Lee et al. Hum. Cent. Comput. Inf. Sci. (2017) 7:31
DOI 10.1186/s13673-017-0112-7

Human-centric Computing and Information Sciences

RESEARCH Open Access

Analyzing of incremental high utility pattern mining based on tree structures

Judae Lee, Unil Yun* and Gangin Lee

*Correspondence: yunei@sejong.ac.kr
Dept. of Computer Engineering, Sejong University, Seoul, South Korea

Abstract

Since the concept of high utility pattern mining was proposed to solve the drawbacks of traditional frequent pattern mining approach that cannot handle various features of real-world applications, many different techniques and algorithms for high utility pattern mining have been developed. Moreover, several advanced methods for incremental data processing have been proposed in recent years as the sizes of recent databases obtained in the real world become larger. In this paper, we introduce the basic concept of incremental high utility pattern mining and analyze various relevant methods. In addition, we also conduct performance evaluation for the methods with famous benchmark datasets in order to determine their detailed characteristics. The evaluation shows that the less candidate patterns make algorithms faster.

Keywords: Data mining, Pattern mining, Utility mining, Incremental mining, High utility patterns, Tree-based algorithms

Introduction

Data mining is a series of tasks for finding interesting knowledge from various types of data, and its merits have attracted much research attention such as clustering [5], a phishing url detection system [7], and other interesting studies [3, 4, 11, 12]. Pattern mining is one of data mining techniques to discover useful information from huge databases, and it extracts such information in pattern forms. Although traditional pattern mining [1, 6] that discover patterns which are frequent has played an significant role in the data mining field, this approach not only treats all items in databases with the same importance but also represents item occurrence as a binary form, i.e., 0 or 1. In addition, it cannot reflect characteristics of real-world databases that can be incremented, deleted and modified to pattern mining completely. High utility pattern mining has been proposed and researched to overcome the limitations. Furthermore, database sizes become larger incrementally in various real-world applications and previous general static methods are not suitable for finding meaningful information efficiently from such databases. Incremental high utility pattern mining [2, 8, 10, 14] has been studied to mine essential information from dynamic databases that new records are inserted continuously by considering real characteristics of them. In this paper, we provide how tree-based pattern mining methods deal with incremental environments. In addition, we conduct analysis on tree-based methods that discover high utility patterns in incremental environments.

Through the various tests, performances of the methods were evaluated, and we analysis of what causes performance difference.

The remainder of this paper is organized as follows. In "Related work" section, we describe influential researches related to tree-based incremental high utility pattern mining. In "Tree-based incremental high utility pattern mining" section, we analyze characteristics of recent tree-based methods for incremental high utility pattern mining through experiments for performance evaluation with real datasets and study direction of improvements based on experimental results. Lastly, in "Performance analysis" section, we summarize contributions of this paper.

Related work

In this section, we describe two types of related studies: traditional frequent pattern mining and high utility pattern mining for static databases.

Frequent pattern mining

Apriori [1] and FP-Growth [6] are well-known BFS and DFS frequent pattern mining algorithms respectively. The former utilizes a candidate generation-and-test approach, which causes the performance degradation since this method makes a large number of candidates and requires multiple database scans. The latter conducts a series of mining processes through only two database scans without candidate generation by employing a divide-and-conquer manner. In general, FP-Growth-like algorithms outperform Apriori-based ones. In pattern mining, meanwhile, the anti-monotone property [1] is used for efficient mining. This suggests that if a pattern is not valid due to its smaller frequency than a given threshold, then its super patterns are not also valid. When an invalid pattern is found in mining processes, the pattern's search space is can be eliminated, which improves mining efficiency. Therefore, satisfying this property is an essential criterion in pattern mining. To discover frequent patterns more efficiently, various studies such as applying cellular learning automata [13] have been conducted.

