具有網頁連結預測能力的智慧型搜尋引擎之設計

陳林志
台灣科技大學資管所博士班
cayley0881@sinamail.com

陸承志
元智大學資管系
imcjluh@saturn.yzu.edu.tw

摘要

在本論文中，我們根據使用者行為的考量探討一個具有網頁連結預測能力的智慧型搜尋引擎之設計。首先，我們發展一個搜尋引擎投票機制，這個機制能讓每個網站的排名合理，同時網站的排名不會因某些特定搜尋引擎的商業定位導致排名"誤導"。接著，我們發展一個全新的智慧連結預測方法，能夠預測使用者可能點擊的所有連結。經過初步的評估，我們發現使用者對於系統的滿意度高。

關鍵字: 搜尋引擎向量投票、連結預測、搜尋引擎、MetaSearch、使用者行為函數

1. 前言

搜尋引擎設計的主要目的是:如何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。根據資料顯示，使用者平均每次輸入 term 的個數是 1.5[1]，如何根據這些 terms 取得相關資料即是各個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 [2]，然後主要根據文件中所包含的關鍵字進行文件排名，但是這樣處理是有問題的，因為這樣會鼓勵網站管理者在文件中加入許多的關鍵字，即可將網站的排名提前，形成"關鍵字灌水"的情形。

為了避免上述的情形，Li 發展出 HVV 的方法 [3]，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 提出一個不同的觀念，亦即在相同查詢下，代表性站台所評比網站的權重往往較大 [4]，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。但這樣的方法還是同樣面臨到第 1 個問題。根據文獻資料，網頁數量以每天 3 億個網頁成長 [5]，在這些資料中，我們很難判斷需要多少評比站台才足夠，除非我們所收集的網頁超過一定的數量，評比的效果才可能比較好。

從另一個觀點，我們可以利用一些搜尋引擎的結果進行評比，這樣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.避免搜尋引擎介面不同造成使用者學習上的困難。這樣的搜尋技術稱為 MetaSearch [6,7]，例如: 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)，其主要精神是由市場上幾個知名的搜尋引擎進行向量投票，保證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。

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幫助。Nick 和 Themis 運用 Intelligent Agent 的方法希望搜尋引擎能夠擁有智慧且能提供一些網頁給使用者 [8]，但是它需要使用者提出一些範例，這對使用者是很困難的，同時也可能因為特定使用者的喜好輸入不合適的範例，最後造成建議不合適的網頁。

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(Link Prediction, LP)，其主要精神是: 當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的網頁而且網站的擷取層數也應較多。因為使用者針對那些有興趣的站台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，然後介紹這兩個方法，最後我們介紹初步系統實作的結果。

2. 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(LP，Link 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 sets)、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 LP)、網頁分析(Pages analysis)、彙總分析(Summary SVV 及 Summary LP)、搜尋引擎向量投票分析(SVV analysis)及連結預測分析(LP 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，使用者輸入查詢後，系統將送至 6 個標準搜尋引擎: 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，系統判斷有那些網頁需要擷取，這些擷取回來的網頁除了可以在 LP 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。網路連結主要是透過網路，將連結預測分析後的網址擷取回來，擷取的網頁會儲存成庫存網頁，同時將網頁內的相關資訊儲存於資料庫。

在內部模組中，系統將會根據不同使用者介面以便回應適當的資料，從系統架構中，我們知道 LP 的輸入是搜尋引擎向量投票分析之後的結果，由於 LP 完成時，SVV 也一定完成，因此當使用者選擇 LP，也可隨時切換到 SVV，此時使用者的選擇性變多。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。反之當使用者選擇 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所定義的網站排名，LP 所需的資料是網頁內的連結)。彙總分析將內部模組處理完

的資料以適當的版面回應給使用者。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。

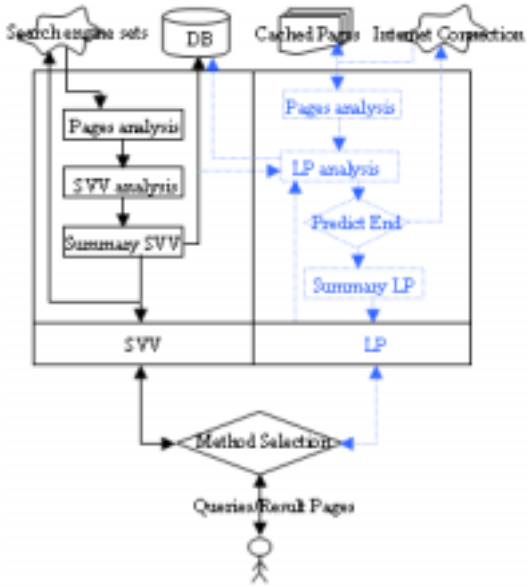

圖 1. Cayley 搜尋引擎系統架構

3. 搜尋引擎向量投票分析

在 SVV 方法中，我們主要是採用投票系統，亦即由 6 個標準搜尋引擎來進行投票。在計算網站權重之前，我們瞭解一般使用者對於排名前面網站的喜好度會比後面網站來得大，而且喜好度會逐漸遞減，因此我們根據一般使用者的行為定義出下列使用者行為函數 (User Behavior Function, UBF):

