ISSN: 2277-9655
Impact Factor: 1.852

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

Frequent Pattern Mining Algorithm for Crime Dataset: An Analysis

D.Usha[*1], K.Rameshkumar[2]
[*1]Assistant Professor, [2]Associate Professor, Hindustan University, Chennai, India
dusha@hindustanuniv.ac.in

Abstract

The mining of frequent item set from uncertain data is the great task and it consumes more time. One of the frequent pattern mining algorithm called Apriori algorithm can be used to mine the data from bulk of uncertain dataset. But it can give only minimum support constraint in mining the large amount of uncertain dataset. The experimental behavior of different types of algorithms is very different in the uncertain case as compared to the deterministic case. In particular when compare to Apriori algorithm and other Apriori based algorithm, each and every algorithm has their own advantages when compare to other algorithm. Some algorithm shows robustness with respect of both efficiency and memory usuage. We will test the approach on a number of real and synthetic datasets and show the effectiveness of the proposed algorithm.

Keywords : Frequent pattern mining, Apriori algorithm, Uncertain dataset

Introduction

Data mining of uncertain data has become an active area of research recently. The data collected may be certain or uncertain. But it is difficult to mine the uncertain data which not in an proper form. In this paper, we will study the problem of frequent pattern mining with uncertain data and various Apriori based algorithm which is suitable to mine the uncertain data, which also may have some drawbacks when compare to other algorithm. The problem of frequent pattern mining with uncertain data has been studied up to some extent[1] and variety of algorithm are examining the relative behavior of various algorithm and extensions of well known classes of deterministic algorithm.

One observation from our extensions to the uncertain case is that the respective algorithms do not show similar trends to the deterministic case. For example, in the deterministic case, the FP-growth algorithm is well known to be an extremely efficient approach. However, in our tests, we found that the extensions of the candidate generate and test as well as the hyper structure based algorithms are much more effective. Furthermore, many pruning methods, which work well for the case of low uncertainty probabilities and do not work very well for the case of high uncertainty probabilities. This is because the extensions of some of the algorithms to the uncertain case are significantly more complex, and require different kinds of trade-offs in the underlying computations.

The next section defines the significant of association analysis. We will also discuss the various frequent pattern mining algorithms to the uncertain version. The remainder of the paper analyzed the improvement given by the various frequent mining algorithm when compare to one another.

Association Analysis

Association analysis is one of the most significant data mining techniques. Market-basket analysis is one of the fine example for Association analysis where dataset consists of number of tuples and attributes, each contains the items that a customer has purchased in a transaction. To discover associations among different items, the given dataset is analyzed. An important step in the mining process is the extraction of frequent item set, or set of items that co-occur in a major fraction of the transactions. Besides market-basket analysis, frequent item set mining is also a core component in other variations of association analysis, such as association-rule mining[6] and sequential-pattern mining[7]. As an example, a crime dataset contains the age of victims, weapons used and place of committing crime and various other attributes. Applying association analysis[9] on such dataset discover correlations and shows the probability among the commitment of crimes and the victims

ISSN: 2277-9655
Impact Factor: 1.852

Age of victims	Weapons Used	Time of committing crime	Place of committing crime	Probability of commitment of crime
Young	Knife	Morning	Train	80%
Middle	Riffle	Afternoon/Night	Public places	70%
Old	Sharp weapons	Afternoon	House	90%

Table 2.1 Crime Dataset

Frequent Pattern

There are various existing algorithm to mine frequent pattern from precise transaction database[16]. Each transaction may contain collection of items which is stored in rows and columns (structured format). Each of these items usually takes only one of the binary states. The item is either present in, or absent from the transaction. Numerous algorithm have been proposed to mine frequent patterns from precise data and use the mined patterns to form interesting association rules.

As we are living in an uncertain world, the data may not be precise always. In some cases, the data may be uncertain. This leads to the mining of frequent patterns from uncertain dataset, in which users are not sure about the presence of domain items in transactions of the dataset. One way to express the uncertainty is to associate each transaction item with an existential probability value which indicates its likelihood of being present in that transaction. In recent years, researchers have proposed algorithms to mine frequent patterns from uncertain data. Recently, Leung reviewed the most recent developments in mining frequent patterns from uncertain data. In the remainder of this paper, we will give a high-level overview of some notable algorithms designed for mining frequent patterns from precise data as well as for uncertain data.

