可以預測網頁連結的智慧型搜尋引擎之設計

陳林志
台灣科技大學資管所

陸承志
元智大學資訊管理系

摘要

本文提出一個依據使用者行為進行預測網頁的搜尋引擎，它具有兩種搜尋機制：搜尋引擎投票向量(SVV)以及連結預測(HLP)。SVV 的方法是對六個知名的搜尋引擎進行英文關鍵字搜尋，當一個網頁能夠從這些搜尋引擎取得較多且較前面的排名，則該網頁的權重自然較高。至於 HLP 的方法是根據 SVV 的結果進行深度連結預測，以推薦使用者最可能點擊的連結。經過初步的評估，使用者對於我們搜尋引擎的滿意度高於其他一般搜尋引擎。本搜尋引擎目前只支援英文關鍵字的搜尋，未來將增加中文關鍵字的搜尋功能。

關鍵字： 向量投票、連結預測、搜尋引擎、使用者行為函數

An Intelligent Search Engine for Hyperlink Prediction

Cayley Chen
Department of Information Management,
National Taiwan University of Science and Technology

Cheng-Jye Luh
Department of Information Management, Yuan-Ze University

Abstract

This paper presents an intelligent search engine which is capable of predicting web pages on the base of user preferences. Two search methods are implemented in this search engine. First, a search engine vector voting (SVV) method is developed to rank the web pages returned from six well known search engines. A web page will be ranked high if it is listed near the top in several search engine results. Based on the URLs in SVV results, a hyperlink prediction (HLP) method is then developed for predicting all the hyperlinks on which users are most likely to click. Preliminary user study results indicate that users are more satisfied with our search engine than others. The work presented in this paper is mainly on English keyword search, and Chinese keyword search will be developed in future research.

Keywords:Search Engine Vector Voting, Hyperlink Prediction, MetaSearch, User Behavior Function.

壹、導論

搜尋引擎設計的主要是從使用者輸入的查詢找到使用者真正想要的資料，但這是一項困難的工作，因為使用者輸入的資訊太少了，所以很難找到合適的資料。根據文獻資料顯示，使用者每次輸入關鍵字的平均個數是 2.3 (Spink et al. 2001)，如何根據這些關鍵字取得相關資料即是搜尋引擎的核心技術。傳統的 Information Retrieval 根據使用者輸入的查詢和文件內容進行相似分析，採用的模式包括向量空間模式、機率模式、和模糊邏輯模式等 (Harman 1992)，然後根據文件內容與查詢關鍵字的符合程度與數量進行文件排名，但是這樣處理會鼓勵網站管理者在文件中重複加入許多的關鍵字，即可將網站的排名提前，形成"關鍵字灌水"的情形。

為了避免上述的灌水情形，Li 發展出 HVV 的方法 (Li 1998)，其主要的精神是: 網站的排名是由其它站台評等，亦即當有比較多的站台投票給該網站時，它的權重將會比較大。但是這樣的方法會有下列問題: 1.我們到底要用多少外部站台來評比才具代表性；2.沒有考慮好的站台及差的站台所評比的差異，亦即所有評比站台的權重都一樣，這樣容易造成評比不公正。

針對第二個問題，Henzinger 探討一個不同的 Page Ranking 觀念，亦即在相同查詢下，優良站台所評比網站的權重應該較大 (Henzinger 2001)，這樣將能確保較佳網站，其所評比的站台能夠在搜尋引擎中取得較前面的排名。但這樣的方法在理論上還是同樣面臨到上述的第 1 個問題，亦即需要收集足夠多的文件訓練後才有足夠好的搜尋品質。根據文獻資料，網頁數量以每天 3 億個網頁成長(Chakrabarti et al. 1999)，我們很難從這些資料中判斷需要多少評比站台才足夠顯現評比的效果。另一方面，使用 recursive 的方法計算大量頁面的 Page Rank，所需花費的計算時間將是非常可觀。

從另一個觀點來看，如果我們可以利用一些搜尋引擎的結果進行評比，將會達到下列的好處: 1.在最短時間達到優良的評比效果；2.避免搜尋引擎介面的不同造成使用者學習上的困難。這樣的搜尋技術一般稱為 MetaSearch (Dreilinger & Howe 1996; Selberg & Etzioni 1997)，知名的例子包括: Ixquick、Metacrawler、MetaSearch、Dogpile 及 SavvySearch。因此我們所發展的第一個方法，稱之為搜尋引擎向量投票 (Search Engine Vector Voting, SVV)，其主要精神是利用市場上幾個知名的搜尋引擎進行向量投票，所以網站權重在下列兩種情形將會較大: 1.當有較多搜尋引擎投票給它；2.它在單一搜尋引擎取得較佳的排名，如此可以避免 HVV 可能產生的二個問題。

目前網路上的搜尋引擎大概有將近百個，理論上我們是可以將所有的搜尋引擎納入，但在資源（例如頻寬）有限的限制下，因此我們只選擇 6 個市場上有名的搜尋引擎。根據 Nielson NetRating: Search Engine Rating (Nielson 2003)結果顯示我們選擇的 Google, Yahoo, Overture, Altavista, Lycos, Looksmart 都排在前 12 名，其他未選的如 MSN, AOL, Earthlink, Netscape 等都不是以 search engine 技術見長）。這些市場上知名的搜尋引擎共同的特色是 1. 他們都有適當的 ranking algorithm 將 search results 做適當排序，2. 他們都

有 spamming check and penalty 的 mechanism 處理灌水問題，所以這些搜尋引擎幾乎不會出現灌水的情形，但少數排在前面的 sponsor links，我們會在 SVV 之前先過濾掉。

再者，搜尋引擎若能根據使用者輸入的查詢與搜尋的結果建議最適合的網頁給使用者，則使用者將會得到很大的助益。Nick 和 Themis (2001)運用 Intelligent Agent 的方法輔助搜尋引擎提供一些網頁給使用者，但是它需要使用者"持續"的提出一些範例並重覆的進行分析，以便調整合適的 user profile，以及相對應的搜尋結果。對使用者而言，要先輸入範例是一個難以克服的障礙，因為使用者一般都是對此查詢沒有太多觀念，同時也可能因為使用者輸入不合適的範例，造成系統建議不合適的網頁。

