在教學網站的環境中發掘熱門的學習路線
Mining Frequent Traversal Patterns in a Web Training Environment

顏秀珍
輔仁大學資訊工程學系
sjyen@csie.fju.edu.tw

李御璽
銘傳大學資訊管理學系
leeys@mcu.edu.tw

徐家馴
輔仁大學資訊工程學系

摘要

網際網路的蓬勃發展，尤其利用全球資訊網來規劃輔助學習的環境，已成為未來的趨勢，然而，無論是教師或個別學習者，在網際網路的學習環境下，並不能有效地掌握大多數學習者的學習行為與學習狀態，因此，如何讓學習者從網際網路的教學網站中有效率地獲得學習輔助資訊，即成為首要的課題。目前，大多數的學習資訊網站已有功能來收集學習者上網瀏覽的時間和網頁，但並沒有進一步地加以分析出有用的學習特性，也就是學習者的學習行為與教學網頁的關連性等。本論文提出一個演算法，從學習者的學習路徑中有效率的發掘出大部分學習者的學習行為特性，以瞭解學習者的學習狀態並提昇學習者的學習效率。

我們的演算法適用於網際網路上教學網站的環境，其特色在於保留原始的學習者學習行為記錄，以及發掘出大部分學習者的學習行為，我們稱為*熱門學習路線*，讓學習者能快速且立即的獲得學習效能。我們使用模擬資料庫來進行實驗，以驗證在實際環境上執行的效率，而實驗的結果顯示，使用我們的演算法所發掘出的資訊，不但可以符合我們的要求，也能得到較佳的執行速度。

關鍵詞：資料探勘，熱門學習序列，頻繁瀏覽序列，教學網站

1. 概論

在資料庫的研究領域中，資料探勘(data mining)[6][12][13][17][21]目前是一個熱門的研究領域，其目的是從大量的資料中，將潛在有用的資訊挖掘出來。而在網路盛行的今天，網路環境充斥著許多的資料，如何將這些大量的資料轉換成有用的資訊便成為當前重要的研究方向。

目前的網路教育網站系統，大部份僅是將課程與書本的章節做成單純的網頁，以靜態的方式提供資訊服務，並沒有考慮到學習者真正的學習方向與學習的興趣喜好，而網路學習者在學習的過程中遇到最大的問題就是摸索與等待，因為教師或教學網站設計者並無法與網路學習者形成互動的關係，學習者仍然需花費很多的時間和精力，去找尋自己真正需要的學習資訊與學習方向，常常讓學習者降低學習的意願與效率。因此，就網際網路上的學習教學領域而言，如果能對學習者瀏覽網頁的行為有所了解，就能掌握學習者的學習情形。我們認為要達到這些目的，必須先了解使用者最常學習瀏覽的網頁路線，以便在適合的網頁中提供適當的訊息，或是輔助後來的學習者增加學習的效率與減少學習摸索及等待的時間。

首先，我們對我們所研究的問題做一些定義：一個網頁（page）對應於網路上的瀏覽網頁，一位學習者可瀏覽許多網頁，將瀏覽的網頁依照時間先後順序排列稱為一個*瀏覽序列(traversal sequence)*，我們表示成 $<s_1, s_2, \ldots, s_n>$，其中 s_i 為一個網頁。假設兩個瀏覽序列$<a_1, a_2, \ldots, a_n>$和$<b_1, b_2, \ldots, b_m>$，若存在一有序整數 $i_1 < i_2 < \ldots < i_n$，$1 \leq i_k \leq m$，使得 $a_1 \subseteq bi_1$，…，$a_n \subseteq bi_n$ 我們稱瀏覽序列$<b_1, b_2, \ldots, b_m>$*包含*（*contain*）瀏覽序列$<a_1, a_2, \ldots, a_n>$，同時，$<a_1, a_2, \ldots, a_n>$為$<b_1, b_2, \ldots, b_m>$之長度為 n 的*子瀏覽序列*。在一個瀏覽序列的集合中，如果一個序列 s 為*最大的瀏覽序列*（maximal traversal sequence），則瀏覽序列 s 不為其他的瀏覽序列所包含。

一個*學習序列*為一位學習者從進入教學網站瀏覽網頁到離開教學網站所產生的瀏覽序列。*學習*

*序列資料庫*則是由許多學習者的學習序列所組成，因此學習序列資料庫中所記載的資料包含了瀏覽網頁、學習者編號及瀏覽時間等。一個網頁的*支持度*（support for a page）為所有包含該網頁之學習者的個數與所有學習者個數的比值。而一個網頁的支持度若滿足使用者定義的門檻值（threshold）(稱之為*最小支持度*（minimum support）)，則此網頁稱為*頻繁網頁*（frequent page）；同樣地，如果瀏覽序列 *s* 被一個學習者的學習序列所包含，則稱此學習者支持（support）瀏覽序列 *s*。一個瀏覽序列的*支持度*（support for a traversal sequence）為所有支持該瀏覽序列之學習者的個數與所有學習者個數的比值。而一個瀏覽序列的支持度若滿足最小支持度，則此瀏覽序列稱為*頻繁瀏覽序列*（frequent traversal sequence），一個瀏覽序列的*長度*（length）為構成此序列之網頁的個數，長度為 *k* 的瀏覽序列我們表示為 *k-瀏覽序列*，且一個長度為 *k* 的頻繁瀏覽序列表示為 *k-頻繁瀏覽序列*，若瀏覽序列中的網頁允許重複，稱為*非簡單的瀏覽序列*（non-simple traversal sequence）；反之，若瀏覽的網頁不可以重複的出現，稱之為*簡單的瀏覽序列*（simple traversal sequence）。

在本篇論文中，我們就是要在網際網路的教學環境下，從網路學習者的學習序列資料庫中，以學習者為導向，找出所有*非簡單的頻繁瀏覽序列*，以下我們仍簡稱為頻繁瀏覽序列。因為頻繁瀏覽序列間可能會有包含的關係，我們須從中找出最大的頻繁瀏覽序列。在找尋頻繁瀏覽序列時，我們會預先產生一個瀏覽序列的集合，透過我們的演算法，對每個瀏覽序列作支持度的計算，決定其是否為頻繁瀏覽序列，我們稱這個集合中的瀏覽序列為*候選瀏覽序列*（candidate traversal sequence）。