Apriori Algorithm

For learning association rules the classic algorithm Apriori is a vital tool. The object of Apriori algorithm is to identify association between different sets of data, and to find out patterns in data. It is sometimes referred to as Market Basket Analysis[14]. The Apriori algorithm is an old algorithm for finding patterns in data[10]. It is based on a really simple observation. For example, if very few people go to Pizza hut and McDonald's on the same day, then there can't be a lot of people going to Starbucks, McDonald's, and Domino's on the same day. So, to find the combinations of three stores that lots of people go to on the same day, you don't have to look at combinations that include two stores that very few people go to on the same day. This tremendously reduces the number of combinations you need to look at. As for where it is used best, in proves the- concept of toy applications. It's not particularly efficient and in real-world applications, more efficient algorithms such an Eclat are used. We knew that most people who bought diapers also bought baby powder and infant formula, but at the grocery store, very few people who bought both diapers and baby powder also bought infant formula. So, we know better which customers to advertise it to. If someone buys A, B, but not C, and A, B, and C associate, an ad or coupon for C has the best chance of working.

"Bottom up" approach is used in Apriori, where frequent subsets are extended one item at a time and groups of candidates are tested against the data which is known as candidate generation. When no further successful extension are identified, the algorithm itself terminates. Each set of data has a number of items and is called a transaction. The output of Apriori is set of rules that tell us how often items are contained in sets of data. The following is the example: each line is a set of items

alpha beta gamma

alpha beta theta

alpha beta theta

1. 100% of sets with alpha also contain beta
2. 25% of sets with alpha, beta also have gamma
3. 50% of sets with alpha, beta also have theta

Apriori algorithm relies on generate and test approach and an important property the Apriori property. This property is also known as anti-monotone property, and it is a basic pillar of the Apriori algorithm. It states that all non-empty subsets of a frequent itemset must be frequent. For example, if item set 1, 2, 3 is a frequent item set[17], then all of its subsets 1, 2, 3, 1, 2, 2, 3 and 1, 3 must be frequent. In other words, if an item set is not frequent, then none of its supersets can be frequent. As a result, the list of potential frequent item set eventually gets smaller as mining progresses.

In order to find frequent patterns[15], Apriori makes first pass over the database to find the frequent 1-itemsets. Once this pass is completed, the algorithm generates candidate 2-itemsets based on these frequent 1-itemsets. Then it scans the database to find frequent 2-itemsets. In the next step, the Apriori algorithm generates candidate 3-itemsets by using frequent 2-itemsets.The algorithm then scans the database to find frequent 3-itemsets from these candidates. This process is repeated until no larger frequent item set are found. The below mentioned diagram illustrates how Apriori finds frequent patterns from a sample database.

ISSN: 2277-9655
Impact Factor: 1.852

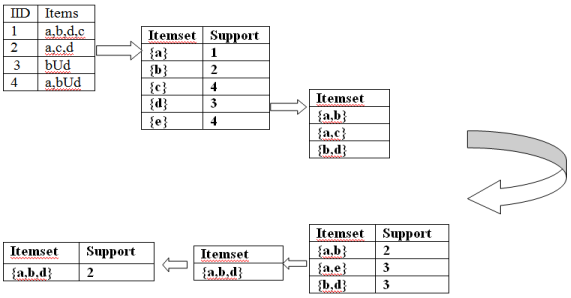

Fig. 3.1 Finding frequent patterns from sample database

Apriori results good performance when dealing with very sparse databases. Even it is one of the old method data mining algorithm, it has a benefit of achieving 100% accurate results. Unfortunately, when databases get denser it degrades much faster, because the algorithm scans database as many times as the longest frequent pattern. Apriori generates candidate item sets of length k from item sets of length k-1, as it uses breadth first search and a Hash tree structure to count candidate item sets potentially. After that it prunes the candidates which have an infrequent sub pattern. The candidate set contains all frequent k-length item sets as per the downward closure lemma. And then, it scans the transaction database to determine frequent item sets among the candidates.

