利用網站使用性探勘於網站人機界面改進之研究

許銀雄
銘傳大學資訊管理研究所
桃園縣龜山鄉德明路 5 號
hsong@mcu.edu.tw

周世俊
銘傳大學資訊管理研究所
桃園縣龜山鄉德明路 5 號
benchou@seed.net.tw

摘要

人機界面是網站成功最重要的因素之一。為了確保網站人機界面的可用性，必須透過可用性評估方法檢視網站。然而一般常用的檢視法與實驗法有不夠精確、耗時與成本高的缺點，不適合使用於變化快速的網站可用性評估，因此本論文發展了一個方法，利用資料探勘的技術，探勘網站記錄檔，自動找出使用者經常瀏覽的連續瀏覽路徑以及使用者在瀏覽路徑上各個網頁的瀏覽時間，並且設計了這個方法中所需要的兩個演算法。藉由這個方法所得到的資訊，可以幫助網站維護人員瞭解使用者使用網站的真實狀況以及是否在瀏覽過程中遭遇問題等，並進而改善其網站之人機界面。

關鍵字: 資料探勘、網站探勘、人機界面、可用性評估。

一、緒論

使用者使用網際網路的主要目的是以查詢資訊為主[4]，所以呈現現有資訊以符合使用者需求是網站最重要的目的之一，能夠讓使用者越快找到資訊的網站設計，就越能滿足使用者的需求。網站的人機界面必須符合可用性(usability)的原則，亦即令使用者感覺到容易使用、便於瀏覽、可以快速達成目的等等。透過可用性評估方法檢視網站，可以改善網站的設計與結構。然而因為網站使用者以及使用環境的多元，目前常用的評估方法仍只能找出部分的問題，要完全排除可用性的問題幾不可能[2]，且由於環境變化的迅速，資訊及使用者的需求一直在變化，要以傳統的人機界面評估方法來動態維繫這樣的需求會有點力有未逮。

網站伺服器記錄檔（web server log）記錄使用者存取網站的真實過程。資料探勘(data mining)技術則可以在大量資料中，發掘出有用的資訊，如果將資料探勘技術應用到網站伺服器記錄檔來進行分析，可以發掘出使用者使用這個網站的實使用情形，幫助網站管理人員瞭解網站的使用情形。

考量網站的特性以及評估成本，採用資料探勘技術的可用性評估方法，對於降低評估的成本與增進評估效益有正面的幫助。本論文的目的便是透過資料探勘技術的使用，為網站管理者從網站記錄檔中粹取使用者的使用情形做為評估網站可用性的資訊，包含從中找出連續瀏覽路徑以及瀏覽路徑的瀏覽時間，並找根據這些結果評估網站人機界面設計上的一些問題。

二、相關研究

網站使用性探勘（web usage mining）是針對伺服器記錄檔做探勘，了解使用者的行為[9]，可以利用這些資訊將網站重新結構化以求網站的使用能更有效率[15]。

目前資料探勘技術在網站上的應用現況有 Park 等人使用關聯樣式的方法來探勘使用

者瀏覽路徑樣式[14]。陳武宏等人使用順序樣式的方法探勘代理伺服器的記錄檔，實作新的搜尋引擎，協助使用者透過其瀏覽目的查詢便可以找到相關的網站[3]。另外，也有研究使用關聯樣式的技術，探勘常常被使用者一起讀取的網頁，再利用群集分析方法，群集未直接以超鏈結相連卻常常被使用者一起讀取的網頁，改良網站的架構，在特定位置顯示使用者感興趣的資料[1]。在網站評估方面，也有研究提出如何使用資料探勘技術，評估網站的設計，不但有效且能節省成本，其主要是以瀏覽路徑與路徑相關資訊判斷使用者的使用行為，提供給設計者改善網站的依據[15]。

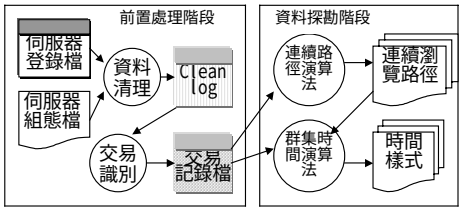
在探勘瀏覽路徑的演算法中，最常使用關聯樣式與順序樣式的概念。關聯樣式將同一個使用者瀏覽的網頁依照時間排序，視為是一個瀏覽路徑，使用關聯樣式的演算法將可以找出一起出現的網頁所構成的瀏覽路徑，也就是使用者瀏覽路徑樣式[14]。而將使用者瀏覽的網頁視為被依序購買的商品，便是使用順序樣式的演算法之概念[3][12]。

關聯樣式的演算法包括 Apriori[6]、DHP[13]、以及 Max-Miner[8]等。順序樣式的演算法有 Agrawal 等人的 *AprioriAll*、GSP 演算法[7]、SPADE 演算法[16]、SLP 演算法與 SSLP 演算法[5]。Han 等人有鑑於關聯樣式與順序樣式的探勘方式，皆遵循 Apriori 演算法的概念，對於探勘的效率並未能有跳躍式的改善，因此提出新的概念探勘瀏覽路徑，這種方法稱為 WAP 演算法 (Web Access Patterns)，其高度壓縮的樹狀結構 WAP-tree 與有效率的探勘方法 WAP-mine，有效減少掃描資料庫的次數，加速探勘的時間[12]。

群集探勘從過去的研究中大體上可以分類為 Partitional Clustering 與 Hierarchical Clustering。Han 等人的研究屬於前者，所有群集中的資料點，以群集中的一個中心點代表，目標是找出 k 個使規範函數最佳化的群集[11]。Zhang 等人的研究則是屬於後者，先群集數個子群集，合併兩個距離最近且距離不超過 ε 的子群集，直到群集數達到 k 個為止[17]。Guha 等人提出的 CURE（Clustering Using Representatives）演算法綜合以上兩種方式以中心點代表群集中的所有資料，加上不斷合併距離最近的群集，用以區隔出不同的群集。