High utility pattern mining

In utility mining, maintaining the anti-monotone property is not easy and Two-Phase [9] is the first algorithm that satisfies the property by applying the overestimation concept, called transaction weighted utilization (twu), to mining processes. This model generates candidates with no smaller overestimation values than a given threshold in the first phase and then identifies actual high utility patterns by computing their utility values through an additional database scan. Since the twu concept was suggested, although various static algorithms with the overestimation model have been developed, they are not appropriate for handling dynamic databases. Table 1 is an example of a utility database, and Table 2 is a set of the corresponding external utility values. Transaction utility (TU) of a transaction is a sum of utility values of all items in the transaction. There are recent studies such as the EFIM algorithm [15] to improve mining performance when discovering high utility patterns.

Table 1 Example database for high utility pattern mining

TID	Transaction	TU
001	(C, 2) (E, 3)	31
002	(A, 2) (B, 1) (D, 1)	14
003	(C, 1) (D, 2) (E, 1)	24
004	(A, 2) (B, 1) (F, 4)	12
005	(G, 2)	18

Table 2 External utilities of the example database

Item	A	B	C	D	E	F	G
External utility	3	2	5	6	7	1	9

Tree-based incremental high utility pattern mining

In contrast to the previous static approaches where the whole mining processes have to be conducted whenever a target database is increased, incremental high utility pattern mining methods simply reflect new information to the current data structures for dealing with the dynamic database.

Below are the fundamental definitions of high utility pattern mining.

Definition 1 *(Utility of an item in a transaction)* Given an item, i_p, belonging to a transaction, T_i, the utility of i_p is denoted as $u(i_p, T_i)$ and calculated by multiplying its internal utility, $iu(i_p, T_i)$, and its external utility, $eu(i_p)$. In the example of Tables 1 and 2, $u(\mathrm{C}, \mathrm{TID{:}001}) = 2 \times 5 = 10$.

Definition 2 *(Utility of a pattern in a transaction)* Given a pattern, $P = \{i_1, i_2, \ldots, i_k\}$, included in a transaction, T_i, the utility of P is denoted as $u(P, T_i)$ and calculated as follows: $\sum u(i_p, T_i)$, where $i_p \in P$ and $P \subseteq T_i$. For example, $u(\mathrm{CE}, \mathrm{TID{:}001}) = u(\mathrm{C}, \mathrm{TID{:}001}) + u(\mathrm{E}, \mathrm{TID{:}001}) = 10 + 21 = 31$.

Definition 3 *(Utility of a pattern)* For pattern $P = \{i_1, i_2, \ldots, i_k\}$, belonging to a given database, DB, its utility, denoted as $u(P)$, is calculated as follows: $\sum u(P, T_i)$, where $P \subseteq T_i$ and $T_i \in DB$. For example, $u(\mathrm{CE}) = u(\mathrm{CE}, \mathrm{TID{:}001}) + u(\mathrm{CE}, \mathrm{TID{:}003}) = 31 + 12 = 43$.

In incremental high utility pattern mining, algorithms build global data structures with the original databases, generate candidate patterns, and identify actual high utility patterns from the candidates. After that, when new transaction information is added to the original databases, the incremental approaches update the data structures, optimize them, and conduct mining processes. Figure 1 shows a simple overall process of incremental high utility pattern mining. In the figure, there are two types of data, the original and incremented ones. As explained above, a set of high utility patterns is mined from the original data and then updated pattern results are extracted by handling only the incremented data.

