第參章 網頁-動作路徑的資料挖掘演算法

3.1 網頁-動作路徑

本研究主要的分析目標是全球資訊網中使用者所瀏覽網頁及其在瀏覽網頁上所做的動作。所以本節主要要來介紹網頁-動作路徑的資料結構及其所擁有的一些特性。首先要說明的是，本文假設使用者所瀏覽的網頁及動作的記錄，可以藉由網頁系統的記錄程式、Cookie 或會員制等方式將其留存在系統端裡。

3.1.1 網頁-動作路徑的資料結構

基本上，網頁-動作路徑資料是由二部份的 Log 所組成的。一部份是記錄使用者曾經瀏覽過那些網頁的 Log，另一部份是記錄使用者曾經運作過那些動作的動作記錄檔。表 3-1、表 3-2 顯示了這二個 Log 檔會收集的資料。

表 3-1 使用者瀏覽網頁路徑的 Log

Page_Log			
Session_Id	**Page_Id**	**StartTime**	**EndTime**
S001	A	2004021	200402100010
S001	C	200402100011	200402100030
S001	B	200402100031	200402100050
…	…	…	…

表 3-2 使用者瀏覽動作的 Log

Action_Log		
Session_Id	**Action_Id**	**ExeTime**
S001	b	200402100003
S001	a	200402100009
S001	c	200402100018
S001	a	200402100025
S001	b	200402100027
S001	b	200402100040
…	…	…

當瀏覽網頁的使用者進入網頁後，系統會給與這個使用者任務一個獨特的序列號，也就是表 3-1、表 3-2 中的 Session_Id。這個序列號可用來判定那些網頁及動作的瀏覽記錄資料是屬於同一次瀏覽任務所產生的。經由比對記錄資料，我們可以將原始的記錄資料利用時間排列出所需的序列資料。以表 3-1 與表 3-2 中可見的資料為例，我們可以排列出<AbaCcabBb>這樣的序列資料，並將其儲存在瀏覽及動作路徑序列的資料庫中(表 3-3)。

表 3-3 瀏覽頁面及動作路徑序列的資料庫

MainSequence	
Session_Id	**PA_Sequence**
S001	AbaCcabBb
S024	AcBabCaEacFx
…	…

將 Log 轉換成序列的邏輯做法如下：

```
ADD {SELECT DISTINCT Session_Id FROM [Page_Log]} into SList;
//將不同的 Session_Id 加入 SList 這個 List 中。
//List 這個物件有一個指標，初始時指在第一個內容的前一筆資料。
While ( 如果 SList 指標後還有資料 ) {
  將指標指向 SList 中的下一個 Session_Id;
  STemp="";
  PList.Clear;                    //將 STemp 與 PList 二個變數清空。
  ADD {SELECT Page_Id,StartTime,EndTime FROM [Page_Log] WHERE _
       Session_Id = SList.Session_Id ORDER BY EndTime} into PList;
  //將同個 Seesion_Id 的 PageCode 與時間資訊加入 PList 中。
  While ( 如果 PList 指標後還有資料 ) {
    將指標指向 SList 中的下一個內容;
    STemp=STemp + PList.Content;
    //先寫 PageCode 進入 STemp 中。
    ADD {SELECT Action_Id FROM [Action_Log] _
           WHERE Session_Id = SList.Session_Id _
             AND ExeTime > PList.StartTime _
             AND ExeTime < PList.EndTime _
           ORDER BY ExeTime} into AList;
    //將同一 Seesion 且執行時間在這個 Page 的開始與結束時間的動作碼加入 AList
    While (AList.Index.haveNext) {
      AList.Index.MoveNext;
      STemp=STemp + AList.Action_Id;
      //再將 ActionCode 寫入 STemp 中。
    }
  }
  INSERT INTO MainSequence(Session_Id,PA_Sequence) VALUES(SList.Session_Id,STemp);
  //完成一筆資料的轉換，將資料寫入資料庫中。
}
```

由以上作法可知，PA_Sequence 是不會出現<AbcAca>這樣的序列組合的。如果動作 bc 與 ca 都是在同一頁 A 所做，且他們是上下頁的關係的話，我們是會將

其視為在 A 頁當中連續運作了 bcca 這些動作。以下我們會討論一些與 PA_Sequence 相關的規則：

PA_Sequence 是由 Page_Id 以及 Action_Id 所組成的。假設 P 是所有的 Page_Id 的集合，A 是所有 Action_Id 的集合，S 是所有可能的序列組合，則：

Rule 3.1.1 任一個 $s \in S$ ，s 中不會存在沒有所屬頁碼概念的動作。

Rule 3.1.2 任一個 $s \in S$ ，s 中不會存在完全沒有動作的頁碼。

也就是說，不可能會出現有如<aAbc>或是<ABca>這樣的序列。這二個規則我們也可以說成：當用頁碼來切割一個 PA_Sequence 的時候。一定能將這個 PA_Sequence 轉換成 $p_1 + A_1 + p_2 + A_2 + \ldots + p_n + A_n$ 這類的模式。其中 p_1、p_2、…p_n 屬於 P 中的一個 element，而 A_1、A_2、…A_n 則是 A 中不為空值的子集合。這裏我們所談的+符號，代表的是直接連結(concatenate)運算，也就是說<Aab>+<Cc>的結果即為<AabCc>。

3.1.2 網頁-動作路徑的 Path 與長度

如果我們忽略 PA_Sequence 中所有的動作碼，會產生出使用者的網頁瀏覽路徑。假設我們的 PA_Sequence 的資料為 s，則 PAGE(s)就代表這個序列所瀏覽的網頁路徑。相反的，如果我們忽略 PA_Sequence 中所有的網頁碼，則會產生出使

用者所執行動作的路徑，我們以 ACTION(s)來代表這個動作路徑。如前文所述，PAGE(s)並不會出現二個頁碼相同且相鄰的狀況，但 ACTION(s)卻有可能出現二個動作碼相同且相鄰的狀況。

因為 PA_Sequence 有網頁瀏覽路徑及動作路徑這二個維度，所以它會有多種的長度計數方式。一是由瀏覽網頁的路徑 PAGE(s)來計算，另一則是由動作路徑來計算 ACTION(s)。舉例來說，當 $s \in S$ ， $s =< AabBacCba >$ 時，則 $|s|_P$ 稱為 s 的網頁路徑長度，在此例中為 3；$|s|_A$ 稱為 s 的動作路徑長度，此例中為 6，$|s|_P + |s|_A$ 則稱為序列 s 的絕對長度，我們用這個 $\|s\|$ 符號來表示，在這個例子中，s 的絕對長度為 9。

3.2 網頁-動作路徑演算法(Page-Action Algorithm)

我們的演算法與 Apriori(Agrawal and Srikant,1994)求關聯規則一樣，需要求取各個候選子序列(Candidate Sequence,CS)的 support 值。再判斷那些候選子序列的 support 值是否超過門檻值(Large Threshold)，藉以產生所謂的 Large Sequence(LS)。完成了 LS 之後，接著用 LS_k 產生 CS_{k+1}。擁有 CS_{k+1} 之後，再重覆以上動作，直到不能再找出 LS 為止。我們的研究是基於這種的概念，而發展出能配適在上節所述資料結構的新演算法。以下我們將演算法中的幾個重要元件與動作做分別的闡述。

3.2.1 網頁-動作路徑的 Pattern (P-A Pattern)