For frequent item mining the Apriori employs level wise search, i.e. breath first search, where it uses frequent k item set to discover the (k+1) item set. To find out the support count of each item, a scan of database is performed while preprocessing the Apriori. In the final stage all those items whose support count is less the minimum support threshold, that is all infrequent 1 item set are removed from the database. The aim of Apriori is to find out frequent item set[18] from a transaction dataset and derive association rules. Finding frequent item set is not trifling because of its combinatorial explosion. Once it is obtained, it can generate association rules with confidence larger than or equal to a user specified minimum confidence. Apriori is a influential algorithm[13] for finding frequent item set using candidate generation [18]. It is characterized as a level-wise complete search algorithm using anti-monotonocity of item set, "if an item set is not frequent, any of its superset is never frequent". Let us set the frequent item set of size k be F_k and their candidates be C_k. Apriori scans the database and searches for frequent item set of size 1 by accumulating the count for each item and collecting those that satisfy the minimum support requirement. The following three steps iterate it and extracts all the frequent item set.

1. Generate C_{k+1}, candidates of frequent item set of size k +1, from the frequent item set of size k.
2. Scan the database and calculate the support of each candidate of frequent item set.
3. Add those item set that satisfies the minimum support requirement to F_{k+1}.

The function apriori generates C_{k+1} from F_k in the following two step process:

1. Join step:

Generate R_{K+1}, the initial candidates of frequent item set of size k + 1 by taking the union of the two frequent item set of size k, P_k and Q_k that have the first k-1 elements in common.

$R_{K+1} = P_k \cup Q_k = \{iteml, . ., itemk, itemk-1 , itemk-2\}$

$P_k = \{iteml , item2, . . . , itemk-1, itemk \}$

$Q_k = \{iteml , item2, . . . , itemk-1, itemk-2\}$, where, iteml < item2 < · · · < itemk < itemk-1.

2. Prune step:

Check if all the itemsets of size k in R_{k+1} are frequent and generate C_{k+1} by removing those that do not pass this requirement from R_{k+1}. This is because any subset of size k of C_{k+1} that is not frequent cannot be a subset of a frequent itemset of size k + 1. Function subset in line 5 finds all the candidates of the frequent itemsets included in transaction t. Apriori, then, calculates frequency only for those candidates generated this way by scanning the database. It is evident that Apriori scans the database at most kmax+1 times when the maximum size of frequent itemsets is set at kmax. The Apriori achieves good performance by reducing the size of candidate sets. However, in situations with very many frequent itemsets, large itemsets, or very low minimum support, it still suffers from the cost of generating a huge number of candidate sets and scanning the database repeatedly to check a large set of candidate itemsets. In fact, it is necessary to generate 2100 candidate itemsets to obtain frequent itemsets of size 100.

FP Growth Algorithm

Han et al proposed a pattern growth approach to avoid the problem of numerous database scans and candidate generate –and-test process. The corresponding algorithm is called FP Growth Algorithm. To obtain the information about the database, it requires two scans only. Frequent patterns are mined from the tree structure, since contents of the database are captured in a tree structure. Specifically, FP-growth starts by scanning the database once to find all frequent 1-itemsets. Afterwards, the algorithm makes a ranking table, in which items appear in descending frequency order.

ISSN: 2277-9655
Impact Factor: 1.852

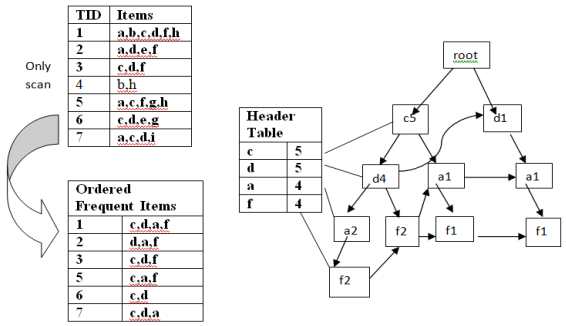

Fig 3.2 Ranking Table

All infrequent items are then discarded. In the second pass, the algorithm orders the items in each transaction according to the ranking created in the first pass. At the same time, infrequent items in the transaction are ignored. Frequent items are inserted in a tree structure called FP-tree by following the rank order. Because of all the transactions follow the same order and share the same prefix, they can be merged. FP growth algorithm constructs a conditional FP-tree for each frequent item so that all frequent patterns can be found by just traversing the structure. It can also be applied to small database. The above mentioned algorithm usually our perform Apriori based variations in runtime. The worst case scenario for FP-tree occurs when mining large but very sparse database. Here, the tree becomes very big. Array based structure can be used to reduce the number of traversals of FP-tree so that it improves the above mentioned case.

