<br>PUF-growth algorithm |

### *A. U-Apriori Algorithm :*

U-Apriori is the uncertain version of Apriori algorithm. To deal with the problem of frequent itemsets under the uncertain data the Apriori algorithm was modified to U-Apriori algorithm. R.Agrawal was the first who had proposed this U-Apriori algorithm[7].

**Step -1 :** Scan the transaction database to get the support S of each 1-itemset, compare S with Minimum Support (MS), and get a set of frequent 1-itemsets, $L_1$.

**Step -2 :** Use $L_{k-1}$ Join $L_{k-1}$ to generate a set of candidate k item sets. And use Apriori property to prune the non frequented k-item sets from this set.

**Step -3 :** Scan the transaction database to get the support S of each candidate k-item set in the final set, compare S with MS, and gets a set of frequent k-item sets, $L_k$.

**Step -4 :** For each frequent item set l, generate all nonempty subsets of l.

**Step -5 :** For every nonempty subset s of l, output the rule " s => (l-s)" if confidence C of the rule,
" s => (l-s)"

However, in situations with prolific frequent patterns, long patterns, or quite low minimum support thresholds, an U-Apriori algorithm may suffer from the following two nontrivial costs: It is costly to handle a huge number of candidate sets. For example, if there are 104 frequent 1-itemsets, the Apriori algorithm will need to generate more than 107 length-2 candidates and accumulate and test their occurrence frequencies. And it is tedious to repeatedly scan the database and check a large set of candidates by pattern matching, which is especially true for mining long patterns[7].

### *B. UF-growth Algorithm :*

To effectively represent uncertain data, we propose a **UF-tree** which is a variant of the FP-tree. Each node in our UF-tree stores (i) an item, (ii) its expected support, and (iii) the number of occurrence of such expected support for such an item. Our proposed UF-growth algorithm constructs the UF-tree as follows. It scans the database once and accumulates the expected support of each item. Hence, it finds all frequent items (i.e., items having expected support ≥ *minsup*). It sorts these frequent items in descending order of accumulated expected support. The algorithm then scans the database the second time and inserts each transaction into the UF-tree in a similar fashion as in the construction of an FP-tree except for the following:

The new transaction is merged with a child node of the root of the UF-tree only if the same item and the same expected support exist in both the transaction and the child nodes. With such a tree construction process, UF-tree possesses a nice property that the occurrence count of a node is at least the sum of occurrence counts of all its children nodes[11].

### *C. UFP- growth Algorithm :*

UFP-growth extends the initial FP-growth algorithm. The FP-tree construction needs two scans of the dataset. The first scan collects the frequent items and their support and in the second scan every transaction is accommodated in the FP-tree structure. The frequent itemsets are then generated recursively from the FP-tree. In order to adapt this algorithm to our method, the first scan computes the expected support of every itemset exactly by computing their support as the sum of existential probabilities in every transaction where it occurs. In the second scan, every transaction is instantiated n times, according to the existential probability of the items in the transaction and then it is inserted in FP-tree structure. The algorithm then extracts the frequent itemsets the same way as the FP-growth algorithm.

### *D. UH-Mine Algorithm :*

The UH-Mine algorithm works as follows: It

**Step -1 :** prunes the initial database such that all singleton infrequent items are removed;

**Step -2 :** divides the pruned database into equal chunks;

**Step -3 :** mines these chunks separately using the UH-Mine algorithm. To mine frequent patterns, UH-Mine maintains an H-Struct, which contains pointers to transaction items. At each step, the algorithm adjusts these pointers and does not incur the

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

overheads, associated with the FP(UF)-Tree construction;
**Step -4 :** joins the results;
**Step -5 :** scans the pruned database once again to remove false positives and obtain the actual counts.

### *E. PUF-growth Algorithm :*

To reduce the size of the UF-tree and UFP-tree, Leung has proposed the **prefix-capped uncertain frequent pattern tree (PUF-tree)** structure, in which important information about uncertain data is captured so that frequent patterns can be mined from the tree. The PUF-tree is constructed by considering an upper bound of existential probability value for each item when generating a k-itemset (where **k>1**). The upper bound of an item $x_r$ in a transaction $t_j$ is the **(prefixed) item cap** of $x_r$ in $t_j$, as defined below,