在我們的研究中，Pattern 就是可能成為 Candidate Sequence 以及 Large Sequence 的元件。Pattern 與 Log 資料雖然一樣都是由頁面碼及動作碼所組成；但是 Pattern 的解釋及架構卻和 Log 資料有明顯的不同。

首先我們先定義一個 Pattern 的基本構成要素：

Definition 3.2.1.1 一個頁面碼加一個動作碼所組成的單元稱為「PA 元素。而 Pattern 則是由一或多個 PA 元素所組合而成的。

基於 Definition3.2.1.1 所說，我們可以很容易的知道一個 P-A 的 Pattern，他的絕對長度一定是偶數。

依 Definition3.2.1.1 所定義的，[AaBb]可以是一個 Pattern，這個 Pattern 在本研究的解釋意義是指「使用者在 A 這一頁中曾執行過動作 a，且 A 頁瀏覽後下一頁是 B，在 B 頁曾執行過動作 b。」

一個 Pattern 的網頁路徑是指這個 Pattern 會走過的網頁瀏覽順序。由於 Pattern 與 PA_Sequence 不同，是可能出現[AaAb]這種模式的。所以要求出一個 Pattern 的網頁路徑，除了要去除動作碼之外，還要將相連重覆的網頁碼去除。以下是找出 Pattern 網頁路徑的演算方法：

```
String PATH(PAT as String) { //此函數輸入要找 PATH 的 Pattern，會傳出路徑碼。
    PATPageCode as String;
    For i = 0 to (PAT.lengh()/2)-1    //迴圈執行該序列擁有的 PA 元素數次
     if (i> 0) {
         If (PAT[2*i]==PAT[2*(i-1)] ) {Continue;}
         //如果頁碼相同就不增加頁碼
     }
     PATPageCode=PATPageCode+PAT[2*i];
     //增加頁碼
    Next i
    Return PATPageCode;
    }
```

3.2.2 網頁-動作路徑的 Pattern 掃描(P-A_ScanX)

在我們產生候選子序列之後，要去計算整個資料庫，共有幾組滿足這個候選子序列所陳述的狀況。這就叫做掃描。一般 Sequence Pattern 掃描可分兩種，一種是只考慮 element 是否與 pattern 出現在序列的順序一樣，不考慮其間的連續出現性。以[ACE]這個 Pattern 為例，Sequence Pattern 中的掃描會認為序列<ABCDE>與<ACEBD>都是滿足[ACE]這個 Pattern。另一種掃描法，它考慮的就是這個 Pattern 必須完整連續出現在序列中，才算是滿足。同樣以[ACE]這個 Pattern 為例，在網頁瀏覽路徑的掃描中，<ACEBD>是滿足[ACE] Pattern 的，而<ABCDE>則不滿足。

連續出現的掃描模式，一般使用於只想找某個連續出現關係的時候。由於掃描要連續出現才算滿足，所以能找出的規則就只在連續出現的關係上。而不需連續出現的掃描模式，則可以找出某個序列先後關係的規則。但是，因為強調先後

關係，對於連續出現的規則，就無法特別的指出來了。

在我們的研究裏，也面臨了網頁與動作二個維度該選用什麼掃描方式的問題。表 3-4 顯示了四種可能的掃描選擇。

表 3-4 四種可能的掃描方式

網頁維度 / 動作維度	連續出現	前後出現
連續出現	Type Ⅰ	TypeⅡ
前後出現	TypeⅢ	TypeⅣ

在選擇要使用的掃描方法前，我們可以先分析一下這四種方式各會找出具有什麼解釋意義的規則。假設我們找到了一個 AbBc → AbBca 50%這樣的規則。如果這個規則是由 Type I 的掃描方式中找到，我們會將其解釋為「當使用者在 A 頁中做了 b 動作，且在下一頁 B 頁中做了 c 動作，使用者有 50%的機率在 B 頁的 c 動作做完後接著做 a 動作。」

如果規則是由 TypeⅡ的方式找到，我們會將規則解釋為「當使用者已在 B 頁前曾在 A 頁中做了 b 動作，之後在 B 頁做了 c 動作時，將有 50%的機率使用者會在做完 c 之後直接做 a 動作。」

如果規則是由 TypeⅢ的方式找到，我們則會將規則解釋為「當使用者已先在 A 頁中做了 b 動作，之後在 B 這一頁做了 c 動作時，將有 50%的機率使用者之後會在同頁做 a 動作。」

最後，如果規則是由 TypeⅣ的方式所找到，我們則會將規則解釋為「當使用者已在 B 頁前曾在 A 頁中做了 b 動作，之後在 B 頁做了 c 動作時，將有 50% 的機率使用者之後會在 B 頁做 a 動作。」

理論上，這四種方法都是可以研究的。但是在目前本文所鎖定的範圍中，我們選擇了一個較合適也較有趣的方式來掃描。由於我們希望找出可提供重組網站結構或廣告配置參考的規則。在這個前提下，我們認為網頁路徑的掃描方法應以概念較固定、強調同時出現路徑的連續掃描較好。網頁採用「連續出現」的掃描方法，可以找出某一張網頁的上下頁關係，對於改善網站配置應較有幫助。也可以避免因為網頁瀏覽路徑過長，找到許多較無關緊要的先後關係。

另一方面。針對現有網站架構與使用者一般瀏覽網站的行為看來，使用者在一張網頁中，可看到多個可實行的動作，使用者操作這些動作的自由度相對於網頁的瀏覽來的高的多。對於一張網頁中動作配置的討論，也大概只有方便與操作性，並不一定有重要的先後關係。另外以現在一般網頁的設計看來，使用者常常做了一二個動作，就可能被引導到其它的呈現頁去。每頁可收集到的動作數目，恐怕只會有二、三個。綜合以上兩點，在本文中，我們採用在概念上較寬的「不連續掃描」作法來處理動作路徑。除了較合乎動作在網頁中可能的不連續性外，也可以增加找出的規則。對於想瞭解使用者的行為，應較有幫助。

所以，本研究採取 TypeⅢ二種模式混合存在的掃描方式。在網頁元件(序列中屬於網頁代碼的元件)間的關係，我們採用「連續出現」的掃描法，而網頁中

的動作元件(序列中屬於動作代碼的元件)間的關係，我們則採用「不連續出現」的模式來掃描。在 3.4 節中，我們會用較類似真實例子的資料來說明，這樣的選擇應該是合理的。實際掃描的方式，我們用表 3-5 舉例說明。

表 3-5 部份 MainSequence 資料庫的資料

Session_Id	PA_Sequence
S005	AacbBacAbc
S010	AaBcabCac
S015	BcaCabcAcb
S020	AcbCacbcaBca

假設我們產生了一個[AcBc]的 Pattern，這個 Pattern 的意思代表「在 A 這頁瀏覽時，使用者有做過動作 c，他下一頁瀏覽 B 頁，而且使用者做了 c 動作。」依造這樣的說法，由 Table3.2.2 中，我們知道，只有 S005 滿足這樣的條件。值得一提的是，S020 中，雖然在 A 頁中使用者有做 c、在 B 頁中使用者也有做 c 的動作，但是因為這二頁中使用者還到過 C 頁，所以並不滿足我們的掃描法則，所以不算 support 值。

另一種可能會產生的 Pattern 是[AcAb]這類的 Pattern，這類 Pattern 是由二個(以上)一樣的頁面代碼相鄰所產生的 Pattern。我們解釋他的意義為「同樣在 A 這一頁，使用者曾經先做過 c 這個動作再做 b 這個動作。」依造這樣的說法，S005、S015、S020 皆滿足這樣的掃描法則。

