行政院國家科學委員會補助專題研究計畫成果報告

互動式勘測大型資料庫中循序樣式之研究

計畫類別：□個別型計畫    ■整合型計畫
計畫編號：NSC 90-2213-E-009-078
執行期間：90年8月1日至91年7月31日

主持人：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

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91年10月30日
Interactive Discovery of Sequential Patterns in Large Databases

一、中文摘要
循序樣式的勘測，是從序列資料庫的序列中，挖掘出所有具循序關係的項目集。通常使用者必須指定一最小支持。然而，使用者通常必須不斷地嘗試各不同的最小支持數值並觀察勘測結果，經由這種互動式循序樣式勘測，找到理想的結果。本計畫建構一個高效率的互動式循序樣式勘測方法。以建立知識庫的方式，來改善互動式循序樣式探勘的效率。減少使用者各次探勘所需的執行時間，並減少整個互動式勘測所需之時間。我們所設計的資料結構與方法，可以提供有效率的、符合多使用者需求的、互動式循序樣式勘測。

關鍵詞：資料探勘、互動式勘測、知識庫

Abstract

The discovery of sequential patterns has become a challenging task due to its complexity. Essentially, a user would specify a minimum support threshold with respect to the database to find out the desired patterns. The mining process is usually iterative since the user must try various thresholds to obtain the satisfactory result. In order to minimize the total execution time and the response time for each trial, we propose a knowledge base assisted algorithm for interactive sequence discovery, called KISP. KISP constructs a knowledge base accumulating the pattern information in individual mining, eliminates considerable amount of potential patterns to facilitate efficient support counting, and speeds up the whole process. In addition, we further optimize the algorithm by direct generations of the reduced candidate sets and concurrent counting of variable sized candidates. The conducted experiments show that KISP outperforms GSP by several orders of magnitudes for interactive sequence discovery.

Keywords: data mining, Interactive discovery, knowledge base

二、緣由與目的

Mining sequential patterns, which finds out temporal associations among item-sets in the sequence database, is an important issue in data mining. A classic application of the problem is the market basket analysis whose database contains purchase records, where each record is an ordered sequence of itemsets (sets of items) bought by a customer. The objective is to discover the itemsets in future purchase after certain itemsets were bought. The mining technique can be applied to various domains such as discovering the relationships between the symptoms and certain diseases in medical applications. In comparison to the mining of association rules [3], sequential pattern mining is more complicated because not only the frequent itemsets but also the temporal relationships must be found.

The mining process is very difficult and time-consuming because patterns could be formed by any permutation of itemsets formed by any combination of possible items in the

plans.
database. In order to distinguish the interesting patterns, a user must supply a *minimum support* threshold (abbreviated *minsup*) for the mining. The result of mining finds out the set of patterns having *supports* greater than or equal to the *minsup*. The *support* of a pattern is the percentage of sequences (in the database) containing the pattern. The discovered patterns are called sequential patterns or frequent sequences. Most approaches focused on minimizing the search space of *potential sequential patterns* (called *candidates*), or on minimizing the required disk I/O due to the multiple database scanning. All these approaches discover the patterns by directly executing the mining algorithms once a *minsup* is specified.

However, the mining process is typically iterative and interactive since a user may specify a *minsup* value that results in too many or too few patterns. Usually, the user must try various *minsup* values until the result is satisfactory. Nevertheless, most approaches are not designed to deal with repeated mining under such circumstance so that each *minsup* invokes a re-mining from scratch. Some approaches solved the interactive problem by pre-processing using an assumed least *minsup*. Nevertheless, the lengthy pre-processing has to be executed again if a user supplies a *minsup* below the assumed least value.

Therefore, we propose a simple approach, called KISP, to improve the efficiency of sequential pattern discovery with changing supports. KISP utilizes the information obtained from prior mining processes, and generates a knowledge base (abbreviated KB) for further queries about sequential patterns of various *minsup*. When the results cannot be directly derived from the knowledge base, KISP incorporates KB into a fast sequence discovery by eliminating the candidates existing in KB before support counting. Unlike those approaches assuming a least *minsup* for pre-processing before iterative mining, KISP accepts any *minsup* value and has no difficulty in mining huge databases even with a small main memory. The conducted experiments on well-known synthetic data show that KISP effectively improves the interactive mining performance.

三、文獻回顧

The problem of interactive association discovery was addressed in [1]. The method in [1] preprocesses the data in the transactional database, and stores frequent itemsets in an adjacency lattice. Online repeated queries about association rules are answered by graph theoretic searching on the lattice.

