International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2015): 78.96 | Impact Factor (2015): 6.391

A Survey on Mining High Utility Itemsets from Transactional Databases

Riswana .P .P[1], Divya .M[2]

[1]AWH Engineering College, KTU University, Department of Computer science & Engineering, Kuttikkatoor, Kozhikode, India

[2]AWH Engineering College, KTU University, Department of Computer science & Engineering, Kuttikkatoor, Kozhikode, India

Abstract: ***Mining high utility itemsets from a transactional database refers to the discovery of itemsets with high utility like profits. Frequent itemset mining (FIM) is one of the most fundamental problems in data mining. In this work, we propose a novel strategy based on the analysis of item co-occurrences to reduce the number of join operations that need to be performed (FHM: Faster High-Utility Miner). A better approach in which we characterize a differentially private FIM algorithm based on the FP-growth algorithm, which is referred to as PFP-growth. The PFP-growth algorithm consists of a preprocessing phase and a mining phase. AS another commitment, we incorporate utility into sequential pattern mining, and a generic framework for high utility sequence mining is defined. An efficient algorithm, USpan, is presented to mine for high utility sequential patterns.***

Keywords: Frequent Itemset Mining, Co-Occurrence Pruning, differential privacy, High-Utility Mining, Sequential pattern Mining

1. Introduction

Data mining is the process of revealing nontrivial, previously unknown and potentially useful information from large databases. Discovering useful patterns hidden in a database plays an essential role in several data mining tasks, such as frequent pattern mining, weighted frequent pattern mining, and high utility pattern mining. Frequent itemset mining (FIM) is one of the most fundamental problems in data mining. It has practical importance in a wide range of application areas such as decision support, Web usage mining, bioinformatics, etc. Given a transaction database, FIM consists of discovering frequent itemsets. i.e. groups of items (itemsets) appearing frequently in transactions [2]. However, an important limitation of FIM is that it assumes that each item cannot appear more than once in each transaction and that all items have the same importance (weight, unit profit or value).

To address this issue, the problem of FIM has been redefined as High-Utility Itemset Mining (HUIM) to consider the case where items can appear more than once in each transaction and where each item has a weight (e.g. unit profit). The goal of HUIM is to discover itemsets having a high utility (e.g. generating a high profit). FHM is based on the observation that although HUI-Miner performs a single phase and thus do not generate candidates as per the definition of the two-phase model, HUI-Miner explores the search space of itemsets by generating itemsets and a costly join operation has to be performed to evaluate the utility of each itemset. To reduce the number of joins that are performed, we propose a novel pruning strategy named EUCP (Estimated Utility Co-occurrence Pruning) that can prune itemsets without having to perform joins. This strategy is easy to implement and very effective.

A better approach in which we present private FP growth (PFP-growth) algorithm, which consists of a preprocessing phase and a mining phase. In the preprocessing phase, we transform the database to limit the length of transactions. The preprocessing phase is irrelevant to user-specified thresholds and needs to be performed only once for a given database In the mining phase, given the transformed database and a user-specified threshold, we privately discover frequent itemsets.

Sequential pattern mining has emerged as an important topic in data mining. The selection of interesting sequences is generally based on the frequency/support framework: sequences of high frequency are treated as significant. Under this framework, the downward closure property (also known as Apriori property) [18] plays a fundamental role for varieties of algorithms designed to search for frequent sequential patterns [20, 21, 23]. In practice, many patterns are identified by frequent sequential pattern mining algorithms. Most of them may not be informative to business decision-making, since they do not show the business value and impact. This has led to high utility pattern mining [22], which selects interesting patterns based on minimum utility rather than minimum support. A sequence is of high utility only if its utility is no less than a user-specified minimum utility. An efficient algorithm, USpan, is presented to mine for high utility sequential patterns. In USpan, we introduce the lexicographic quantitative sequence tree to extract the complete set of high utility sequences and design concatenation mechanisms for calculating the utility of a node and its children with two effective pruning strategies.

2. The FHM Algorithm

The main procedure in FHM takes as input a transaction database with utility values and the *minutil* threshold. The algorithm first scans the database to calculate the TWU of each item. Then, the algorithm identifies the set I* of all items having a TWU no less than *minutil*. The TWU values of items are then used to establish a total order $\succ$ on items, which is the order of ascending TWU values(as suggested in [1]). A second database scan is then performed.