P-A_Scan 依計算 support 的方式不同，區分成 PA_Scan1 與 PA_ScanN 二種演算法，PA_Scan1 是指滿足條件的資料，不論單筆資料其中可滿足多少次的 Pattern

數，皆算一次 support 值。而 PA_ScanN 的演算法，則可對滿足多次 Pattern 的資料，計算對應次數的 support 值。例如：Pattern : [AaBb] 在資料<AabBabCcAaBb>中，採用 PA_Scan1 只算一次的 support，但是如果在 PA_ScanN 的演算規則之下，將計算二次的 support 值。在這裏要特別提出來的是，如 Pattern : [Aa]在序列<AabaBacAac>中，採用 PA_ScanN 描掃法的話，Support 只會計算 2 次，因為[Aa]的解釋是在 A 頁中做了 a 動作的模式，在第一頁的序列中<Aaba>已滿足了在 A 頁中做了 a 的動作，雖然做了 a 的動作二次，但是在我們的研究中，不去計算動作部份的加權值。而最後一段的子序列<Aac>又滿足了一次 Pattern。所以這個任務的 Support 計算了 2 次。

以下是 PA_Scan1 與 PA_ScanN 兩種掃描法的邏輯做法：

P-A_ScanX(PAT as String) as Integer {　//本函數會傳入欲比較的 Pattern，會傳回 Support 值

```
Add { SELECT PA_Sequence AS PS FROM MainSequence } Into PAList;
//將每筆的序列 Record 加入 PAList 這個 List 中
 PATPageCode=PATH(PAT);          //進行 Pattern 的 PATH 轉換。
 Count=0;

 While (如果 PAList 還有沒讀過的資料) { //開始讀取所有的序列資料
    將指標移到 PAList 沒讀過的資料;

    If (PLengh(PAList.PS)< PATPageCode.lengh() ) {Continue;}
    //如果序列資料的網頁路徑長度比 Pattern 的長度短，就不用計算，直接再找下一筆。

    PAtemp as String;
    PAtempC[] as Integer;
    j = 0; k = 0;
```

```
//開始將序列資料的網頁路徑儲存到 PAtemp 這個字串裏
//而在 PAtempC[]中，則儲存對應頁碼的 ActionCode 數目。
 For i = 0 to PAList.PS.lengh()-1     //一個碼一個碼的讀
   If (PAList.PS[i]  是 PageCode ) { //如果讀進來的碼是 PageCode
     If (k>0) {
       PAtempC[j]=k;                  //將上一個碼的 ActionCode 數目儲存
       k=0;
       j=j+1;
     }
     PAtemp= PAtemp + PAList.PS[i];   //將這個網頁碼存到字串裏
   }
   Else { k=k+1;}
 Next i

 If ( Exist(PAtemp,PATPageCode)==0 ) {Continue;}
 //如果序列的網頁路徑中不存在 Pattern 的網頁路徑，就不用計算，直接再找下一筆序列。

 k=0;
 key = 0;
 For i = 0 to lengh(PAtemp)-1   //開始正式進行比對
     key = 0;

     If (PAtemp[i]==PATPageCode[0]) { //判斷此頁是否為可能的符合 Pattern 的第一頁。
         Key=1;
         For j=1 to PATPageCode.lengh()-1   //判斷之後幾頁是否也符合 Pattern 所需的頁碼
           If (PAtemp[i+j]!=PATPageCode[j]) {Key=0;}
         Next j
     }
     //如果在此時，Key 為 1，代表序列裏第 i 個頁面存在一個可能符合 Pattern 的連續頁
     IF (Key==1) { //進入可能的連續頁裏
         K2 = 0;
         For j = 0 to PAT.lengh()/2-1   //頁面指標
           For o = 1 to PAtempC[j]-k2   //動作指標
             If (PAList.PS[k+o+k2] == PAT[2*j+1]) {   //一個動作碼比對成功
                 If (PAT[2*j]== PAT[2*j+2]) {
                     k2=o;
                 }
```

```
                    Else { k2=0; }
                    Key=Key+1;     //比對成功一碼，加一個 Key 值
                  }
                Next o
                If ((Key-1)!=j+1) {j=PAT.lengh()/2;}
            //如果一個頁面的動作碼都比對完了，但是尚未比對成功，則這個連續頁失敗。
                If (PAList.PS[k+o+k2] != PAT[2*j+1]) {k=k+1+PAtempC[j];}
              Next j
              IF ((Key- PAT.lengh()/2)==1) {
                Count=Count+1;     //Support + 1
                i = lengh(PAtemp);
                //當演算是是 PA_Scan1()時，要加這一行。
                //反之，如果演算法是 PA_ScanN()時，不要加這一行，就會重覆算 support。
               }
              //已比對成功，Counter 加一。
            }
            Else {k=k+1+PAtempC[j];}
            //移到下一頁的位置碼。
        Next i
      }
 Return Count;          //將符合的總數傳回去
 }
```

3.2.3 網頁-動作路徑的門檻值(Large Threshold in P-A Algorithm)

我們採取與傳統相同的方式，一個 Pattern 在資料庫中出現的次數要超過門檻值，我們才稱這個 Pattern 是 Large。對應於 3.2.2 節中，所說的二種 Scan 法，在訂定門檻值也有二種不同的做法。

如果我們採用 PA_Scan1(不重覆計次的掃描法)，我們採用與(Agrawal and Srikant, 1994)中將門檻值訂為資料庫中資料總數的百分比值，這一 Large Threshold 是一個固定的數值。

如果我們採用 PA_ScanN(重覆計次的掃描法)，我們認定 Large 的方式與學者陳仕昇、許秉瑜與陳彥良(民 88)採用的 Next Pass Large Threshold 與 Next Pass Large Sequence 類似，首先，我們得先對序列資料庫計算所有的可能 Support，再利用其所有可能的 Support 值來設定 Min. Support。在 PAScanN 這裏有一點必須要說明的，在我們的研究中，每一階掃描所產生出的 Pattern，雖然 PA 元素的數目都一樣，但是 Pattern 的網頁路徑長度並不會一樣。舉例來說，Pattern [AaBa] 與[AaAc]都是可能在第二階掃描中產生出的 Pattern，但是[AaBa]是網頁路徑長度是 2，而[AaAc]的網頁路徑長度是 1。所以在我們的 PA_ScanN 演算法中，會發生同一階裏，會對應到不同的 Min. Support 值。為了避免過多不必要的計算，我們採用維護一個名叫 Total Possible Support 的資料庫，這個資料庫儲存著序列資料庫在特定的網頁路徑長度時，所擁有的所有可能 Support 數。這個最大的 Support 數乘上百分比型態的 Min. Support 值，就可做為特定網頁路徑長度的門檻值了。

3.2.4 網頁-動作路徑的 Join (P-A_Join)

本研究的產生 Candidate Sequence 的方式採用類似(Srikant and Agrawal 1996) 的 join 方法，也就是利用 LS_k join LS_k 產生 CS_{k+1}。我們稱 LS_k 為原生 Pattern；而稱 CS_{k+1} 為生成 Pattern。以下我們定義原生 Pattern 與生成 Pattern 的關係