Uncertain Data

Data is known fact or information. Database is a collection of related information. Data may also uncertain[2] because of measurement inaccuracy, sampling discrepancy, outdated data sources or other errors. For example, in the scenario of moving objects, it is impossible for the database to tract the exact locations of all objects at all time instants. Hence, the location of each object is associated with uncertainty between updates. The different sources of uncertainty have to be considered in order to produce accurate query and mining results[5]. Uncertain data[3] may be in Structured format or unstructured format.

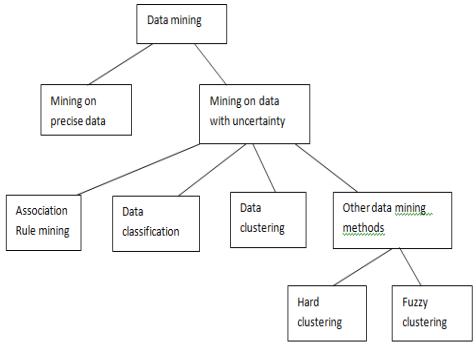

Fig 4.1 Taxonomy of Uncertain Data mining

Structured Data

Structured data refers to data that is certain because it is organized in a structure. The general form of structured data is a database where specific information is stored based on a methodology of columns and rows, so called a table structure.

The term structured data also refers to data that has a defined length and format for massive volume of data. Numbers, dates and groups of words and numbers called strings are the examples of structured data. It is usually stored in a database. Normally structured data refers to data kept in a "database" form rather than "free form". In view of technical sense, structured data is built using information that is stored in fixed fields within a record or file. These fields can be referenced by all others since they are in an organized format. Structured data is also searchable by data type within content. Structured data is understood by computers and is also efficiently organized for human readers. In contrast, unstructured data has no identifiable structure. Examples of structured data would be relational databases and spreadsheets.

Unstructured Data

The term unstructured data refers to any data that has no identifiable structure. For example, images, videos, email, documents and text are all considered to be unstructured data within a dataset. While each individual document may contain its own specific structure or formatting that based on the software program used to create the data, unstructured data may also be considered "loosely structured data" because the data sources do have a structure but all data within a dataset will not contain the same structure. In contrast, unstructured data is information that is brought together in a non-structured format, like a PDF document, or the text in a chart note. It is considered "free form" and does not follow any sort of organizational pattern. It is not possible to read and interpret information that is free form, since it does not built in an organized way.

ISSN: 2277-9655
Impact Factor: 1.852

Frequent Pattern Mining From Uncertain Data

Direct Hash and Pruning Algorithm (DHP)

One of the potential problem of Apriori algorithm is the huge number of candidate k-itemsets generated and tested by the algorithm. To deal with the potential problem faced by the Apriori algorithm, Direct Hash and Pruning (DHP) algorithm was developed. The DHP algorithm uses a hash table to prune away infrequent candidate *k*-item set. At the beginning of each level *k*, the DHP algorithm hashes each item set to a bucket by using a hash function. Once all item set have been hashed, the counter at each bucket is checked. If the count is smaller than the minsup value, all candidates in that bucket are discarded since they cannot be frequent. As a result of having fewer candidates to check for, the hashing technique speeds up the mining process and reduces the number of candidates to be tested. The performance of DHP depends on the size of the hash table and of the number of infrequent item set being hashed into the same bucket. For example, if several distinct infrequent item set are being hashed into the same bucket, the count of the bucket may exceed minsup. Consequently, DHP cannot prune away these (infrequent) item set, which can be considered as false positives in the intermediate levels.

Perfect Hashing and Pruning Algorithm (PHP)

A variation of DHP is the Perfect Hashing and Pruning (PHP) algorithm (Ozel & Guvenir, 2001), which uses perfect hashing to avoid false positives in the intermediate levels of the mining process. As a result, each bucket shows the actual support of every itemset, and thus saves some computation.

Matrix Apriori Algorithm

Matrix Apriori algorithm was proposed by Pavon et al. to speed up the mining process. It reduces the number of candidate item set by utilizing matrix and vector structures.

Partition Algorithm

Many Apriori-based algorithms (including DHP and PHP) require numerous database scans, which incur high I/O costs, and thus slow down the mining process. The Partition algorithm is another technique proposed to improve Apriori-based algorithms by dividing the database in a number of non-overlapping segments. After the first database scan, item set that are frequent locally in each segment can be found. For an item set to be globally frequent in the database, it must be locally frequent item set in at least one partition (or segment). So, after gathering all local frequent item set, the Partition algorithm scans the database for the second and last time to check which of those local frequent item set are actually frequent globally in the whole database. As a result, this technique reduces drastically the number of scans needed by Apriori-based algorithms to only two. So, Partition algorithm always depends on the data distribution and the number of segments.