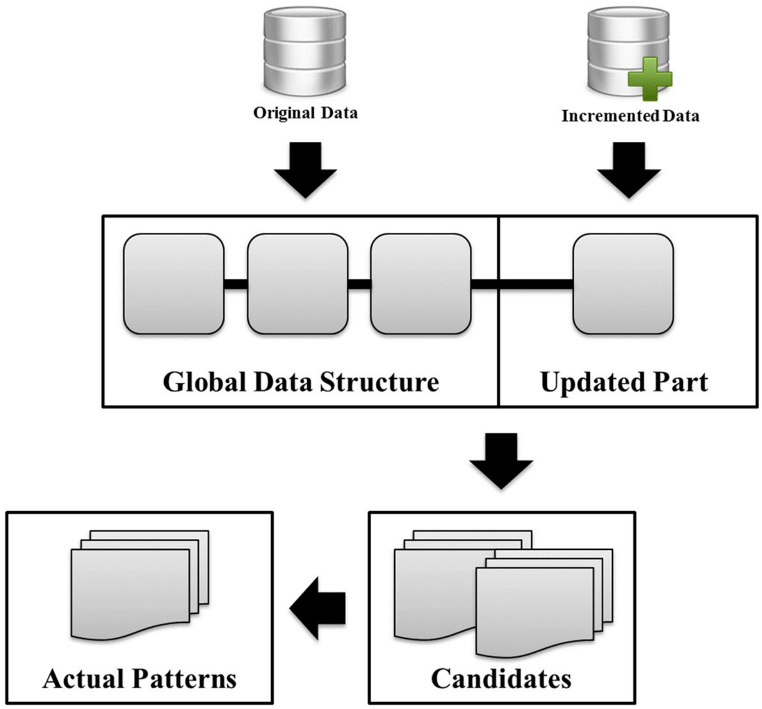

Fig. 1 Overall process of incremental high utility pattern mining

FUP-HU [8] is an Apriori-based incremental high utility pattern mining algorithm and can reflect characteristics of real-world databases to pattern mining. However, this method requires a lot of database scans and operations. To address this issue, FP-Growth-based algorithms with tree data structures, incremental high utility pattern (IHUP) [2] and high utility patterns in incremental databases (HUPID) [14], were proposed. IHUP applies the overestimation model of the Two-Phase algorithm [5] to its mining processes. For incremental high utility pattern mining, IHUP firstly constructs a tree data structure through a single database scan, extracts candidate patterns, and identifies actual high utility patterns from them with an additional scan. Although this algorithm mines valid patterns faster than the previous Apriori-based ones by utilizing the FP-Growth approach, it still generates a large number of candidates and consumes high computational time for the identification of actual patterns. To address this issue, HUPID extracts the fewer number of candidate patterns by reducing the overestimation values, through which it improves mining performance of tree-based incremental high utility pattern mining.

Performance analysis

In this section, we evaluate mining performance of tree-based incremental high utility pattern mining algorithms, IHUP [2] and HUPID [14], in terms of runtime and the number of generated candidate patterns with the Chain-store dataset (http://cucis.ece.northwestern.edu/projects/DMS/MineBench.html), where real utility information is included. All performance experiments were conducted on an experimental environment with a 4.0 GHz Intel processor and 32 GB memory, running the 64 bit Windows 7 OS. For the performance evaluation in an incremental environment, we first used only 20% of the dataset, and then added each 20% of the rest to the initial data. With the incremental data, we repeatedly constructed tree data structures of the algorithms, restructured them, and conducted mining processes.

Figures 2 and 3 show experimental results of the performance evaluation for IHUP and HUPID with the incremental Chain-store dataset in terms of runtime and the number of candidates when a minimum utility threshold is 0.03%. From the figure, we can observe that runtime is proportional to the number of extracted candidate patterns. In the results, moreover, HUPID mines the same number of high utility patterns faster than IHUP by generating the smaller number of candidates.

Figures 4 and 5 present the performance evaluation results of the algorithms with respect to the Accidents dataset, where the minimum utility threshold is set to 40%. As in the case of the previous test for Chain-store, HUPID shows better performance than that of IHUP. The reason why HUPID guarantees faster runtime performance is that the algorithm can extract a smaller number of candidate patterns. Recall that tree-based high utility pattern mining methods cannot directly actual high utility patterns since the utility factor does not satisfy the anti-monotone property. For this reason, they