Definition 3.2.4.1 X 與 Y 是二個 Pattern，如果 Z 是 X 與 Y 共同生成的 Pattrn，則 Z 必然同時滿足 X 與 Y 的各別意義。

因為本研究的特殊問題，所以我們必須採用較特別的方法實作 Join 的動作。我們發現二個 Pattern 必須是可接合或融合的，才可以 Join 出生成 Pattern。以下我們就可接合與可融合的 Pattern 分項討論：

3.2.4.1 一對可接合的 Pattern

當一個 Pattern 去掉頭一個 PA 元素(包含一個網頁碼及一個動作碼)，與另一個 Pattern 去掉末一個 PA 元素是完全相同時，我們稱這對 Pattern 是一對可接合的 Pattern，比如[AbBa]與[BaCc]的 Pattern 或是[AbBcBa]與[CcAbBc]，這些 Pattern 可以頭尾接合而產生如[AbBaCc]或是[CcAbBcBa]這種比他原生 Pattern 多出一個 PA 元素的新 Pattern。這是 P-A_Join 生成下一階 CS 的第一種做法。

3.2.4.2 一對可融合的 Pattern

P-A_Join 生成下一階 CS 的第二種做法，我們稱之為融合。這種情形則發生於一個 Pattern 的網頁路徑是另一個 Pattern 網頁路徑的子序列，而且這二個 Pattern 只有一個 PA 元素是不同的。就像是[AbBa]與[AcBa]的 Pattern 或是

[AbCaBa]與[AbCbBa]這樣的 Pattern。這類型的 Pattern，我們處理的方式是融合，將相同的部份保留，然後在不同的位置上分別加入不同的部份。加完了之後，還要再考慮排列的問題。就像是第一對的 Pattern 會產生[AbAcBa]與[AcAbBa]這二個下一階的 CS。而第二組的二個 Pattern 則會產生[AbCaCbBa]與[AbCbCaBa]這二個 CS。

接下來我們將証明，任一個第三階以上的 pattern，定是由上一階的兩個父母 pattern，由接合或融合這二種方法其中之一所生成的。

假設 pattern Z 是任一個 k 階 pattern，X、Y 則是 k-1 階的 pattern，X 與 Y 是生成 Z 的父母。由於 X 與 Y 屬於 k-1 階，所以 X 與 Y 是由 k-1 個 PA 元素所組成的。而 Z 是 k 階的 pattern，所以 Z 是由 k 個 PA 元素所組成。按照 Definition 3.2.4.1 的定義，pattern Z 會滿足 X 與 Y 個別的意義。因此 X 是 Z 差一 PA 元素的子集合。Y 也是 Z 差一 PA 元素的子集合。

假設 Z 的生成模式，不是接合也不是融合。也就是說 X 與 Y 的網頁路徑，如果有一個是另一個的子集合，X 與 Y 就不會只差一個 PA 元素(情況一)。如果當一個並不是另一個的子集合時，一個 pattern 去掉頭個 PA 元素並不會與另一個 pattern 去掉最後的 PA 元素相等(情況二)。

當 X 與 Y 的狀況是情況一時，因為 X 與 Y 與皆是 Z 去掉一個 PA 元素的子集合，所以 X 與 Y 最多只可能差一個 PA 元素，與假設矛盾。所以，如果 X 與 Y 是情況一時，Z 一定是接合或融合所生成的。

當 X 與 Y 的狀況是情況二時，因為 X 與 Y 最多只可能差一個 PA 元素，而且並不是在一個的頭與另一個的尾。所以 X 與 Y 有差距的部份必定在 pattern 的中間或者在兩方的頭或尾。如果差異處在中間，我們假設 pattern X = (α)X_pX_a(β)、pattern Y= (α)Y_pY_a(β)，其中(α)是 X 與 Y 相同 PA 元素的前半部份，而(β)是 X 與 Y 相同 PA 元素的後半部份；而 X_p 與 Y_p 代表不同 PA 元素的 PageCode；而 X_a 與 Y_a 代表不同 PA 元素的 ActionCode。也就是說，X、Y 的樣子就類似(AaBc) X_pX_a (Cc)、(AaBc) Y_pY_a (Cc)。因為 X 與 Y 一個的網頁路徑並不是另一個的子集合，所以 X_p 與 Y_p 除了一定不能相同之外，二個 PageCode 也不能同時屬於(α)最後一個 PA 元素的 PageCode 或是(β)最前一個 PA 元素的 PageCode。因為 Z 是由 X 與 Y 生成出來的，所以 Z 必須同時滿足 X、Y 的各別意義。如果(α)=(β)，則 Z 的最簡單型式為 (α)X_pX_a(α)Y_pY_a(α)，Z 的長度必然大於 k，與假設矛盾；如果(α)≠(β)，則 Z 的最簡單型式為 (α)X_pX_a(β)(α)Y_pY_a(β)，Z 的長度亦必然大於 k，與假設矛盾。如果差別之處在二方的頭或尾，假設差別處在二方的頭 pattern X = X_pX_a(β)、pattern Y= Y_pY_a(β)。因為一個的網頁路徑不會是另一個的子字串，所以 X_p 與 Y_p 不能相等，而且 X_p 與 Y_p 不能等於(β)第一個 PA 元素的網頁碼。所以，Z 的最簡單型態是 X_pX_a(β) Y_pY_a(β)，Z 的長度也是必然大於 k，與假設矛盾；同理可証，差別處在二方尾部時，Z 的長度也是必然大於 k 與假設矛盾。所以，如果 X 與 Y 是情況二時，Z 也一定是接合或融合所生成的。綜合情況一與情況二來看，Z 定是 X、Y 由接合或融合所生成的。

3.2.4.3 P-A_Join 的過程

一開始系統會進行兩、兩比對的工作。比對完後就依類別進行接合或融合的動作，並將生成出的 Pattern 加入CS_{k+1}中。當所有的可能都掃描過後，還要清除CS_{k+1}中相同的資料，才算完成 P-A_Join 的工作。

整個 P-A_Join 的演算邏輯如下：

```
P-A_Join(LSList) as CSList {   //本函數需輸入 LSList，會產生 CSList

  For i = 0 to LSList.Index.Count-1
    For j = i to LSList.Index.Count-1
      If (比對二 Pattern 是否只有一 PA 元素不同) {
        If ( ISSubString(PagePath(LSList.Index[i]) ,PagePath(LSList.Index[j])) OR_
            ISSubString(PagePath(LSList.Index[j]) ,PagePath(LSList.Index[i]))) {
            //如果某一個 Pattern 的網頁路徑是另一個網頁路徑的子序列的話。就是融合。
            進行融合程式;
            將融合結果加入 CSList;
        }
        Else If(不同的 PA 元素，一個在頭一個 PA 元素、另一個在末一個 PA 元素) {
                進行接合程式;
                將接合結果加入 CSList;
        }
      }
    Next j
  Next i

  //以下是將重 CSList 重覆的資料清除
  CSList.Index.MoveFirst;
  While (CSList.Index.haveNext) {
     CSList.Index.MoveNext;
     IF (CSList 的內容不是維一的) {
        將這個 CSList，標示為重覆的。
     }
```

```
    }
    將已標示為重覆的 CSList 移除;
}
```

3.3 網頁-動作路徑演算法的演算實作

表 3-6 是一個較完整的 MainSequence 資料庫的資料，在以下二小節中我們將參造表 3-6 來展示 PAScan_1 與 PAScan_N 這兩種方式的網頁-動作路徑演算法的演算實作。

