行政院國家科學委員會專題研究計畫成果報告

網頁日誌探勘在改進網頁瀏覽效率上之研究

A Study on Web Log Mining for Improving Web Navigation

計畫編號：89-2218-E-009-016-

國立交通大學資訊科學系及研究所

計畫主持人： 楊維邦

摘要

資料探勘最常用於商店交易資料庫和醫院中病患就診資料庫，這一類的資料庫有一個共通特性，即每個交易記錄裏所包含的項目沒有先後順序。然而在應用環境中如能考慮項目間的順序，可得到許多隱藏的重要訊息。本計劃根據相關規則及其延伸問題定義新的探勘模式(mining pattern)，即參考序列，並定義一個在有序交易紀錄環境的探勘問題--大參考序列探勘，以提出一個利用反向索引技術的快速探勘大參考序列演算法。最後討論此探勘模式在全球資訊網的應用。

Abstract

Data mining is mostly used in retail databases and medical record databases. In this kind of database, the transactions are not sequential. However, if considering the order in practical environment, we can get some information that is unobvious but important. This project is based on association rule mining and its extended problems to define the new mining pattern, also called reference sequence. Then we define large reference sequence mining problem, a mining issue in sequential transaction records. Moreover, we propose an efficient mining algorithm with inverted index technique, and its applications on WWW are also discussed.

第一章　　簡介

資料探勘最常用於商店交易資料庫和醫院中病患就診資料庫，這一類的資料庫有一個共通特性，即每個交易記錄(在醫院中，即代表每個病人每一次就診的症狀)裏所包含的項目沒有先後順序。然而在應用環境中如能考慮項目間的順序，可得到許多隱藏的重要訊息。例如：在全球資訊網(WWW)的環境裏，瀏覽者在進入此網站時所觀看的內容有先後的分別；在就診資料庫考慮病人症狀出現順序；同一顧客在商店多次不同交易之間購買物品的順序。

近年來，全球資訊網的發展迅速，資訊傳播已不是傳統的平面媒體的特權，人們已經漸漸習慣於在網頁上查資料，尋找自己想要知道的資訊，訂購所需的用品等。面對琳瑯滿目的媒體資訊站，使用者如果對某網站不是很熟悉常不知如何以最有效的方式去利用，該往哪兒連結，該在哪兒停留，就像到大的購物中心霧裏看花一般，我們稱此現象為網頁瀏覽問題(navigation problem)。

茲舉出一些使用者常遇到網頁瀏覽問題：(1)網頁裏的連結數量過大，要選擇下一步的方向非常難下決定。(2)使用者事先已有預設想找的目標，但因無直接之連結可以到達，需要一番努力才能找的到通往路徑。(3)管理者設計此網頁的頁面分類方法和資料連結跟使用者的想法相違背，使用者需要去學習如何的適應這個網站的方式。(4)當網站長期經營，必然會走向資料越來越多的傾向，這時候有效率的管理是非常重要的，如果站太大有沒有適當的處理，那麼使用者經常會在裏頭迷失方向。由以上的分析可知，這些問題幾乎都是管理人員忽略一個很大的因素，亦即使用者的想法、行為和習慣。如果能知道使用者瀏覽網頁的行為傾向，這些問題將不復存在；再者，不同使用者有不同的瀏覽習性，即使相同的使用者隨著時間與環境的變遷其行為反應亦可能有所改變，導致於傳統網頁靜態式的架構無法動態地滿足使用者的瀏覽需求。

第二章　　研究方法及進行步驟

本計畫分為以下步驟來進行：

(1) 參考序列的新命題

(2) 探勘演算法 DDT

(3) 演算法效能評估

(4) 在網頁上的應用

以下就各研究步驟分別做詳細說明：

一、 參考序列的新命題

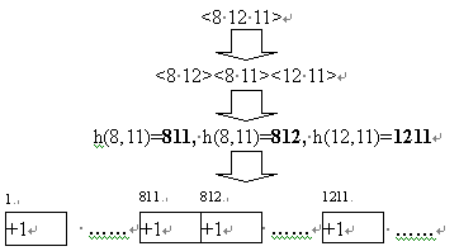
相對於大項目集而言，在此問題我們把大於 minsup 的參考序列稱作大參考序列，長度為 k 的參考序列稱之 k-參考序列，同樣的大於 minsup 的 k-參考序列稱之大 k-參考序列。所有的大 k-參考序列的集合為 L_k，所有可能為大 k-參考序列的候選序列所成的集合為 C_k。

同樣的也把問題分成兩個子問題，提出演算法針對第一子問題，亦即如何在 D 快速的尋找任何大於 minsup 的大參考序列。

大參考序列探勘可以拿大項目集探勘的演算法(見第二章第二節)將其稍做更改就可以達到探勘的目的。我們將 Apriori，和 DHP 裏的項目集定義和運算改成參考序列來實作，並設計一個新的演算法 DDT 與之比較。