以下我們將介紹一些與我們研究問題相關的文獻：*挖掘序列型樣*(Mining sequential patterns)的目的是要從商品交易資料庫中，找出大部份顧客購物的循序行為，也就是頻繁序列。Apriori All 演算法[1][2]利用一次又一次地掃描分析交易資料庫的方式，從長度 1 的頻繁序列集找起，再由長度為 k 的頻繁序列集組合成長度為 k+1 的候選序列集，漸近地找出所有長度的頻繁序列集，直到無法產生新的頻繁序列為止，因此無法避免多次重覆地掃描交易資料庫，且所有的候選序列集存放到記憶體，因此，此演算法是相當花費時間和空間的。

挖掘路徑瀏覽樣式(Mining path traversal patterns)是在網際網路的環境下，尋找出大多數的網路瀏覽者的瀏覽行為，當我們了解大多數使用者在網際網路上的瀏覽行為，便可以針對發掘出的資訊做不同的應用，譬如可以提供網站設計者改善網站設計，提昇網站的效率、亦可提供使用者瀏覽網頁的預先存取及瀏覽路線的建議等等。相關的研究有 M.S. Chen [4] 提出的 FS (Full Scan) 及 SS (Selective Scan) 演算法，其演算法主要分為兩個部分： 先利用最大向前參考序列（Maximal Forward reference)演算法把資料由 log data 轉換成最大向前參考序列的資料庫，再運用 FS 或 SS 演算法來發掘出所有長度的頻繁瀏覽序列集。所謂最大向前參考是指使用者一直向前瀏覽網頁的動作時，發生有回頭瀏覽(backward reference)的行為，即切斷剛剛向前瀏覽的路線。

FS 和 SS 演算法是利用 DHP(Direct Hashing and Pruning)演算法[5]的觀念。DHP 演算法與 AprioriAll [1][2]一樣是多次掃描資料庫的演算法，第一次產生 1-頻繁序列集，第二次產生 2-頻繁序列集，第 k 次產生 k-頻繁序列集，以此類推。主要差異是在於 DHP 利用雜湊表(Hash table)來減少 2-候選序列集的個數，接著當計算 k-候選序列集的支持度時，利用 k-候選序列集來刪減資料庫。DHP 的優點在於它會減少 2-候選序列集的數目及縮減資料庫的大小，藉此加快演算法的速度，而 SS 是類似 FS 演算法，不同於 FS 演算法是 SS 演算法它並不是每次都掃描資料庫，而是利用之前的候選序列集直接產生下一個長度的候選序列集，再一起掃描資料庫，進而減少 disk I/O 的時間，而 FS 及 SS 演算法的缺點是經過 MFR 演算法切割後，可能會產生一些隱藏資訊的遺失，且若*最小支持度*(minimum support) 訂得太低，會讓 DHP 無法減少過多的 2-

候選序列集及縮減資料庫的大小，使得 DHP 演算法的速度變慢。

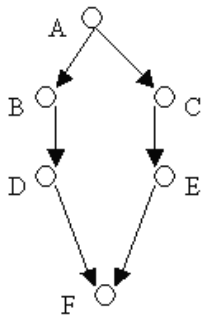
另一個相關於挖掘路徑瀏覽樣式(Mining path traversal patterns)的研究是 J. Han [11]所提出的 WAP mining 演算法，完全不同於其他的演算法是因為它並不產生候選序列集，而是運用 WAP-tree 資料結構。WAP-tree 演算法的優點在於它不產生候選序列集，所以它可以避開候選序列集暴增的情形，另外，它只需掃描資料庫兩次，因此速度上較 Apriori All 演算法為快。但其缺點為需要很大的記憶空間，因為該演算法等於把資料庫中的所有交易全部轉換 WAP-tree 結構而存放在記憶體中，特別是當資料庫中瀏覽序列重複出現的情形很低時，所需要的記憶體更是難以估算，使得記憶體的負荷量大增。由於 WAP-tree 的資料結構在實作上記憶體使用實在太大，因此後面的實驗無法進行。

2. 挖掘非簡單的頻繁瀏覽序列

教學網站本身的結構是一位或多位這方面領域或學科的專家設計而成，因此在教學網站中兩兩相連的網頁，必然有其關聯性，也有其引導學習者的預定流程與學習建議，而我們希望透過所有學習者的學習行為記錄，來找出大多數學習者共同的學習行為，以了解學習者的真正學習行為模式，而教學網站設計的專家們，也可以透過發掘出的資訊，明瞭網路學習者的學習情形。

在先前討論提出的演算法中，大多數的研究領域受限於網頁不能重複出現在同一個瀏覽序列，因此發掘出來的資訊為簡單瀏覽序列。再者，由於學習是一種重複閱讀的動作，有些學習科目網頁是相互參考，沒有所謂前後性的分別，例如{資料結構}與{程式設計}兩個課程的學習網頁是互相參考而沒有前後性的分別，因此教學網站的學習者可能會多次瀏覽到相同的網頁，而我們的演算法所找出的學習瀏覽路線，允許同一學習網頁的重複出現，所發掘出的資訊為非簡單瀏覽序列。這篇論文針對網際網路上教學網站的環境與應用特性，提供了一個有效率的演算法來找出非簡單瀏覽路徑，以幫助使用者在學習上快而立即的取得學習資訊，並輔助使用者有效率的學習，以期能得到較佳的學習效能。