三、探勘架構

本論文將分兩階段處理網站記錄檔，一為前置處理階段，另一個階段為資料探勘階段，如圖一所示。



圖一 網站使用性探勘架構圖

前置處理階段有二個步驟，分別為資料清理與交易識別。資料清理將讀取組態檔，做為清理記錄檔的依據。本論文所需的資料為使用者的網路位址 (IP address)，如：「192.72.211.121」、使用者提出要求的時間，如：「01/Jun/2001:18:30:50」與要求的網頁網址，如：「/Mysql/Mysql.htm」。一些重複或不需要的資料，將會刪除。所需的資料將置放於 Clean Log 檔案中。Clean Log 中的一筆記錄可以描述為(C,T,P)，C 為使用者的網路位址，T 為時間點，照時間順序排列，P 為網頁網址。

交易識別的任務在於將 Clean Log 中的記錄，集合成一個個使用者某一次瀏覽的網頁集合，亦即一個個的交易。本論文使用 User Session 的概念，視使用者每一次的瀏覽行為為一個 User Session，並利用 timeout 的機制，區分同一使用者的多次瀏覽行為，同一使用者在不同時段的瀏覽行為將視為不同的 User

Session。其格式可以表示為「C_i （T_1,P_1）（T_2,P_2）…（T_n,P_n）」。使用者瀏覽路徑將記錄在交易記錄檔中，做為探勘連續瀏覽路徑與群集時間樣式的輸入資料。在資料探勘階段將提出連續瀏覽路徑演算法與群集時間演算法。找出使用者的連續瀏覽路徑樣式，以及使用者花費在此路徑的時間樣式。

四、連續瀏覽路徑樣式演算法

瀏覽路徑集合使用者在一次使用期間所依序瀏覽過的網頁，可以表示為 S = $p_1 p_2 \ldots p_n$，p_i 代表某網頁，$p_i \in R$，R 為網站中所有網頁的集合，而且 $1 < i < n$；S 的長度即為網頁的個數 n；因為使用者在一次瀏覽的過程中，可能會重複瀏覽同一網頁，因此瀏覽路徑中可以包含重複的網頁，例如{a, b, a, c}代表瀏覽路徑，其中即包含了重複的網頁 a。

當兩個瀏覽路徑 S = $p_1 p_2 \ldots p_n$ 與 S'=$p'_1 p'_2 \ldots p'_l$ 具有下列情形時，我們稱 *S'*包含於 *S*，表示為 $S' \subseteq S$，亦即 $p_{ij}=p'_j$，$1 < i_1 < i_2 < \ldots < i_l < n$ 與 $1 \le j \le l$，亦可稱 *S'*為 *S* 的子路徑(sub-path)。此外，*S'*所包含的網頁在 *S* 中為連續者，稱 *S'*為 *S* 的連續瀏覽子路徑，例如 S = $p_1 p_2 \ldots p_n$，若 $S'=p_1 p_2 p_3$ 則 *S'*即為 S 的連續瀏覽子路徑。

所有瀏覽路徑的集合，稱為交易記錄檔 *TD*（Transaction Database），以 $TD = \{S_1 S_2 \ldots S_m\}$ 表示，$1 \le i \le m$，S_i 為一個瀏覽路徑。

*S'*的支持度（Support）為：*S'*在 S_i 中出現次數的總合除以交易總數，其中 $1 < i < m$、$S_i \in TD$ 且 $TD = \{S_1 S_2 \ldots S_m\}$， *S'* 若重複出現於 S_i，則支持度也將予以累計。在探勘的過程中，會有一個支持度的門檻限制，稱為最小支持度（Minimun Support），亦即確保瀏覽路徑的出現次數在交易記錄檔中佔有相當的比例。當網頁或瀏覽路徑的支持度超過最小支持度限制時，我們稱這些網頁為頻繁網頁(Frequent Page)，瀏覽路徑 *S'* 的支持度 $Sup(S') \ge Min\_Sup$ 且為連續時，稱 *S'*為連續瀏覽路徑樣式。

表 1 交易記錄檔

交易代號	瀏覽路徑
100	abdac
200	bdacf
300	bdcfa
400	fbdacc
500	abdca
600	bbdaca

因此探勘連續瀏覽路徑的問題可以描述為：給定一個交易記錄檔與最小支持度限制，找出所有存在於交易記錄檔中所有符合最小支持度限制的連續瀏覽路徑樣式。

由於 WAP 演算法將交易記錄檔載入記憶體，以減少讀取檔案的次數之概念，與本論文的想法不謀而合，而且探勘的效率良好，因此本論文在探勘連續瀏覽路徑的演算法方面將運用到 WAP 演算法的概念，並加以改變為適合連續瀏覽路徑之探勘。

本演算法主要的特色在於使用一個高度壓縮的樹狀結構 UATree（User Access Tree）與使用條件式搜尋(Conditional Search)的方式探勘。只需存取登錄檔兩次，且不必經由組合的過程，即可找到所有長度的連續路徑樣式。同時，連續瀏覽路徑具有數個特性。

特性 1：若一支持度為 ε 的瀏覽路徑 S'，任何包含此路徑 S'的父路徑，其支持度不可能大於 ε。

以表 1 的交易記錄檔為例，若有一網頁的支持度只有 3，將不可能出現在支持度為 4 的連續瀏覽路徑樣式之內。如網頁 f，支持度為 3，則 f 將不會出現在任何連續瀏覽路徑樣式中。因此，本演算法會在第一次掃描交易記錄檔 TD 時，即刪除不符合支持度限制的網頁。表 2 為頻繁網頁集，TD 中的瀏覽路徑若包含未存在於頻繁網頁集的網頁，則將此瀏覽路徑切斷成為兩區段，結果如表 3。交易代號為 300

的瀏覽路徑 bdcfa，因為網頁 f 並非頻繁網頁，所以必須忽略網頁 f，區分 bdcfa 為 bdc 與 a；以及交易代號為 200 與 400 的網頁 f，將會直接刪除。

表 2 頻繁網頁集

網頁	a	b	c	d	最小支持度
支持度	9	7	7	6	4

特性二：C 為一符合最小支持度的連續瀏覽路徑樣式。若有一網頁 p，以 pC 形成的連續瀏覽路徑出現在 TD 中的次數大於最小支持度，則 pC 也是符合最小支持度的連續瀏覽路徑樣式。

