網頁流探勘之技術、應用與系統實作

Web-flow Mining Techniques, Applications and System Implementations

陳武宏　　吳宜鴻　　陳良弼

國立清華大學資訊工程系

Email: alpchen@cs.nthu.edu.t

摘要

由於網頁的快速激增，使得如何在全球資訊網上快速而有效地取得使用者感興趣的資訊，成爲一個非常重要的研究課題。本論文主要研究是透過分析代理伺服器的記錄檔，蒐集使用者瀏覽網頁的行爲習性，並進一步分析這些蒐集到的瀏覽行爲，以便從中找出有意義且可用於導覽的網頁流。在本論文中，我們將提出一個循序樣式資料探勘的演算法，藉此推導出所有的網頁流，並顯示推導後的結果。此外，我們也設計了一套使用者的查詢介面，讓使用者可以透過輸入關鍵字下查詢以取得相關的導覽流程，或在瀏覽搜尋引擎的查詢結果時，適時地建議相關的網頁串列。

關鍵字：網頁流、資料探勘、全球資訊網、搜尋引擎、循序樣式。

1 簡介

由於全球資訊網（World Wide Web）逐漸地被廣泛的使用，愈來愈多人利用瀏覽器（browser）取得他們感興趣的資料。然而，網頁數量的持續激增，往往造成使用者必須花費大量時間尋找真正感興趣的資料。因此，如何提供一個方便而有效率的方法，讓使用者能夠快速的找到感興趣的資料，已經成爲當前一個重要的研究課題。

目前已經有許多解決方案被提出來，其中，*搜尋引擎*（search engine）是最普遍被採用的作法。但是，在這類系統中，當使用者給定一個關鍵字例如「玉山」，往往代表著多個意義如「玉山國家公園」或「玉山銀行」等。所以，搜尋引擎所回傳的結果，通常會包含大量與查詢目的不相關的網頁，使用者必須再額外花費一些時間才能找到真正感興趣的資料。截至目前爲止，在全球資訊網上較爲著名的搜尋引擎有 Alta Vista，Yahoo，Yam，Kimo，Openfind 等。

使用者上網的*瀏覽目的*（information goal），往往要藉由蒐集許多資料來達成；而這些資料通常分屬於不同的領域，我們稱之爲不同的*主題*（subject）。就傳統的搜尋引擎而言，使用者必須針對每個主題給予不同的查詢，並分別從查詢結果中擷取他們感興趣的資料，最後再予以整合。以「週末旅遊計劃」之瀏覽目的爲例，使用者可能需要當地旅遊資訊，交通與天氣等不同主題的資料，使用者便必須依不同主題下查詢，如關鍵字「天氣預報」、「鐵路局訂票系統」等，再從查詢結果中找出相關的資料。因此，完成包含多個主題的瀏覽目的，經常是相當費時的。有鑑於此，我們希望能夠利用一個或最少的查詢，便可以讓使用者得到所有與「週末旅遊計劃」相關的網頁資訊。

在另一方面我們也觀察到，在不同的主題之間，使用者經常會依照某種次序瀏覽或尋找網頁。以上述的旅遊爲例，可能會有下列的次序性：

當地旅遊資訊→天氣預報→交通工具→旅館

其含意正說明了使用者會先到氣象局的網站，查看當地的天氣情形，再根據天氣的好壞，自鐵路局網路訂票系統訂購火車票，然後才藉由當地旅館的網頁訂房。很明顯地，傳統的搜尋引擎並不能滿足一次查詢多個主題並且提供瀏覽次序之建議的需求。

針對上述需求，我們首先將一群具有瀏覽次序的網頁（或目錄樹的概念節點，我們將在第三節對此做更深入的說明）定義爲一個*網頁流*（Webflow）。接著，我們藉由分析*代理伺服器記錄檔*（proxy server log），推導有意義的網頁流，並提供使用者查詢介面，讓使用者透過關鍵字即可找到所有相關的網頁流。舉例來說，當使用者給定關鍵字「墾丁」，可能會得到如下的網頁流：

1. http://www.ktnp.gov.tw/ 歡迎蒞臨墾丁國家公園→
   http://www.cwb.gov.tw/ 中央氣象局全球資訊網→
   http://www.caesarpark.com.tw/caesar/kenting.ht
   墾丁凱薩飯店
   →http://www.railway.gov.tw/taiwan/tai17.html
   臺灣鐵路局--網路訂票
2. 墾丁國家公園→天氣預測→旅館→鐵路網路訂票

由上述可知，網頁流所構成的查詢結果，不僅提供了相同瀏覽目的但不同主題的網頁資訊，也更進一步提供了瀏覽次序的建議；如此一來，便可節省使用者在搜尋引擎中由於不同主題所導致大量的查詢時間。