表 3-6 MainSequence 資料庫的資料

Session_Id	PA_Sequence	$\|s\|_p$
S001	AbcBcCb	3
S002	BcCbAc	3
S003	AbcBcAb	3
S004	BbcCb	2
S005	AcbBaCb	3
S006	CbAbcBcAc	4
S007	AcBcAbBc	4
S008	CaAbcCab	3
S009	AbcBcCaAbc	4
S010	AbCc	2

3.3.1 不重覆計算的演算方式(PA_Scan1 Algorithm)

我們設定 Large Threshold 為 33%，也就是 CS 的 support 要超過 3.3 (10*33%)，才算是 Large。

首先我們先做第 0 階的預先掃描，這個做法在資料變異大時可能可以馬上消除許多種 pattern，留下可能會 Large 的 Pattern 來進行產生 CS1 的工作。這裏我

們介紹兩個抽象化的表示法，一個是[A_]；一個是[?a]，前者的底線可代表任一的動作碼，而後者的問號則可代表任一的頁碼。經過掃描後，我們可以產生第 0 階 Candidate Sequence 如表 3-7。在表中，我們可以發現 pattern [?a]並沒有達到設定的 Large 數目，所以可以不用再考量之後由 a 動作路徑產生出的各種 Pattern。

表 3-7 Candidate Sequence 0 及其 support 值：CS_0

Candidate Sequence	**PA_Scan1 support**
A_	9
B_	8
C_	8
?a	3
?b	10
?c	10

從觀察 表 3-7 的 support 值，我們可以產生 LS_0 如 表 3-8。

表 3-8 Large Sequence 0 及其 support 值：LS_0

Large Sequence	**PA_Scan1 support**
A_	9
B_	8
C_	8
?b	10
?c	10

在 LS_0 中的 Pattern，其實代表著上列的頁面碼與動作碼是值得我們繼續研究的，當資料庫中有許多網頁或是許多動作碼不常使用時，先實作 CS_0 是可以有效減少 CS_1 數量的。隨著資料庫的性質不同，省略 CS_0 直接將頁碼與動作碼做 Join 來產生 CS_1 進行演算也是可以的。這部份在 PA_ScanN 演算法也有類似的想法，我們會在下一小節詳述。

接著我們將 LS_0 中有列的出的頁碼與動作碼以組合方式產生 CS_1，CS_1 如表 3-9。

表 3-9 Candidate Sequence 1 及其 support 值：CS_1

Candidate Sequence	**PA_Scan1 support**
Ab	8
Ac	8
Bb	1
Bc	7
Cb	6
Cc	1

在執行過 PA_Scan1 的掃描後，可以得到相對應 Patern 的 support 值。按造我們的設定，只有 support 超過 3.3 的 Pattern 可以算是 Large，所以 Ab、Ac、Bc 與 Cb 四個 Pattern 是 Large 的，我們產生了表 3-10 的 Large Sequence 1。

表 3-10 Large Sequence 1 及其 support 值：LS_1

Large Sequence	**PA_Scan1 support**
Ab	8
Ac	8
Bc	7
Cb	6

在 LS1 產生後，我們開始重覆使用 LS_{k-1} P-A_Join LS_{k-1} 產生 CS_k，產生出的結果(CS_2)如表 3-11。按造表 3-11 的 CS_2 我們可以再產生出 LS_2，這時 LS_2 只剩下三筆資料，如表 3-12。

表 3-11 Candidate Sequence 2 及其 support 值：CS_2

Candidate Sequence	**PA_Scan1 support**
CbCb	0
CbBc	0
CbAb	1
BcCb	2
BcBc	3
BcAb	0
BcAc	2
AbCb	1
AbBc	1
AbAb	5
AbAc	0
AcCb	5
AcBc	1
AcBc	5
AcAb	1
AcAc	0

表 3-12 Large Sequence 2 及其 support 值：LS_2

Large Sequence	PA_Scan1 support
AbAc	5
AbBc	5
AcBc	5

再重覆產生 CS_3、LS_3、CS_4、LS_4…直到無法產出新的 CS 為止，表 3-13 是其他所有的 CS 與 LS 表。

表 3-13 其他所有的 CS 與 LS 表

CS_3

Candidate Sequence	PA_Scan1 support
AbAbBc	0
AbBcBc	0
AbAcBc	4
AbBcAc	1
AcAbBc	0
AbAbAc	0
AbAcAc	0
AcAcBc	0
AcBcBc	0

LS_3

Large Sequence	PA_Scan1 support
AbAcBc	4

CS_4

Candidate Sequence	PA_Scan1 support
AbAcBcBc	0
AbAbAcBc	0
AbAcAcBc	0

當產生出所有的 LS_x 之後，就可以開始找尋關聯規則了。找尋規則的方法我們在 3.3.3 節會仔細的介紹。

3.3.2 重覆計算的演算方式(PAScan_N Algorithm)

重覆計算與不重覆計算在解釋意義上有很大的不同。重覆計算可以對於在資料庫中出現較頻繁的 Candidate Sequence 計算達到加成的效果(陳仕昇、許秉瑜與陳彥良, 民 88)。在我們的研究中，重覆計算的演算方式與不重覆的演算方式有差別的地方除了 3.2.2 節中已介紹的 Scan 方式有所不同之外，如 3.2.3 節中所介紹，重覆計算的演算法在判斷是否為 Large Sequence 的方式也不相同。以下我們就整個 PAScan_N 演算法做一系列的說明。

在這個任務中，我們設定 Min. Support 為 15%，而且要「往後看一步」。首先，我們得先計算這個序列庫每個 PV 值的總合可能 Support 值(Total Possible Support)。在 表 3-6 中 $|\mathbf{s}|_\mathbf{p}$ 代表 PA_Sequence 的網頁路徑長度。在 $|\mathbf{s}|_\mathbf{p}$ 中的最大值就是這個序列庫的 PV 值上限，在這十筆資料庫裏，這個上限是 4。之後，我們一一計算當 PV=1、2…一直到 4 時，每一個可能的 support 總合值。以 AbcBcCb

為例，當 PV 為 1 時，這個序列可能有[A_],[B_],[C_]三個可能。所以 AbcBcCb 對 PV=1 的貢獻計次就為 3。同樣的，當 PV=2 時，這個序列可能有[A_B_], [B_C_] 兩個可能，所以它對 PV=2 的貢獻計次就為 2。當 PV=3 時，AbcBcCb 只有[A_B_C_]一種可能，

這個序列要 PV=3 的貢獻就只有 1。而當 PV=4 時，該序列沒有可能滿足 PV=4 的 Pattern。所以貢獻是 0。一一將所有的序列計算完畢後，就可能完成 表 3-14 的 TPS 表，而 TPS 與 Min. Support(百分比型態)的乘積就是該網頁路徑所需參造的實際門檻值。

表 3-14 序列庫的 TPS 表

PV	1	2	3	4
TPS	31	21	11	3
TPS*MinS	4.65	3.15	1.65	0.45

完成 TPS 之後，我們也可以與 PAScan_1 一樣，先對系統做預先掃描，因為我們的任務是「往後看一步」所以預先掃描 Min. Support 所參照的 TPS 表除了 PV=1 的 4.65 之外，在 Support 為 3.15 到 4.65 之間的 Pattern，我們雖然不認為它是 Large，但是有可能在下一步會 Large，所以還是保留它生成下一階 Pattern 的能力。表 3-15 是預先掃描的結果。