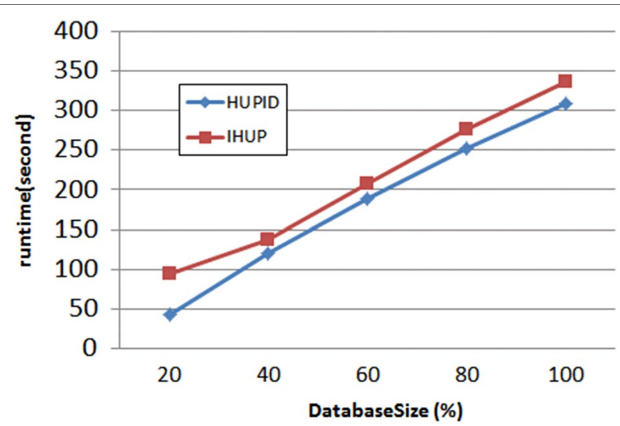

Fig. 2 Experimental results of runtimes on Chain-store

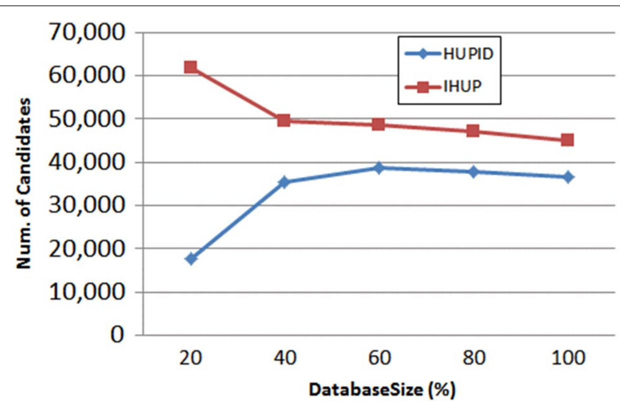

Fig. 3 Experimental results of number of candidates on Chain-store

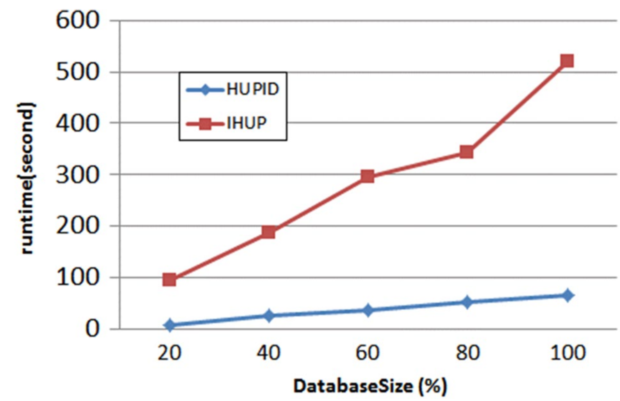

Fig. 4 Experimental results of runtimes on Accidents

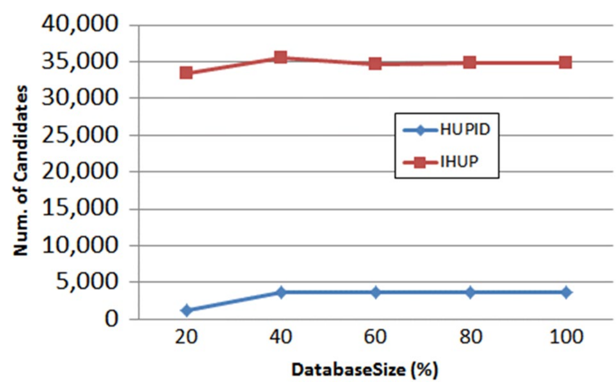

Fig. 5 Experimental results of number of candidates on Accidents

have used overestimation techniques to maintain this property. Since the overestimation approach has no choice but to extract a number of candidate patterns larger than that of actual high utility patterns, it has been considered as an important issue to reduce the number of generated candidates as many as possible. If the number of mined candidates becomes lower, the corresponding algorithm performance is naturally improved. In this regard, there is no doubt that HUPID outperforms IHUP since the algorithm can mine a smaller number of candidate patterns as shown in the figure.

Figures 6 and 7 show the experimental results for the Retail dataset, where the minimum utility threshold is set to 0.08%. When the database size is 20%, the runtime gap between the algorithms is very large, but the gap decreases as the database size becomes large because of the number of candidate patterns. Nevertheless, the performance of HUPID is always better than that of IHUP.