本論文以下各章節說明如下，第二節我們對相關的研究進行討論，第三節描述基本觀念，第四節說明我們所提出探勘網頁流的演算法，第五、六節是實驗結果以及使用者查詢介面的實作。然後在最後第七節做個結論。

2 相關研究

首先提出*循序樣式探勘*（Sequential pattern mining）問題的是[2]，其定義是在*交易資料庫*（transaction database）中，如何找出大多數顧客購買商品的次序性。以錄影帶出租店爲例，大部分的顧客租了「倩女幽魂」錄影帶之

後，隔一段時間，會繼續租「倩女幽魂 2」，接下來則是「小倩」。隨之，相同作者繼續提出比先前演算法快上 20 倍的新演算法 GSP，並且加入了最大與最小*時間間隔*（gap）的限制、*滑動窗*（sliding window）與*目錄樹*（taxonomy）等因素。這些演算法皆使用較複雜的*雜湊樹*（hash tree）結構，其主要的探勘過程分爲兩部分，先產生*候選 k-循序樣式*（candidate k-sequential pattern），再重新讀取資料庫一次，以計算其相對應的*支持度*（support）。因此，如果資料量大或降低*最小支持度*（minimal support）的時候，由於讀取資料庫的量與次數增加，相對地執行時間也會大幅的增加。

另外，[6]利用*相關圖*（association graph）提出 DSG 演算法，做法是以*常見 1-循序樣式*（frequent 1-sequential pattern）建立圖中的節點，利用交集產生常見 2-循序樣式構成圖的邊，接著利用遊走相關圖產生所有的*常見循序樣式*（frequent sequential pattern）。雖然其作法只需要讀取資料庫一次，但是當資料庫的資料量相當大時，DSG 演算法需要的記憶體也相對的增大，這是需要解決的問題。

[8]在最近提出 SPAD 演算法，比[4]所提的 GSP 演算法大約快上兩倍。作者利用*等價類別*（equivalence class）的觀念，將原來的問題分解成可以個別獨立在記憶體中執行的子問題，另外再使用*框格*（lattice）結構以及簡單的*結合*（join）運算，來進行循序樣式的探勘。此外，所有的常見循序樣式只需要讀取資料庫三次即可求出，是目前已知文獻中最快的方法。

以上所有描述有關循序樣式探勘的演算法，除了 GSP 外，都只針對交易資料庫中的*項目集*（itemset）進行探勘，並沒有引入目錄樹的概念。傳統的*關聯法則*（association rule）探勘上，[1]首先觀察到並提出目錄樹的想法可以應用到探勘上。接著[3][7]便引用目錄樹的資訊進行探勘，做法是將最原始的資料提升到目錄樹中某個特別的*階層*（level）來產生關聯法則。本論文所採用的目錄樹是一個分類概念樹（如，人∈哺乳類∈動物的概念關係），使得我們在探勘時，需要往上進行一般化或往下進行特殊化的動作時，仍可保持其概念間的包含特性；另外，我們也允許資料可以存在於任何一個概念節點上。

3 基本觀念

本研究希望能從代理伺服器所儲存的瀏覽記錄，蒐集使用者瀏覽網頁的瀏覽軌跡，進一步分析這些蒐集到的瀏覽軌跡，從中找出有用的循序樣式，並計算相對應的支持度（亦即包含各循序樣式的瀏覽軌跡數目）；若大於最小支持度，我們便稱該循序樣式爲有意義的網頁導覽流程，簡稱網頁流。

本章節我們先針對原始資料做一個分析說明，藉以取得*瀏覽軌跡*（navigation path），接著討論*交叉層次循序樣式探勘*（Cross-level sequential pattern mining）時所需的*語意目錄樹*（semantic taxonom ）。

3.1 瀏覽軌跡

```
890984441.324 0 140.117.11.12 UDP_MISS/000 76
ICP_QUERY http://www.yam.org.tw/ b5/yam/...
```

上列爲代理伺服器記錄檔的記錄，包含來源機器的 IP 位址「140.117.11.12」、目的網頁的網址「http://www.yam.org.tw/b5/yam/」、資料處理的時間點「890984441（爲 UNIX 作業系統的時間標籤）」、與其他狀態值等。

假設同一個來源機器的 IP 位址代表同一個使用者，我們先從記錄檔擷取「來源機器的 IP 位址」、「資料處理的時間點」和「目的網頁的網址」，並根據不同使用者得到不同的*瀏覽序列*（browsing sequence）；接著我們再利用[5]所提出的*使用期間*（user session）概念，透過*逾時*（time-out）的機制區分同一個使用者的不同使用期間，也就是說同一位使用者在不超過逾時限制內所瀏覽的任兩張網頁才算是相關的。於是我們給定一最大逾時限制，藉此檢查每個瀏覽序列中任兩張相鄰網頁的瀏覽時間差；如果超過最大逾時限制的話，我們就從瀏覽序列中分割得到一個新的瀏覽軌跡，如此我們便可蒐集到所有的瀏覽軌跡，如表格 1所示。