由於一些挖掘序列型樣(Mining sequence patterns)[1][2][10][18][20][24]以及挖掘路徑瀏覽樣式(Mining path traversal patterns) [3][4][7][9][19]的演算法，都有其不同應用的方向，如修改網頁連結設計，以減少瀏覽者在網站上的摸索時間及提供網頁的預先存取等等，而本篇論文所著重的方向，是在教學網站上的應用，如圖 2-1，一個簡單的網站架構，當我們利用 AprioriALL 演算法[1][2]發掘出{AF}這個頻繁瀏覽序列時，在網際網路的環境下，瀏覽者或學習者並沒有辦法確定如何由 A 網頁走到 F 網頁，且在網際網路的教學環境上，直接由 A 網頁加入一個連結到 F 網頁也不一定是一個完整的學習行為，例如 A 網頁是{三角函數首頁}, F 網頁是{三角函數查表}, 發掘{AF}的瀏覽行為就不具意義，因此，我們的研究著重於所找出的頻繁瀏覽序列中，相鄰的網頁必定有直接的連結關係，也就是有直接連結的非簡單瀏覽序列，以便提供學習者在網路學習環境中，迅速連結到下一個正確的網頁，得到快而立即的學習資訊，以減少學習者學習摸索時間，增加學習效能。



圖一: 一個簡單的網站架構

2.1 建立學習者的學習序列

在網際網路的環境下，大多數的伺服端已經有記錄網路上來訪者記錄的功能，但由於大多數網路使用者為了節省等待的時間，會在用戶端設定快取 Proxy，也因為如此，會造成伺服端所收集到的資料不齊全與不正確，因為在做學習者行為分析之前，取得學習者正確的瀏覽行為是非常重要的。我們提出兩個會造成資料不正確的原因及其解決的方

式：第一個原因是不同的學習者在自己的瀏覽器輸入快取 Proxy 後，會有相同 IP address[8]的問題，我們可以利用在伺服端撰寫相關應用程式來取得最原始的來源 IP address (目前 HTTP 1.1 Protocol 及 ASP 已經可以取得最原始的 IP address)，以解決不同使用者透過同一個 Proxy(相同 IP address)可能產生的問題，在教學網站的環境下，由於大多數的教學網站都需要學習者輸入自己的帳號和密碼，來確定學習者的身分，因此對於相同網路位址，不同的學習者擁有相同 IP address 的問題便可迎刃而解。

另一個會造成資料不正確的原因是瀏覽器本身有快取記憶體的功能，會將之前瀏覽過的網頁保留在自己的電腦上，造成在建立學習者的學習序列時，無法呈現完整的學習序列，也就是當學習者看過 A 網頁後，再瀏覽 B 網頁，當要回頭看 A 網頁時，網路伺服端並沒有真的收到要求看 A 網頁的訊息，而是由學習者自己的瀏覽器透過快取記憶體的功能，在學習者的瀏覽畫面上，重複出現 A 的網頁畫面，針對這個問題，我們可以透過學習伺服端，將學習網頁提供"No Cache "的機制，當使用者點選到之前瀏覽過的網頁時，還是必須通知伺服端重新取回 A 的網頁資料，以達到學習序列的完整性。

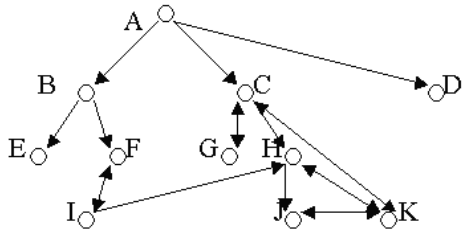
2.2 挖掘有直接連結之非簡單瀏覽序列

我們提出 FDLP(Frequent Direct Learning-Path)演算法來挖掘出有直接連結之非簡單瀏覽序列。FDLP 演算法在實作上會參考教學網站本身的網站架構，並會找出適合教學網站學習者的瀏覽資訊，也就是非簡單且有直接連結的瀏覽路線，主要執行步驟如表一所示。

產生長度(K+1)-候選瀏覽序列(K≥1)的方法如下[4]：對於任意兩個不同的 K-頻繁瀏覽序列 $<r, S_1, S_2, \ldots, S_{k-1}>$和$< S_1, S_2, \ldots, S_{k-1}, t>$，可產生(k+1)-候選瀏覽序列$< r, S_1, S_2, \ldots, S_{k-1}, t>$。

表一: FDLP 演算法主要執行步驟

執行步驟	執行動作
第一次掃描資料庫	(1) 求出所有的 1-頻繁瀏覽序列集 (2) 產生 2-候選瀏覽序列集 (3) 透過原先的網站架構，只保留有直接連結的 2-候選瀏覽序列集
第二次掃描資料庫	(1) 求出所有的 2-頻繁瀏覽序列集 (2) 在掃描資料庫的過程中，刪除所有在學習序列中的非 1-頻繁瀏覽序列集 (3) 產生 3-候選瀏覽序列集
第 n 次掃描資料庫	利用特殊的資料庫掃描方式，可以從學習序列中產生出所有在網站上有直接連結之長度為 n 的瀏覽序列，以求出所有長度為 n 的直接連結之非簡單頻繁瀏覽序列集
	直到沒有頻繁瀏覽序列產生為止



圖二: 一個網站的架構

為了說明 FDLP 演算法，我們舉例如下：假設網站架構如圖二，學習序列資料庫如表二所示，其最小支持度為 40%：

表二: 學習序列資料庫

User ID	學習序列
User 001	ABEBFIH
User 002	ABFIHKCHKCHK
User 003	ACGCGCKJKC
User 004	ACHKJKC
User 005	BFIFH

首先，我們掃描一次學習序列資料庫，計算每一個網頁的支持度，求出所有的 1-頻繁瀏覽序列為網頁 A、B、C、F、H、I、J 和 K。接著產生所有的 2-候選瀏覽序列，對於網站本身的架構，我們針對此架構中每一個網頁與其有直接連結的網頁做記錄，例如與 A 網頁有直接連結的是 B、C 和 D 網

頁，我們利用稀疏矩陣的資料結構來記錄這個資訊，在 A 網頁這一列陣列上，將對應到 B、C 和 D 網頁都設為 1，表示 A 網頁和 B、C 和 D 網頁有直接連結的關係，其他則為 0。

