行政院國家科學委員會補助專題研究計畫成果報告

※ ※※※※※※※※※※※※※※※※※※※※※※※
※ ※
※ ※
※ 大型資料庫中循序樣式之勘測與維護 ※
※ ※
※※※※※※※※※※※※※※※※※※※※※※※※

計畫類別：Ｙ**個別型計畫**　　□**整合型計畫**

計畫編號：NSC－89－2213－E－009－006

執行期間：88 **年** 08 **月** 01 **日至** 89 **年** 07 **月** 31 **日**

計畫主持人：李素瑛

著作人：李素瑛

本成果報告包括以下應繳交之附件：

□**赴國外出差或研習心得報告一份**

□**赴大陸地區出差或研習心得報告一份**

□**出席國際學術會議心得報告及發表之論文各一份**

□**國際合作研究計畫國外研究報告書一份**

執行單位：國立交通大學資訊工程學系

中　華　民　國 89 **年** 10 **月** 31 **日**

行政院國家科學委員會專題研究計畫成果報告
大型資料庫中循序樣式之勘測與維護
Discovery and Maintenance of Sequential Patterns in Large Databases

計畫編號：NSC 89-2213-E-009-006
執行期限：88 年 8 月 1 日至 89 年 7 月 31 日
主持人：李素瑛　國立交通大學資訊工程學系
著作人：李素瑛　國立交通大學資訊工程學系
計畫參與人員：林明言 國立交通大學資訊工程學系

一、中文摘要

關聯規則與循序樣式的勘測是重要的資料探勘問題。在銷售資料庫中，循序樣式的勘測是要找出所有同一個顧客的交易間，所銷售的商品之循序性關係。其探勘包括了關聯規則的勘測，以及項目集間排列組合的順序關係。

由於可能序列的個數，相當影響探勘的演算法效率。因此，一個有效率的演算法，必須能夠涵蓋所有頻繁序列的可能序列，並減少資料庫檢測的次數或資料筆數的循序樣式勘測方法。同時，隨著資料庫新資料的加入、舊資料的更新及修改，必須對現有的循序樣式加以維護。本計畫設計了一個新演算法，利用現有的循序樣式資訊，減少可能序列的產生，同時降低重複的資料庫檢視，有效改善因資料庫更新導致愈來愈長的探勘執行時間。

關鍵詞：資料探勘、循序樣式、時間性關聯規則、序列勘測、序列維護

Abstract

The discovery of sequential patterns is to find out the sequence relationships among transaction items of inter-transactions in a sales database. The mining of sequential patterns comprise of not only the discovery of associations but also the permutation of sets of items.

The project aims to devise an efficient algorithm for the mining and the maintenance of sequential patterns. Since the number of candidate sequences is one of key factors that affect the performance of mining algorithm, we have developed a method for speeding up the mining process. Our new algorithm utilizes previous knowledge about frequent patterns to reduce the number of candidate sequences, and to eliminates the redundant scanning of customer sequences. The performance of mining new patterns, i.e., pattern maintenance, is effectively improved.

Keywords: Data Mining, Sequential Pattern, Temporal Association Rule, Sequence Discovery, Sequence Maintenance

二、緣由與目的

Data Mining and Knowledge Discovery in Databases (KDD) thus has been recognized as a promising field of AI, statistics and database researches [1]. Mining of sequential patterns was first introduced in [2]. The purpose is to discover sequential patterns in a database of customer transactions, consisting of records having customer id, transaction time, and transaction items. A sequential pattern indicates a sequence of transactions that usually occurred serially in time.

The issue of maintaining sequential patterns becomes essential because database transactions may be updated over time. Due to new transactions, some existing sequential patterns would become invalid after database update since they might no longer have sufficient supports, while some new sequential patterns might appear. However, there is not much work on incremental updating of sequential patterns. In order to ascertain sequential patterns up to date for the updated database, re-execution of mining algorithm on the updated database is required.

Nevertheless, because of appended database transactions, re-execution of mining algorithm demands more time than previous mining. Moreover, the effort of mining last time is wasted if all discovered sequential patterns in the original database were ignored.

In this project, we propose a new algorithm that can utilize information about discovered sequential patterns, and efficiently reduce candidate sequences for mining in updated database. The objective of this work is to solve the update problem of sequential patterns after a nontrivial number of new transactions have been appended to original database. Assuming that minimum support keeps the same, existing frequent sequences and their supports in original database could be utilized for the mining of updated database. Through effective reuse of previous derived knowledge in each pass, the number of candidate sequences is substantially reduced. Instead of counting all candidate sequences for full updated database, counting reduced candidate sequences for original database and for increment database, could achieve better performance. For transactions that have same customer id in both original database and increment database, we extract old transactions from original database and merge into increment database. Since the size of increment database is smaller than original database in general, better performance thus could be retained.