Pid	Navigation Path
1	(A,9) (D,11) (F,24) (E,27) (S,41)
2	(D,10) (E,13) (S,18) (A,28)
3	(K,66) (F,72)
4	(D,15) (A,17) (E,22) (S,25)
5	(F,42) (A,46) (J,49) (S,57)

表格 1. 原始資料庫

3.2 語意目錄樹

由於全球資訊網上的網頁數目極爲龐大，假如直接以網址作爲資料探勘的資料，可能由於支持度太小的緣故，無法得到有意義的循序樣式。此外，我們也希望探勘結果是由*語意化概念*（semantic concept）所形成的網頁流。所以我們利用語意目錄樹進行交叉層次循序樣式探勘，來解決支持度不夠的問題，同時也藉此找出較具語意的網頁流。

舉例來說，我們希望得到如下較具語意的概念：

墾丁國家公園→天氣預測→旅館→鐵路網路訂票

而不只是單純的網址所形成的網頁流：

http://www.ktnp.gov.tw/→http://www.cwb.gov.tw/
→http://www.caesarpark.com.tw/caesar/kenting.htm
→http://www.railway.gov.tw/taiwan/tai17.html

在語意化網頁流的問題上，我們除了利用目錄樹進行交叉層次的循序樣式探勘之外，我們也透過網頁的標題，來輔助網頁流的語意描述。所以本研究所探討的網頁流除了包含單純網址外，還加上網頁標題以及目錄樹的概念節點。以上述單純網址所形成的網頁流爲例，其包含網頁標題所形成的網頁流如下：

http://www.ktnp.gov.tw/ 歡迎蒞臨墾丁國家公園
→http://www.cwb.gov.tw/ 中央氣象局全球資訊網
→http://www.caesarpark.com.tw/caesar/kenting.ht
墾丁凱薩飯店
→http://www.railway.gov.tw/taiwan/tai17.html
臺灣鐵路局--網路訂票

4 循序樣式探勘

這個章節我們將詳細地說明探勘網頁流的演算法，我們將此演算法命名爲 *SPII*（Sequential Patterns discovery using Inverted-list and Intersection operation），藉此演算

法推導所得的常見循序樣式即爲上述之網頁流。第一小節先說明儲存使用者瀏覽軌跡與常見循序樣式的*反轉表*（inverted list），第二小節介紹產生（k+1)-循序樣式與計算支持度的*交集運算*（intersection operation），接著第三小節才詳細描述 SPI 演算法產生網頁流的完整程序，第四小節則說明利用目錄樹進行交叉層次循序樣式探勘的作法。

4.1 反轉表

我們以網頁或已經探勘出的常見循序樣式爲鍵值，相對應於每個鍵值爲一連串成對的編號與時間；編號代表包含該網頁或常見循序樣式的瀏覽軌跡，而時間則是該網頁或常見循序樣式被瀏覽的最後時間。我們將此串列稱爲反轉表，藉此可建立一個*反轉資料庫*（inverted database）。將瀏覽軌跡的原始資料庫轉成反轉資料庫，爲 SPI I演算法的第一個步驟。

我們首先針對每個瀏覽軌跡編號 i，讀取原始資料庫中瀏覽軌跡的資料，對於軌跡中每個節點（u, t），以（i, t）插入於鍵值爲 u 的反轉表中。以表格 1的編號 1 軌跡爲例，我們先讀取的是（A, 9），於是插入（1, 9）於鍵值爲 A 的反轉表中；第二個節點爲（D, 11），因此插入（1, 11）於鍵值爲 D 的反轉表中，如此反覆，便可得到探勘時所需的反轉資料庫了。如表格 2即爲表格 1所轉成的反轉資料庫。

Key	Inverted List *(pid, tid)*
A	(1,9)(2,28)(4,17)(5,46)
D	(1,11)(2,10)(4,15)
E	(1,27)(2,13)(4,22)
F	(1,24)(3,72)(5,42)
J	(5,49)
K	(3,66)
S	(1,41)(2,18)(4,25)(5,57)