針對所有的 2-候選瀏覽序列，透過網站架構之資料結構的過濾刪除，只保留有直接連結關係的 2-候選瀏覽序列集，例如我們結合 1-頻繁瀏覽序列 A 網頁和 C 網頁，產生 2-候選瀏覽序列{AC}。因為 A 網頁有直接連結到 C 網頁，因此我們保留{AC}這個 2-候選瀏覽序列。另外，我們結合 1-頻繁瀏覽序列 A 網頁和 F 網頁，產生 2-候選瀏覽序列{AF}，因為 A 網頁和 F 網頁並沒有直接的連結關係，因此不產生 2-候選瀏覽序列{AF}，以此類推下去，我們可以得到下列的 2-候選瀏覽序列集：{AB, AC, BF, CH, CK, FI, HJ, HK, IF, IH, JK, KC, KH, KJ}，比起原先 AprioriALL 演算法的任意組合(P(8,2) = 56 個 2-候選瀏覽序列)會少許多，因此可以節省很多儲存候選序列的記憶體空間，且存放網站架構之資料結構的記憶空間在完成找出所有直接連結關係的 2-候選瀏覽序列集後，便可移除。

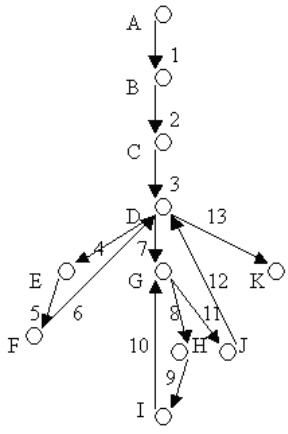
表三: 刪除非 1-頻繁瀏覽網頁後的資料庫

User ID	學習序列
User 001	ABFIH
User 002	ABFIHKCHKCHK
User 003	ACKJKC
User 004	ACHKJKC
User 005	BFIFH

第二次掃描資料庫時，我們計算每一個 2-候選瀏覽序列的支持度，所求出的 2-頻繁瀏覽序列集為{AB, AC, BF, CH, FI, HK, IH, JK, KC, KJ}，且這些 2-頻繁瀏覽序列在網站上都有直接的連結關係，在掃描第二次學習序列資料庫的過程中，我們也刪除所有在學習序列中的非 1-頻繁瀏覽序列，如表二中的第一筆學習序列{ABEBFIH} ，我們可以將非 1-頻繁瀏覽序列 E 從第一筆學習序列中移除，因 E 網頁的前後都是相同的網頁 B，且同一個網頁在同一個學習序列中連續重複出現是沒有意義的，所以我們刪除其中一個網頁 B，最後得到新的學習序列為{ABFIH}，如此，我們可以得到新的學習序列資料庫如表三所示。

由於在產生 2-頻繁瀏覽序列集時，我們只保留在網站上有直接連結的瀏覽序列，當學習序列資料庫在刪除非 1-頻繁瀏覽序列後，可能會產生出在網站上沒有直接連結的瀏覽序列去比對候選瀏覽序列集，但由於我們所產生的 2-頻繁瀏覽序列，在網站上必定都有直接連結的關係，因此所產生的候選序列也必定有直接連結關係，所以即使在學習序列中存在沒有直接連結的瀏覽序列，也無法在候選瀏覽序列集中找到相同的瀏覽序列做計數，因此可以滿足我們所要找尋的資訊。我們利用已經求出的 2-頻繁瀏覽序列集，產生長度為 3 的候選瀏覽序列集為{ABF, ACH, BFI, CHK, FIH, HKC, HKJ, IHK, JKC, JKJ, KCH, KJK}。

接下來，我們提出一個特殊的資料庫掃描方式，來產生所有在網站上有直接連結的 k-頻繁瀏覽序列(k ≥ 3)，我們先舉一個例子來說明此特殊的資料庫掃描方式，例如：有一位學習者的學習序列為{ABCDEFDGHIGJDK}，網站的局部結構為圖三，其學習序列的瀏覽網頁順序如表四所示。



圖三: 網站的局部結構

當要從此學習序列產生出在網站上有直接連結之長度為 3 的瀏覽序列來比對候選瀏覽序列集時，我們的資料庫掃描方式可以用*直接連結瀏覽序列表*來說明，如表五所示。

表四: 學習序列的瀏覽網頁順序

瀏覽網頁	A	B	C	D	E	F	D
瀏覽順序	1	2	3	4	5	6	7
瀏覽網頁	G	H	I	G	J	D	K
瀏覽順序	8	9	10	11	12	13	14

表五: 直接連結瀏覽序列表(產生 3-瀏覽序列)

瀏覽順序	瀏覽網頁	預備序列	瀏覽序列	瀏覽順序	瀏覽網頁	預備序列	瀏覽序列
1	A			9	H		DGH
2	B			10	I		GHI
3	C		ABC	11	G	DG	HIG
4	D		BCD	12	J		IGJ
5	E		CDE				DGJ
6	F		DEF	13	D	CD, FD	GJD
7	D	CD	EFD	14	K		JDK
8	G		FDG				CDK
			CDG				FDK

當掃描到此學習序列的第三個網頁 C 時，會循序產生瀏覽序列{ABC}去比對候選瀏覽序列集，如果在候選瀏覽序列集中，存在{ABC}這個候選瀏覽序列，則我們將{ABC}這個候選瀏覽序列計數加一，但如果對於同一學習者，若其學習序列已經產生過{ABC}這個瀏覽序列，則不重複計數。同樣地，掃描到第四個網頁 D 的時候，循序產生瀏覽序列{BCD}去比對候選瀏覽序列集，以此類推掃描下去；值得注意的是，當掃描到第七個網頁 D 的時候，由於 D 網頁在此學習序列中的第四個瀏覽網頁曾經出現過，而其前一個瀏覽網頁為 C，此時我們會產生一個預備序列{CD}存放在瀏覽順序第七的位置。當掃描到第八個瀏覽網頁 G 的時候，首先循序產生瀏覽序列{FDG}去比對候選瀏覽序列集，再將預備序列{CD}與此網頁 G 結合成有直接連結的瀏覽序列{CDG}去比對候選瀏覽序列集，因為瀏覽序列{CDG}在網站上是有直接連結的關係，而光從學習序列上無法看出，因此我們產生此預備序列{CD}，以便在掃描下一個網頁時產生有直接連結的瀏覽序列。接下來，我們將預備序列{CD}的第一個網頁刪除，加上第八個瀏覽網頁 G，產生新的預備序列{DG}，但由於在掃描到第九個網頁 H 時，預備序列{DG}結合 H 網頁與循序產生的瀏覽序列{DGH}相同，所以刪除{DG}這個預備序列，也就是在產生長度為 3 的瀏覽序列時，預備序列不必再與下一個瀏覽網頁結合產生新的預備序列，掃描完所有學習序列後，即可得到所有支持長度為 3 的候選瀏覽序列之學習者個數，進而得到所有長度為 3 的直接連結之頻繁瀏覽序列。