與 link prediction 相關研究一般採用 stochastic approach 針對網站的 access log 以 Markov chains 建立使用者的瀏覽型樣(access patterns)模式，以便預測使用者未來會點選的連結(Lempel & Moran 2000; Sarukkai 2000)。由於 Markov chain 模式的精華在於它的訓練資料，亦即網站的 access log，通常在單一網站中使用者瀏覽的熱門路徑較明顯，因此瀏覽型樣比較容易建立；但在多網站分析時，由於各個網站的性質與網頁設計理念的不同，一般不容易出現跨網站共通的熱門路徑，瀏覽型樣不容易建立，其結果對未來預測幫助不大。再者，stochastic approach 在預測 link 之後，就不再提供更進一步的資訊，接下來使用者就必須進入該 link 瀏覽，以判斷該頁面中的那些 link 是他想要的。

假若系統能根據大多數人的可能行為去建議網頁而不用輸入範例，其所找到的結果將能夠避免上述可能發生的問題，因此我們發展出一個全新的網頁推薦技術，稱之為連結預測(HyperLink Prediction, HLP)，其主要精神是: 對搜尋引擎投票向量中權重較大的網站，我們不只需要擷取較多網站內部的網頁而且網站的擷取層數也應較多，因為使用者對有興趣的網站可能拜訪較多的內部連結，而且觀看的資料也會較深入。

本研究實作一個搜尋引擎，稱之為 Cayley 搜尋引擎，它具有搜尋引擎向量投票及連結預測兩個機制，使用者可以根據本身所需要的英文資料，決定使用那一種搜尋機制。在論文後續部份，我們將先介紹系統架構，然後詳細介紹這兩個方法，最後我們探討初步系統實作與使用者評估的結果。

貳、系統架構

在 Cayley 系統中，我們假設使用者的資料需求分成廣度及深度，廣度代表使用者需要較多的網站，深度代表使用者需要較多關鍵性站台的網頁內容。根據上述觀念，我們提出兩個搜尋機制:搜尋引擎向量投票(SVV，Search engine Vector Voting)及連結預測(HLP，HyperLink Prediction)。圖 1 是我們的系統架構，其中包括外部模組及內部模組。外部模組包括: 搜尋引擎集合(Search engine set)、資料庫(DB)、庫存網頁(Cached pages)以及網路連結(Internet Connection)。內部模組包括:使用者介面(SVV 及 HLP)、網頁分析(Pages analysis)、彙總分析(Summary SVV 及 Summary HLP)、搜尋引擎向量投票分析(SVV analysis)及連結預測分析(HLP analysis)。

在外部模組中，搜尋引擎集合是我們 SVV 方法中進行 MetaSearch 的對象，使用者

輸入英文關鍵字查詢後，系統將送至 6 個知名的搜尋引擎: Google、Yahoo、AltaVista、LookSmart、Overture 以及 Lycos 進行投票分析。資料庫主要是儲存查詢的相關資訊，避免相同查詢因分析及擷取網頁所造成的時間延遲，一旦查詢存在資料庫中，系統將直接顯示給使用者。庫存網頁主要是每次連結預測分析後，系統判斷需要擷取的網頁，這些擷取回來的網頁除了可以在 HLP 結果中讓使用者參考之外，也可以做為更進一步連結預測的依據。網路連結主要功能是透過網路將連結預測分析後的網頁擷取回來，並將之儲存成庫存網頁，同時將網頁內的相關資訊儲存於資料庫。

在內部模組中，使用者介面提供使用者搜尋機制的選項，系統根據不同的搜尋選項回應適當的資料。從系統架構中，我們知道 HLP 的輸入是搜尋引擎向量投票分析之後的結果，因此 HLP 進行處理時，SVV 必須先行完成，因此選擇 HLP 的使用者，也可切換檢視 SVV 分析結果，亦即使用者的選擇變多了。但若查詢尚未被系統處理過，由於每當擷取一個網頁後，即需再判斷其內部連結有那些需擷取，因此系統的回應時間將會較長。反之，若使用者選擇 SVV，系統的回應時間將較快。在論文後續部份，我們會討論如何減少系統回應時間。頁面分析可以將擷取回來的網頁做適當的處理，並取得搜尋引擎向量投票分析及連結預測分析所需的資料(例如:SVV 所需的資料是每個搜尋引擎回傳結果的網站排名，HLP 所需的資料是網頁內的連結)。搜尋引擎向量投票分析將根據搜尋引擎彼此之間的投票行為建立適當的網站排名。在連結預測分析中，系統將會持續擷取網頁內的連結，每當擷取一定數量的連結網頁後，系統會判斷是否需要繼續擷取下一層網頁，若判斷需要則繼續擷取其它網頁並做連結預測分析，否則透過彙總分析將結果顯示給使用者。

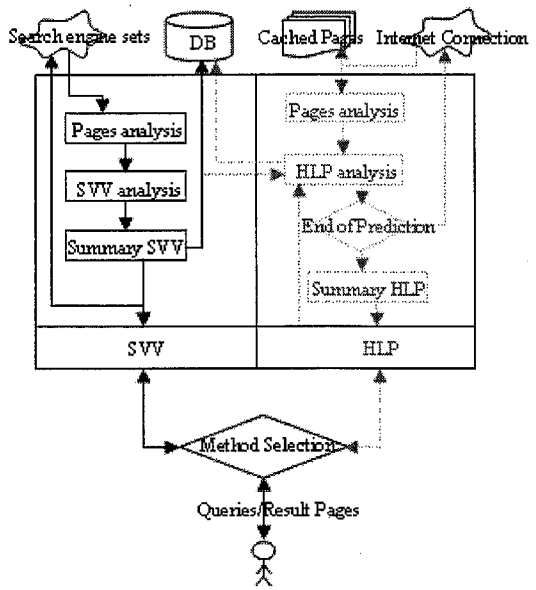

圖 1：Cayley 搜尋引擎系統架構

叁、搜尋引擎向量投票分析

在 SVV 方法中，我們採用 6 個市場上常用的搜尋引擎的搜尋結果來進行投票分析。在計算網站權重之前，我們瞭解一般使用者對於排名的前面網站喜好度會比後面網站來得大，而且喜好度會逐漸遞減，這種特性在心理學稱為「先前效用」 (Primacy effect)，亦即使用者會偏好前面的資訊且偏好程度會逐漸降低 (Morris et al. 2002)，其他研究學者亦有類似的觀察 (Lempel & Moran 2000; Paepcke et al. 2000)。