二、 探勘演算法 DDT

圖四列出 DDT 的主程式。DDT 有一個雜湊表 H，每個籃子(bucket)是一個計數，DDT 在第一次掃瞄資料庫時對所有交易的 2-子序列的雜湊值所對應到雜湊表的籃子最累計。我們所設計的雜湊函數在不同 2-序列的雜湊值也會不相同，是一個沒有碰撞(no-collision)的雜湊。此雜湊函數為項目個數乘上第一個項目的編號再加上第二個項目編號。例：如圖一所示，假設總共有 100 個項目，所以雜湊函數 h(i,j)為 i*100+j 在交易序列<8 12 11>共有 2-子序列有<8 12>、<8 11>和<12 11>，則此三做雜湊函數 h(8,12)、h(8,11)和 h(12,11)得到的值分別為 812、811 和 1211，所以將雜湊表 H[812]、H[811]和 H[1211]值各加一。



圖一：DDT 雜湊概念

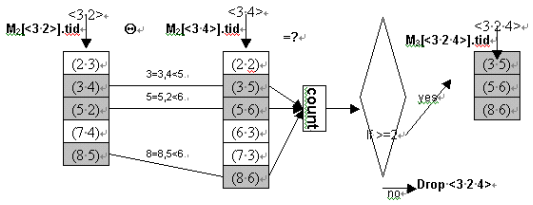
雜湊表中每個籃子代表特定一 2-序列，所以檢查籃子是否大於所指定的最小支持度即可判斷此籃子所對應的 2-序列是否為一大 2-參考序列。透過檢查所有籃子裏的計數可以找出所有大 2-參考序列。

有了所有大 2-參考序列，在進行第二次資料庫搜尋時，記錄每個大 2-參考序列所出現在的交易記錄的識別碼，每個大 2-參考序列會有一個由交易識別碼所組成的串列，整個集合我們叫做(M_2)。原本資料庫是交易序列所形成集合，以交易識別碼做為索引，而 M_2 是識別碼串列所成的集合，以大 2-序列做索引，所以我們稱之為倒置資料庫(inverted database)，gen_inverted 副函式輸入 H 雜湊表，建構 M_2 倒置資料庫，如圖五所示。每個串列除了記錄所出現的交易記錄識別碼外還記錄此大 2–參考序列的第二個元素在此交易記錄出現的位置。圖二為 M_2 的一個例子，以序列<1 2>為例，分別出現在交易記錄 1、交易記錄 3 和交易記錄 5，2 項目分別在第 2、第 3 和第 4 位置出現。

L2	交易識別碼串列
<1 2>	(tran1,pos2)(tran3,pos3)(tran5,pos4)
<2 1>	(tran2,pos3)(tran4,pos5)
<2 3>	(tran2,pos4)(tran3,pos4)
<3 2>	(tran4,pos4)(tran5,pos4)
<3 4>	(tran1,pos4)(tran2,pos5)

圖二：倒置資料庫 M_2

由倒置資料庫 M_2 可直接判斷得出 L_3 並產生以 L_3 為索引的倒置資料庫 M_3，M_3 又可以產生 L_4 和 M_4，一直反覆到$|L_k|$(或$|M_k|$)=0。圖六的 gen_next 輸入 M_k 產生 M_{k+1}。連結 L_k*L_k 產生 C_{k+1}，每產生一個 $c \in C_{k+1}$，就直接判斷 c 是否為大(large)。假設 $c=<a_1 \ldots a_k\ b_k>$，$a,b \in L_k$，如果 a 和 b 的交易識別碼串列裏包含相同的交易識別碼且 a_k 和 b_k 出現的位置為 $a_k<b_k$ 的個數大於最小支持度，則 c 為一大 k+1-參考序列，將之加入 M_{k+1} 並將 a 和 b 一同出現的交易識別碼當作其交易識別碼串列裏的元素，c_{k+1} 的位置則採用 b_k，gen_next 裏的'n'運算即指此由 a 和 b 的交易識別碼串列產生 c 的交易識別碼串列。圖三是由<3 2> <3 4>大 2-參考序列的交易識別串列判斷<3 2 4>是否為大參考序列，並形成其交易識別碼串列。可清楚見到<3 2>和<3 4>同時在交易 3、交易 5 和交易 8 出現，而且 2 在 4 的前面出現(4<5，2<6，5<6)，所以<3 2 4>一共在交易 3、交易 5 和交易 8 三個交易記錄出現，大於最小支持度 2，則<3 2 4>是一個大參考序列，它的交易識別碼序列為(3 5)(5 6)(8 6)。