表 3-15 可重覆的 Candidate Sequence 0 及其 support 值：CS_0

Candidate Sequence	Pattern's PageLen	門檻值 正常/往後	PA_ScanN support	狀態
A_	1	4.65 / 3.15	13	Large
B_	1	4.65 / 3.15	9	Large
C_	1	4.65 / 3.15	9	Large
?a	1	4.65 / 3.15	4	Next1 Large
?b	1	4.65 / 3.15	17	Large
?c	1	4.65 / 3.15	19	Large

經過預掃後，雖然[?a]不是 Large，但因為是 Next1 Large 所以還保留他參與下一階生成新 Pattern 的權力。按此規則，我們產生出來下一階如表 3-16 的 CS_1。

表 3-16 可重覆的 Candidate Sequence 1 及其 support 值：CS_1

Candidate Sequence	Pattern's PageLen	門檻值 正常/往後	PA_ScanN support	狀態
Aa	1	4.65 / 3.15	0	
Ab	1	4.65 / 3.15	10	Large
Ac	1	4.65 / 3.15	10	Large
Ba	1	4.65 / 3.15	1	
Bb	1	4.65 / 3.15	1	
Bc	1	4.65 / 3.15	8	Large
Ca	1	4.65 / 3.15	3	
Cb	1	4.65 / 3.15	6	Large
Cc	1	4.65 / 3.15	1	

LS_1

Large Sequence	**PA_ScanN support**	**狀態**
Ab	10	Large
Ac	10	Large
Bc	8	Large
Cb	6	Large

按造這樣的作法，重覆運算，我們可以得到 表 3-17 的各個 CS 與 LS 表。

表 3-17 可重覆演算法其他的 CS、LS support 值：

CS_2

Candidate Sequence	Pattern's PageLen	門檻值 正常/往後	PA_ScanN support	狀態
CbCb	1	4.65 / 3.15	0	
CbBc	2	3.15 / 1.65	0	
CbAb	2	3.15 / 1.65	1	
CbAc	2	3.15 / 1.65	2	Next1 Large
BcCb	2	3.15 / 1.65	3	Next1 Large
BcBc	1	4.65 / 3.15	0	
BcAb	2	3.15 / 1.65	2	
BcAc	2	3.15 / 1.65	1	
AbCb	2	3.15 / 1.65	1	
AbBc	2	3.15 / 1.65	5	Large
AbAb	1	4.65 / 3.15	0	
AbAc	1	4.65 / 3.15	6	Large
AcCb	2	3.15 / 1.65	1	
AcBc	2	3.15 / 1.65	5	Large
AcAb	1	4.65 / 3.15	1	
AcAc	1	4.65 / 3.15	1	

LS_2

Large Sequence	PA_ScanN support	狀態
AbBc	5	Large
AbAc	6	Large
AcBc	5	Large
CbAc	2	Next1 Large
BcCb	3	Next1 Large

CS_3

Candidate Sequence	Pattern's PageLen	門檻值 正常/往後	PA_ScanN support	狀態
CbCbAc	2	3.15 / 1.65	0	
CbAcAc	2	3.15 / 1.65	0	
CbAcCb	3	1.65 / 0.45	0	
AcBcAc	3	1.65 / 0.45	1	Next1 Large
AbCbAc	3	1.65 / 0.45	0	
CbAbAc	2	3.15 / 1.65	1	
BcBcAb	2	3.15 / 1.65	0	
BcAbAb	2	3.15 / 1.65	0	
BcAbBc	3	1.65 / 0.45	1	Next1 Large
AbBcAb	3	1.65 / 0.45	1	Next1 Large
BcAbAc	2	3.15 / 1.65	0	
BcBcCb	2	3.15 / 1.65	0	
BcCbCb	2	3.15 / 1.65	0	
BcCbBc	3	1.65 / 0.45	0	
AbAbBc	2	3.15 / 1.65	0	
AbBcBc	2	3.15 / 1.65	0	
AcAbBc	2	3.15 / 1.65	0	
AbAcBc	2	3.15 / 1.65	4	Large
AbBcAc	3	1.65 / 0.45	1	Next1 Large
AcAcBc	2	3.15 / 1.65	0	
AcBcBc	2	3.15 / 1.65	0	
AbAbAc	1	4.65 / 3.15	0	
AbAcAc	1	4.65 / 3.15	0	

LS_3

Large Sequence	PA_ScanN support	狀態
AbAcBc	4	Large
AcBcAc	1	Next1 Large
BcAbBc	1	Next1 Large
AbBcAb	1	Next1 Large
AbBcAc	1	Next1 Large

CS_4

Candidate Sequence	Pattern's PageLen	門檻值 正常/往後	PA_ScanN support	狀態
AcAcBcAc	3	1.65 / 0.45	0	
AcBcBcAc	3	1.65 / 0.45	0	
AcBcAcAc	3	1.65 / 0.45	0	
BcBcAbBc	3	1.65 / 0.45	0	
BcAbAbBc	3	1.65 / 0.45	0	
BcAbBcBc	3	1.65 / 0.45	0	
BcAbBcAb	4	0.45	0	
AbBcAbBc	4	0.45	0	
AbBcAcBc	4	0.45	0	
AbAbBcAb	3	1.65 / 0.45	0	
AbBcBcAb	3	1.65 / 0.45	0	
AbBcAbAb	3	1.65 / 0.45	0	
AbAbBcAc	3	1.65 / 0.45	0	
AbBcBcAc	3	1.65 / 0.45	0	
AbBcAcAc	3	1.65 / 0.45	0	
AbAbAcBc	2	3.15 / 1.65	0	
AbAcAcBc	2	3.15 / 1.65	0	
AbAcBcBc	2	3.15 / 1.65	0	

3.3.3 找出關聯規則(Find The Association Rules)

當我們擁有各階層的 LS 之後，我們就可以進行尋找關聯規則的工作了。我

們可以列出所有可能的 X→Y，而 X 是 Y 的 SubSequence，例如，我們觀察在不重覆計算演算法出產生的 LS_1 與 LS_2(表 3-10 與表 3-12)，我們可以發現如表 3-18 中所列的關聯規則：

表 3-18 LS_1 與 LS_2 可產生出的規則

LHS	RHS	發生機率	規則類型
Ab	**Ab**Ac	5/8=62.5%	→後推
Ab	**Ab**Bc	5/8=62.5%	→後推
Ac	Ab**Ac**	5/8=62.5%	←前導
Ac	**Ac**Bc	5/8=62.5%	→後推
Bc	Ab**Bc**	5/7=71.4%	←前導
Bc	Ac**Bc**	5/7=71.4%	←前導

其中發生率的演算法與數學中條件機率的演算法一樣，也就是發生 X 之後，又緊接著發生 Y 的機率是 $P(Y|X)=\frac{P(X\cap Y)}{P(X)}=\frac{\#of(X\cap Y)}{\#ofX}$，也就是等於 RHS 的 support 值，除於 LHS 的 support。

我們算出發生機率之後，我們必須判定這個規則的發生機率是否常見，所以我們會定義一個最小的信心值，一但找出的規則超過這個信心值，我們就認為這個規則是常發生的，必須被研究者所注意。我們假設發生機率超過 70%才是我們所認為需要注意的，所以在做完這部份的工作後，我們可以知道 Bb→AbBc 與 Bc→AcBc 這兩條規則是常出現的規則。