因此我們定義出下列的使用者行為函數(User Behavior Function, UBF):

$$UBF = \alpha_i x_{i,s}^{\beta} \quad (1)$$

其中α_i表示使用者對第 i 個引擎所搜尋出第一個網站的喜好度(一般來說第一個網站是搜尋引擎給使用者的第一個印象，因此對使用者而言是很重要)，$x_{i,s}$ 表示網站 s 在引擎 i 中的排名，β(我們定義β<0)代表使用者對網頁偏好度，當|β|愈大使用者偏好下降程度愈強烈(例如圖 2 中，β= -0.9 時的下方曲線)，代表使用者偏好效應降低將較快速。

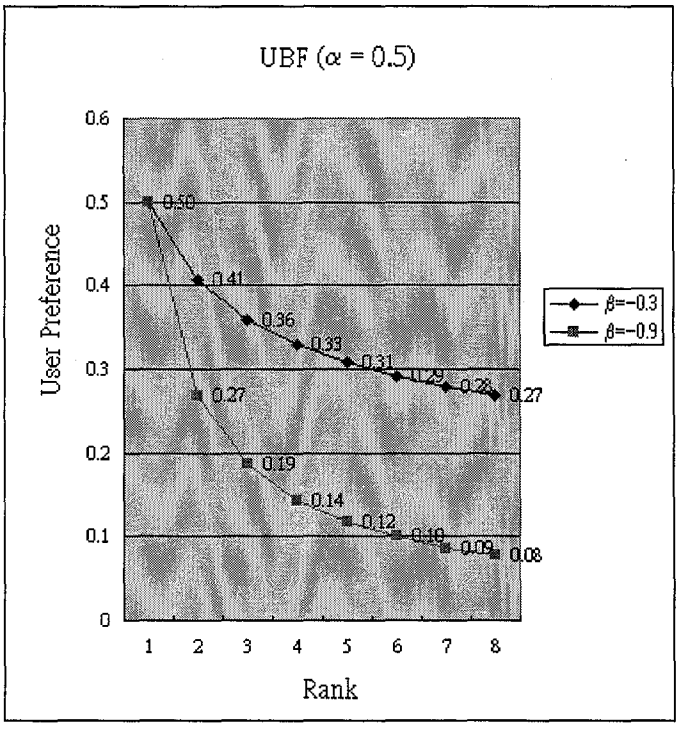

圖 2. 當α=0.5，β= -0.3 及β=-0.9 時的使用者行為函數

SVV 計算出的網站權重是由每個搜尋引擎所表達的使用者行為函數透過向量內積計算而得。我們定義每個網站的權重如下:

$$w_s = \sum_{i=1}^{6} \alpha_i x_{i,s}^{\beta} \qquad (2)$$

其中 w_s 代表網站 s 的權重。根據這個定義，某個特定網站在下列兩種情形的權重會較大: 1. 有較多搜尋引擎投票給它，2. 此網站在單一搜尋引擎中排名較前面。底下我們以符合關鍵字 "php"的範例 URLs 說明公式 2 的運作，這些 URLs
在搜尋引擎的排名情形如表 1 所示 (0 表示該 URL 不在該搜尋引擎的搜尋結果之中):

表 1. 範例 URLs 在各個搜尋引擎回傳結果的排名

	Google	Yahoo	AltaVista	LookSmart	Overture	Lycos	SVV 權重
http://www.php.net	1	1	1	3	1	1	4.78353
http://www.php.com	5	4	11	1	7	8	2.29317
http://phpnuke.org	3	2	3	0	3	4	2.19225

上述每個 URL 的 SVV 權重是透過公式 2 計算得到的，其中公式 2 有二個參數: α及β，我們剛開始的做法是在α及β的上限及下限範圍之間取一個亂數當成該引擎的實際參數(根據 UBF 我們定義β<0)，其數值如下表所示:

表 2.系統開始運作時各個搜尋引擎的α及β參數值

	Google	Yahoo	AltaVista	LookSmart	Overture	Lycos
α上/下限	0.8~0.95	0.8~0.95	0.8~0.95	0.8~0.95	0.8~0.95	0.8~0.95
β上/下限	-0.3~-1	-0.3~-1	-0.3~-1	-0.3~-1	-0.3~-1	-0.3~-1
實際α	0.895259	0.844789	0.811069	0.93683	0.905779	0.889514
實際β	-0.77304	-0.77304	-0.77304	-0.77304	-0.77304	-0.77304

以表 1 中的 http://www.php.net 為例，其 SVV 權重是: $0.895259*1^{-0.77304} + 0.844789*1^{-0.77304} + 0.811069*1^{-0.77304} + 0.93683*3^{-0.77304} + 0.905779*1^{-0.77304} + 0.889514*1^{-0.77304} = 4.78353$。

其次，為了給予使用者更多決策時的參考，我們亦提供每個網站在 6 個搜尋引擎間的投票傾向 P_s，其公式如下:

$$P_s = w_s / \sum_{i=1}^{6} \alpha_i \qquad (3)$$

例如：http://www.php.net 的投票傾向是 4.78353/(0.895259 + 0.844789 + 0.811069 + 0.93683 + 0.905779 + 0.889514) = 0.96，在 SVV 結果網頁中(圖 3)，我們以長條顏色框表示此一投票傾向，因此其左側的長條顏色框填滿程度為 96%。若有一網站在六個搜尋引擎的搜尋結果中都排第一的話，其長條顏色框將是百分之百填滿顏色。