以表 1 中的交易記錄檔為例，瀏覽路徑 ac 的支持度為 4，並符合最小支持度，所以 ac 為符合最小支持度的連續瀏覽路徑樣式；另外其中的網頁 d 與連續瀏覽路徑樣式 ac 排在一起出現在 TD 中總共有 4 次，因此瀏覽路徑 dac 也將是一個符合最小支持度的連續瀏覽路徑樣式。

表 3 修正後之交易記錄檔

交易代號	頻繁路徑
100	abdac
200	bdac
300	bdc, a
400	bdacc
500	abdca
600	bbdaca

本演算法的另一個特性即是將交易記錄檔中有用的資料壓縮到一個 UATree 中，UATree 合併相同的瀏覽路徑，有效地將交易記錄壓縮至記憶體中，樹中記載了使用者的瀏覽路徑與支持度。建立了 UATree 之後，透過 UATree 的走訪，可以計算瀏覽路徑的支持度，有效減少檔案的讀取次數。之後，根據特性二與 UATree，使用條件式搜尋的方式，遞迴地探勘 UATree，即可以找出所有長度的連續路徑樣式。

在說明條件式搜尋之前，需先了解前綴（Prefix）與後綴（Suffix）的定義。假設存在一個瀏覽路徑 $S = p_1 p_2 \ldots p_k p_{k+1} \ldots p_n$，瀏覽路徑 $S_{prefix} = p_1 p_2 \ldots p_k$ 與 $S_{suffix} = p_{k+1} \ldots p_n$ 是瀏覽路徑 S 的子路徑；若有一連續瀏覽路徑樣式 $C = p'_1 p'_2 \ldots p'_j$，是瀏覽路徑 $S_{suffix} = p_{k+1} \ldots p_n$ 的子路徑且 $p'_1 = p_{k+1}$，則 $S_{prefix} = p_1 p_2 \ldots p_k$ 稱為在瀏覽路徑 S 中連續瀏覽路徑樣式 C 的前綴。同理，連續瀏覽路徑樣式 C 為 $S_{prefix} = p_1 p_2 \ldots p_k$ 在瀏覽路徑 S 中的後綴。例如，連續瀏覽路徑樣式 ac 是瀏覽路徑 $S200_{suffix}$ = acf 的子路徑，而且瀏覽路徑 $S200_{suffix}$=acf 是瀏覽路徑 S200=bdacf 的子路徑，因此在瀏覽路徑 S200=bdacf 中，$S200_{prefix}$=bd 是連續瀏覽路徑樣式 ac 的前綴，而連續瀏覽路徑樣式 ac 則是 $S200_{prefix}$=bd 在瀏覽路徑 S200=bdacf 中的後綴。

而條件式搜尋是指，在搜尋的過程中以所有頻繁的瀏覽路徑中後綴相同的子路徑為搜尋對象。因為根據特性二，若後綴為連續瀏覽路徑樣式，則與其形成連續瀏覽路徑的任何前綴，支持度若大於最小支持度，必也是連續瀏覽路徑樣式。

因此，連續路徑樣式演算法，將依照下列步驟進行探勘；輸入的資料包括交易記錄檔與最小支持度 ε：

步驟 1：掃描交易記錄檔，得到頻繁網頁集。

步驟 2：掃描交易記錄檔，建立 UATree。

步驟 3：使用條件式搜尋，遞迴地探勘 UATree。

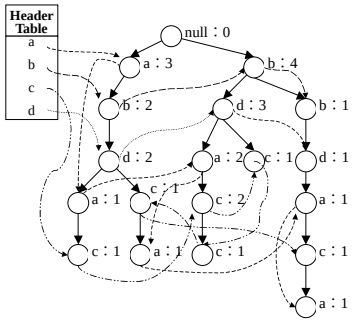
以下分別說明 UATree 的建立過程以及條件式搜尋的方法。

(一) UATree 的建立

UATree 的節點中存放的資訊包括標籤(Label)與支持度(Support)，標籤記錄網頁的名稱或代碼，支持度記錄著以此節點為後綴的瀏覽路徑之個數；根節點則是一個虛擬的節點，其標籤為空、支持度為零。

依序將所有交易記錄檔中的瀏覽路徑，插入至 UATree 中，若有相同的路徑則合併。假

設瀏覽路徑的第一個頻繁網頁 e，而根節點有一個子節點的標籤也為 e，則指標指到子節點 e，並將子節點 e 的支持度加 1；若根節點並未有一個子節點的標籤為 e，則產生新的子節點，標籤為 e，並將支持度設為 1。如此重複將一筆瀏覽路徑加入到 UATree 中，直到所有的瀏覽路徑皆插入至 UATree，即完成 UATree 的建構。若交易中的網頁並非頻繁的網頁，則指標即指向根節點，並略過此網頁。另外，所有存在於 UATree 中具有相同標籤的節點，會被置於一個佇列中，佇列的標籤與節點的標籤相同，如串聯網頁 e_i 的佇列稱為 e_i-queue，藉由這個佇列可以走訪樹中所有標籤相同的節點，找出所有以為 e_i 後綴的子路徑。而 e_i-queue 都存放在 Header Table 中。由表 3 的例子可以建立 UATree 如圖二所示。建立 UATree 的詳細演算法如圖三所表示。