同樣地，當要從此學習序列產生出在網站上有直接連結之長度為 4 的瀏覽序列來比對候選瀏覽序列集時，其資料庫掃描方式我們也用直接連結瀏覽序列表來說明，如表六所示。

表六: 直接連結瀏覽序列表(產生 4-瀏覽序列)

瀏覽順序	瀏覽網頁	預備序列	瀏覽序列	瀏覽順序	瀏覽網頁	預備序列	瀏覽序列
1	A			11	G	CDG	GHIG
2	B					FDG	
3	C			12	J		HIGJ
4	D		ABCD			DGJ	CDGJ
							FDGJ
5	E		BCDE				
6	F		CDEF	13	D	BCD	IGJD
7	D	BCD	DEFD			EFD	
8	G		EFDG				DGJD
		CDG	BCDG	14	K		GJDK
9	H		FDGH			CDK	BCDK
			CDGH			FDK	EFDK
10	I		DGHI				

當掃描到第四個網頁 D 時，會循序產生瀏覽序列{ABCD}去比對候選瀏覽序列集，如果在候選瀏覽序列集中，存在{ABCD}這個候選瀏覽序列，則我們將{ABCD}這個候選瀏覽序列計數加一，但如果對於同一個學習者其學習序列已經產生過瀏覽

序列{ABCD}，則不重複計算次數；同樣地，掃描到第五個網頁 E 的時候，循序產生瀏覽序列{BCDE}去比對候選瀏覽序列集，以此類推掃描下去；與產生長度為 3 的瀏覽序列方式不同的是，當掃描到第七個網頁 D 的時候，由於 D 網頁在此學習序列中的第四個瀏覽網頁曾經出現過，則將其與前兩個瀏覽網頁結合，產生長度為 3 的預備序列{BCD}，記錄在直接連結瀏覽序列表瀏覽順序第七的位置中。

當掃描到第八個瀏覽網頁 G 的時候，首先循序產生瀏覽序列{EFDG}去比對候選瀏覽序列集，再將目前的預備序列{BCD}與網頁 G 結合成有直接連結的瀏覽序列{BCDG}去比對候選瀏覽序列集，之後將預備序列{BCD}的第一個網頁刪除，加上第八個瀏覽網頁 G，產生新的預備序列{CDG}，當掃描到第九個網頁 H 時，除了循序產生瀏覽序列{FDGH}外，也將預備序列{CDG}結合 H 網頁產生{CDGH}，因為瀏覽序列{BCDG}與{CDGH}在網站上是有直接連結關係的，而光從學習序列上無法看出，因此我們記錄此預備序列，以便掃描下一個網頁時產生有直接連結的瀏覽序列，同樣的，將預備序列{CDG}的第一個網頁刪除，再加上第九個網頁 H，產生新的預備序列{DGH}，當掃描到第十個網頁 I 時，由於新的預備序列{DGH}與網頁 I 結合所產生的瀏覽序列和循序產生的瀏覽序列{DGHI}相同，所以刪除{DGH}這個預備序列，也就是在產生長度為 4 的瀏覽序列時，預備序列只需刪除第一個網頁與加入新的網頁一次，產生新的預備序列，同理，若產生長度為 K (K≥3)的瀏覽序列時，預備序列則須連續刪除第一個網頁與加入新掃描到的網頁(K-3)次。

當掃描到第十一個瀏覽網頁 G 的時候，會產生兩個預備序列{CDG、FDG}，其中{FDG}的產生方式與前面相同，而{CDG}的產生是因為網頁 G 在此學習序列的第八個瀏覽網頁曾經出現過，而此位置有一個預備序列{CDG}。在掃描到第十二個瀏覽網頁 J 的時候，產生瀏覽序列{HIGJ、CDGJ、FDGJ}及預備序列{DGJ}。在掃描到第十三個瀏覽網頁 D 的時候，產生瀏覽序列{IGJD、DGJD}及預備序列{BCD、EFD}，以此類推下去，在掃描完所有學習序列後，即可得到所有支持長度為 4 的候選瀏覽序列之學習者個數，進而得到所有長度為 4 的直接連結之非簡單頻繁瀏覽序列。

對於第 n 次(n>=3)的資料庫掃描方式，我們可以用圖四的演算法來說明。

```
// (PrepareSequence 表示為預備序列，
    TraversalSequence 表示為瀏覽序列)
取得一個學習序列
While(I=1;I<=學習序列長度;I++)
{
  GetPage = 學習序列中的第 I 個網頁;
  GetSequence = GetSequence + GetPage;
  If (Have_PrepareSequence == TRUE)
      Check_PrepareSequence();
  If (GetSequence_Len >=n)
      Gen_TraversalSequence();
}
```

圖四: FDLP 掃描學習序列資料庫演算法

```
Gen_TraversalSequence()
{
從 GetSequence 中取出最後 n 個網頁所形成的瀏
覽序列，並比對候選瀏覽序列集，若有此候選序
列，則其計數加一;
If(GetPage 在 GetSequence 中重複出現)
   {prepare_Sequence[I]= 在 GetSequence 中每
     個與 GetPage 相同的網頁，及其前(n-2)個網
     頁;
  Have_PrepareSequence = TRUE;}
}
```