此外，為了兼顧例外的情況，例如當搜尋引擎對某個網站的投票傾向不一致時，該網站的長條顏色框會幾乎是空白，將造成使用者決策時的困難。因此我們亦將提供每個網站與查詢關鍵字的關聯程度，以提供使用者更多決策的幫助。我們利用下列公式計算每個網站與查詢關鍵字的關聯程度:

$$R_s = \begin{cases} High, & w_p > \overline{w} + 3\sigma_w \\ Middle, & \overline{w} < w_p \le \overline{w} + 3\sigma_w \\ Low, & otherwise. \end{cases} \quad (4)$$

其中 R_s 代表網站 s 與此查詢相關程度，$\overline{w}$ 代表在此查詢下所有回傳網站的平均權重，σ_w 代表在此查詢下所有網頁權重的標準差。此處我們根據 Chebyshev's 定理 (Walpole & Myers 1993)，得知網站權重>平均權重+3 倍權重標準差的機率小於九分之一，因此該網站與查詢高度相關。我們分別以綠、黃、紅三種不同色球顯示高、中、低度相關，例如: 經過計算後符合"php"的 URLs 的平均權重是 0.26766452825035，標準差是 0.47376845087598，其中 http://www.php.net 的權重是大於平均值＋3 倍標準差(4.78353 > 0.26766452825035 + 3*0.47376845087598)，因此有綠球顯示於前。

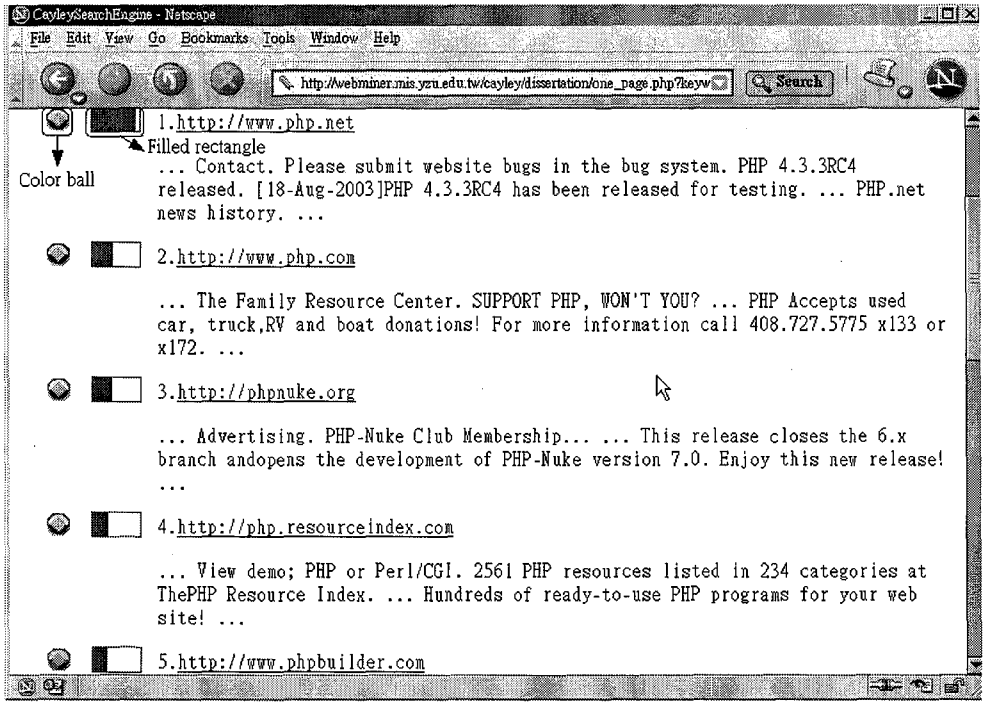

圖 3. SVV 搜尋結果範例

肆、連結預測分析

針對 SVV 的搜尋結果，使用者將會比較偏好權重較大的網站，除了閱讀較多該網站內的網頁而且拜訪的網站層數也會較多，因此我們發展出連結預測(HLP)的機制。HLP 基本上是一個獨立的連結預測機制，它的輸入來源除了 SVV 的搜尋結果外，也可以是任一搜尋引擎。

當連結預測進行分析時，必須重複做三個判斷: (1)根據權重判斷某個網頁是否符合條件可以擷取其後代；(2)當某個網頁符合第 1 個條件時，我們可以在它網頁內的連結擷取那些後代連結(亦即有那些後代連結可以繼續進行連結預測)；(3)最多需擷取幾層連結。在討論這三個判斷之前，我們需要瞭解每個網頁的權重將會遺傳給該網頁內的後代連結(亦即後代連結將繼承父代網頁的權重)。上述的第一個判斷是根據下列公式:

$$P_c = \begin{cases} \log_{10}(T_{p,l}) & if\,(T_{p,l} > 10) \\ 1, & otherwise. \end{cases} \qquad (5-1)$$

$$Q_p = \begin{cases} 1, & if \left\lceil \frac{\alpha_p}{P_c} \right\rceil \geq 1 \\ 0, & otherwise. \end{cases} \qquad (5-2)$$

其中 Q_p 是網頁 p 的擷取條件值(1 代表網頁 p 該擷取、0 代表網頁 p 不需擷取)，P_c 是網頁 p 的懲罰成本，α_p 代表父代的權重(當第一次預測每個網站所需擷取的連結時，我們令 $\alpha_p = w_s$，亦即第一層的網站將權重遺傳給其內部連結)，$T_{p,l}$ 代表從網站首頁(亦即 SVV 的網站)到網頁 p 所經過的累積連結總和。因為連結個數愈多將造成使用者做決策時的負擔，亦即形成懲罰成本，此處我們選擇以 10 為基底。