3.4 一個可能真實例子的操作

在本節裏，我們將針對一個較實際的例子來實作演算法。為了能夠較方便的

來表示較真實範例的網頁動作路徑。我們還是採用一些代號來表示可能的網頁碼與動碼。表 3-19、3-20 就是我們在這個例子中會使用到的基本代號。

表 3-19 一個可能真實的 PageCode 與其代表

PageCode	**代表頁面**
Home	首頁
PInfo	商品資訊頁面
OnSale	特價品資訊頁面
Laws	法規資訊頁面
Submit	訂單確定頁面

表 3-20 一個可能真實的 ActionCode 與其代表

PageCode	**代表動作**
H01	查詢最新消息
H02	登錄會員
P01	檢視詳細商品資訊
P02	查閱商品討論
O01	點選廣告資訊
O02	點選推薦品資訊
L01	查詢購物(交易)需知
L02	查詢會員需知
S01	將商品加入購物車
S02	將商品從購物車移除
X00	送出購物單，完成訂貨

在實際的情境中，我們假設使用者沒有做任何動作就會自動轉址的網頁是無效的。也就是說，對我們有意義的網頁，使用者至少要做一個動作。另外，在實務上，網站上某張網頁可執行的動作已經被網站經營者設定好了。系統能取得使

用者的網頁與動作路徑是會在限定的網站地圖中的。了解一個網站其中的設計，可以有助於事前的資料清洗與事後的規則評估。表 3-21 是一段可能真實的網頁動作路徑資料。S001 使用者的 PA_Sequence「Home(H02、H01)→OnSale(O02、S01)→Submit(X01)」代表，使用者在系統首頁中先登錄會員，然後查詢了最新消息的資訊。之後使用者跳到了特價品資訊頁，查詢了一個推薦的特價品資訊，之後他將商品加入購物車當中，最後他跳到了訂單確定頁，送出購物單後完成訂貨的動作。

表 3-21 一段可能真實的 MainSequence 資料

Session_Id	**PA_Sequence**	$\|s\|_p$
S001	Home(H02、H01)→OnSale(O02、O01、S01)→Submit(X01)	3
S002	Home(H01、H02)→PInfo(P01、P02、S01、P02)→Laws(L01)→Submit(S01、X01)	4
S003	Home(H01)→OnSale(O02、P01)	2
S004	Home(H02)→PInfo(P01、O01、P02、S01)→OnSale(O02、S01、O02、S01)→Submit(S02、X01)	4
S005	OnSale(O02、O01、S01)→Laws(L02)→Submit(S02)	3
S006	Home(H01)→OnSale(O01、O02、P01)→Laws(L01)	3
S007	Home(H02、O01)→OnSale(O02、O01、S01)→PInfo(P01、P02、S01)→Submit(S02)	4
S008	PInfo(P01、O01、P02)→Laws(L02、L01)→Home(O01、H01)	3
S009	Home(H02、O01、H01)→OnSale(O02、S01)→PInfo(P01、P02)	3
S010	Home(H02)→PInfo(P01、O02、S01、O01)→Submit(X01)	3

有了原始資料之後，我們就可針對這個資料進行網頁動作路徑的資料挖掘。在這裏我們採用的是 PAScan1 的演算法，門檻值設為 30%，超過 80%發生機率的規則，才算是常發生的。表 3-22 是本範例預先掃描的結果。預掃的結果顯示

L02(查詢會員需知)這個動作並未達到門檻需求。可不列入第一階 PA_Join 的處理單元中。

表 3-22 本範例的預先掃描 CS_0

Candidate Sequence	PA_Scan1 support	Large?
Home_	9	L
PInfo_	6	L
OnSale_	7	L
Laws_	6	L
Submit_	4	L
?H01	6	L
?H02	6	L
?P01	8	L
?P02	5	L
?O01	8	L
?O02	8	L
?L01	3	L
?L02	2	
?S01	7	L
?S02	3	L
?X01	4	L

完成了預掃之後，就開始進入 PA_Scan1 各階的重覆演算中。表 3-23、3-24、3-25 為本範例產出的各階 Large Sequence。

表 3-23 本範例的 Large Sequence 1 及其 support 值：LS_1

Large Sequence	What	PA_Scan1 support
Home(H01)	在首頁查詢最新消息	6
Home(H02)	在首頁登錄會員	6

Home(O01)	在首頁中點選廣告資訊	3
PInfo(P01)	在商品資訊頁檢視商品資訊	6
PInfo(P02)	在商品資訊頁查閱商品討論	5
PInfo(O01)	在商品資訊頁中點選廣告資訊	3
PInfo(S01)	在商品資訊頁將商品加入購物車	4
OnSale(S01)	在特價品資訊頁將商品加入購物車	5
OnSale(O01)	在特價品資訊頁點選廣告資訊	4
OnSale(O02)	在特價品資訊頁點選推薦品資訊	7
Laws(L01)	在法規資訊頁查詢購物需知	3
Submit(S02)	在訂單確認頁移除購物車內商品	3
Submit(X01)	在訂單確認頁中完成訂貨	4

表 3-24 本範例的 Large Sequence 2 及其 support 值：LS_2

Large Sequence	**What**	**PA_Scan1 support**
Home(H01)OnSale(O02)	在首頁查過最新消息後，接著去特價品頁點選推薦品資訊	4
Home(H02)OnSale(S01)	在首頁登錄會員後，接著去特價品頁將商品加入購物車	3
Home(H02)OnSale(O02)	在首頁登錄會員後，接著去特價品頁點選推薦品資訊	3
Home(H02)PInfo(P01)	在首頁登錄會員後，接著去商品資訊頁檢視商品資訊	3
Home(H02)Pinfo(S01)	在首頁登錄會員後，接著去商品資訊頁將商品加入購物車	3
PInfo(P01)PInfo(P02)	在商品資訊頁中曾先檢視商品資訊，再查閱商品討論資訊	5
PInfo(P01)PInfo(S01)	在商品資訊頁中曾先檢視商品資訊，再將商品加入購物車	4
PInfo(P02)PInfo(S01)	在商品資訊頁中曾先查閱商品討論，再將商品加入購物車	3
PInfo(P01)PInfo(O01)	在商品資訊頁中曾先查閱檢視詳細商品資訊，再將點選廣告資訊	3
OnSale(O02)OnSale(S01)	在特價品頁中曾先點選推薦品資訊，然後會將商品加入購物車	5

OnSale(O01)OnSale(S01)	在特價品頁中曾先點選廣告資訊，然後會將商品加入購物車	3
OnSale(O02)OnSale(O01)	在特價品頁中曾先點選推薦品資訊，然後再點選廣告資訊	3
OnSale(O01)OnSale(S01)	在特價品頁中曾先點選廣告資訊，然後再將商品加入購物車	3
OnSale(O02)OnSale(S01)	在特價品頁中曾先點選推薦品資訊，然後再將商品加入購物車中	5

表 3-25 本範例的 Large Sequence 3 及其 support 值：LS_3

Large Sequence	**What**	**PA_Scan1 support**
Home(H02)OnSale(O02)OnSale(S01)	在首頁登錄會員後，接著來到特價品頁中曾先點選推薦品資訊，然後會將商品加入購物車	3
Home(H02)PInfo(P01)PInfo(S01)	在首頁登錄會員後，接著來到商品資訊頁中，曾先檢視商品資訊，然後會將商品加入購物車	3
PInfo(P01)PInfo(P02)PInfo(S01)	在商品資訊頁中，檢視詳細商品資訊後，再查閱商品討論，將商品加入購物車	3
OnSale(O02)OnSale(O01)OnSale(S01)	在特價品資訊頁中，曾先點選推薦品資訊，之後會點選廣告資訊後，最後將商品加入購物車	3