圖五: 循序產生瀏覽序列之子函式

FDLP 掃描學習序列資料庫演算法開始讀入一個學習序列，當掃描到第 n 個瀏覽網頁時，開始產生瀏覽序列(呼叫 Gen_TraversalSequence()子函式，如圖五)，去比對候選瀏覽序列集；當掃描到的網頁重複出現在同一個學習序列時，會產生預備

序列以便掃描到下一個網頁時，產生有直接連結的瀏覽序列。因此，若掃描到的網頁之前一個網頁有預備序列，則將此預備序列與目前掃描到的網頁結合以產生在網站上有直接連結的瀏覽序列去比對候選瀏覽序,也就是呼叫 Check_ PrepareSequence() 子函式，如圖六，直到整個學習序列資料庫被掃描完畢為止。

```
Check_PrepareSequence()
{
Generate 瀏覽序列 "PrepareSequence[I-1]
+GetPage" 並比對候選瀏覽序列集,若有此候選序列，則其計數加一;
對於 PrepareSequence[I-1]中的每一個預備序列刪除其第一個網頁,後端加入 GetPage 放入 PrepareSequence[I];
If (在 prepare_Sequence[I]中，若有預備序列等於 GetSequence 最後(n-1)個瀏覽序列)，則刪除其預備序列;
If 沒有任何預備序列存在，則 Have_PrepareSequence = FALSE;
}
```

圖六: 從預備序列產生瀏覽序列之子函式

接續表二的例子，在透過 FDLP 演算法的特殊掃描資料庫方式，我們可以求出 3-頻繁瀏覽序列集為{ABF, BFI, CHK, FIH, HKC, JKC, KJK}，再利用已經求出的 3-頻繁瀏覽序列集，產生長度為 4 的候選瀏覽序列集為{ABFI, BFIH, CHKC, KJKC}。經過第四次掃描學習序列資料庫，我們得到 4-頻繁瀏覽序列集為{ABFI, BFIH, CHKC, KJKC}，並產生長度為 5 的候選瀏覽序列為{ABFIH}，經過掃描資料庫後我們發現{ABFIH} 為 5-頻繁瀏覽序列，到此，因為無法產生新的候選瀏覽序列，所以停止 FDLP 演算法。最後我們找出的所有最大之直接連結的非簡單頻繁瀏覽序列集為{AC, CHKC, KJKC, ABFIH}。

3. 實驗結果分析和比較

首先，我們先建立一個教學網站的架構，並定義了幾個參數，請詳見表七。

表七: 參數表

Page	總共網頁數目
D	學習瀏覽序列總數（千筆）
/I/	每筆瀏覽序列平均的網頁個數
Min_S	最小支持度
N_FS	最大可能的頻繁瀏覽序列個數
R	瀏覽網頁時，返回上一個網頁會重複之百分比(%)
J	瀏覽網頁時，會進行跳躍式的瀏覽之百分比(%)
Ns	瀏覽網頁時，並不是由入口網頁進入之百分比(%)

我們利用這些參數，產生學習序列資料庫，由於大多數學習者會進行重複性瀏覽網頁的學習行為，因此我們針對 R 著手設計實驗，R 表示的是瀏覽網頁時，返回上一個網頁的重複之百分比(%)，因此我們固定了資料庫中的其他參數，只變動 R，我們所設計之資料庫的參數如表八所示。

我們總共使用 4 個資料庫 Backward(10%)、Backward(20%)、Backward(30%)及 Backward(40%)分別表示 R = 10%, 20%, 30%, 40%。我們將 AprioriAll、MFR 結合 FS 與 FDLP 三種演算法的執行時間比例列於表九。而對於每一種資料庫及三種演算法所發掘出最長之頻繁瀏覽序列的長度，如表十所示。

表八: 參數設定值

Page	1000 個
D	100（千筆）
/I/	每筆瀏覽序列平均 15 的網頁
Min_S	0.5%
N_FS	1000 個
J	瀏覽網頁時，會進行跳躍式的瀏覽為 3%
Ns	瀏覽網頁時，並不是由入口網頁進入為 8%

表九: 三種演算法的執行時間比例

資料庫	Apriori All	FDLP	MFR 結合 FS
D100-I15-**R10**-J3-Ns8	298	1	0.92
D100-I15-**R20**-J3-Ns8	275	1	0.96
D100-I15-**R30**-J3-Ns8	260	1	0.94
D100-I15-**R40**-J3-Ns8	251	1	0.91

表十: 三種演算法所產生頻繁序列的最大長度

資料庫	Apriori All	FDLP	MFR 結合 FS
D100-I15-**R10**-J3-Ns8	9	7	7
D100-I15-**R20**-J3-Ns8	8	8	7
D100-I15-**R30**-J3-Ns8	7	8	5
D100-I15-**R40**-J3-Ns8	6	8	5

由於 AprioriAll 演算法所發掘出的頻繁瀏覽序列中，包含了在網站上有直接與間接連結的瀏覽序列，且其瀏覽序列中出現的網頁不重複，所以在拆解學習序列的過程中，花費了很多的時間和空間，而 FDLP 演算法所發掘出來的瀏覽序列和 AprioriAll 演算法是不相同的，執行效率也比 AprioriAll 演算法快很多。相對於 MFR 結合 FS 演算法，由於 FDLP 演算法發掘的資訊較多，所產生的候選序列集與針對每一筆學習序列拆解的次數也比較多，所以在執行的效率上，FDLP 演算法會比 MFR 結合 FS 演算法較差一些。值得注意的是，如表十第三個和第四個資料庫(返回上一個網頁的重複之百分比較高時)，MFR 結合 FS 演算法由於先分割每一筆學習序列，所以造成平均的學習序列長度縮短，使得發掘出來的頻繁瀏覽序列的長度會比較短，可能造成一些頻繁瀏覽序列的流失，不符合教學網站上學習者重複閱讀網頁的習性。

接著，我們再針對較少的網頁數量來作實驗，因為一般的教學網站，網頁的數目並不會很多，而是網頁的內容比較來的大。我們將網頁個數由 1000 個改為 500 個，再針對參數 R(瀏覽網頁時，返回上一個網頁的重複之百分比)的變動來進行實驗，我們所設計之資料庫的參數如表十一所示，實驗結果如表十二和表十三所示。