上述的第二個判斷主要是使用競賽方式來處理。我們將連結分成勝部及敗部(亦即敗部也有復活的機會) 兩部份，其中勝部的定義為: 那些連結權重>平均連結權重+3 倍連結權重標準差的連結；敗部為: 那些非勝部的其餘連結。接著面臨的兩個議題為: (1)勝部及敗部各需擷取的連結個數，(2)該擷取那些連結。在處理上述議題之前，我們瞭解網頁內連結的權重也會符合使用者行為函數(亦即使用者會比較偏好排名前面的連結)，因此我們使用下列公式計算網頁內的每個連結的權重:

$$w_{p,l} = \alpha_p x_{p,l}^{\beta} \qquad (6)$$

其中 $w_{p,l}$ 表示網頁 p 中連結 l 的權重($w_{p,l}$ 會遺傳至連結 l 所指向的網頁，同時成為該網頁的 α_p)，$x_{p,l}$ 代表連結 l 在網頁 p 的排名。

接下來我們透過下列公式來解決勝部的第 1 個議題，亦即判斷所需擷取連結的個數:

$$V_s = \{l \text{ if } w_{p,l} > \overline{w_p} + 3\sigma_{w_p}, \text{ for } \forall_{p,l}\} \quad (7\text{-}1)$$

$$V_p = |V_s| \quad (7\text{-}2)$$

其中 V_p 代表網頁 p 內所需擷取的勝部連結個數，$\forall_{p,l}$ 代表網頁 p 的所有連結，$\overline{w_p}$ 代表網頁 p 中所有連結權重的平均值，σ_{w_p} 為網頁 p 中所有連結權重的標準差。簡言之，公式 7-1 及 7-2 定義勝部所需擷取連結的個數為那些權重大於平均值+3 倍標準差的連結之總個數。

接著，我們處理勝部的第 2 個議題，定義在勝部中擷取的連結是 "V_p 個權重較大且未曾擷取過的連結"。為什麼是未曾擷取過的連結而非第 1 個問題中的所有符合條件的連結，亦即上述的 V_s? 主要是為了避免"連結灌水"的情形，例如: 在圖 4 中 P_1 網頁與 P_2 網頁內符合擷取條件都是 $P_{1,1}$，$P_{1,2}$ 及 $P_{1,3}$ 連結，而且 P_1 是 P_2 的祖先網頁，我們在處理 P_1 網頁時就該擷取這三個連結，而 P_2 網頁應將擷取的機會留給其餘未曾擷取但權重較大的連結。如此一來，可以確保優良連結會被擷取而且不會有重複擷取的情形。

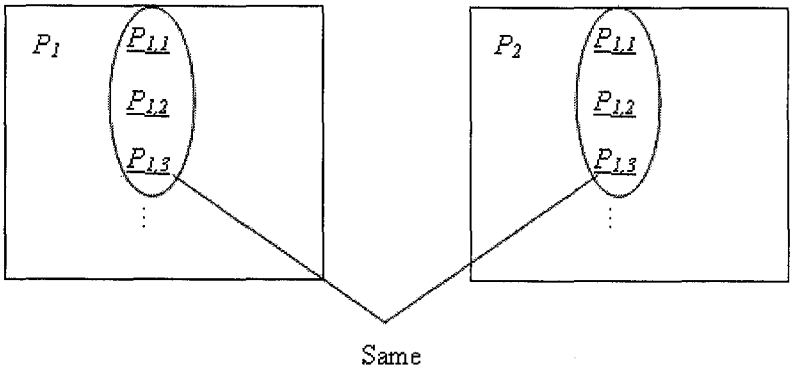

圖 4. 兩個網頁具有部份相同的連結

關於敗部的第 1 個議題，我們將敗部中所需擷取的連結個數分為兩部份: (1) 網頁有遺傳父代網頁的權重(亦即目前網頁的層數大於 1)； (2) 網頁無遺傳父代網頁的權重(亦即目前網頁的層數為 1)。我們利用下列公式來解決上述問題的第一部份:

$$F_p = \left\lceil \frac{\frac{\alpha_p}{\sum \forall_{p,l}(w_{p,l})} \times T_f \times (1+\beta)}{\log_{10}(T_f)} \right\rceil \quad (8)$$

其中 F_p 代表網頁 p 所需擷取的敗部個數，T_f 代表敗部連結的個數(亦即#$\forall_{p,l}$ - V_p，其中#代表個數)。公式 8 表示父代網頁相對權重愈大時，從該網頁抓取的敗部連結就愈多，此處所謂相對權重是指該網頁的權重α_p在所有連結權重總和($\sum \forall_{p,l}(w_{p,l})$)的一定比率，至於β是影響使用者觀看網頁的偏好程度，當β愈大時，使用者會看更多連結網頁，因此β愈大則該敗部連結抓取的個數就愈多(因為β<0，因此我們這裡定義為 1+β)，同時在這裡我們也考慮到懲罰成本的觀念(當敗部連結數愈多，連結被抓取的機會就會降低)，例如若要判斷 http://www.php.net/downloads.php 所需抓取的敗部個數(它在第 2 層)就是

$$\left\lceil \frac{\frac{4.78353}{221.48511872068} \times 211 \times (1+-0.77304)}{\log_{10}(211)} \right\rceil$$

關於上述問題的第二部份，即無遺傳父代的網頁，也就是 SVV 結果中每個 URL 所指的網頁，我們用下列公式計算:

$$F_p = \left\lceil \frac{V_P \times (1+\beta)}{\log_{10}(T_f)} \right\rceil \quad (9)$$

亦即敗部擷取的個數為勝部個數的一定比例。例如在 http://www.php.net 所需抓取的敗部連結個數(它在第 1 層)就是$\left\lceil \frac{4 \times (1+-0.77304)}{\log_{10}(167)} \right\rceil$。

關於敗部網頁的第 2 個問題: 判斷那些連結需要擷取，我們使用基因演算法中輪盤式選擇法(Goldberg 1999; Sullivan 1997; Obitko 1998)。輪盤式選擇法保證權重較大的連結其被選中的機率較大，這樣可以確保權重較大的連結即使落入敗部，其被選中的機率還是比較大(亦即敗部連結權重較大者，其敗部復活的機率較大)。