圖二 完整的使用者存取資料樹

(二) 條件式搜尋

條件式搜尋在搜尋過程中，它利用一個稱為條件庫（Conditional Base）的資料結構來存放搜尋對象。一個後綴為 e_i 的條件庫表示為 CB | e_i，此條件庫是一些配對(p : s)的集合，配對中的 p 是具有相同後綴 e_i 的子路徑，而 s 則是其支持度。藉由 Header Table 中的 e_i-queue 之走訪，將可建立條件庫。以圖二為例，以 a 為後綴的條件庫 CB | a＝{(abd：1) (bd：2) (abdc：1) (bbd：1) (bbdac：1)}。另外，條件庫中子路徑的最後一個網頁節點，稱為樹葉節點（Leaf node），如子路徑(abd：1)的網頁節點 d 即為樹葉節點，樹葉節點的集合則稱為樹葉節點集（Leaf node pageset）。因為條件庫中的所有子路徑都有相同的後綴，所以它具有下面的特性。

```
Build UATree Algorithm
Input：交易記錄檔 TD，頻繁網頁集 FP
Output：UATree
Method：
Step1：
    Create a root node for UATree
Step2：
    For each transaction t in the TD do
      Let t=p1p2…..pn，where pi (1 ≤ i ≤ n ) are
                                        pages in t.
      Let current_node point to the root of UATree.
      for i = 1 to n do {
        if  pi ∉ FP {
          current_node point to the root
        }else{
          if  ( current_node has a child labeled pi ){
            increase the support of pi by 1.
            current_node point to pi.
          }else{
            create a new child node (pi：1).
            current_node point to the new node.
            insert pi into pi-queue.
          }
        }
      }
Step3：
    Return (UATree)
```

圖三 建立 UATree 的演算法。

特性三：條件庫的子路徑中，惟有是樹葉節點的網頁方可與後綴形成連續的路徑樣式。

根據特性三，在進行條件式搜尋時，只有樹葉節點才有可能與後綴具有連續的形式。如 abd 存在於後綴為 a 的條件庫中，唯有樹葉節點 d 可以與後綴 a 形成連續。而樹葉節點必須要大於最小支持度，才能稱為連續路徑樣式。因此條件庫還會有下面一個特性。

特性四：條件庫中的子路徑若要與後綴形成連續的路徑樣式，條件庫中子路徑的樹葉節點網頁，其數量必須超過最小支持度。數量超過最小支持度的樹葉節點，稱為頻繁樹葉節點（Frequent Leaf node）。當條件式搜尋在遞迴進行搜尋時的下一輪遞迴中，Header Table 將只需要存放頻繁的樹葉節點，也只需要走訪頻繁的樹葉節點之佇列即可。

而根據特性一，所有條件庫中的單一網頁也必須是條件庫中的頻繁網頁（Conditional Frequent Page），條件庫中頻繁網頁的集合則

稱為頻繁網頁集（Conditional Frequent Pageset）。因為，若條件庫中的某一路徑，包含了非頻繁網頁，則此路徑中非頻繁網頁之前的所有網頁與形成此條件庫的後綴，將會因非頻繁網頁的關係而無法形成連續，也因為如此，所以條件庫的另一個特性。

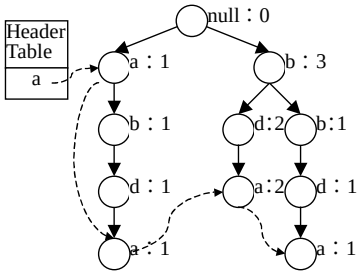
特性五：條件庫中某路徑若包含非頻繁網頁，則此路徑中非頻繁網頁之前的網頁皆須刪除，包括非頻繁網頁在內。例如 abda 是條件庫中的瀏覽路徑，若網頁 b 在條件庫中不是頻繁網頁，則路徑 abda 中，包含網頁 b 的前綴都必須刪除，只留下路徑 da 存在於條件庫中。

剔除了條件庫中不符合條件的子路徑之後，剩下的瀏覽路徑才具有繼續建樹遞迴的資格。以圖二中的 UATree 為例，在第一次建樹之後產生的 Header Table 中有(a, b, c, d)，則(a, b, c, d)即為用來進行遞迴探勘的後綴。利用 e_i-queue 走訪 UATree，為所有的後綴(a, b, c, d)建立條件庫，進行遞迴，直到所有的後綴(a, b, c, d)皆找到所有的連續瀏覽路徑樣式為止。

再以圖二的 Header Table 中 c 為例，走訪使用者存取資料樹的結果產生條件庫{(abda：1) (bda：2) (bd：1) (bdac：1) (abd：1) (bbda：1)}，根據特性四，條件庫中樹葉節點的個數{(a：4) (d：2)}，只有網頁 a 符合特性四。而根據特性五，條件庫中單一網頁的支持度為{(a：7) (b：8) (c：1) (d：7)}，所以網頁 c 並未超過最小支持度。因此，條件庫中樹葉節點不為 a 的路徑必須刪除，與包含網頁 c 的路徑中，網頁 c 與其前面的網頁也必須刪除。結果網頁 c 的條件庫{(abda：1) (bda：2) (bbda：1)}將做為繼續遞迴建樹與探勘的條件庫。

網頁 c 的條件庫可以表示為 *CB* | c，意指以 c 為後綴的條件庫。使用建立存取資料樹的方法將 *CB* | c 建立 c 的存取資料樹，如圖四所示。Header Table 中只存在符合資格的後綴。

經過不斷地遞迴，走訪 UATree，直到條件庫 *CB* | bdac ={ ϕ }，bdac 的條件庫為空集合為止，即可得到以網頁 c 為後綴的連續瀏覽路徑樣式{bdac}。連續瀏覽路徑樣式的詳細演算法如圖五所表示。



圖四 *CB* | c 的存取資料樹

```
Conditional Search Algorithm
Input：使用者存取資料樹 UATree，最小支持度 ε
Output：連續瀏覽路徑樣式的集合 UAP
Method：
Step1：
    Let UAP= φ，insert every pages in Header Table to
    UAP
Step2：
for each page p_i in Header Table
  Construct CB | p_i following the p_i-queue in UATree.
  Count and construct CFP (conditional frequent
                          pageset，根據特性五)
  Count and construct FLP (frequent leaf-node pageset.，
                          根據特性四)
  Adjust CB | p_i by CFP and FLP.
  if  (CB | p_i ≠ φ ) {
    Build UATree for CB | p_i using Build UATree
                    Algorithm(CB | p_i , CFP).
    Call conditional search
                Algorithm(UATree for CB | p_i , ε ).
  }
  Combine return pages with p_i and insert to UAP
Step3：
    return UAP.
```