表格 2. 由表格 1所轉成的反轉資料庫

4.2 交集運算

以下分成兩個部分說明，第一部份描述由常見 1-循序樣式產生常見 2-循序樣式的步驟，第二部分則是由常見 k-循序樣式產生常見(k+1)-循序樣式的程序。

第一部份，假設有兩個常見 1-循序樣式分別爲 A 和 B，其反轉表分別爲一連串（p_i, t_a）與（p_j, t_b）的串列。我們先選出兩個反轉表中擁有相同的瀏覽軌跡編號（p_i=p_j）的節點，比較其瀏覽時間的前後順序，如果滿足 t_a<t_b的話，我們便產生新的 2-循序樣式「A→B」，並將（p_i, t_b）插入此樣式的反轉表中；反之，如果滿足 t_a>t_b的話，我們會產生新的 2-循序樣式「B→A」，同樣也將（p_i, t_a）插入反轉表中，如表格 3所示。

在本論文中，我們將 k-循序樣式扣除最後一個節點後所形成的*子樣式*（sub-pattern），稱爲原樣式的(k-1)-*前序子樣式*（prefix sub-pattern）。例如：節點 D 爲 1-循序樣式，而 D→E 是 2-循序樣式，則我們將節點 D 稱爲 D→E 的 1-前序子樣式；同理，D→E→H 是 3-循序樣式，D→E 便可稱爲 D→E→H 的 2-前序子樣式。

第二部份的做法是針對常見 k-循序樣式，先從反轉資料庫進行兩兩比對，找出擁有相同(k-1)-前序子樣式的某兩個樣式，再進行交集運算。假設我們找到兩個擁有相同(k-1)-前序子樣式 W 的常見 k-循序樣式分別爲 W→A 和 W→B，其中 A 和 B 是兩張不同的網頁，其對應的反轉表可以分別用一連串的（P_i, t_a）與（P_j, t_b）串列表示。則交集運算的進行步驟如下，其複雜度爲 $O(n^2)$：

if $\exists P_i \in W \rightarrow A$ and $P_j \in W \rightarrow B$ such that $P_i = P_j$ **then**

1. {t_a is the timestamp which P_i access the URL 'A' following (k-1)-prefix 'W'}
2. {t_b is the timestamp which P_j access the URL 'B' following (k-1)-prefix 'W'}

if $t_b > t_a$ **then**

1. Generate the (k+1)-sequential patterns W→A→B
2. Insert the (P_i, t_b) into the Inverted List of W→A→B in inverted database

if $t_a > t_b$ **then**

1. Generate the (k+1)-sequential patterns W→B→A
2. Insert the (P_i, t_a) into the Inverted List of W→B→A in inverted database

範例

假設有兩個常見 3-循序樣式分別爲「D→H→A」和「D→H→B」，其 2-前序子樣式皆爲「D→H」，且個別的反轉表分別爲「(1, 2) (6, 4) (9, 10 (10, 11 」和「(1, 3) (2, 5) (6, 7) (9, 8) (10, 14)」。

結果

交集後會產生兩個 4-循序樣式分別爲「D→H→A→B」與「D→H→B→A」，其對應的反轉表分別爲「(1, 3) (6, 7) (10, 14)」與「(9, 10 」。此外，我們也求出其*支持度計數*（support count）分別爲 3 和 1；假設最小支持度計數爲 3，則可得知「D→H→A→B」爲一個常見 4-循序樣式，而「D→H→B→A」則否。

4.3 SPII 演算法

有了反轉表與交集運算的概念之後，接著我們將於這一節，完整地描述 SPI 演算法。我們將演算法分爲三個步驟，首先在步驟一，我們先得到常見 1-循序樣式，接下來的步驟二取得常見 2-循序樣式，而最後步驟則是推導出所有的常見 k-循序樣式（k>2）。

步驟一：取得常見 1-循序樣式

首先我們將原始資料庫轉成反轉資料庫，這個步驟已經在 4.1 小節說明過。當我們一方面將瀏覽軌跡轉換成相對應網頁的反轉表時，我們也同時將每個網頁的支持度計數累加下來；直到轉換結束後，我們便可根據最小支持度，決定哪些網頁屬於常見 1-循序樣式，並儲存其反轉表以供下一個步驟使用。

以表格 1的瀏覽軌跡爲例，假設最小支持度計數爲 3，則最後得到的常見 1-循序樣式分別爲「A，D，E，F，S」，其對應的反轉表如表格 2所示。

步驟二：取得常見 2-循序樣式

在這個步驟中，我們直接將所有的常見 1-循序樣式兩兩做交集運算，以產生 2-循序樣式與對應的反轉表。如果

總共有 n 個常見 1-循序樣式，則必須進行 $C(n, 2)$次的交集運算，其中每個反轉表需要被讀取 $(n-1)/2$ 次；在另一方面，兩兩進行交集運算不需要大量的記憶體，且交集後可直接得到新的 2-循序樣式、對應的支持度計數、以及其反轉表。如表格 3即爲經過此步驟後所得到的新反轉表。