In the results of the Connect dataset, HUPID also has the best performance as shown in Figs. 8 and 9, where the minimum utility threshold is set to 92%. As shown in the figure, HUPID presents stable runtime performance regardless of the database size. On the other hand, IHUP shows different runtime results depending on the size settings. As the size of the Connect dataset becomes larger, its density feature becomes stronger. Therefore, the number of generated candidate patterns also decreases as the database

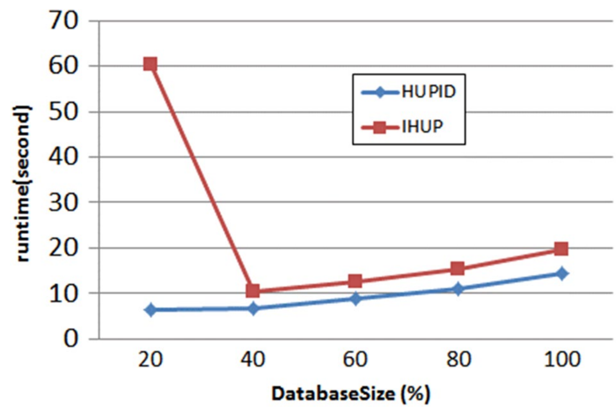

Fig. 6 Experimental results of runtimes on Retail

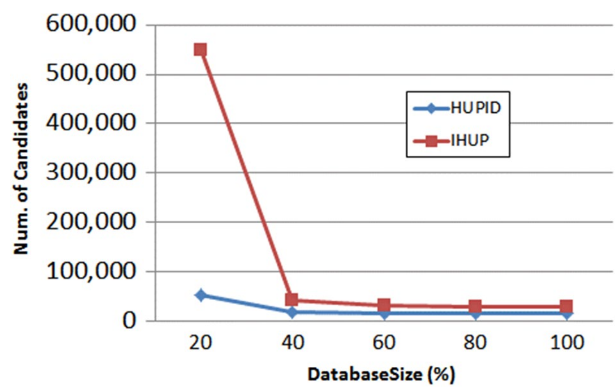

Fig. 7 Experimental results of number of candidates on Retail

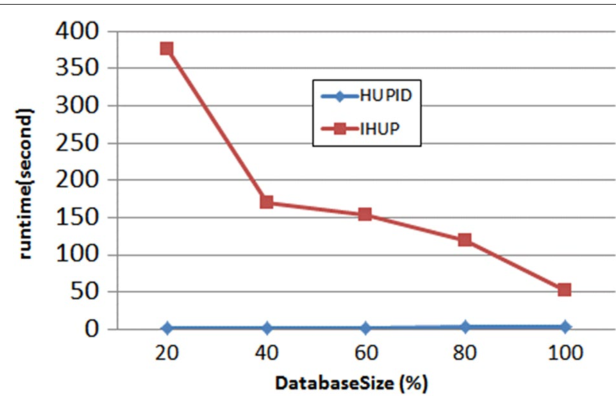

Fig. 8 Experimental results of runtimes on Connect

size becomes larger. However, since HUPID always extracts a smaller number of candidates in every case, its runtime performance is also always better than that of IHUP.

From the above experimental results, we can learn that performance of incremental high utility pattern mining algorithms depends on how many candidate patterns are

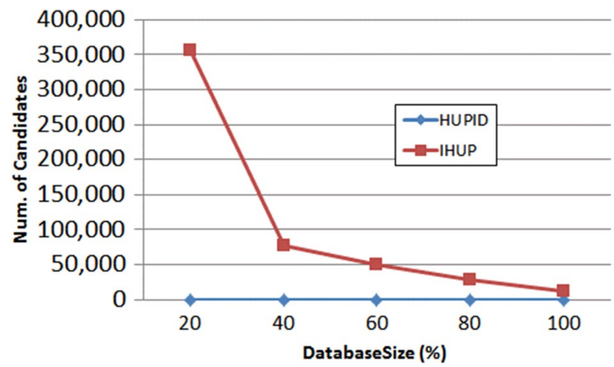

Fig. 9 Experimental results of number of candidates on Connect

extracted, and as a result developing an effective method for decreasing their number can be a good future work.