圖五 條件式搜尋演算法

本論文採用與 WAP 演算法類似的精神，但因為目的的不同，而有下列幾點差異。首先為將交易記錄檔轉換為頻繁路徑記錄檔的轉換方式不同。本論文尋找的是連續瀏覽路徑，因此交易記錄檔中的單一網頁若小於最小支持度並不能直接於瀏覽路徑中刪除，而是將一條瀏覽路徑根據小於最小支持度的網頁分割為數條頻繁路徑；然而，WAP 演算法所尋找的是順序樣式，因此交易記錄檔中的單一網頁若小於最小支持度，即直接於瀏覽路徑中刪除即可。如瀏覽路徑 abcde，若網頁 c 小於最小支持度，本論文將其分割為兩條子路徑分別為 ab 與 de；而 WAP 的方式是直接刪除網頁 c。

其次是在條件庫的限制條件方面，本論文

有兩個限制條件，第一個是計算單一網頁在條件庫的支持度，第二個限制條件為：條件庫中的網頁 e_j，位於條件庫中最後一個網頁的數量必須超過最小支持度，唯有如此方能確保其為連續。而 WAP 只有一個限制條件，即條件庫中的網頁必須是條件庫中的頻繁網頁，其會刪除條件庫中的非頻繁網頁。

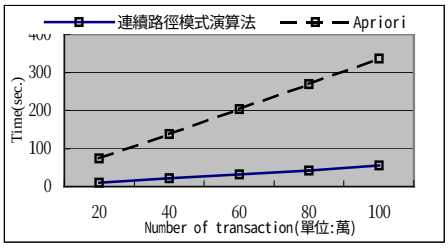
第三是在 Header Table 中的網頁方面，本論文只有將頻繁樹葉節點列入 Header Table 中；而 WAP 的方式是將條件庫中的頻繁網頁皆列入 Header Table 中。

最後是在遞迴停止的條件方面，WAP 的停止條件是當 WAP-tree 只有單一分枝時，即回傳分枝節點的組合，如單一分枝 ab，即回傳 {(a) (b) (ab)}。而本論文則是在條件庫為空集合時停止繼續遞迴，並傳回值。

(三) 連續瀏覽路徑樣式演算法實驗結果

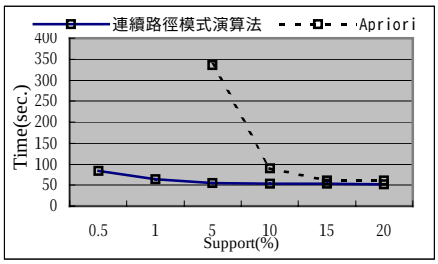
本論文以連續瀏覽路徑樣式演算法與 Apriori 演算法相互比較。並利用自動產生網站記錄檔的模擬程式產生記錄檔。在實驗環境的建構上，利用 Java 2 為主要的系統開發工具，而主要的研究測試平台為 Intel Pentium 4 1.2-GHz PC 及 256 megabytes 主記憶體，作業系統為 Microsoft Windows 2000。

第一部分的實驗目的在於兩個演算法，在交易量不同的情況下之執行效率，交易量分別為二十萬、四十萬、六十萬、八十萬與一百萬筆交易的網站登錄檔，最小支持度限制設定為為 5%。其結果如圖六所示。



圖六 多種交易量下演算法的執行速率圖

第二部份的實驗想要比較在不同的支持度限制之下，兩個演算法的執行效率，使用者每次平均瀏覽 10 張網頁，交易量為一百萬筆交易。圖七即為執行的結果。



圖七 不同支持度下之執行時間圖

五、時間樣式演算法

我們希望找出使用者在連續瀏覽路徑樣式中所花的平均瀏覽時間，然而由於不同類型的使用者其瀏覽的時間會很不一樣，所以本演算法的目的在於，對某一路徑樣式中的所有交易之瀏覽時間做群集，希望能得到某一路徑樣式中，各種類型使用者的平均瀏覽時間。

時間分為「要求時間」與「瀏覽時間」；要求時間代表使用者向網站伺服器提出要求的時間點；瀏覽時間則是使用者在某網頁停留的時間長度。兩個交易瀏覽相同網頁的瀏覽時間之差即為距離（distance），距離只針對相同網頁的瀏覽時間差距而言。而連續瀏覽路徑樣式由網頁所組成，網頁之間具有被連續瀏覽的特性，路徑樣式中的第 p 個網頁，稱為 p 網頁節點 (p page-node)。

為了不讓平均瀏覽時間因不同類型的使用者而產生誤差，因此需要將瀏覽時間相似的交易的劃分為相同的群組，以求取路徑樣式的平均瀏覽時間，其平均時間才不會有所偏差；而所謂瀏覽時間相似即表示交易之間的距離小於某個程度；每一個群組以一個平均瀏覽時間代表群組的中心，也代表某一類型使用者的平均瀏覽時間。

因此，探勘瀏覽時間樣式的問題在於，對路徑樣式、路徑樣式所包含的交易與每個交易的瀏覽時間，使用群集分析的技術，將每一個

網頁節點中的所有交易做群集的處理，求得群組之平均瀏覽時間，再根據群集的結果執行下一個網頁節點的群集分析，直到路徑樣式的所有網頁節點完成群集處理為止，最後即獲得此路徑樣式中各類型使用者的平均瀏覽時間，提供網站維護者改善網站人機界面的參考。

(一) 時間樣式演算法

CURE 演算法以中心點代表群組的做法，恰能解決本論文欲求取各類型使用者平均瀏覽時間的問題，因此本論文在求取時間樣式的演算法方面，將利用 CURE 演算法的概念與精神，發展適合探勘時間樣式的演算法。

時間樣式演算法的特色在於，對一路徑樣式的網頁節點分別做群集，並且依照上一個網頁節點群集的結果，再對群組內的交易做群集。每一個群組皆有一個平均時間做為代表點，做為群組在合併之時，合併公式的依據。群組之中尚包含了屬於此群組的交易，而交易的個數即為群組的支持度，支持度不足的群組，在下一網頁節點時即可不必再繼續群集。

時間樣式演算法首先根據路徑樣式演算法的結果，讀取交易記錄檔，將所有交易記錄檔中包含此路徑樣式的交易資料，經由瀏覽時間的計算之後，將交易代號與瀏覽時間，存放至路徑資料表中，做為時間樣式演算法的輸入資料。而路徑資料表如表 4，Path 代表路徑樣式，Time 代表瀏覽時間，而 transaction 則代表交易代號；每個路徑樣式將有一個屬於自己的路徑資料表。