During this database scan, items in transactions are reordered according to the total order $\succ$, the utility-list of each item $I \in I^*$ is built and our novel structure named EUCS (Estimated Utility Co-Occurrence Structure) is built. This latter structure

is defined as a set of triples of the form (a, b, c) ∈ I* × I*×R. A triple (a,b,c) indicates that TWU({a,b})=c. The EUCS can be implemented as a triangular matrix or as a hashmap of hashmaps where only tuples of the form (a, b, c) such that c≠ 0 are kept. Building the EUCS is very fast (it is performed with a single database scan) and occupies a small amount of memory, bounded by | I*| × | I*|, although in practice the size is much smaller because a limited number of pairs of items co-occurs in transactions. After the construction of the EUCS, the depth-first search exploration of itemsets starts by calling the recursive procedure Search with the empty ∅ itemset ;, the set of single items I* , *minutil* and the EUCS structure.

Tid	Transactions
T_1	(a,1)(c,1)(d,1)
T_2	(a,2)(c,6)(e,2)(g,5)
T_3	(a,1)(b,2)(c,1)(d,6),(e,1),(f,5)
T_4	(b,4)(c,3)(d,3)(e,1)
T_5	(b,2)(c,2)(e,1)(g,2)

Item	a	b	c	d	e	f	g
Profit	5	2	1	2	3	1	1

Figure 1: A transaction database (left) and external utility values (right)

Table 1: The FHM algorithm

The FHM algorithm
input : D: a transaction database, *minutil:* a user-specified threshold output: the set of high-utility itemsets 1 Scan D to calculate the TWU of single items; 2 I* each item i such that TWU(i) < *minutil*; 3 Let ≻ be the total order of TWU ascending values on I*; 4 Scan D to built the utility-list of each item i ∈ I* and build the EUCS structure; 5 Search (∅, I*, *minutil*, EUCS);

TID	TU
T_1	8
T_2	27
T_3	30
T_4	20
T_5	11

Item	TWU
a	65
b	61
c	96
d	58
e	88
f	30
g	38

Item	a	b	c	d	e	f
b	30					
c	65	61				
d	38	50	58			
e	57	61	77	50		
f	30	30	30	30	30	
g	27	38	38	0	38	0

Figure 2: Transaction utilities (left), TWU values (center) and EUCS (right)

3. PFP-Growth Algorithm

In this work, we present a private FP-growth algorithm, which is referred to as PFP-growth. The framework of PFP-growth algorithm is shown in Figure below.

PFP-growth algorithm consists of a Preprocessing phase and a mining phase. In the preprocessing phase, we transform the database to limit the length of transactions. The preprocessing phase is irrelevant to user-specified thresholds and needs to be performed only once for a given database. We argue, to enforce such a limit, long transactions should be split rather than truncated. That is, if a transaction has more items than the limit, we divide it into multiple subsets (i.e., sub transactions) and guarantee each subset is under the limit. We devise a novel smart splitting method to transform the database. In particular, to ensure applying ϵ-differentially private algorithm on the transformed database still satisfies ϵ -differential privacy for the original database, we propose a weighted splitting operation. Moreover, to preserve more frequency information in subsets, we propose a graph-based approach to reveal the correlation of items within transactions and utilize such correlation to guide the splitting process.

In the mining phase, given the transformed database and a user-specified threshold, we privately discover frequent itemsets. Despite the potential advantages of transaction splitting, it might bring frequency information loss. In the mining phase, inspired by the double standards method in [15], we propose a run-time estimation method to offset such information loss. In particular, given the noisy support of an itemset in the database transformed by transaction splitting, we first estimate its actual support in the transformed database, and then further compute its actual support in the original database. In addition, by leveraging the downward closure property, we put forward a dynamic reduction method. During the mining process, we dynamically estimate the number of support computations, so that we can gradually reduce the amount of noise required by differential privacy.

Through formal privacy analysis, we show that our PFP-growth algorithm is ϵ -differentially private. . For two databases D and D', they are neighboring databases if they differ by at most one record. A fundamental concept in differential privacy is the sensitivity [12]. The amount of injected noise is carefully calibrated to the sensitivity. The sensitivity of count queries is used to measure the maximum possible change in the outputs over any two neighboring databases. For the computation whose outputs are real, Dwork et al. [12] propose the Laplace mechanism. The Laplace mechanism adds noise drawn randomly from the Laplace distribution. For the computation whose outputs are integer, Ghosh et al. [17] propose the Geometric mechanism.

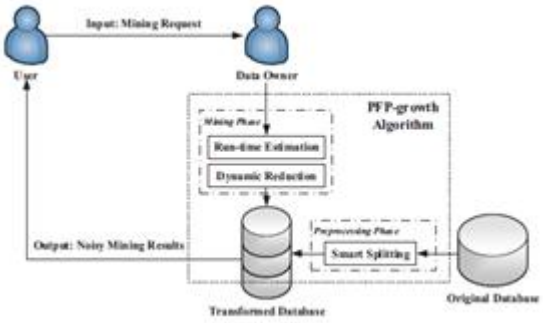

Figure 3: The Framework of PFP-growth Algorithm

4. USpan Algorithm

The USpan algorithm is illustrated in table 2 below. The input for USpan is a database S and a minimum utility threshold ξ; the output includes all the high utility patterns.