Conclusion

In this paper, we described the characteristics of various tree-based high utility pattern mining methods designed to handle incremental stream data. In addition, we also conducted extensive performance evaluation with respect to well-known benchmark datasets in order to analyze detailed features of recent incremental high utility pattern mining approach. The results of the performance analysis showed HUPID generates a smaller number of candidates than the other tree-based method and faster than the other one. The number of generated candidate patterns has a significant effect on runtime performance of algorithms. That is, we determined that an algorithm has better performance as it mines a smaller number of candidates on the tree-based structure. We are scheduled to research a variety of concepts and methods for mining patterns in dynamic environments as our future work.

Authors' contributions

JL suggested the proposed algorithm and wrote the contents of this paper. GL wrote the contents of this paper and reviewed this manuscript. UY provided the main idea of this paper, designed the overall architecture of the proposed algorithm. All authors read and approved the final manuscript.

Competing interests

The authors declare that they have no competing interests.

Availability of data and materials

Data will not be shared because of individual acquisition.

Consent for publication

Not applicable.

Ethics approval and consent to participate

Not applicable.

Funding

This study was funded by the Ministry of Education, Science and Technology of the National Research Foundation of Korea (NRF Nos. 20152062051 and 20155054624).

Publisher's Note

Received: 10 March 2017 Accepted: 14 August 2017
Published online: 09 November 2017

References

1. Agrawal R, Srikant R (1994) Fast algorithms for mining association rules. In: Proceedings of the 20th international conference on very large data bases Santiago de Chile, pp 487–499
2. Ahmed CF, Tanbeer SK, Jeong B-S, Lee Y-K (2009) Efficient tree structures for high utility pattern mining in incremental databases. IEEE Trans Knowl Data Eng 21(12):1708–1721
3. Cho Y, Moon S (2015) Recommender system using periodicity analysis via mining sequential patterns with time-series and frat analysis. J Converg 6(1):9–17
4. Choi J, Shin H, Nasridinov A (2016) A comparative study on data mining classification techniques for military applications. J Converg 7
5. Gaur M, Pant B (2015) Trusted and secure clustering in mobile pervasive environment. Hum Centric Comput Inf Sci. 5(1):1–17
6. Han J, Pei J, Yin Y (2004) Mining frequent patterns without candidate generation: a frequent-pattern tree approach. Data Min Knowl Discov 8(1):53–87
7. Jeeva S, Rajsingh E (2016) Intelligent phishing url detection using association rule mining. Hum Centric Comput Inf Sci. 6(1):1–19
8. Lin C-W, Lan G-C, Hong T-P (2012) An incremental mining algorithm for high utility itemsets. Expert Syst Appl 39(8):7173–7180
9. Liu Y, Liao W-K, Choudhary (2005) AN a Two-Phase algorithm for fast discovery of high utility itemsets. Adv Knowl Discov Data Min. Hanoi 689–695
10. Ryang H, Yun U, Lee G, Kim D, Jung W, Lee J, Gwon G (2016) Performance analysis of incremental high utility pattern mining methods. Korea Internet Inf Soc 17(2):99–100
11. Sato A, Huang R, Yen N (2015) Design of fusion technique-based mining engine for smart business. Hum Centric Comput Inf Sci. 5(1):1–16
12. Sanna G, Angius A, Concas G, Manca D, Eros F (2015) PCE: a knowledge base of semantically disambiguated contents. J Converg 6(2):10–18
13. Sohrabi MK, Roshani R (2017) Frequent itemset mining using cellular learning automata. Comput Hum Behav 68:244–253
14. Yun U, Ryang H (2015) Incremental high utility pattern mining with static and dynamic databases. Appl Intell 42(2):323–352
15. Zida S, Fournier-Viger P, Lin JC-W, Tseng VS (2017) EFIM: a fast and memory efficient algorithm for high-utility itemset mining. Knowl Inf Syst 51(2):595–625