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

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)代表使用者對文件偏好度，當β愈大使用者偏好度愈強(例如圖 2 中，β= -0.3 時的上方曲線)，代表使用者會看較多的網頁，亦即使用者偏好效應降低將較慢。當α及β確定時，使用者對前面網頁的偏好會較後面網頁來得大。

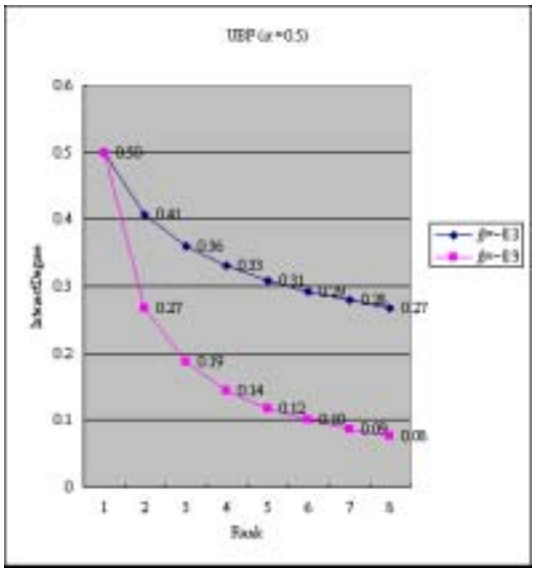

圖 2. 當α=0.5，β= -0.3 及β=-0.9 時的使用者行為函數

SVV 所計算出的每個網站的權重主要是表達 6 個標準搜尋引擎的投票趨向，因此權重是由每個搜尋引擎所表達的使用者行函數透過向量內積計算而得。SVV 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。我們根據權重的定義瞭解下列兩種情形的權重會較大: 1.較多搜尋引擎投票給它，2.網站在單一搜尋引擎中排名較前面。在圖 3 的 SVV 結果中，網站的排名是根據每個網站的權重排序而得。

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趨向度，其公式如下:

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

其中 P_s 代表網站 s 在 6 個搜尋引擎投票趨向的比例，亦即圖 3 的長條顏色框。 因此若有一網站在六個搜尋引擎的搜尋結果中都排第一的話，其長條顏色框將百分之百填滿顏色。

為了兼顧例外的情況，例如搜尋引擎投票趨向不一致或者某個少數網站權重佔了投票的大部份結果，此時若我們只顯示長條顏色框時，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聯程度:

$$R_s = \begin{cases} High, & w_s \geq \overline{w} + 3\sigma_w \\ Middle, & \overline{w} \leq w_s < \overline{w} + 3\sigma_w \\ Low, & elsewhere. \end{cases} \quad (4)$$

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 [13]，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。我們以不同色球表達網站間的相關程度，例如圖 3 中排名 1~4 的網站其相關程度為高，排名 5 的網站其相關程度為中。

圖 3. SVV 的結果

4. 連結預測分析

在 SVV 的結果中，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分析的機制。連結預測針對權重較大網站的處理分為二部份: (1) 擷取較多網站內的連結，(2) 擷取網站內的層數也應較多。

當我們進行連結預測分析時，我們必須做三個判斷: (1)根據權重判斷某個網頁是否符合可以擷取其後代，(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。當我們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後代連結(亦即後代連結將繼承父代網頁的權重)。我們的第一個判斷是根據下列公式:

$$G_p = \begin{cases} 1, & if \left\lceil \dfrac{\alpha_p}{\log_{10}(T_{p,l})} \right\rceil > 1 \\ 0, & elsewhere \end{cases} \quad (5)$$

其中 G_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)，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第二層的內部連結)，$T_{p,l}$ 代表在網頁 p 中從網站首頁(亦即 SVV 的網站)到網頁 p 所經過所有祖先的累積連結總和，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。另一方面，我們考量到 $\log_{10}(T_{p,l})$ 函數的特性，當 $T_{p,l} \leq 10$ 造成 $\log_{10}(T_{p,l}) \leq 1$，此舉將使得 G_p 更容易達成擷取條件，亦即鼓勵使用者灌水，因此我們定義懲罰成本的上限。當 $T_{p,l} \leq 10$ 時，我們令 $T_{p,l}=10$，亦即不到一定比例不懲罰。