表 4 路徑資料表

Path a→b→c

page / transaction	a	b	c
	Time（sec.）		
1	3	9	7
2	5	7	2
3	31	10	8
4	21	20	12
5	5	5	4
6	60	55	32
7	7	9	10
8	9	11	8

演算法由路徑樣式所建立的路徑資料表開始，分別群集各網頁節點的瀏覽時間，區分為數個群組，而在下一輪迴時，根據上一個網頁節點所分的群組繼續群集群組內的交易。如表 4 所示的路徑時間表，包含三個網頁節點 a, b, c，依照 a, b, c 的順序依序群集瀏覽時間。

路徑資料表中，一個網頁節點所包含的交易代號與瀏覽時間都會另外存放至另一個資料結構表格 T 中。透過表格 T，演算法可以知道某個交易所屬的群組，與群組的平均瀏覽時間，做為下一個網頁節點群集處理的依據。

時間樣式演算法從路徑樣式的第一個網頁節點開始做群集，直到最後一個網頁節點為止，針對每個節點，演算法分為五個主要的執行步驟：

步驟 1：依照路徑時間表建立表格 T。

步驟 2：依照表格 T 建立堆積樹 Q(heap-tree Q)，並依照距離升冪排序。

步驟 3：合併最近點、重新計算群組間的距離與調整堆積樹。

步驟 4：由堆積樹 Q 的根節點判斷群組是否需要繼續合併。

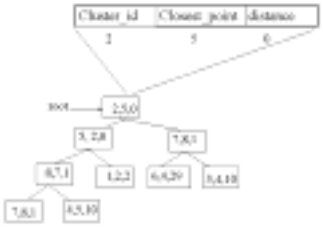
步驟 5：判斷是否為最後的網頁節點。

表格 T 包含三個欄位：Cluster_id、mean 與 transaction。初始時 Cluster_id 與 transaction 是相同的，經由演算法的處理合併之後，Cluster_id 會合併，而群組中所包含的交易數目會增加；mean 則代表群組的平均時間。如表 5 即為根據表 4 路徑時間表的 1 網頁節點 a 所建立的表格 T。

堆積樹如圖八所示，堆積樹 Q 的節點中，記載了群組代號（Cluster_id）、距離本群組最近的群組代號（Closest_point）與群組與最近群組之間的距離（distance）。群組代號即為表格 T 中的 cluster_id；最近的群組代號代表與此群組距離最近的群組之代號；而距離則為兩個距離最近的群組之間的距離。heap-tree 的節點則根據距離作升冪排序，距離最小的置於根節點。

表 5 表格 T

Cluster_id	mean	transaction
1	3	1
2	5	2
3	21	3
4	31	4
5	5	5
6	60	6
7	8	7
8	9	8



圖八 堆積樹

在步驟 3 方面，則從堆積樹的根節點開始進行合併，合併的方式為將根節點 u 與距離其最近的群組 v 合併，合併的公式為：

$$z.mean=\frac{|u.trans|*u.mean+|v.trans|*v.mean}{|u.trans|+|v.trans|}$$

意思為群組 u 的交易個數乘上其平均時間 mean 與群組 v 的交易個數乘上其平均時間 mean，相加之後，再除以 u 與 v 的交易個數總和，即為新群組 z 之平均時間 mean。重新修改表格 T 中的群集資料，刪除合併前的兩個舊的群集 u、v，新增一個合併後的新群集 z；並重新計算群組之間的距離與最近群組，調整堆積樹 Q。重新計算群組距離的方式為，先令新群組 z 的最近群組為任一群組 x；接著對所有存在於表格 T 的群組皆進行計算，若群組 y 與群組 z 的距離比群組 z 中所記載的最近群組短，則群組 z 中所記載的最近群組將改為群組 y；對群組 y 來說，若群組 y 的最近群組為 u 或 v，則需重新尋找最近群組，若否，則比較群組 z 是否比最近群組更接近，詳細的演算法可參考圖九。

在步驟 4 方面，則是判斷堆積樹 Q 的根節點之距離是否大於所設定的群集時間窗；如果根節點的距離小於群集時間窗，則回到步驟 3 繼續合併的處理；如果距離大於群集時間窗則進行步驟 5。

在步驟 5 方面，將群組中包含的交易數目小於最小支持度的群集予以刪除之後，判斷此網頁節點是否是路徑中的最後一個網頁節點；若否回到步驟 1 繼續下一網頁節點的時間樣式探勘；若是則完成此一路徑樣式的時間樣式之探勘。最小支持度的意思，即是確保群組之中包含的交易數量大於最小支持度。

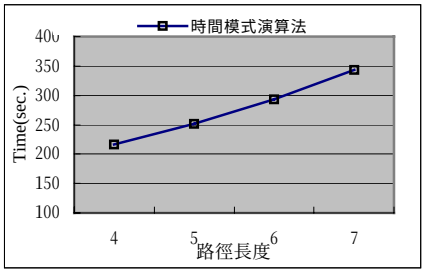
```
        Time Cluster Algorithm
input：路徑時間表PTT，距離時間窗CTW，最小支持
       度ε，網頁節點p
output：時間樣式
procedure cluster(PTT, CTW, ε ,p)
Step1：
Using PTT to build table T.（建立表格T）
Step2：
Using table T to build heap_Q. (建立堆積樹Q)
Step3 and Step4：
While the distance of root < CTW do {
    u is the root of heap_Q
    v is the closest cluster of u.
    Merge cluster u and cluster v.（合併兩最近點）
    Let the closest cluster of z equal to x
    for each x ∈ heap tree Q do {（重算群組距離）
      if distance(z,x) < distance(z, z.closest)
        z.closest := x
      if x.closest is either u or v {
        if distance(x, x.closest) < distance(x, z){
          Re-search the closest cluster of x
        }else{
          x.closest := z
        }
        relocate(Q,x)（調整堆積樹）
      }else
        if distance(x, x.closest) > distance(x, z) {
          x.closest := z
          relocate(Q,x)（調整堆積樹）
        }
      }
      insert(Q,z). （調整堆積樹）
}
Step5：
for all clusters in the p page-node{
                    （判斷群組是否需要繼續合併）
  if the amount of transactions in cluster < ε
    Delete this cluster.
  }
if the p page-node isn't the last one)
                    （判斷p是否為最後的網頁節點）
  Call Time Cluster Algorithm(PTT, CTW, ε , p+1).
```