表十一: 參數設定值

Page	**500** 個
D	100（千筆）
|I|	每筆瀏覽序列平均瀏覽 15 的網頁
Min_S	0.5%
J	瀏覽網頁時，會進行跳躍式的瀏覽為 3%
Ns	瀏覽網頁時，並不是由入口網頁進入為 8%

表十二: 三種演算法的執行時間比例

資料庫	Apriori All	FDLP	MFR 結合 FS
D100-I15-**R10**-J3-Ns8	321	1	1.04
D100-I15-**R20**-J3-Ns8	310	1	1.07
D100-I15-**R30**-J3-Ns8	297	1	1.03
D100-I15-**R40**-J3-Ns8	284	1	0.98

我們發現在網頁數量較少時，會產生較多的頻繁瀏覽序列，以致於對 MFR 結合 FS 演算法來說，並不能有效的減少 2-候選瀏覽序列集，且也不容易在執行過程中縮小學習序列資料庫的大小，使得執行效率有時候會比 FDLP 演算法差一些，直到重複百分比大於 40%，由於 MFR 結合 FS 演算法產生的頻繁序列之最大長度為五，所以只掃描五次資料庫，比 FDLP 演算法掃描八次來得少，以致於執行時間相對又會比較快。

表十三: 三種演算法所產生頻繁序列的最大長度

資料庫	Apriori All	FDLP	MFR 結合 FS
D100-I15-**R10**-J3-Ns8	9	7	7
D100-I15-**R20**-J3-Ns8	8	7	6
D100-I15-**R30**-J3-Ns8	7	8	6
D100-I15-**R40**-J3-Ns8	6	8	5

FDLP 演算法所要發掘出的頻繁瀏覽路徑，是在網站上有直接連結的關係，以便提供學習者在網路學習環境中迅速得到快而立即的學習資訊，所以在拆解學習序列的過程中，不需要將所有可能的瀏覽序列組合都拆解出來，而 AprioriAll 演算法則是將所有可能組合的瀏覽序列都拆解出來。FDLP 演算法所發掘出來的頻繁瀏覽序列雖然沒有 AprioriAll 演算法發掘出來的多，但卻能符合在網際網路教學網站環境下的應用，而 Apriori All 演算法除了候選瀏覽序列集可能過多以外，每一次的學習序列的拆解次數也非常龐大，因此，在執行的效能上，FDLP 演算法會比 Apriori All 演算法快很多。

另外，我們先舉一個例子來比較 FDLP 與 MFR 結合 FS 演算法所發掘之資訊的差異，假設有一位學習者的學習序列為：{ABCDEFDGHIGJDK}，網站的局部結構為圖三，我們分別利用 MFR 結合 FS 演算法與 FDLP 演算法針對此筆學習序列，拆解出其所有可能的長度為 3 之瀏覽序列，比較如表十四所示。

表十四: 執行 MFR 結合 FS 與 FDLP 演算法拆解學習序列的方式之比較

學習序列: {ABCDEFDGHIGJDK}	
MFR 演算法轉換後的學習序列	{ABCDF} {ABCDGHI} {ABCDGJ} {ABCDK}
FDLP 演算法 (保留原始學習序列)	{ABCDEFDGHIGJDK}
FS 演算法拆解出的瀏覽序列	{ABC}, {BCD}, {CDE}, {DEF}, {CDG}, {DGH}, {GHI}, {DGJ}, {CDK}
FDLP 演算法拆解出的瀏覽序列	{ABC}, {BCD}, {CDE}, {DEF}, {EFD}, {FDG}, {CDG}, {DGH}, {GHI}, {HIG}, {IGJ}, {DGJ}, {GJD}, {JDK}, {CDK}, {FDK}

我們發現同一個學習序列{ABCDEFDGHIGJDK}中，在經過 MFR 結合 FS 演算法拆解後，會遺失在網站上有直接連結的瀏覽序列{EFD}、{FDG}、{HIG}、{IGJ}，{GJD}、{JDK}及{FDK}，導致所挖掘出的資訊也會有所遺失，而這些瀏覽序列利用 FDLP 演算法可全部拆解出來，這些瀏覽序列的重要性，我們舉例如下：因為在瀏覽 K 網頁之前，我們不能確定學習者是因為看過 C 網頁、F 網頁或是 J 網頁才會去瀏覽 K 網頁，所以{FDK}與{JDK}這兩個瀏覽序列也應該被拆解出來。

再者，由於 2-候選瀏覽序列集可能非常龐大，FS 演算法利用建立及檢查 Hash Table 的方式來減少 2-候選瀏覽序列的個數，而 Hash Table 的大小會影響到刪減 2-候選瀏覽序列的多寡，如果當網頁的數量不多，而學習序列資料庫很大時，使得頻繁瀏覽序列過多，會造成 FS 演算法不容易裁減 2-候選瀏覽序列集及縮小學習序列資料庫的大小，如表十二的實驗結果。FDLP 演算法則是利用網站本身的架構來減少 2-候選瀏覽序列集，因為在教學網站的環境下，每兩兩相連結的網頁必定有其關聯性，以達到其教學的目的；而兩兩沒有相連結的網頁，則視為彼此沒有關聯性的網頁，因此，透過網站的架構，我們可以直接刪除一些不必要的 2-候選瀏覽序列集。FDLP 演算法針對在教學網站上的應用，除了求出有直接連結的頻繁瀏覽序列集外，也允許有重複的網頁在頻繁瀏覽序列中，如此才能表現出學習者完整的學習行為。

4. 結論

在本篇論文中，我們提出了一個在網際網路教學網站的環境下能輔助學習者有效學習的資訊—有直接連結之非簡單的頻繁瀏覽序列。直接連結的頻繁瀏覽序列，可以提供學習者在網路學習環境中得到快而立即的學習參考路線，而非簡單的頻繁瀏覽序列，能夠表現出學習者完整的學習行為。另外，我們提出 FDLP 演算法來挖掘出所有的有直接連結之非簡單頻繁瀏覽序列，此演算法利用網站本身的架