步驟三：取得常見 k-循序樣式

最後我們將所有的常見 k-循序樣式，經由 4.2 小節所介紹的交集運算，一一產生新的（k+1)-循序樣式、對應的支持度計數、以及其反轉表；同樣可依據最小支持度來決定哪些爲常見(k+1)-循序樣式。累增 k 值重覆此步驟，直到沒有新的常見循序樣式產生，整個演算法方可終止。

Key	Support Count	Inverted List *(pid, tid)*
D→E	3	(1,27)(2,13)(4,22)
D→S	3	(1,41)(2,18)(4,25)
E→S	3	(1,41)(2,18)(4,25)
A→S	3	(1,41)(4,25)(5,57)

表格 3. 常見 2-循序樣式

以表格 3爲例，最後推導出一個常見 3-循序樣式爲 D→E→S，其支持度計數爲 3，反轉表爲「(1,41)(2,18)(4,25)」。由於僅有一個常見循序樣式，無法再透過交集運算產生新的 4-循序樣式，所以整個演算過程便結束於此。更詳細的演算法可參閱[9]。

4.4 交叉層次循序樣式探勘

在交叉層次循序樣式探勘上，我們一樣是採用 SPI 演算法，但由於支持度會隨著如Figure 1的語意目錄樹從下往上累加，所以上層概念節點往往會有比較大的支持度；如此一來，雖然解決了支持度可能不夠的問題，但是卻又產生另一個問題：上層概念節點所代表的語意過於一*般化*（generalization）。

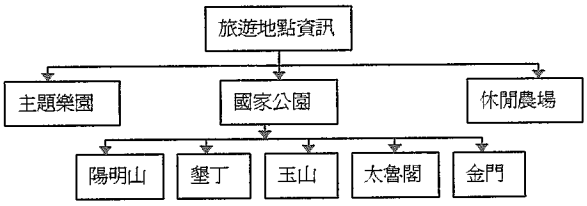

Figure 1. 語意目錄樹

爲了解決上述問題，我們再加入一項*最大支持度*（maximal support）的*臨界*值，藉此可將支持度過大的概念節點或循序樣式*略過*（pruning），以期找出更能發揮導覽作用的網頁流。除此之外，我們也避免產生節點間爲父子關係的網頁流，如Figure 1中的「國家公園」與「玉山」；當發生這類狀況時，由此所有可能衍生的網頁流都將被略過不予產生。

5 實驗

本研究的實驗平台如下：CPU 爲 Pentium 100，32M DRAM，作業系統爲 FreeBSD 2.8，資料庫採用 Mysql。首先，我們對代理伺服器記錄檔進行前置處理，主要有下列兩個動作：

1. 刪除多媒體檔案，如附屬檔名爲 gif, jpg, mov 檔案。
2. 以一小時爲最大間隔，將使用者的瀏覽序列，分割成數個瀏覽軌跡。

實驗中所採用的測試資料，爲取自電腦中心的實際存取記錄，總共一週的測試資料量，可得到 2409 條瀏覽軌跡與 614673 個網址。爲了加速實驗，所有的瀏覽軌跡與目的網址都已事先經過編碼的手續。

5.1 常見循序樣式

以下所列爲最小支持度設爲 24 時，所得到的幾種常見循序樣式。其中循序樣式的表示，以「站內互連」爲例，最上面的一行 「81806-81991-51070-81837 28」表示所得到的循序樣式爲「81806-81991-51070-81837」（每個數字代表某一張網頁的編號），且對應的支持度計數爲 28；緊接著的表格，則包括了每張網頁的編號、所代表的網址、以及網頁的標題。

站內互連：其瀏覽行爲皆在同一個網站內的網頁流

81806-81991-51070-81837 28

81806	Http://www.ftv.com.tw/channel/ftv.cdf	FTV CHANNE
81991	Http://www.ftv.com.tw/index.htm	民視新聞資訊網- 線上即時新聞
51070	Http://www.sosoon.com.tw/ftv.htm	搜神引擎 -- 民視新聞線上搜尋
81837	Http://www.ftv.com.tw/FTVFilter/rindex.htm	民視小品

頁框效應

某些網頁會將其本身分割成數個頁框（frame），每個頁框可放置一張網頁。所以，當我們瀏覽這類網頁時，往往一次瀏覽多張網頁，因而造成了網頁流。

32969-32971-33606-33589 26

32969	Http://www.yshen.com/indexs.html
32971	Http://www.yshen.com/indexs1.html
33606	Http://www.yshen.com/sides.html
33589	Http://www.yshen.com/s3a.html