Similarly, a knowledge cache is used for interactive association discovery in [9]. The knowledge cache contains frequent itemsets and the non-frequent itemsets, if memory space is available, that have been discovered while processing other queries. The study [9] indicated that their *benefit replacement* algorithm is the best caching algorithm.

Although on-line association discovery \[1, 5, 9, 10\] is close to our problem, these approaches aim to interactively find frequent itemsets rather than frequent sequences, which is more complicated. One related work of interactive sequence mining extended the SPADE algorithm [16] into the ISM (Incremental Sequence Mining) algorithm for incremental and interactive sequence mining [11]. All queries are performed on a pre-processed in-memory data structure, the Increment Sequence Lattice (ISL). Therefore, A ‘small enough’ *minsup* must be pre-selected to apply SPADE for pre-processing and saving the results in ISL. Nevertheless, if a query involves a threshold smaller than the pre-selected *minsup*, another (more) lengthy mining process must be performed to generate a new ISL for the new query. Moreover, as described in [11], the ISM might encounter memory problem if the number of the potentially frequent patterns is too large. Without any assumption on the *minsup* value and on the required memory, the proposed algorithm speeds up interactive sequence discovery by using the acquired information with optimizations like direct candidate-generation and concurrent counting.

四、結果與討論

Fig. 1 outlines the KISP algorithm. We further optimized KISP by Theorem 1, which is used to generate the new-candidates in pass \(k\) (denoted by \(X_k')\) directly. **Theorem 1.** \(X_k' = (S_{k-1}[KB.base] \otimes N_{k-1}[\text{minsup}]) \cup (N_{k-1}[\text{minsup}] \setminus S_{k-1})\)
may efficiently discover patterns in large databases with KB. The execution time of KISP increases linearly as the database size increases.

五、計畫成果自評

The problem of interactive sequence mining is extensively studied in the project. The result is a satisfactory accomplishment, the KISP algorithm. The comprehensive experiments also show that the proposed algorithm outperforms current state-of-the-art algorithm and can be used to improve the mining efficiency of interactive sequence mining. We summarize this project in a paper, which is accepted in the HICSS-36 conference. This also confirms the project is successful.

六、參考文獻

Algorithm KISP \((DB, KB, \text{mins}up)\)

**Input:** \(DB\) = the database of data sequences; \(\text{minsup}\) = user specified minimum support; \(KB\) = knowledge base having the supports of all the candidates in prior minings

**Output:** \(S[\text{minsup}]\) = sequential patterns with respect to \(\text{minsup}\); \(KB\) = (new) knowledge base

---

1. \(if\) \(KB = \emptyset\) \(then\) \(KB = \{x \text{ and } x.sup, \forall x \in X_1\}\);
2. \(S[\text{minsup}] = \{x | x \in KB \land x.sup \geq \text{minsup}\}\); // obtain valid sequential patterns from knowledge base
3. \(if\) \(\text{minsup} < KB.base\) \(then\) // mine new patterns and accumulate new knowledge
4. \(k = 2\);  
5. \(generate X_k[\text{minsup}]\) from the frequent \((k-1)\)-sequences in \(S[\text{minsup}]\);
6. \(X_k' = X_k[\text{minsup}] - \{x | x \in KB\}\); // eliminate those candidate \(k\)-sequences in \(KB\)
7. \(while\) \(X_k' \neq \emptyset\) \(do\) // there exist candidate \(k\)-sequences, obtains their supports
8. \(forall\) data sequences \(ds\) in database \(DB\) \(do\)
9. \(increase\ the\ support\ of\ \(x\)\ if\ \(x\)\ is\ contained\ in\ \(ds\);\)
10. \(endfor\)
11. \(endfor\)
12. \(KB = KB \cup \{x \text{ and } x.sup, \forall x \in X_k'\}\); // collect new candidates and their supports
13. \(S[\text{minsup}] = S[\text{minsup}] \cup \{x | x.sup \geq \text{minsup} \land x \in X_k'\}\); // collect new patterns from \(X_k'\)
14. \(k = k+1\);  
15. \(generate X_k[\text{minsup}]\) from the frequent \((k-1)\)-sequences in \(S[\text{minsup}]\);
16. \(X_k' = X_k[\text{minsup}] - \{x | x \in KB\}\); // the reduced set eliminates candidate \(k\)-sequences in \(KB\)
17. \(endwhile\)
18. \(KB.base = \text{minsup}\); // update the counting base of \(KB\)
19. \(endif\)

---

**Fig. 1. Algorithm KISP**