圖三：大參考序列判斷(Θ運算)

```
 1)s = a minimum support;
 2)n = number of items;
 3)h(i,j)=n*i+j;
 4)set all H[i]=0, i=0~n*n;//將雜湊表籃子設為 0
 5)forall transaction t∈D do
 6)     forall 2-subset x=x_i,x_j of t do
 7)          H[h(x_i,x_j)]++; //對二子序列的雜湊值做計數
 8)M_2 = gen_inverted(H);//產生 M_2 倒置資料庫
 9)k = 2;
10)while(|M_k|>0){
11)L_k = {x|M_k[x] exists };
12)M_k+1 = gen_next(Mk);//找出 Lk+1 並產生 Mk+1
13)k++;
14)}
15)Answer = ∪k Lk;
```

圖四：DDT **主程式**

```
Procedure gen_inverted(H)
  forall i,j<n do
    If (H(h(i,j))>=support) then do
      Insert <i j> into L2
  forall transaction t∈D do
    forall 2-subset x of t do
      If (x∈L2) then do
        insert (t.id and position of x2 in t) into
          M2[x].tid //將交易序列的識別碼
    return M2;              //和 x2 在此交易記錄出
    end Procedure                  //現的最後位置記錄
    下
```

圖五：gen_inverted **副函式**

```
    Procedure gen_next(Mk)
      forall z=<x1…xk-1 yk>, x,y∈Lk,
                x1=y1,…xk-1=yk-1,xk≠ykdo
        if (|Mk[x].tid ΘMk[y].tid|>=isupport) then
          Mk+1[z].tid= Mk[x].tid ΘMk[y].tid
          //Θ運算見前一頁
        Return Mk+1
     end Procedure
```

圖六：gen_next **副函式**

三、 演算法效能評估

在這一章中，我們根據[Agrawal94]設計不同的測試數據來評估 DDT 的表現。所使用的平台是 SunOS 5.6，Ultra sparc 模組 60 的 CPU。針對 DDT、T_DHP 和 T_Apriori 做效能分析和比較。其中 T_DHP 為改良 DHP[Park97]的演算法，加入項目集的順序，變更包含的定義使符合為大參考序列探勘，T_Apriori 則是改良 Apriori[Agrawal94]演算法成為大參考序列探勘的演算法。

實驗一：常態資料之效率測量

目的為在一般的參數設定狀態下瞭解此三個演算法在於效率上的優劣，所使用的參數均符合於一般狀態 。

在探勘過程中，大部分的計算時間消耗在交易序列比對候選集上。結果顯示出 T_DHP 遠比 T_Apriori 節省計算時間。而 DDT 的計算量遠小於 T_Apriori，並在任何支持度下也都優於 T_DHP 的表現。

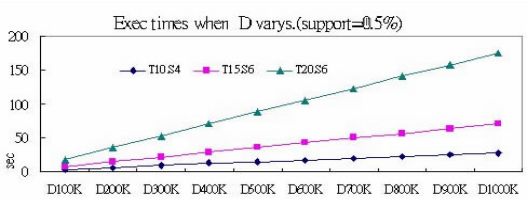
除了處理器時間外，磁碟讀取與輸出的多寡也影響整個效能。T_Apriori 在掃瞄資料庫的次數和大小在三者中屬於最多，T_DHP 因有適時的刪除資料庫中下次掃瞄不必要之項目所以磁碟的輸出入少了許多。但 DDT 最多只需讀取資料庫兩次，在三者中耗費最少時間在磁碟動作中。

在處理器時間上：T_Apriori<T_DHP<DDT，而在磁碟輸出入：T_Apriori<T_DHP<DDT，所以可預期總探勘所需的執行時間也應該和兩者相同，實驗結果說明了此一事實，T_DHP 總執行時間為 T_Apriori 1/2 到 1/4 不等，而 T_Apriori 的 DDT 總執行時間在 T_DHP 1/2 到 8/9 不等。

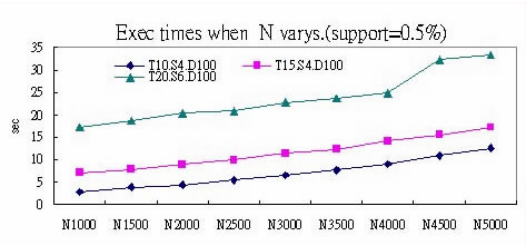
實驗二：非常態線性比例增加資料之效率測量

對某任一變動參數做 DDT 執行時間的記錄。所變動的參數有|D|，|N|。測量 DDT 遇到跟非常態資料(某一變數可能特別突出)時的效率表現。

在交易個數成比例增加，在 DDT 中因為僅需兩次掃瞄交易序列資料，因此，磁碟輸出入所需的時間的成相同的比例增加。所以整個的效能決定在處理器時間增加比例之優劣。由結果知，處理器時間所增加的比例大約就等於|D|所增加的比例，所以整個探勘所需的時間與|D|正好線性比例增加。