***Definition.*** The **(prefixed) item cap** $I^{cap}(x_r,t_j)$ of an item $x_r$ in a transaction $t_j = x_1,\ldots..,x_r,\ldots.,x_h\}$, where $1<r<h$, is defined as the product of $P(x_r,t_j)$ and the highest existential probability value M of items from $x_1$ to $x_{r-1}$ in $t_j$(i.e., in the *proper prefix* of $x_r$ in $t_j$ ):

$$I^{cap}(x_r,t_j) = \begin{cases} P(x_r,t_j) \times M \; if \; h > 1 \\ P(x_1,t_j) \qquad if \; h = 1 \end{cases}, \text{ where M} = max_{1\le q\le r-1} P(x_q,t_j) \quad [2]$$

The cap of expected support $expSup^{cap}(X)$ of a pattern $X=\{x_1,\ldots\ldots,x_k\}$(where k>1) is defined as the sum of all item caps of $x_k$ in all the transactions that contain X:
$expSup^{cap}(X) = \sum_{j=1}^{n}\{(I^{cap}(x_k,t_j)|X \subset t_j)\}$ [1].
In other words, the cap of expected support of a pattern satisfies the downword closure property.

We take an example,

***Table 3 A transactional database with minsup=0.5***

| TID | Transactions | Sorted Transactions (with infrequent items removed) |
|---|---|---|
| t1 | $\{a:0.2, b:0.2, c:0.7,$ | $\{a:0.2, c:0.7, f:0.8\}$ |
| t2 | $\{a:0.5, c:0.9, e:0.5\}$ | $\{a:0.5, c:0.9, e:0.5\}$ |
| t3 | $\{a:0.3, d:0.5, e:0.4,$ | $\{a:0.3, e:0.4, f:0.5,$ |
| t4 | $\{a:0.9, b:0.2, d:0.1,$ | $\{a:0.9, e:0.5, d:0.1\}$ |

Consider an uncertain database with four transactions as presented in the second column in Table 2[1]. The item cap of *c* in *t*1 can be computed as,
$I^{cap}(c,t_1) = 0.7 \times \max\{P(a,t_1),P(b,t_1)\} = 0.7 \times \max\{0.2,0.2\} = 0.7 \times 0.2 = 0.14$

Similarly, the item cap of *f* in $t_1$ is,
$I^{cap}(f,t_1) = 0.8 \times \max\{P(a,t_1),P(b,t_1),P(c,t_1)\} = 0.8 \times \max\{0.2,0.2,0.7\} = 0.8 \times 0.7 = 0.56$

**How to construct a PUF-tree?**

- With the first scan of the database, find distinct frequent items in database
- construct a header table called *I-list* to store only frequent items in some consistent order (e.g., canonical order) to facilitate tree construction.
- The actual PUF-tree is constructed with the second database scan in a fashion similar to that of the FP-tree [7].
- when inserting a transaction item,First compute its item cap and then insert it into the PUF-tree according to the *I-list* order.
- If that node already exists in the path, Update its item cap by adding the computed item cap to the existing item cap.

Otherwise, Create a new node with this item cap value.

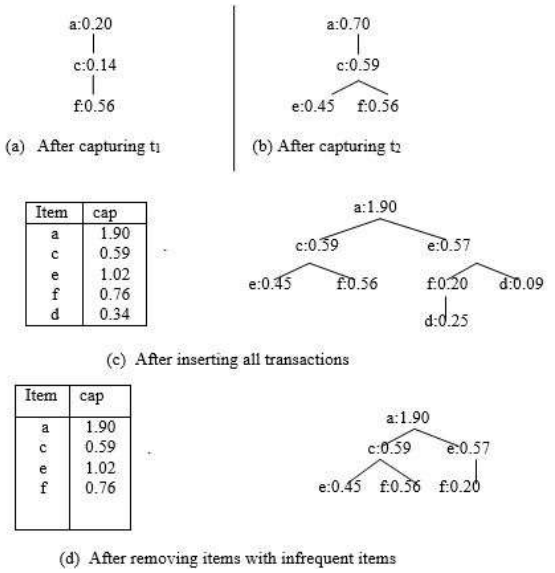


***Fig. 1. PUF – tree for the database in table 2 when minsup = 0.5[1***

***Number of False Positives:***

Both UFP-tree and PUF-trees are compact, their corresponding algorithms generate some false positives. Hence, their overall performances depend on the number of false positives generated. PUF-growth reduces the number of false positives when compared with UFP-growth. The primary reason of this improvement is that upper bounds of expected support of patterns in clusters are *not* as tight as the upper bounds provided by PUF-growth. In a UFP-