三、文獻回顧

There are many excellent algorithms that deal with the mining of association rules [1, 3] and several algorithms were developed for the mining of sequential patterns [2, 9, 10]. Algorithms to discover frequent episodes in a single long sequence and its generalization can be found in [6, 7]. There is no efficient algorithm designed for the maintenance of sequential patterns in large databases. As mentioned earlier, they call for the need of re-mining the whole database. Incremental updating techniques for the maintenance of association rules were proposed in [4, 5]. However, previous work dealing with the incremental updating of sequential patterns cannot be found. Besides, appended transactions induce more complicated problems in sequential pattern mining than in association rule mining. The problem of finding association rules concerns with intra-transaction patterns whereas that of sequential pattern mining concerns with inter-transaction patterns [2]. Appended transactions bear no relation to original database for the former problem, while for the latter problem, different transactions with same customer id in both databases must be sorted into one data sequence.

四、研究成果

We propose a new algorithm, FASTUP, to efficiently keep patterns up-to-date. The basic construct of the *FASTUP* algorithm is similar to that of *GSP*[9], with improvements on candidate generation and support counting. As shown in Figure 1, *FASTUP* reduces the size of C_k (candidate k-sequences) into C_k' and updates the supports of sequences in S^{DB} (frequent k-sequences in *DB*) by simply checking the increment database *db*, which is usually smaller than the original database *DB*. In addition, the *separate counting* technique enables *FASTUP* to accumulate candidates' supports quickly because only the *new* candidates, whose supports are unavailable from S^{DB}, need to be checked against *DB*. In order to avoid duplicate counting, *FASTUP* deals with the required merges by implicit merging technique.

Essentially, *FASTUP* generates candidates and examines data sequences to determine frequent patterns and their supports in multiple passes, like *GSP*. Nevertheless, with the knowledge of previous sequential patterns, *FASTUP* further reduces the number of candidates required for checking, and separates the process of support counting to efficiently counts the supports of candidates during the same database scanning. As shown in Figure 1, Support Counting (I) deals with sequences in the increment database *db* using the full

candidate set C_k, while the Support Counting (II) works with sequences in the original database *DB* using the reduced candidate set C_k'. Moreover, we also integrate the implicit merging into the first support counting process. Figure 2 lists the proposed *FASTUP* algorithm.

In each pass, we initialize the two support counts of each candidate to zero, and read the support count of each frequent *k*-sequence *x*. We then accumulate the increases in support count of candidates with respect to the sequences in *db* by Support Counting (I). Before Support Counting (II) starts, candidates which are also frequent in *DB* but can not be frequent in *UD* according to Lemma 4 are filtered out. The full candidate set C_k is reduced into the set C_k'. Next, the Support Counting (II) calculates the support counts of these promising candidates with respect to the sequences in *DB*. As indicated in Lemma 1, the support count of any candidate *k*-sequence is the sum of the two counts obtained after the two counting processes. Consequently, we can discover S_k by validating the sum of the two counts of every candidate. The S_k is used for generating the complete candidate set for the next pass, employing the same candidate generation procedure in *GSP*. The above process is iterated until no more candidates.

Extensive experiments were performed to compare the execution times of *FASTUP* and *GSP*. The effect on performance with various *minsup*s for datasets having the same number of customers was evaluated first. The experiments shows that *FASTUP* is always faster than *GSP* for all values of minimum supports. In general, *FASTUP* gains less by incremental update at higher increment ratio because bigger increment ratio causes more pattern updatings. *FASTUP* updates patterns more efficiently than *GSP* for all the comeback ratios and that *FASTUP* was efficient with implicit merging, even when the comeback ratio was increased to 100%, i.e., all the transactions in the increment database must be merged. In these experiments, other datasets generating with various combinations of parameters. Still, *FASTUP* was several times faster than *GSP* for distinct combinations of data characteristics.

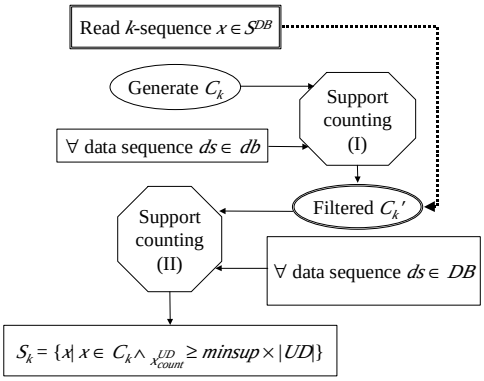

Figure 1. The *k*-th pass of *FASTUP*.