構大量減少長度為 2 的候選瀏覽序列，並改善了最大向前參考演算法切割學習序列及 FS 或 SS[4]演算法找尋頻繁瀏覽序列可能產生之資訊遺失的問題。

在未來研究方面，我們提出兩個方向：第一個方向是學習者的分群[15][16]，透過對學習者分群，我們希望挖掘出局部的熱門路線，以便針對不同類型的學習者提供不同的學習方式，進而吸引非學習會員的加入與參與。由於大多數的學習者學習領域不盡相同，我們先利用學習者的學習行為進行學習者分群，便可以透過每一群組的學習特性，找出此群組學習者的學習行為，以瞭解每一群學習者在學習過程中的局部瀏覽序列，也就是局部的熱門路線。

第二個方向是挖掘出科目之間的關聯性[14]，由於整個教學網站可能不只由一位專家所設計，且網站著重的是課程內容網頁(content-page)，而索引網頁(index-page)扮演的僅是輔助課程學習的角色，有時候我們可以加以忽略索引網頁(index-page)，針對專家與專家之間所設計的課程內容網頁，挖掘出隱藏在某些科目之間的關聯性，透過學習者的學習行為，我們可以找出某個科目的學習網頁與另一個科目的學習網頁存在的關聯性，而挖掘出來的資訊，可以建議網站設計者改善網站架構及提昇網站的整體學習效能之考量。

誌謝

這篇論文的研究成果是國科會計劃（NSC-90-2213-E-030-003）的一部份。我們在此感謝國科會經費支持這個計劃的研究。

參考文獻

[1] R. Agrawal and et al. “Mining Sequential Patterns.” In *Proceedings of International Conference on Data Engineering*, pages 3-14,1995.

[2] R. Agrawal and et al. “Mining Sequential Patterns: Generalizations and Performance Improvements. “ In *Proc. of the Fifth Int'l Conference on Extending Database Technology (EDBT)*, Avignon, France, pages 3-17, March 1996.

[3] M.S. Chen , X.M. Huang, I.Y. Lin ” Capturing User Access Patterns in the Web for Data Mining.” *Proc. of the 11th IEEE International Conference Tools with Artificial Intelligence*, November 9-11, pages 345-348, 1999.

[4] M.S. Chen. “Efficient Data Mining for Path Traversal Patterns.” In *IEEE Transactions on Knowledge and Data Engineering* , pages 209-220, 1998.

[5] M.S. Chen , J.S. Park , and P.S. Yu. *“*An Effective Hash-Based Algorithm for Mining Association Rules.” In *Proceedings of ACM SIGMOD*, 24(2): 175-186, 1995.

[6] M.S. Chen,C.H. Yun, “Mining Web Transaction Patterns in an Electronic Commerce Environment.” *Proc. of the 4th Pacific-Asia Conf. on Knowledge Discovery and Data Mining*, pages. 216-219, April 18-20, 2000.

[7] J. Cho , N. Shivakumar , G.M. Hector. ” Finding replicated Web collections” In *ACM SIGMOD*, pages 355-366, 2000.

[8] R. Cooley , B. Mobasher , and J. Srivastava. ” Data Preparation for Mining World Wide Web Browsing Patterns” In *IEEE knowledge and information system* 1999, pages 5-32.

[9] J. Han, J. Pei, Y. Yin. “Mining Access Patterns Efficiently from Web Logs.” In *PAKDD*, pages 396-407, 2000.

[10] J. Han , J. Pei , M.A. Behzad. ”Frequent Pattern-Projected Sequential Pattern Mining.” In *ACM SIGMOD* ,pages 355-359, 2000.

[11] Jiawei Han, Jian Pei, Yiwen Yin: Mining Frequent Patterns without Candidate Generation. ACM SIGMOD, pages 1-12, 2000.

[12] C. Hidber. “Online Association Rule Mining” In *ACM SIGMOD*, pages 145-156, 1999.

[13] K. Joshi , A. Joshi, Y. Yesha and R. Krishnapuram. “Warehousing and Mining Web Log” In *ACM*

Workshop on Web Information and Data Management, pages 63-68, 1999.

[14] C.H. Lee, H.C. Yang. "A Web Text Mining Approach Based on Self-organizing Map" In *Workshop on Web Information and Data Management* , pages 59-62, 1999.

[15] M. Perkowitz , O. Etzioni. "Towards adaptive Web sites: Conceptual framework and case study" In *Artificial Intelligence Volume 118, Number 1-2 April*, pages 245-275*,* 2000.

[16] M. Perkowitz , O. Etzioni. " Adaptive Web Sites Conceptual Cluster Mining" In *IJCAI* , pages 264-269, 1999.

[17] S.J. Yen and A.L.P. Chen. "The Analysis of Relationships in Databases for Rule Derivation." In *Journal of Intelligent Information Systems*, Vol. 7, No. 3, *November*, pages 232-260, 1996.

[18] S.J. Yen and A.L.P. Chen. Chen. "An Efficient Approach to Discovering Knowledge from Large Databases." In *Priceedings of the International Conference on Parallel and Distributed Information Systems*, pages 8-18,1996.

[19] S.J. Yen "Mining Frequent Traversal Patterns in a Web Environment." In *Proceedings* of International Symposium on Intelligent Data Engineering and Learning , pages 219-224, 1998.

[20] S.J. Yen and C.W. Cho. "An Efficient Approach to Discovering Sequential Patterns from Large Databases," *Lecture Notes in Artificial Intelligence: Principles of Data Mining and Knowledge Discovery (PKDD)*, pages 685-690, 2000.

[21] S.J. Yen and A.L.P. Chen. "A Graph-Based Approach for Discovering Various Types of Association Rules," *IEEE Transactions on Knowledge and Data Engineering*, to appear, 2001.

[22] S.J. Yen and C.W. Cho. "An Efficient Approach for Updating Sequential Patterns Using Database Segmentation," *International Journal of Fuzzy Systems Special Issue on Soft Computing and Data Mining*, pages 422-431, 2001.