我們第二個判斷主要是使用競賽方式來判斷那些連結需擷取。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，我們定義勝部為: 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。現在將面臨到兩個問題: (1)勝部及敗部各需擷取的連結個數, (2)擷取那些連結。在做上述判斷之前，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($w_{p,l}$ 會遺傳至下一層，同時成為下一層的 α_p)，$x_{p,l}$ 代表連結 l 在網頁 p 的排名。接下來我們討論勝部的第 1 個問題, 我們主要是透過下列公式來判斷所需擷取連結的個數:

$$V_p = \sum \forall_{p,l} (w_{p,l} \geq \overline{w_p} + 3\sigma_{w_p}) \qquad (7)$$

其中 V_p 代表網頁 p 所需擷取的勝部個數, $\forall_{p,l}$ 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} 代表網頁 p 中所有連結權重的標準差。

接下來我們討論勝部的第 2 個問題, 我們勝部擷取的連結是 " V_p 中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(亦即連結權重≥平均連結權重+3 倍連結權重標準差)? 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，則我們在處理 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。

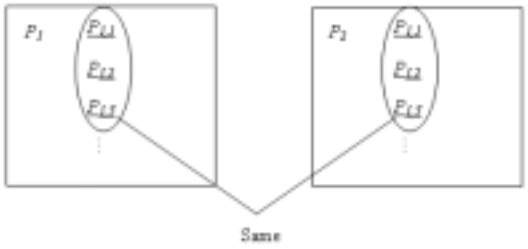

圖 4. 兩個網頁具有部份相同的連結

關於敗部的第 1 個問題, 我們將敗部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)。關於上述問題的第一部份我們用下列公式計算:

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

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)，父代網頁權重較大者其子代可能較優良, 因此擷取的敗部個數將較多, T_f 值愈大將造成使用者決策時的困難, 因此該參數為懲罰成本。關於上述問題的第二部份，即網頁無遺傳父代網頁者，我們用下列公式計算:

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

在這個部份中，因為網頁無遺傳父代權重，此時我們無法瞭解父代權重, 因此敗部擷取的個數為勝部個數的一定比例。

關於敗部網頁的第 2 個問題: 判斷那些連結需要擷取, 主要使用基因演算法中輪盤式選擇法 [9-11]，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)。

我們的第三個判斷，即最多要擷取幾層，主要是考量下列兩個因素: (1)控制系統的資源使用量到一定程度, (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，我們為了避免無止盡的$T_{p,l}<10^{\alpha_p}$，例如以某個網站權重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 個以上的連結。當在最差情形中，每一層網頁只有一個連結(形成 skew tree)，則此時我們將要擷取超過 70000 層，這是不合理。因此我們使用下列公式控制網站的最大擷取層數：

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

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)。

在圖 5 的 LP 結果中，系統將會針對 SVV 權重較大網站進行兩項處理: 1.擷取較多的連結，2.擷取層數也較多。LP 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，系統還會將 LP 所擷取的連結以 Cached 網頁顯示。

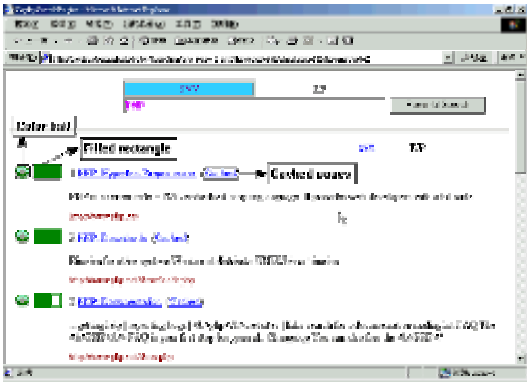

圖 5. LP 的結果

5. 系統測試

我們首先測試使用者對我們系統的滿意度。在這個測試下，我們使用兩個大學部班級當成我們的測試樣本，兩組在不同時間及不同地點，針對相同查詢進行查詢。第一組擁有 28 人，主要是測試系統未處理過的查詢。第二組擁有 31 人，主要是測試系統已處理過的查詢。滿意度給分為 0~5 分，最低 0 分，最高 5 分。我們根據表 1 及表 2 得知，若查詢尚未處理過，則大部份給分為 3，否則大部份給分為 4。