圖九 時間樣式演算法

時間樣式演算法與 CURE 演算法的差異：首先在結束條件方面，本論文是依照群集時間窗的限制，決定是否需要繼續合併，惟有群組之間的距離尚小於群集時間窗時才需要繼續合併；而 CURE 演算法則是執行至參數 k

個群組產生之後，即不再合併。其次在群集執行次數不同的方面，時間樣式演算法需要群集所有路徑樣式所包含的網頁節點，並依照群集的結果，繼續下一個網頁節點的群集；而 CURE 只需執行一次群集的處理即可。最後的差異在於，時間樣式演算法對於群組內所包含的交易數量有支持度的限制，其群組支持度必須大於最小支持度才有繼續群集的必要；而 CURE 卻沒有此項限制。

(二) 時間樣式演算法實驗結果



圖十 網頁節點數與執行的時間圖

本論文測試路徑樣式的網頁節點數與執行時間的關係，交易量為一百萬筆交易，以及連續瀏覽路徑演算法在最小支持度為 1%之下所產生的某一路徑樣式；路徑樣式方面分別為具有四個網頁節點到具有七個網頁節點的路徑樣式，時間窗設定為 60 秒而最小支持度為 20%，實驗的結果如圖十所示。

六、資料探勘於網站人機界面的應用

藉由連續瀏覽路徑樣式與時間樣式這兩方面的資訊，結合網頁的屬性一起推論，便能從使用者的使用情形了解人機界面的設計問題。網頁的屬性可分成兩類：一類是內容頁（Content Page），另一種是瀏覽頁（Navigational Page）[10]。內容頁是網站設計者提供資訊的網頁，使用者通常會在這些網頁上停留較長的時間閱讀；而瀏覽頁則是網站設計者幫助使用者到達內容頁的導覽，使用者通常將這些網頁當成是到達內容頁的跳板，停留的時間會比較短。

使用者通常希望瀏覽頁能夠讓他們清楚的知道資訊所在的位置，以及內容頁的資訊能夠符合他們的興趣。因此，透過連續瀏覽路徑樣式與時間樣式的資訊結合網頁屬性共同推論，將可發掘使用情形異常的情況，進而發現設計問題，朝向容易使用的目標改進。

使用本評估方法評估網站可用性將比傳統的評估方法更貼近使用者，因為網站記錄檔是使用者真正的使用情況之記錄，以真正對網站有需求的使用者做為評估對象，評估的結果比較能夠發現真實使用者所遭遇的可用性問題。另外，相對於目前評估方法的耗時與高成本，本評估方法能有效節省時間與成本，以及面對時常更新的網站，本評估方法能隨著網站更新而動態地進行評估。

(一) 從連續瀏覽路徑樣式推論網站人機界面的問題

連續瀏覽路徑樣式代表有一定數量的使用者依循這個樣式瀏覽網站。如果探勘的結果出現連續瀏覽路徑樣式的長度比網站的深度短的情形時，依照路徑樣式出現的位置，又可分為三種情形。第一種情形：若只出現在網站用於導覽的部分，代表使用者對網站的路徑感到迷惑，而出現嘗試與猜測的行為，導致連續路徑樣式無法連接至內容頁，此時，即需檢視網站的導覽功能是否可以提供使用者正確的方向，以及內容是否做適當的分類，連結的說明是否容易了解。第二種情形：若只出現內容頁的部分，代表使用者到達內容頁的路徑不相同，導致無法產生連續路徑樣式，此時，應該針對內容頁設計最容易到達的路徑。第三種情形：若出現網站的前段與後段，卻不是在同一個連續瀏覽路徑樣式中出現，其意義代表使用者在導覽的過程中曾經迷失，才又找到資訊內容的所在，此時，網站的設計者應該檢視網站的某個網頁是否讓使用者迷惑。

網站設計者在設計網站之時，針對某個瀏覽目的或網站所欲提供的某項資訊，會有一個預設的瀏覽路徑，使用者遵循預設的瀏覽路徑，可以快速找到所需的資訊。若設計者預設的瀏覽路徑並沒有出現在連續路徑樣式中，則代表此預設的瀏覽路徑並無法為使用者所理解，此時設計者便需重新檢視其路徑的設計。

連續路徑樣式與順序路徑樣式都可以用於了解使用者的瀏覽路徑，而連續路徑樣式與順序路徑樣式之間的差異在於，順序瀏覽代表使用者在網頁節點之間或許曾經瀏覽過其他的網頁，而連續瀏覽則否。由上述的差異可知，若某路徑在順序路徑樣式中出現，卻未出現於連續路徑樣式中，代表使用者在順序網頁之間可能發生某些導覽上的困擾。

連續路徑樣式與順序路徑樣式的互相搭配，對於設計問題的改善也有幫助。找出兩個順序網頁節點之間的連續路徑，再加上支持度，便可以了解在順序網頁節點之間較受歡迎的瀏覽路徑，根據這些資訊可以重新修正順序網頁之間的路徑，使其符合使用者的認知。

相同的連續路徑樣式與順序路徑樣式，順序路徑樣式的支持度若高於連續路徑樣式，代表有一定比例的使用者想依照連續路徑樣式瀏覽，卻因某些原因，讓他們在中途瀏覽了其他的網頁，導致只在順序路徑樣式中出現而未在連續路徑樣式出現。原因可能在於這一部分的使用者是新的使用者，對網站架構並不熟悉，應當加強對初次到訪使用者的導覽設計。

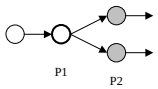
在探勘連續瀏覽路徑樣式時，必須注意最小支持度的設定，若最小支持度設定太高，則連續瀏覽路徑樣式可能無法產生。可以嘗試多種最小支持度的設定，並比較結果，將能發現更多設計上的問題。