單一瀏覽目的：僅僅包含單一主題的網頁流。

124227-124265-124277 25

124227	http://www.railway.gov.tw/	臺灣鐵路管理局
124265	http://www.railway.gov.tw/taiwan/index.html	臺灣鐵路管理局
124277	http://www.railway.gov.tw/taiwan/tai17.html	臺灣鐵路管理局--網路訂票

多重瀏覽目的：同時涵蓋多個主題的網頁流。

207069-90005-90140 24

207069	http://www.music.com.tw/broadway/top10/top10s.html	飛行音樂網路唱片行 TOP20
90005	http://www.cts.com.tw/channel/news/weather/weather.htm	華視氣象台
90140	http://www.cts.com.tw/educ/lesson/lesson.htm	空中商行專課程查詢

5.2 加上目錄樹後的循序樣式

最小支持度計數設爲 100，最大支持度計數訂爲 360，所推導出來的交叉層次循序樣式如下例所示：

100000302-100000702-100000301　126

100000302	搜尋引擎
100000702	軟體廠商
100000301	軟體下載

5.3 其他

Minimal Support	1.00% (24)	1.25% (30)
K=1	812.97	805.59
K=2	13283.76	5035.84
K=3	187.92	20.90
K=4	4.02	0.25
K=5	0.16	
Total	14288.83	5862.58

表格 4. 循序樣式探勘所需花費的平均時間

上表是分別以最小支持度計數 24 與 30 進行探勘，所需花費的平均時間（單位：秒）。我們可以發現在推導 2-循序樣式時所花費的時間，幾乎佔了絕大部份的時間。其主要原因在於 2-循序樣式是由常見 1-循序樣式兩兩交集產生，所以需要比較費時的運算；另外，當最小支持度降低（針對最小支持度計數 30 與 24 而言）時，常見 1-循序樣式的數目會相對增加(約 2 倍，我們可由表格 5 得知)，推導 2-循序樣式的時間也因而增加(約 2.5 倍)。

下表是分別以最小支持度計數 24 與 30，在每個步驟中所推導出來的網頁流個數。我們發現到最小支持度的些微降低，將會爲網頁流個數帶來顯著的增加。

Minimal Support	1.00% (24)	1.25% (30)
K=1	810	422
K=2	766	191
K=3	133	20
K=4	12	

表格 5. 每個步驟推導出來的網頁流個數

6 使用者查詢介面的實作

6.1 專家網頁流

根據實驗結果，我們不難發現推導出來的循序樣式，大多數不容易給定一個較爲明確的瀏覽目的。如前言所提及的，我們希望得到的「墾丁」旅遊網頁流，一方面可以清楚地知道使用者的瀏覽目的；另一方面，流程中的每個節點也可以代表各自獨立的主題。

爲了將推導不出來但具備導覽作用的網頁流，也提供給使用者查詢，在本研究中，我們也允許藉由人工輸入的方式，針對不同的瀏覽目的，產生可用於特定領域的*專家網頁流*（expert webflow）；這種特殊的網頁流，可以是「墾丁旅遊」、「時事」等由專家或系統管理者認定有用的網頁流。如此一來，由記錄檔推導所得的網頁流，是大多數使用者貢獻的結果，顯示大多數的使用者有類似的瀏覽行爲；而專家給定的網頁流，乃是以專家的經驗爲基礎，可提供專業且具權威的導覽建議。透過這兩者的結合，可以讓網頁流的查詢更容易滿足使用者需求。

在系統的實作上，我們加入幾個專家網頁流進行測試，其中一個有關墾丁旅遊的網頁流如下所示：

10000003-10000009-10000005-124277

10000003	http://www.ktnp.gov.tw/ (歡迎蒞臨墾丁國家公園)
10000009	http://www.cwb.gov.tw/ (中央氣象局全球資訊網)
10000005	http://www.caesarpark.com.tw/caesar/kenting.htm(墾丁凱撒飯店)
124277	http://www.railway.gov.tw/taiwan/tai17.html (臺灣鐵路管理局--網路訂票)

以下我們將說明本研究所設計的兩種查詢方案：

1. 網頁流的查詢介面：讓使用者以輸入關鍵字的方式，找到相關網頁流。
2. 整合傳統的搜尋引擎：讓使用者在操作傳統的搜尋引擎時，也能同時接收到相關網頁流的資訊。

我們實作的系統網址爲 http://dbpc14.cs.nthu.edu.t/~wohong/webflow.html。

6.2 網頁流的查詢介面

最基本的查詢功能，其查詢畫面如Figure 2，可以讓使用者直接輸入關鍵字，如網路訂票、墾丁、http://www.cwb.gov.tw/、chinatime 等。