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

tree, if a parent has several children, then each child will use higher cluster values in the parent to generate the total expected support. If the total number of existential probability values of that child is still lower than that of the parent's highest cluster value, then the expected support of the path with this parent and child will be high. This results in more false positives in long run.

| Dataset | minsup | UFP-growth[8] | PUF-growth[1] |
|---|---|---|---|
| u10k5L_80_90 | 0.1 | 61.20% | 22.55% |
| u100k10L_50_60 | 0.07 | 89.32% | 25.99% |

***Table 4 Comparison of false positives (in terms of total # patterns)[1]***

## Comparison of algorithms

***Table 5 Comparison of uncertain frequent pattern mining algorithms***

| Sr. No | Algorithms | Advantage | Disadvantage |
|---|---|---|---|
| 1 | U-Apriori algorithm | This algorithm greatly reduces the size of candidate set. | It scans database many times and thus performance is affected. |
| 2 | UF-growth algorithm | This algorithm uses UF-trees to mine frequent patterns from uncertain databases in two database scans. | It contains a distinct tree path for each distinct item, existential probability pair. |
| 3 | UFP-growth algorithm | This algorithm scans the database twice, As nodes for item having similar existential probability values are clustered into a mega-node, the resulting mega-node in the UFP-tree. | It contains a distinct tree path for each distinct item, existential probability pair. |
| 4 | UH-mine algorithm | This algorithm Stores all frequent items in each database transaction in a hyper-structure called UH-struct. | It Suffer from the high computation cost of calculating the expected support. |
| 5 | PUF-growth algorithm | This algorithm Mines frequent patterns with constructing a projected database for each potential frequent pattern and recursively mine its potential frequent extensions. | It creates some false positives. |

## Conclusion

In this paper, the depth study of the frequent pattern mining algorithms of uncertain data is done and identified many strength and weakness of each. New variants of existing algorithms are compared with classical mining algorithms and results in significant benefits and limitations. This comparison may also fall into various optimization issues that will lead to better performance. Efficiency of the mining algorithms is no longer hindrance but yet there is a need to develop methods to get excellent results.

## References

1. Leung, C.K.-S.and S.K.Tanbeer. "PUF-Tree: A Compact tree structure for frequent pattern mining of uncertain data." In: J. Pei, T., Suzuki, E., Ting, K.M., Inokuchi, A. (eds.) PAKDD 2013. LNCS (LNAI), vol. 7818, pp. 13–25. Springer, Heidelberg (2013).
2. Lakshmanan, L.V.S., Leung, C.K.-S., Ng, R.T.: Efficient dynamic mining of constrained frequent sets. ACM TODS 28(4), 337–389 (2003).
3. Han, J., Pei, J., Yin, Y.: Mining frequent patterns without candidate generation. In: ACM SIGMOD 2000, pp. 1–12(2000).
4. Leung, C.K.-S., Irani, P.P., Carmichael, C.L.: FIsViz: a frequent itemset visualizer. In: Washio, T., Suzuki, E., Ting, K.M., Inokuchi, A. (eds.) PAKDD 2008. LNCS (LNAI), vol. 5012, pp. 644–652. Springer, Heidelberg (2008).
5. Leung, C.K.-S., Mateo, M.A.F., Brajczuk, D.A.: A tree-based approach for frequent pattern mining from uncertain data. In: Washio, T., Suzuki, E., Ting, K.M., Inokuchi, A. (eds.) PAKDD 2008. LNCS (LNAI), vol. 5012, pp. 653–661. Springer, Heidelberg (2008).
6. Leung, C.K.-S., Jiang, F.: RadialViz: an orientation-free frequent pattern visualizer. In:

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

7. Tan, P.-N., Chawla, S., Ho, C.K., Bailey, J. (eds.) PAKDD 2012, Part II. LNCS (LNAI), vol. 7302, pp. 322–334. Springer, Heidelberg (2012).
8. Han, J., Pei, J., Yin, Y.: Mining frequent patterns without candidate generation.In: ACM SIGMOD 2000, pp. 1–12 (2000).
9. Aggarwal, C.C., Li, Y.,Wang, J.,Wang, J.: Frequent pattern mining with uncertain data. In: ACM KDD 2009, pp. 29–37 (2009).
10. Calders, T., Garboni, C., Goethals, B.: Approximation of frequentness probability of itemsets in uncertain data. In: IEEE ICDM 2010, pp. 749–754 (2010).
11. Leung, C.K.-S., Hao, B.: Mining of frequent itemsets from streams of uncertain data. In: IEEE ICDE 2009, pp. 1663–1670 (2009)