(二) 從時間樣式推論網站人機界面的問題

藉由時間樣式與網頁屬性的搭配，網站設計者可以從三個方面的來推論，第一個方面為時間樣式所包含的群組數量，接著是平均的瀏覽時間，最後再從網頁的屬性來推論使用者的使用情形。

在時間樣式所包含的群組數量方面，首先，對只產生單一群組的時間樣式，搭配平均瀏覽時間，則有三種情形：第一種情形：路徑樣式中某一個網頁節點的平均時間明顯高於其他的網頁節點之平均時間。若此網頁屬於導覽頁，則便需要檢討網頁的設計是否太過於繁雜或對於連結的描述不清楚，令使用者無法快速找到資訊所在位置。第二種情形：路徑樣式中網頁節點的平均瀏覽時間都很短時，在內容資訊的提供方面顯示並不足以吸引使用者。第三種情形：路徑樣式中網頁節點的平均瀏覽時間都很長時，導覽頁在人機界面的設計便需要改善。

第二，若時間樣式在某個網頁節點產生分裂，如圖十一的情形，在網頁節點 P2 時分裂為兩種時間樣式，這代表了此連續路徑樣式有不同類型的使用者。有兩種情形產生：第一種情形：此分裂的網頁節點 P2 為內容頁時，平均瀏覽時間較短的群組，代表這一類型的使用者對此網頁節點缺乏興趣或此群組的使用者是曾閱讀過的使用者，可能是導覽不明確，讓使用者誤會網頁的內容；也可能是因為使用者是曾閱讀過，必須注意是否因為網頁內容太久未更新使舊有的使用者失去興趣所導致的情形。第二種情形：為導覽頁時，平均瀏覽時間長的群組，代表使用者不熟悉網站的設計，便需要加強導覽，幫助使用者熟悉網站的架構；而平均瀏覽時間短的群組，代表網站吸引使用者再度瀏覽，必須找出吸引使用者的關鍵。



圖十一 群組於某一網頁節點分裂示意圖

在應用時間樣式時應該注意是否因為時間窗的設定太高，以至於只產生單一群組或是設定太低而產生太多群組，混沌了真正的設計

問題。

七、結論

將資料探勘技術應用於網站記錄檔之分析，透過這些資訊的提供，可以幫助網站維護人員瞭解使用者使用網站的真實狀況、是否在瀏覽過程中遭遇問題等，並進而改善其網站之人機界面。對於未來的研究尚有需要改進的部分：多面向的探勘資訊：網站使用性的探勘除了路徑與時間，尚有許多不同的角度可以互相比較與參考，如何將更多角度的資訊納入網站可用性評估方法之中，是未來可以改善的部分。更多實際網站的探勘應用：如果可以將此系統應用到真實網站記錄檔，應可以對改善網站人機界面的效果更加瞭解。

八、參考文獻

[1] 翁頌舜，董惟鳳，「應用資料探勘方法建構適性化資訊網站」，第五屆資訊管理研究暨實務研討會，頁 476~483。

[2] 許銀雄，詹榮昌，「全球資訊網網站可用性評估之研究」，展望新世紀國際學術研討會，銘傳大學，資訊組論文集，民國 89 年，pp.389-402 。

[3] 陳武宏，「利用全球資訊網之使用者瀏覽行為探勘網頁流之技術探討」，國立清華大學碩士論文，民國八十八年。

[4] 資策會 FIND 網際網路情報中心，http://www.find.org.tw/

[5] 顏秀珍，左聰文，「從大型資料庫中挖掘序列型樣的有效率之演算法」，第十一屆全國資訊管理學術研討會，民國八十九年。

[6] R. Agrawal and R. Srikant, “Fast Algorithms for Mining Association Rules”, In Proceedings of the 20th International Conference on VLDB, Santiage, Chile, SEP, pp.487-499, 1994.

[7] R. Agrawal, and R. Srikant, “Mining Sequential Patterns：Generalizations and Performance Improvements”, In Proceedings of Fifth International Conference on Extending Database Technology（EDBT）, Avignon, France, March 1996.

[8] R. J. Bayardo Jr, “Efficiently Mining Long Patterns from Databases”, Proceedings of ACM SIGMOD international conference on Management of data , Seattle, WA USA, pp.85-93, 1998.

[9] R. Cooley, B. Mobasher, and J. Srivastava, “Web Mining: Information and Pattern Discovery on the World Wide Web”, Proceedings of the 9th International Conference on Tools with Artificial Intelligence (ICTAI '97), 1997.

[10] R. Cooley, B. Mobasher, and J. Srivastava, “Grouping Web Page References into Transactions for Mining World Wide Web Browsing Patterns”, proceeding of knowledge and data engineering exchange workgroup, pp.2-9, 1997.

[11] J. Han and T. N. Raymond, “ Efficient and effective clustering methods for spatial data mining”, Proceedings of the VLDB Conference, Santiago, Chile, pp.144-155, 1994.

[12] J. Han, J. Pei, B. Mortazavi-Asl, and H. Zhu, “ Mining Access Pattern efficiently from Web logs”, Proc. 2000 Pacific-Asia Conf. on Knowledge Discovery and Data Mining (PAKDD'00), Kyoto, Japan, 2000.

[13] 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.

[14] J. S. Park, M. S. Chen and P. S. Yu, “Efficient Data Mining for Path Traversal Patterns”, IEEE Transactions on Knowledge and Data Engineering Vol 10, No.2, pp.209-221 1998

[15] M. Spiliopoulou, “Web Usage Mining for Web Site Evaluation”, Communication of the ACM Vol.43, No.8, pp.127-134, 2000.

[16] M. J. Zaki, “Efficient Enumeration of Frequent Sequences”, CIKM, Bethesda MD USA, pp.68-75, 1998.

[17] T. Zhang, R. Ramakrishnan and M. Livny, “BIRCH：an efficient data clustering method for very large databases”, Proceedings of the ACM SIGMOD international conference on Management of data, Montreal Canada, pp.103-114, 1996.