1) /* Initially, each item is a candidate *1*-sequence */
2) $C_1 = \{x \mid x = <(\sim)>, \sim \in V\}$; /* Let *x* be a candidate *k*-sequence */
3) $k = 1$; /* Start from pass 1 */
4) repeat /* Find frequent *k*-sequences in the *k*-th pass */
5) for each $x \in C_k$ do $x_{count}^{DB} = x_{count}^{db} = 0$; /* Initialize counters */
6) Read S_k^{DB}; /* S_k^{DB}= {frequent *k*-sequences in *DB* */
7) Check sequences in *db* by Support Counting (I) ; /* See Figure 8 */
8) /* Prune candidates: (1) counted in S_k^{DB} (2) insufficient "new" counts */
9) $C_k' = C_k - \{x \mid x \in S_k^{DB}\} - \{x \mid x_{count}^{db} < minsup \times (|UD| - |DB|)\}$;
10) Check sequences in *DB* by Support Counting (II) ;/* See Figure 8 */
11) /* Frequent *k*-sequences in *UD* found */
12) $S_k = \{x \mid x \in C_k \wedge x_{count}^{DB} + x_{count}^{db} \geq minsup \times |UD|\}$;
13) $k = k + 1$;
14) Generate C_k with S_{k-1} ; /* Generate candidates for next pass */
15) until no more candidates
16) Answer $S^{UD} = \bigcup_k S_k$;

Figure 2. FASTUP Algorithm

五、結論

The problem of sequential pattern mining is more complicated than association discovery due to sequence permutation. Without maintenance, validity of discovered patterns may change and new patterns may emerge after updates on databases. Keeping sequential patterns current by re-execution of mining algorithm on the whole database takes more time than last mining because of the additional sequences. We have devised the *FASTUP* algorithm utilizing discovered knowledge to resolve the problem efficiently by incremental updating without re-mining from scratch. The performance improvements result from effective implicit merging, early candidate pruning, and efficient separate counting.

Implicit merging ensures that *FASTUP* employs correctly combined data sequences while preserving previous knowledge useful for incremental updating. Candidate pruning after updating pattern supports against the increment database further accelerates the whole process, since fewer but more promising candidates are generated by just checking counts in increment database. Eventually, efficient support counting of promising candidates over the original database accomplishes the discovery of new patterns. *FASTUP* both updates the supports of existing patterns and finds out new patterns for the updated database. Our simulation shows that the proposed incremental updating mechanism is several times faster than re-mining using the *GSP* algorithm, no matter what combination or what proportion of transactions in the increment database and the original database. *FASTUP* runs slightly slower than *GSP* only when the increment database is bigger than the original database, i.e. the maintenance period is too long.

The *FASTUP* algorithm currently solves the pattern updating problems with the constraint of fixed minimum support. Further researches could be extended to problems of dynamically varying minimum supports. Generalized sequential pattern problems [2], such as patterns with *is-a* hierarchy or with sliding-time window property, are also worthy of further investigation since different constraints induce diversified maintenance difficulties. In addition to the maintenance problem, constantly updated database generally create a pattern-changing history, indicating changes of sequential patterns at different time. It is challenging to extend our algorithm to efficient exploration for the history of pattern updates.

六、參考文獻

[1] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen and A. I. Verkamo, Fast Discovery of Association Rules, Advances in Knowledge Discovery and Data Mining, edited by U. M. Fayyad et al, AAAI/MIT Press, pp. 307-328, 1996.

[2] R. Agrawal and R. Srikant, Mining Sequential Patterns, Proceedings of the 11th International Conference on Data Engineering (ICDE'95), pp. 3-14, Taipei, Taiwan, 1995.

[3] S. Brin, R. Motwani, J. Ullman and S. Tsur, Dynamic Itemset Counting and Implication Rule for Market Basket Data, Proceedings of the 1997 SIGMOD Conference on Management of Data, pp. 255-264, 1997.

[4] D. W. Cheung, J. Han, V. T. Ng and C. Y. Wong, Maintenance of Discovered Association Rules in Large Databases: An Incremental Updating Technique, Proceedings of International Conference on Data Engineering, 1996.

[5] S. D. Lee, D. Cheung, B. Kao, A General Incremental Technique For Maintaining Discovered Association Rules, Proceedings of the 5th International Conference On Database Systems For Advanced Applications, pp. 185-194, Melbourne, Australia, Apr. 1997.

[6] H. Mannila and H. Toivonen, Discovering Generalized Episodes using Minimal Occurrences, Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD'96), pp. 146-151, Portland, 1996.

[7] H. Mannila, H. Toivonen and A. I. Verkamo, Discovering Frequent Episodes in Sequences, Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD'95), pp. 210-215, Montreal, Canada, 1995.

[8] J. S. Park, M. S. Chen, and P. S. Yu., Using a Hash-Based Method with Transaction Trimming for Mining Association Rules, IEEE Transactions on Knowledge and Data Engineering, Vol. 9, No. 5, pp. 813-825, 1997.

[9] R. Srikant and R. Agrawal, Mining Sequential Patterns: Generalizations and Performance Improvements, Advances in Database Technology–5th International Conference on Knowledge Discovery and Data Mining (KDD'95), pp. 269-274, Montreal, Canada, 1995.

[10] M. J. Zaki, Fast Mining of Sequential Patterns in Very Large Databases, Technical Report 668, The University of Rochester, New York, Nov. 1997.