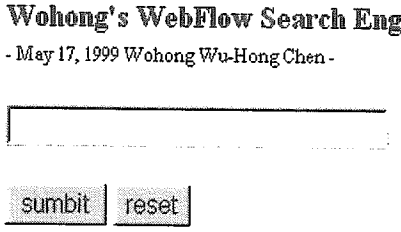

Figure 2. 網頁流的查詢介面

系統處理完查詢後會傳回相關的網頁流，其中包含所輸入的關鍵字、以及在目錄樹中相對應的父概念節點；舉例來說，如果使用者給定「中時電子報」關鍵字，則回傳結果會是包含「中時電子報」數個網頁流。如：

新聞→http://www.kimo.com.tw/inde.html→
http://home.netscape.com/ (Netcenter)→
http://www.chinatimes.com.tw/ （中時電子報）

因爲「中時電子報」的父概念節點爲「報紙」，所以回傳結果中也會有包含「報紙」的網頁流。如：

http://www.kimo.com.tw/index.html→新聞→
生活資訊→報紙

查詢結果的輸出畫面如Figure 3所示，使用者可點選網頁流中任何一個節點；比如點選「中時電子報首頁」，則右邊的頁框便會顯示中時電子報的首頁。此外，若該節點是目錄樹中的一個概念節點，則右邊的頁框則會顯示所包含的子概念節點，直到節點爲實際的網頁位址。

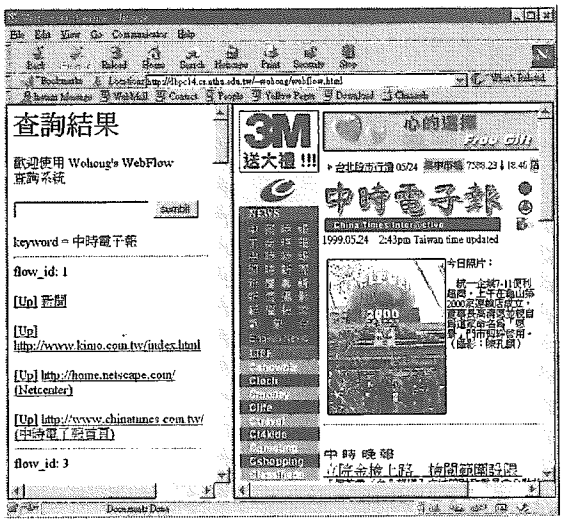

Figure 3. 網頁流查詢介面的查詢結果

在另一方面，使用者若點選節點左邊的*[Up]*鏈結，則會顯示該節點在目錄樹中的父概念節點。如此一來，使用者不僅可以根據關鍵字取得相關網頁流，還可以直接在目錄樹中遊走，以便選擇最符合自己需求的語意化網頁流。

6.3 整合傳統的搜尋引擎

由於上述的網頁流查詢介面，並不像傳統的搜尋引擎，需要先分析網頁內容，並將有用的關鍵字建成索引；取而代之的，只需爲網址或網頁標題建立索引。當使用者採用關鍵字進行查詢時，往往會因爲索引資訊不足，無法查詢到使用者感興趣的網頁流。有鑑於此，我們利用搜尋引擎的全文檢索功能，來輔助網頁流的查詢處理；也就是，當使用者自搜尋引擎回傳結果中，點選了某一個超鏈結之後，系統便到資料庫中尋找是否有網頁流包含該網址，如果有的話，便將相關的網頁流提供給使用者參考。

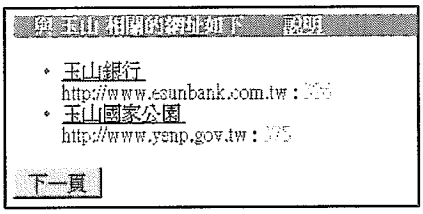

Figure 4. 搜尋引擎查詢結果

舉例來說，Figure 4是對搜尋引擎給定關鍵字「玉山」的查詢結果。假設使用者點選了「玉山銀行」，則除了顯示玉山銀行的首頁外，系統也會搜尋網頁流的資料庫，找出包含該網址的網頁流。Figure 5所顯示的即爲此查詢結果相關的網頁流。

假如資料庫中並不具有包含該網址的網頁流，但有包含其父概念節點的網頁流時，系統也可以提供包含其父概念節點的網頁流給使用者。以關鍵字「墾丁」爲查詢例子，其搜尋引擎的查詢結果如Figure 6所示，第一個查詢結果爲「歡迎蒞臨墾丁國家公園」。在網頁流資料庫中，存在一個專家網頁流如下：

http://www.ktnp.gov.tw/ (歡迎蒞臨墾丁國家公園)
→http://www.cwb.gov.tw/ (中央氣象局全球資訊網)
→http://www.caesarpark.com.tw/caesar/kenting.ht (墾丁凱撒飯店)
→http://www.railway.gov.tw/taiwan/tai17.html (臺灣鐵路管理局--網路訂票)