U-Apriori Algorithm

A classic Apriori algorithm for uncertain data[12] is called U-Apriori. The process is almost the same as in the original algorithm, but now the expected support of a given pattern is incremented by the product of all existential probabilities of the items in the pattern. Expecting the performance of U-Apriori to be even worse than that of the original Apriori because of the effect of multiplying small numbers several times. Chui et al. proposed a trimming strategy to reduce the database by removing items with low probability.

Decremental Pruning (DP) Algorithm

Decremental Pruning (DP) technique[11] was developed in order to further improve the performance of U-Apriori. DP scans the database once to estimate bounds for each 1-itemset and stores this value in a decremental counter for all patterns that contain this item. As the database is scanned, this counter is updated by subtracting the corresponding "over-estimate" for each item in the pattern. If the counter gets below the minimum support, any pattern containing that item cannot be frequent and hence can be pruned. DP—with its two improvements—is a very effective technique and it improves both runtime and memory requirements of U-Apriori. Even though it is still bounded by the generate-and-test approach limitations, the application of the decremental technique (known as UCP-Apriori algorithm) is a reasonable Apriori-based adaptation for uncertain data.

H-Mine Algorithm

H-Mine algorithm[18] was developed by Pei et al, that uses dynamic linked list to maintain a hyperlink array structure called H-struct. By using this structure, the algorithm tries to improve the mining time. Once the H-struct is constructed, the H-Mine algorithm[4] just needs to maintain and update the numerous links that point from one transaction to the next that contains the same set of items. Since H-Mine keeps all transactions that contain frequent items in memory, there is no need to read the database more than once. From that point on, all information is extracted from the H-struct. H-Mine outperformed Apriori by finding frequent patterns quicker and requiring less memory than FP-growth, especially with small minimum support threshold.

Comparison

The analysis shows how uncertain data provides different scenario and most algorithms give very different performances than their counter parts with precise data, U-Apriori inherits the problems of generating-and-testing large number of candidates. UCP-Apriori detects infrequent candidates, support the

minimum value and improves the performance better than U-Apriori. Even though UF-growth may suffer from the problem of having very big trees as a result of many different probability values for the same items, its improvements truncate probability values and thus merge more nodes. At the end, they decrease the chance of having very big trees and the algorithm needs small memory space. However, they require longer runtime than U-Eclat. U-Eclat is the algorithm that requires less memory to mine frequent patterns from uncertain data when taking few database samples. However, the more samples were taken by U-Eclat, it would take long time to finish.=

The UF-Growth algorithm modifies the FP-Growth algorithm by the way of building the transaction tree. FP-Growth uses the FP-Tree, a tree-based data structure, to store a compact representation of the transaction database that contains information about all frequent items. To overcome the drawback of FP-Tree which does not store existential probabilities, associated with items, UF-Tree is proposed. Each node stores an item, its expected support, as well as the number of occurrence for each item. To merge the transaction with the child node in UF-Tree, UF-Growth requires both the item and its corresponding existential probability to match. Hence UF-Tree algorithm have lower compression ratio then FP-Tree. The UH-Struct structure uses the linkage behaviour among transactions corresponding to a branch of the FP-Tree(UF-Tree) without actually creating a projected database. This approach is better than FP-Tree even in the deterministic case, when compression from FP-Tree is not high. This turns out to be particularly true for the uncertain case, as discussed earlier. H-struct also stores the probability of each item besides the link and the item itself.

UFIMT (Uncertain Frequent Itemset Mining) contains three representative algorithms: UApriori [1], UFP-growth [1], and UH-Mine [1]. UApriori is the first expected support-based frequent item set mining algorithm which extends the well-known Apriori algorithm to the uncertain environment and employs the generate-and-test framework to find all expected support-based frequent item set. UFP-growth algorithm extends the well-known FP-growth algorithm. Similar to the traditional FPgrowth algorithm, UFP-growth algorithm also builds an index tree, called UFP-tree to store all information of the uncertain database. Then, based on the UFP-tree, the algorithm recursively builds conditional sub-trees and expected support-based frequent item set. UH-Mine is also based on the divide-and-conquer framework. The algorithm is extended from the H-Mine algorithm which is a classical algorithm in deterministic frequent itemset mining. Similar to H-Mine, UH-Mine[19] first builds the special data structure, UH-Struct, and then recursively discovers the expected support-based frequent itemsets based on the DFS strategy. Many of the pattern finding algorithms such as decision tree, classification rules and clustering techniques that are frequently used in data mining have been developed in machine learning research community.