完成了各階的 LS 之後，接著就是應用 3.3.3 節所述的方法，找尋出滿足條件的規則。表 3-26 是此例發生機率超過 80%的所有規則。將這些規則整理後，並用人類了解的語言書寫後，可得表 3-27。

表 3-26 此例發生機率超過門檻值(80%)的規則

#	LHS	RHS	發生機率	規則類型
1	Home(H02)OnSale(O02)	**Home(H02)OnSale(O02)**OnSale(S01)	100%	→後推型
2	Home(H02)OnSale(S01)	**Home(H02)**OnSale(O02)**OnSale(S01)**	100%	→←中夾型
3	OnSale(S01)	OnSale(O02)**OnSale(S01)**	100%	←前導型
4	PInfo(P02)	PInfo(P01)**PInfo(P02)**	100%	←前導型
5	PInfo(S01)	PInfo(P01)**PInfo(S01)**	100%	←前導型
6	PInfo(O01)	PInfo(P01)**PInfo(O01)**	100%	←前導型
7	Home(H02)PInfo(P01)	**Home(H02)PInfo(P01)**PInfo(S01)	100%	→後推型
8	Home(H02)PInfo(S01)	**Home(H02)**PInfo(P01)**PInfo(S01)**	100%	→←中夾型
9	PInfo(P02)PInfo(S01)	PInfo(P01)**PInfo(P02)PInfo(S01)**	100%	←前導型
10	OnSale(O01)OnSale(S01)	OnSale(O02)**OnSale(O01)OnSale(S01)**	100%	←前導型
11	OnSale(O02)OnSale(O01)	**OnSale(O02)OnSale(O01)**OnSale(S01)	100%	→後推型
12	PInfo(P01)	**PInfo(P01)**PInfo(P02)	83.3%	→後推型

表 3-27 此例發生機率超過門檻值(80%)規則的解釋

#	What
1	已知一人[先在首頁登錄會員後，接著在特價品資訊頁中檢視過推薦品資訊]，有 100%的機率 他會在特價品頁[將商品加入購物車]
2	已知一人[先在首頁登錄會員後，接著在特價品資訊頁中將商品加入購物車]，有 100%的機率 他曾在特價品資訊頁中[檢視過推薦品資訊]
3	已知一人[在特價品資訊頁中將商品加入購物車]，有 100%的機率 他之前曾在同一頁[檢視過推薦品資訊]
4	已知一人[在商品資訊頁曾查閱相關討論]，有 100%的機率 他之前曾在同一頁[檢視過產品資訊]
5	已知一人[在商品資訊頁面曾將商品加入購物車]，有 100%的機率 他曾在同一頁中[檢視過商品資訊]
6	已知一人[在商品資訊頁面曾點選過廣告資訊]，有 100%的機率 他曾在同一頁[點選過商品資訊]
7	已知一人[在首頁曾登錄過會員，而且他接著在商品資訊頁執行過商品資訊的查詢]，有 100%的機率 他會[在商品資訊頁將商品加入購物車裏]
8	已知一人[在首頁中曾登錄會員，而且他接著在商品資訊頁將商品加入購物車裏]，有 100%的機率 他在加入購物車前[曾在商品資訊頁執行過商品資訊的查詢]
9	已知一人[在商品資訊頁面曾查閱商品討論，之後並已將商品加入購物車]，有 100%的機率 他在同一頁中曾在查閱討論前先做過[查詢商品資訊]

10	已知一人[在特價品資訊頁面中曾點選廣告資訊，之後並已將商品加入購物車]，有 100%的機率 他在同一頁中曾在點選廣告前已先做過[點選過推薦品資訊]
11	已知一人[已在特價品點選推薦品資訊，之後又曾點選過廣告資訊]，有 100%的機率 他會在同頁中[將商品加入購物車]
12	已知一人[在商品資訊頁檢視過產品資訊]，有 83.3%的機率 他之後會在同一頁[查閱相關討論]

表 3-27 整理出的八個規則，就是隱藏在這個範例資料中在我們演算法運作之下所能找到的規則。根據這些規則，網站的經營者可用改善網頁的配置或是廣告的位置。例如，經營者得到了規則 1 之後，可以針對在首頁登錄後，而又跳至特價品資訊頁的使用者，增加生動的廣告誘使他能夠去點閱推薦特價品的資訊。這樣也許就能夠有效增加使用者將商品放至購物車的機率。這類規則是無法用一般只考慮網頁路徑演算法找出的。另外，我們觀察規則 5，這個規則可讓經營者知道，會將商品加入購物車的使用者，他會先做那些動作。但我們仔細觀查表 3-21 可以發現，沒有一筆資料是先查閱商品資訊後，下個動作就加入購物車的。如果我們對於動作路徑的掃描採用「連續出現」的模式，這樣的掃描法就無法找出這類的規則。

3.5 本演算法的效率與效益

由於我們同時進行二種不同編碼的序列規則萃取，所以在找尋規則的每個步驟都較一般的網頁瀏覽規則演算法來的複雜。所以在執行效率上會較普通的網頁瀏覽規則來得低。(模擬測試相關的數據資訊，在 4.2 節會深入探討。)

但是就效益來談的話，本演算法可以找出網頁中與網頁間使用者行為動作的關聯規則，這是一般演算法所無法達到的，以下我們詳述本演算法可以找出的規則類型，及其應用：

一、常出現的使用者網頁-動作路徑：我們的演算法可以找出使用者常用的網頁路徑及其中使用者會進行的動作。網站的經營者可以藉由這樣的規則來調整選單的位置，或是在適當的位置設置適當的廣告，以提昇效益。在系統面，網站管理者如果知道這樣的資訊，就可以瞭解這個使用者的行為路徑是很重要的，所以與這個行為路徑有關的程式或是系統快取，應該特別加以優化。

二、網頁間的網頁-動作關聯規則：我們的演算法可以找出使用者執行到某一個階段時，其前推或是後推的網頁-動作會是什麼。也許我們會找出「當使用者在 商品資訊頁 執行 觀看付款規則 的動作時，有 80%的使用者下一步會去 線上訂購頁 進行 下單訂購 的動作」這樣的規則。這種後推型的規則，網頁經營者可以預測使用者在其網站上的行為，在滿足某些瀏覽動作規則時，經營者可以安排出現某些會引吸使用者的消息或廣告，可以增加網站的效益。前推型的規則是指「已知使用者在 A 網頁做了 a 動作，那他可能有 70%的機率是由 B 網頁而來，在其中做過 b 這個動作。」在效益上，也是可以幫助網站經營者瞭解使用者的行為。

三、網頁中的網頁-動作關聯規則：我們的演算法也可以找出使用者在同一頁中所做動作的先後關聯。比方說，我們可以找出「當使用者在 商品資訊頁 先

執行了 手動輸入機型 這個動作，他在同一頁還會在之後執行 將資料寄送給我 動作的機率為 80%」，網頁經營者瞭解了這樣的狀況，對於這兩個功能選單的網頁配置，就應該更注意；可能是要將這兩個功能選單放在附近，或是在使用者手動輸入了機型之後，就出現在資訊結果的旁邊。