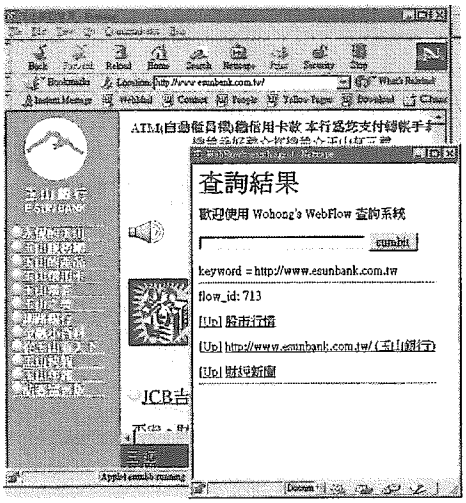

Figure 5. 整合搜尋引擎的查詢結果

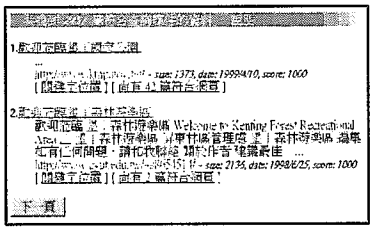

Figure 6. 關鍵字「墾丁」的查詢結果

因此，在顯示墾丁國家公園的首頁時，系統也會同時顯示上述網頁流。至於第二個查詢結果「歡迎蒞臨墾丁森林遊樂區」，由於資料庫中並沒有包含該網址的網頁流，也沒有包含其父概念節點的網頁流，所以系統將不會顯示任何的網頁流。

7 結論

本研究針對使用者的瀏覽目的中可能包含許多不同的主題，且主題間存在某種瀏覽次序的問題，提出一個解決的方法。我們的作法是對瀏覽軌跡進行循序樣式探勘，找出有意義的網頁流，最後提供查詢介面及建議視窗，讓使用者可以進行查詢與接受建議。

在本論文中，我們提出了一個循序樣式探勘的 SPI 演算法，利用反轉表與交集運算來完成循序樣式的探勘，找出大多數使用者所貢獻的網頁流。爲了讓網頁流對於使用者而言更有意義，我們也將網頁流中每個節點網頁的標題（title）一併顯示；另一方面，我們也利用目錄樹進行交叉層次的循序樣式探勘，找出由目錄樹概念節點所形成的網頁流，提供另一種具有語意的網頁流。

此外，由於網頁流的產生受限於記錄檔中的資料，所以我們額外加入專家給定的網頁流。如此一來，我們可以同時提供大多數使用者共同的瀏覽行爲，以及具有特定導覽作用的網頁流，將使網頁流的查詢與建議更能滿足使用者需求。

未來的研究，我們將繼續改進演算法，以加快探勘的速度並減少所需花費的記憶體；尤其在處理大量的記錄檔時，這兩方面的改進將顯得更加迫切需要。另外，我們也將考慮記錄檔每天新增的部分，找出新增的網頁流或修改已經存在的網頁流。在語意目錄樹方面，我們也需要一個足以滿足大多數使用者需求的語意描述方式，而建構這樣的目錄樹將是一個未來的重要課題之一。

參考文獻

[1] R. Agrawal and R. Srikant, "Fast algorithms for mining associateion rules," *Proceedings of International Conference on Very Large Data Base*, 1994.

[2] R. Agrawal and R. Srikant, "Mining sequential patterns," P*roceedings of International Conference on Data Engineering*, pp. 3-14, 1995.

[3] J. W. Han and Y. J. Fu, "Discovery of multiple-level association rules from large databases," *Proceeding of International Conference on Very Large Data Base*, 1995.

[4] R. Srikant and R. Agrawal, "Mining sequential patterns: Generalizations and performance improvements," 1996.

[5] T. W. Yan, M. Jacobesn, H. Garcia-Molina, and U Dayal, "From user access patterns to dynamic hypertext linking," *Proceedings of Internat ional World Wide Web Conference*, 1996.

[6] S. J. Yen and A. L. P. Chen, "An efficient approach to discovering knowledge from large databases," *Proceedings of IEEE/ACM Conference on Paralle and Distributed Information Systems*, 1996.

[7] S. J. Yen and A. L. P. Chen, "A graph-based approach for discovering various types of association rules," *NTHU Technical Report*, 1998.

[8] M. J. Zaki, "Efficient enumeration of frequent sequences," *Proceedings of ACM Conference o Information and Knowledge Management*, 1998.

[9] 陳武宏, "利用全球資訊網之使用者瀏覽行爲探勘網頁流之技術探討," *清華大學碩士論文*, 1999.