Conclusion

In this paper, we analyzed the most well known algorithms to find frequent patterns from uncertain data. We also explained clearly about uncertain data that consist of both structured and unstructured data. The traditional Apriori algorithm is the referent algorithm for generating frequent pattern candidates and checking the database to keep those that are indeed frequent. We also compare various Apriori based algorithm and conclude that each of the algorithms that we have described in this paper possesses very different features, and the performance of each depends heavily on the characteristics of the dataset. We sure that researches and data miners can utilize this paper at their level best.

References

[1] Charu C. Aggarwal, et al. (2009), " Frequent pattern mining with uncertain data", International Conference on Knowledge Discovery and Data Mining, Paris, June 2010, pp.29-38.

[2] Ming Hua, Jian Pei (2008), "Mining uncertain and probabilistic data: problems, challenges,methods, and applications", International Conference on Knowledge Discovery and Data mining", Las Vegas, September 2008.

[3] Carson Kai-Sang Leung, Christopher Lee Carmichael, and Boyu Hao (2007), "Efficient mining of frequent patterns from uncertain data", International Conference on Data Mining Workshops", USA, October 2007, pp.489-494.

[4] Jian Pei et al. (2001), "H-Mine: Hyper-structure mining of frequent patterns in large databases", International Conference on Data Mining, California, USA, November 2001, pp.441-448.

[5] Calin Garboni, Toon Calders and Bart Goethals (2010), "Efficient pattern mining from uncertain data with sampling", Pacific-Asia Conference on Knowledge Discovery and Data Mining, Hyderabad, June 2010, pp.480-487.

[6] Agrawal, R. and Srikant, R. (1994), "Fast algorithms for mining association rules in large databases", International Conference on Very

ISSN: 2277-9655
Impact Factor: 1.852

Large Data Bases (VLDB), Santiago de Chile, Chile, pp.487-499.

[7] Agrawal, R. and Srikant, R. (1995), "Mining Sequential Patterns", International Conference on Data Engineering (ICDE), Taipei, Taiwan, pp.3-14.

[8] Agrawal, R., Imielinsky, T., and Swami, A. (1993), "Database mining: a performance perspective", IEEE Transactions on Knowledge and Data Engineering , pp.914–925.

[9] Cheng, H., and Han, J. (2009), "Frequent itemsets and association rules in Encyclopedia of Database Systems", Springer, pp.1184-1187.

[10] Cheng, J., Ke, Y., and Ng, W. (2007), "A survey on algorithms for mining frequent itemsets", Knowledge and Information Systems , pp.1–27.

[11] Chui, C.K., and Kao, B. (2008), "A decremental approach for mining frequent itemsets from uncertain data", 12th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), Osaka, Japan, pp.64-75.

[12] Chui, C.K., Kao, B., and Hung, E. (2007), "Mining frequent itemsets from uncertain data". 11th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), Nanjing, China, pp.47–58.

[13] O. Maimon, and L. Rokach, "Data Mining and Knowledge Discovery Handbook", 2nd ed., pp. 321-338. Springer.

[14] Han, J., & Kamber, M. (2011), "Data Mining, Concepts and Techniques", 3rd ed.

[15] Han, J., Cheng, H., Xin, D., & Yan, X. (2007), "Frequent pattern mining: current status and future directions", pp.55–86.

[16] Juan J. Cameron, Carson K. Leung, "Mining Frequent Patterns From Precise And Uncertain Data", Canada.

[17] Yongxin Tong, Lei Chen, Philip S. Yu, "UFIMT: An Uncertain Frequent Itemset Mining Toolbox", USA.

[18] Jian Pei et al. (2007), "H-Mine: Hyper-Structure Mining of Frequent Patterns in Large Databases", Hong Kong, pp. 39, 593–605.

[19] Jian Pei et al. (2007), "H-Mine: Fast and space-preserving frequent pattern mining in large databases", Hong Kong.