最後，我們的第三個判斷，即最多要擷取幾層，主要考量下列兩個因素: (1)控制系統資源的使用量；(2) 避免某些網站因"連結灌水"造成無止盡的擷取層數，例如: 在公式 5 中為了滿足$T_{p,l} < 10^{\alpha_p}$，若以某個網站權重α_p為 4.8586 為例，它必須從網站首頁累積到本身網頁的連結個數超過 70000 個以上。在最差的情形下，每一層網頁只有一個連結(即形成 skew tree)，則此時需要擷取超過 70000 層，這是不合理的。因此我們使用下列公式控制網站的最大擷取層數:

$$ML_s = \lceil w_s \rceil \quad (10)$$

其中 ML_s 代表網站 s 最大允許的擷取層數(亦即當處理每一層連結時，我們將與 ML_s 比較，若目前層數小於 ML_s 則該層繼續使用競賽方式判斷後代連結)。例如從 http://www.php.net 需抓取的層數為$\lceil 4.78353 \rceil$，亦即 5 層。

根據公式 7~10 的定義，我們瞭解 HLP 從重要網站抓取的內部連結將會較多，而且抓取層數也會較多。例如在圖 5 針對關鍵字"php"的 HLP 搜尋結果中，前三項 URLs 均來自 SVV 處理後權重最大的網站 http://www.php.net。此外，HLP 除了能夠顯示與 SVV 相同定義的高中低相關及長條顏色框外，系統還將 HLP 所擷取的連結以 Cached 網頁顯示。

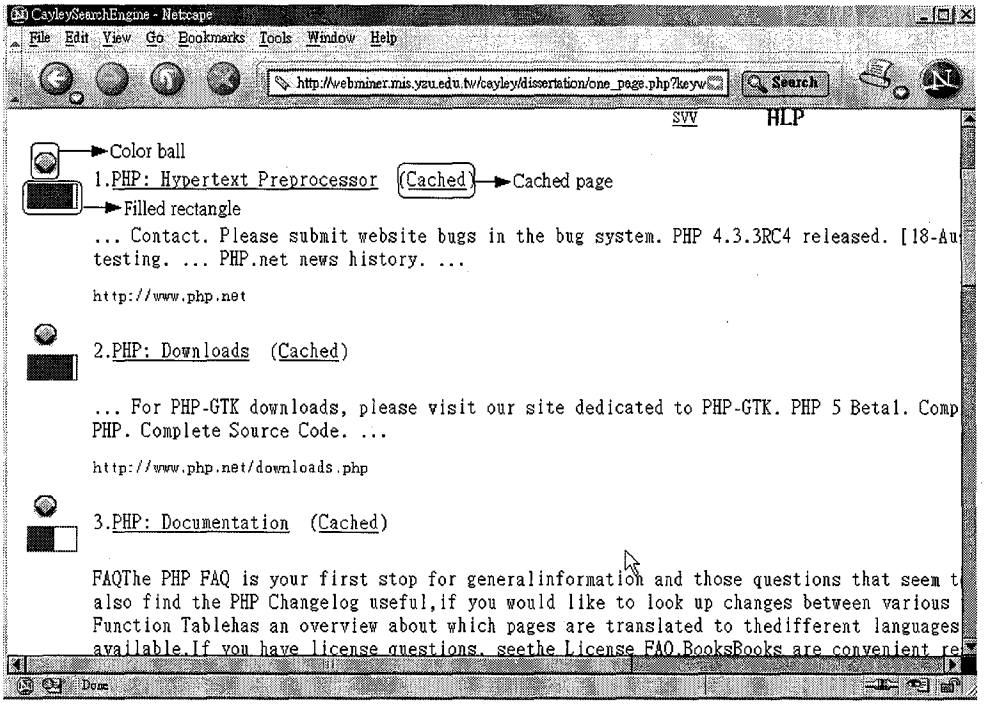

圖 5. HLP 搜尋結果範例

伍、系統評估測試

本節討論兩項系統測試結果，首先我們要驗證在網頁權重>平均權重+3 倍權重標準差的條件下，使用者最滿意我們的搜尋結果；接著，我們比較使用者對 SVV、HLP 及上述六個搜尋引擎的滿意度。

一、最佳權重條件評估

本小節主要在驗證在網頁權重>平均權重+ n 倍權重標準差的條件下，當 n=3 時(用於公式 4 、7)，使用者最滿意我們的搜尋結果。我們的測試對象是一班有 31 位學生的大學部班級，每位學生獨立的使用相同查詢關鍵字對 n=2..6 等 5 種案例進行查詢，然後對每個案例給一個滿意度 0~5 評分。

從圖 6 的滿意度分數分布顯示，有 25 位學生在 $n=3$ 時給了 4 分，是所有案例中最高的。由於 $P(w_p > \overline{w}+3\sigma_w) < \frac{1}{9}$，此時我們處理勝部的資料量低於總數的 11%，測試結果顯示使用者對這樣的資料量感到最滿意。至於 n = 2 時，系統將於勝部產生過多的資料，一般使用者不僅無法消化，反倒成為做決策時的負擔。至於 n = 4, 5, 6 時，系統產生的勝部資料極少，此時大部份資料都是從敗部選出，對使用者的幫助不大。