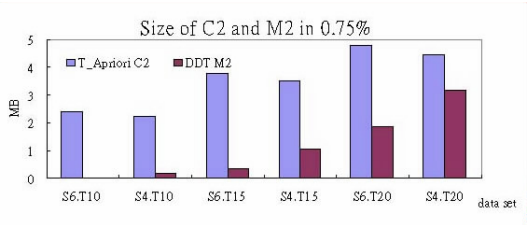
|N|增加所代表的意義為：項目的種類增加會使探勘結果減少，因為相同的交易記錄長度所分配相同項目的機率減少了。所以除了第一次搜尋資料庫(|C_2|變大)所耗費的時間會增加，其餘次所需時間會減少。DDT 在|N|變動的測試會隨著|N|增加而少許增加執行時間大約是 0.8|ΔN|，曲線緩慢上升。原因來自於 DDT 探勘所消耗的時間大部分是來自第一次掃瞄交易資料庫做雜湊以及尋找出 L_2。



實驗三：DDT 的 M_2 與 T_Apriori 的 C_2 記憶體比較

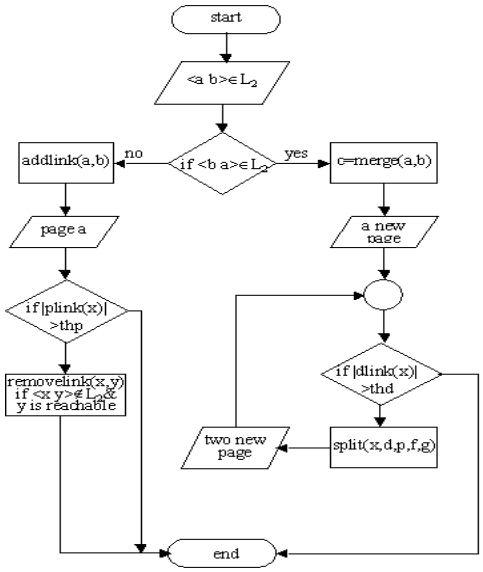
對各個不同的測試資料組測量 T_Apriori 之 C_2 和 DDT 之 M_2 所佔用記憶體的紀錄。我們將支持度固定在 0.75 和 0.5 百分比，測試資料參數設定範圍為 |D|={100K}、|S|={4，6}、|T|={10，15，20}、N={1000} 和|L|={2000}。對於兩者記憶體資源耗損做個比較。

DDT 將所有探勘所需的資訊在第二次掃瞄時換成 M_2，令人擔心記憶體用量過大，為消除此疑慮，設計此實驗。實際上，M_2 記憶體使用過多的問題是出現在所設定的最小支持度不恰當，倒致探勘出太多規則，過多的規則對於分析是無效的，所以反過來說，如將探勘出的規則控制在一定的數量下，則 M_2 將永遠不會太大。



四、 在網頁上的應用

本計劃提出一個簡單調整網頁的流程，讀者亦可配合自行解釋規則和運算之間的關係來設計自己的調整方法。所有屬於 L_2 的序列<a b>即代表觀看網頁 a 的使用者有很高的頻率會去觀看 b，如果<b a>不屬於 L_2 就 addlink(a,b)，如果<b a>屬於 L_2，代表了網頁 a 和網頁 b 常會被一起瀏覽，則施行 merge(a,b)。此方法以 addlink 和 merge 為優先施行的運算，直到一個網頁的對外連結超過超過值 thp，則施行 removelink，將所有不存在於 L_2 的連結刪除，若網頁裏的資料集合超過值 thd，則施行 split。



第三章 完成工作項目及具體成果

計畫執行的成果如下：

定義新的探勘模式，即參考序列，並以參考序列定義一個在有序交易紀錄環境的探勘問題-大參考序列探勘，提出一個探勘演算法。大參考序列探勘遠比大項目集探勘[Agrawal94]和另一大參考序列探勘命題[Park97]耗費時間。

此探勘模式在全球資訊網可作為調整網頁配置及線上及時指引使用者的參考。本論文提出一個的網頁架構表示法及其如何在上面應用此探勘模式的概念及簡單的調整方法。

參考資料

•[Agrawal94] R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules. Proc. of the 20th VLDB Conference, 1994.

• [Park97] J. S. Park, M. S. Chen,and P. S. Yu. Using a Hash-Based Method with Transaction Trimming for Mining Association Rules. IEEE Trans. On Knowledge and Data Engineering, vol. 9, no. 5, 1997.