Lines 1 describes the depth pruning strategy. A depth pruning strategy stops USpan from going deeper by identifying the leaf nodes in the tree. Imagine the following scenario: the pivots are approaching the end of q-sequences;

meanwhile, the maximum utility of the sequence is much less than ξ. The gap is so large that even if all the utilities of the remaining q-items are counted into the utility of the sequence, the cumulative utility still cannot satisfy ξ. In this situation, we use the depth pruning strategy to backtrack USpan instead of waiting to go deeper and returning with nothing. We use the notation urest(i, s) to refer to the remaining utility at q-item i (exclusive) in q-sequence s.

$$\sum_{i \in s' \wedge s' \sim t \wedge s' \subseteq s \wedge s \in S} \left(urest(i,s) + u(s') \right) \quad (1)$$

A node is judged as a leaf or not based on the comparison between the value of Equation (1) and ξ; if it is lower than ξ then it returns to its parent nodes. Lines 2 to 4 are the scanning subroutine with the width pruning in Line 5. To avoid selecting the unpromising items, we propose a width pruning strategy for the scanning subroutine. This is based on the sequence-weighted downward closure property (SDCP). Sequence-weighted utilization (SWU) of a sequence.

$$SWU(t) = \sum_{s' \sim t \wedge s' \subseteq s \wedge s \subseteq S} u(s) \quad (2)$$

If SWU (t') ≥ ξ, we say item i is a promising item to t. Otherwise, i is called an unpromising item. In the implementation, to test whether an item is promising, we do not have to generate the new sequence to test whether an item is promising. We simply add the utilities of all the sequences; this is equal to the SWU of the new sequence. For utility-based sequences, we adapt the concept of the Lexicographic Sequence Tree in [20] to the characteristics of q-sequences, and come up with the Lexicographic Q-sequence Tree (LQS-Tree) to construct and organize utility based q-sequences.

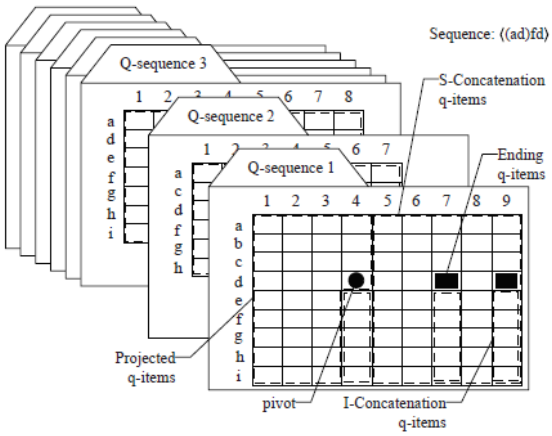

Figure 4: Data Representation in USpan

Suppose we have a k-sequence t, we call the operation of appending a new item to the end of t to form (k+1)-sequence concatenation. If the size of t does not change, we call the operation I-Concatenation. Otherwise, if the size increases by one, we call it S-Concatenation.

Table 2: USpan Algorithm

Algorithm USpan(t, v(t))
Input: A sequence t, t's utility v(t), a utility-based sequence database S, the minimum utility threshold ξ.
Output: All high utility sequential patterns
1: if p is a leaf node then return
2: scan the projected database S(v(t)) once to:
3: a).put I-Concatenation items into *ilist*, or
4: b).put S-Concatenation items into *slist*
5: remove unpromising items in *ilist* and *slist*
6: for each item i in *ilist* do
7: (t', v(t')) ← I-Concatenate(p,i)
8: if $u_{max}(t') \geq \xi$ then
9: output t'
10: USpan(t', v(t'))
11: for each item i in *slist* do
12: (t', v(t')) ← S-Concatenate(p,i)
13: if $u_{max}(t') \geq \xi$ then
14: output t'
15: USpan(t', v(t'))
Return

Once the concatenation items are collected, the unpromising items are omitted from the respective lists. Lines 7 and 12 construct the I-Concatenation and S-Concatenation children respectively. It invokes the concatenation to generate the utilities of sequences; the positions are also maintained. USpan then outputs the high utility sequences if qualified, and recursively invokes itself to go deeper in the LQS-Tree.

5. Conclusions

In this work, we have presented a novel algorithm for high-utility itemset mining named FHM (Fast High-Utility Miner). This algorithm integrates a novel strategy named EUCP (Estimated Utility Co-occurrence Pruning) to reduce the number of joins operations when mining high-utility itemsets using the utility list data structure. We additionally presented another private FP-growth (PFP-growth) algorithm, which consists of a preprocessing phase and a mining phase. In the preprocessing phase, to better improve the utility-privacy tradeoff, we devise a smart splitting method to transform the database. In the mining phase, a run-time estimation method is proposed to offset the information loss incurred by transaction splitting. Finally we have provided a systematic statement of a generic framework, and an efficient algorithm, USpan, for mining high utility sequential patterns. In fact, USpan stores the positions and utilities of the candidates, a range of different functions can be applied on them with different purposes in such a framework.