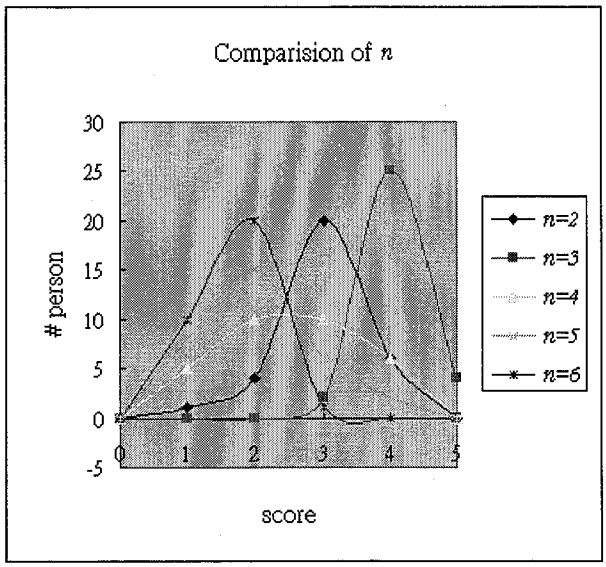

圖 6. 不同標準差對使用者的滿意度比較

二、搜尋引擎比較測試

本實驗主要在測試使用者對我們的 SVV 和 HLP 方法與上述六個搜尋引擎的評比。我們從網路上邀請 23 個自願者當成我們的測試對象，分別對 Google[1]、Yahoo、AltaVista、LookSmart、Overture、Lycos 以及我們的搜尋方法 SVV 及 HLP 進行測試評比。根據平時測試結果，我們發現六個引擎中的 Google 及 Yahoo 搜尋品質較佳，因此在 SVV 中有

[1] Google 最近兩年已經不允許所有來自 MetaSearch 的查詢，為了能夠對 Google 進行存取，我們透過修改 HTTP-Header 中的 User-Agent 為"Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0; T312461)"，將 Crawler 模擬成 IE 5.5 的形式。

關這兩個搜尋引擎的參數設定值較高，以便讓 SVV 符合大部份使用者的搜尋預期。此外，為了讓SVV搜尋結果排名合理，我們去除了AltaVista、LookSmart、Lycos以及Overture搜尋結果中的廣告連結。

測試者被要求使用從 Lycos 50 (2003)選取的熱門關鍵字進行查詢，這些關鍵字包括 "Dragonball"、 "Pamela Anderson"、 "Britney Spears"、 "Las Vegas"、 "Harry Potter"、 "Kazaa"、 "WWE"、 "Jennifer Lopez"、 "Final Fantasy"、 "Yu-Gi-Oh!"、 "The Bible"、 以及 "NBA"。同時，測試者獨立的對每個搜尋引擎的搜尋結果進行總體性滿意度的評分，評分範圍為 1 到 5。

測試結果如表 3 所示，表中的每一儲存格(不包含最後一列)是所有使用者評給特定搜尋引擎對特定查詢的平均分數，例如：Google 在"Dragonball"查詢的平均得分是 4.18。表 3 的最後一列則表示每個搜尋引擎的總平均得分。總括來說，測試的結果符合我們的期望，亦即 HLP 得分高於 SVV，然後 SVV 又比其它搜尋引擎略高。 HLP 得分 4.03 是在所有評比的搜尋引擎之中最高的，SVV 得分 3.88 與 Google 的 3.81 及 Yahoo 的 3.72 相當，這樣的結果符合我們在 SVV 中對 Google 和 Yahoo 的參數設定。這個測試結果顯示使用者確實比較偏好深度的資訊推薦，因此對 HLP 的滿意度高於其他搜尋引擎。

表 3. SVV、HLP 及其它搜尋引擎的滿意度比較

	Google	Yahoo	AltaVista	LookSmart	Overture	Lycos	SVV	HLP
Dragonball	4.18	4.23	3.08	3.57	2.25	3.51	4.18	4.33
Pamela Anderson	3.84	3.51	2.11	3.74	2.90	3.56	3.79	4.04
Britney Spears	4.09	3.43	2.78	4.02	1.77	3.33	3.93	4.34
Las Vegas	3.81	4.23	3.36	3.55	1.32	3.74	4.02	4.18
Harry Potter	3.15	2.69	3.70	3.77	1.33	3.27	3.86	3.77
Kazaa	4.11	3.86	3.71	3.95	2.34	3.86	4.03	4.07
WWE	3.97	3.74	3.38	3.73	2.25	3.51	3.95	4.22
Jennifer Lopez	4.29	3.95	3.02	3.16	2.67	3.66	4.02	3.45
Final Fantasy	3.51	4.01	3.52	3.66	2.28	3.37	3.83	4.33
Yu-Gi-Oh!	4.30	4.52	2.41	3.33	2.71	3.29	4.33	4.46
The Bible	2.73	2.63	2.09	2.86	2.21	2.90	2.57	2.79
NBA	3.71	3.82	3.77	3.79	1.08	3.35	4.04	4.35
Average	3.81	3.72	3.08	3.59	2.09	3.45	3.88	4.03

不過，HLP 在某些特殊的查詢關鍵字得到比較低的分數(例如："Jennifer Lopez")，這是因為當網頁沒有描述文字或包含許多的圖檔、聲音及電影等非文字的元素時，HLP 無法有效預測連結。例如在 "Jennifer Lopez" 查詢時排在最前面的網址

http://www.jenniferlopez.com/ 中就沒有包含任何的描述文字可供連結預測。

此外，在所有搜尋引擎之中，Overture 得到最低的 2.09 分，主要原因在於其搜尋結果前半部放置太多廣告連結。另一方面，"The Bible" 這個關鍵字在每個搜尋引擎的分數都不高，其主要原因是我們的測試對象之中有大部份是要找尋程式語言主題中的 "The Bible Book"。

陸、結論

本論文提出兩種搜尋引擎的技術: 搜尋引擎向量投票(SVV)和連結預測(HLP)。在 SVV 中，網站的排名是由搜尋引擎彼此的投票行為決定，當某個網站能夠獲得較多搜尋引擎的投票或者它能在搜尋引擎得到較好的排名，則它的權重將較大。在 HLP 中，我們根據使用者的行為偏好，針對搜尋結果中權重較大的網站，使用者可能點閱較多的網頁而且可能觀看較多層數來進行連結預測。使用者測試結果顯示使用者對 HLP 和 SVV 的滿意度高於其他搜尋引擎。

未來仍有許多工作尚待進一步研究，首先，我們希望能在 SVV 及 HLP 的成果中直接預測出使用者心中可能想要的答案，以節省使用者瀏覽與判斷網頁的時間，例如當查詢為 php 時，使用者心中很可能想要瞭解 "What is php?"，如果我們能從 HLP 所回傳的網頁中找到一個段落包含"... php is ..."，同時將相關的段落組合成一個新的網頁剪報，這樣將對使用者產生更大的助益。同時，我們也希望進行更大規模的使用者測試，藉以微調系統機制，以便連結預測結果更貼近使用者的需求。最後，本搜尋引擎目前只支援英文關鍵字的搜尋，我們將持續加入 openfind, Yahoo 中文等搜尋引擎做為資料搜尋的來源，即可提供中文關鍵字的搜尋功能。

註:我們搜尋引擎網址是 http://cayley.sytes.net/dissertation/ 或
http://webminer.mis.yzu.edu.tw/cayley/dissertation/

致謝

本研究部分接受國科會研究計畫案 NSC 91-2213-E-155-039 資助，特此致謝。

參考文獻

1. Chakrabarti, S., et al. "Mining the Web's Link Structure," *IEEE Internet Computing*, August 1999, pp. 60-67
2. Dreilinger, D. and A. Howe, "An Information Gathering Agent for Querying Web Search Engine," Tech. Report CS-96-111, Computer Science Dept., Colorado state Univ., Fort Collins, Colo., 1996
3. Goldberg, D. E., *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison Wesley, 1999
4. Harman, D., "Ranking Algorithm," in *Information Retrieval Data Structure and Algorithm*, Prentice Hall, 1992
5. Henzinger, M. R., "Hyperlink Analysis for the Web," *IEEE Internet Computing*, Jan.-Feb. 2001, pp. 45-50
6. Lempel R. and Moran S., "The stochastic approach for link-structure analysis (SALSA) and the TKC effect," *Computer Networks*, 33, 2000, pp. 387-401
7. Li, Y., "Toward A Qualitative Search Engine," *IEEE Internet Computing*, July-August 1998, pp. 24-29
8. Lycos 50, ""Lycos50 with Aaron Schatz – Lycos.com," http://50.lycos.com/, 2003
9. Morris, C.G., A. Levine and A. A. Maisto, Psychology: An Introduction, 11th ed., Prentice-Hall, New Jersey, 2002.
10. Nick, Z. Z., and P. Themis, "Web Search Using a Genetic Algorithm," *IEEE Internet Computing*, Mar.-Apr. 2001, pp. 18-26
11. Nielson NetRating: Search Engine Rating, January 2003, available at http://searchenginewatch.com/reports/article.php/2156451
12. Obitko, M., "Introduction to genetic algorithms with Java applets", available online at http://cs.felk.cvut.cz/~xobitko/ga/, 1998.
13. Paepcke A., Garcia-Molina H., Rodriguez-Mula G. and Junghoo Cho, Beyond Document Similarity: Understanding Value-Based Search and Browsing Technologies, *SIGMOD Record*, 29, 2000, pp. 80-92
14. Pinkerton, B., "Finding What People Want: Experiences with the WebCrawler," Proceedings of Second International WWW Conference, 1994; available online at http://www.thinkpink.com/bp/WebCrawler/WWW94.html.
15. Selberg, E., and O. Etzioni, "The MetaCrawler Architecture for Resource Aggregation on the Web," *IEEE Expert*, Jan.-Feb. 1997, pp.11-14; also available at http://www.selberg.org/homes/speed/papers/ieee/ieee-metacrawler.ps

16. Spink, A. D., Wolfram, B. J. Jansen, and T. Saracevic, "Searching the Web: The Public and Their Queries," *Journal of the American Society of Information Science* **53**, No. 2, 226–234, 2001
17. Sullivan, D., "What People Search For", INT Media Group©, available online at http://searchenginewatch.com/facts/searches.html, 2002.
18. Sullivan, M., "An Introduction to Genetic Algorithms," available online at http://www.cs.qub.ac.uk/~M.Sullivan/ga/ga_index.html, 1997.
19. Walpole, R. E., and R H. Myers, *Probability and Statistics for Engineers and Scientists,* Macmillan, 1993.

附錄一

SVV 搜尋的虛擬碼

```
Algorithm search (query, method)
{
    Switch (method)
    {
        Case SVV:
            Concurrently collect the search results to the query from 6 search engines;
            Do a page analysis to collect a website's rank from the above collection;
            Do a SVV analysis on the results of the page analysis;
            Set the SVV results to be the analysis results;
            Break;
    }
    Do a Summary analysis on the analysis results
    {
        Sort the analysis results according to their weights;
        Layout the results depending on the specific requirement of each method;
        Combine the results and the web pages to be final results;
        Show final results;
    }End of Do;
}End of Algorithm;
```

附錄二

HLP 搜尋的虛擬碼

```
Algorithm search (query, method)
{
    Switch (method)
    {
        Case HLP:
            If (the SVV result is NULL)
                Do search(query, SVV);
            Set the SVV results to be the input set of the HLP analysis;
            Do a HLP analysis
            {
                Divide the input set into winner set and loser set;
                Do a tournament competition on the winner and loser sets;
                Concurrently collect the web pages according to the result of tournament
                  competition;
            }End of Do;
            Set the number of current layer to 1;
            While (the number of current layer ≤ max(the weight for all websites))
                {
                For all collected pages
                {
                    If (a web page satisfies hyperlink collection condition and the weight
                      of this web page ≤ the number of current layer)
                    {
                      Do a page analysis to collect all hyperlinks from this web page;
                      Compute the weight of these hyperlinks;
                      Set these hyperlinks to be the input set of the HLP analysis;
                      Do a HLP analysis;
                    }
                }End of For;
                The number of current layer plus one;
```

```
            }End of While;
            Set all collected hyperlinks to be the analysis results;
            Break;
    }End of Switch;
    Do a Summary analysis on the analysis results
    {
        Sort the analysis results according to their weights (SVV or HLP);
        Layout the results depending on the specific requirement of each method;
        Combine the results and the web pages to be final results;
        Show final results;
    }End of Do;
}End of Algorithm;
```