References

[1] Liu, M., Qu, J.:Mining High Utility Itemsets without Candidate Generation. In Proceedings of CIKM12, pp. 55{64 (2012).
[2] Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: Proc. Int. Conf. Very Large Databases, pp. 487–499, (1994)
[3] Tseng, V. S., Shie, B.-E., Wu, C.-W., Yu., P. S.: Efficient Algorithms for Mining High Utility Itemsets from Transactional Databases. In: IEEE Trans. Knowl. Data Eng. 25(8), pp. 1772{1786 (2013).

[4] Ahmed, C. F., Tanbeer, S. K., Jeong, B.-S., Lee, Y.-K.: Efficient Tree Structures for High-utility Pattern Mining in Incremental Databases. In: IEEE Trans. Knowl. Data Eng. 21(12), pp. 1708–1721 (2009).
[5] Li, Y.-C., Yeh, J.-S., Chang, C.-C.: Isolated items discarding strategy for discovering high utility itemsets. In: Data & Knowledge Engineering. 64(1), pp. 198–217 (2008)
[6] Fournier-Viger, P., Nkambou, R., Tseng, V. S.: RuleGrowth: Mining Sequential Rules Common to Several Sequences by Pattern-Growth. In: Proc. ACM 26th Symposium on Applied Computing, pp. 954–959 (2011)
[7] Fournier-Viger, P., Wu, C.-W., Gomariz, A., Tseng, V. S.: VMSP: Efficient Vertical Mining of Maximal Sequential Patterns. In: Proc. 27th Canadian Conference on Artificial Intelligence, Springer, LNAI, pp. 83-94 (2014)
[8] Liu, Y., Liao, W., Choudhary, A.: A two-phase algorithm for fast discovery of high utility itemsets. In: Proc. PAKDD 2005, pp. 689–695 (2005)
[9] Fournier-Viger, P., Gomariz, A., Campos, M., Thomas, R.: Fast Vertical Sequential Pattern Mining Using Co-occurrence Information. In: Proc. 18th Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer, LNAI, (2014)
[10] C. Dwork, "Differential privacy," in ICALP, 2006.
[11] L. Sweeney, "k-anonymity: A model for protecting privacy," Int. J. Uncertain. Fuzziness Knowl.-Base Syst, 2002.
[12] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in TCC, 2006.
[13] C. Zeng, J. F. Naughton, and J.-Y. Cai, "On differentially private frequent itemset mining," in VLDB, 2012.
[14] J. Vaidya and C. Clifton, "Privacy preserving association rule mining in vertically partitioned data," in KDD, 2002.
[15] M. Kantarcioglu and C. Clifton, "Privacy-preserving distributed mining of association rules on horizontally partitioned data," TKDE, 2004.
[16] L. Bonomi and L. Xiong, "A two-phase algorithm for mining sequential patterns with differential privacy," in CIKM, 2013.
[17] A. Ghosh, T. Roughgarden, and M. Sundararajan, "Universally utility-maximizing privacy mechanisms," SIAM Journal on Computing, 2012.
[18] R. Agrawal and R. Srikant, Mining sequential patterns, ICDE 1995, pp. 3-14.
[19] C. F. Ahmed, S. K. Tanbeer, J. Byeong-Soo and L. Young-Koo, Efficient tree structures for high utility pattern mining in incremental databases, TKDE 2009, vol. 21, pp. 1708-1721.
[20] J. Ayres, J. Flannick, J. Gehrke and T. Yiu, Sequential PAttern mining using a bitmap representation, ICDM 2002, pp. 429-435.
[21] J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal and M.C. Hsu.,PrefixSpan: mining sequential patterns efficiently by prefix-projected pattern growth, ICDE 2001, pp. 215-224.
[22] H. Yao, H. J. Hamilton and C. J. Butz, A foundational approach to mining itemset utilities from databases, ICDM 2004, pp. 482-486.
[23] M. J. Zaki, SPADE: An Efficient Algorithm for Mining Frequent Sequences, Machine Learning, 2001, vol. 42, pp. 31-60.

Author Profile

Riswana. P. P received B-Tech degree in Information Technology from Calicut University in 2010. Currently pursuing M Tech in Computer Science and Engineering from KTU University, respectively.

Divya. M received the B Tech and M Tech degrees in Computer Science and Engineering from University of Calicut in 2014 respectively. Currently she is working as assistant professor in AWH Engineering College.