表 1.第一組滿意度得分分佈

Score	0	1	2	3	4	5
Number	0	1	1	20	6	0

表 2.第二組滿意度得分分佈

Score	0	1	2	3	4	5
Number	0	0	0	2	25	4

為什麼同樣的查詢給分差距會如此之大? 主要原因是使用者在滿意度有不同的考量因素。一般使用者對於搜尋引擎滿意度的考量因素可以分三部份: (1)精確率，(2)幫助性，(3)回應時間。第一組因查詢尚未處理過，系統需要將網頁擷取回來，造成系統回應時間變長，因此，使用者對於滿意度將產生偏差。第二組因查詢已處理過，此時使用者對於滿意度考量將只剩下精確率及幫助性。在這樣的環境下，系統將最能表達使用者的滿意度。搜尋引擎在開始操作將會面臨到擷取網頁的問題，一般的解決方法有兩種: (1) 預先輸入常用的查詢[12]，(2)使用 Robot 機制擷取使用者可能點擊的網頁，我們也將同時運行這兩個機制以便降低系統回應時間。

在我們的第二個測試中，我們要驗證網頁權重≥平均權重+3 倍權重標準差對使用者幫助性最大，此處的測試樣本為上述測試的第二組樣本。根據 Chebyshev's 定理[13]，我們知道 $P(w_s \geq \bar{w} + n\sigma_w) < \frac{1}{n^2}$，接下來我們要顯

示當 $n = 3$ 時，使用者的滿意度將會最大。圖 6 顯示在不同標準差，使用者滿意度的差別，我們主要測試當 n 為 2，3，4，5 及 6 的情形。

圖 6 顯示當 $n = 3$ 時，使用者的滿意度將達到最高，這是為什麼呢？因為我們知道 $P(w_p \geq \overline{w} + 3\sigma_w) < \frac{1}{9}$，此時我們處理勝部資料時只顯示低於總數的 10%，這些資料量對使用者滿意度將得到最大，亦即勝部資料量不會太多也不會太少。至於 $n = 2$ 時，系統將於勝部產生過多的資料，一般使用者不需那麼多的勝部，反倒會讓使用者做決策時的負擔。至於 $n = 4$ 時，系統產生的勝部資料極少，此時因大部份資料都是從敗部選出，這樣代表使用者選取的資料大部份將從敗部選取，如此對使用者的幫助性將不會太大。因此當 n=3 時，系統所產生的資料將對使用者幫助最大。

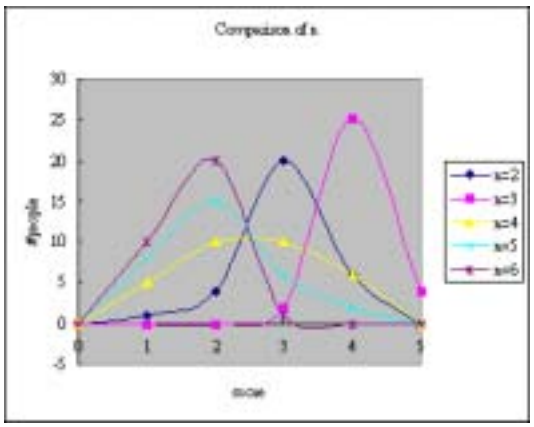

圖 6. 不同標準差對使用者的滿意度比較

6. 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(LP)。在 SVV 中，網站的排名是表達搜尋引擎彼此的投票行為，當某個網站能夠被較多搜尋引擎投票或它能在搜尋引擎得到較好的排名，則它的權重將較大。在 LP 中，我們根據使用者的行為進行網頁預測，然後根據 SVV 中權重較大的網站，使用者可能點閱較多的網頁且可能觀看較多層數來進行連結預測。

未來仍有許多工作尚待完成，首先，我們希望能在 SVV 及 LP 的成果中直接預測出使用者心中可能想要的答案，節省觀看網頁所需做的額外判斷。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 LP 所尋得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幫助。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。

註：我們搜尋引擎網址是 http://cayley.dorm.ntust.edu.tw/dissection/

8.參考文獻

1. B. Pinkerton， "Finding What People Want: Experiences with the WebCrawler，" Proc. Second Int'l WWW Conf.， 1994; available online at http://www.thinkpink.com/bp/WebCrawler/WWW94.html.
2. D. Harman， "Ranking Algorithm，" Information Retrieval Data Structure and Algorithm， Prentice Hall， 1992.
3. Y. Li， "Toward A Qualitative Search Engine，" IEEE Internet Computing， July-August 1998， pp. 24-29.
4. M R. Henzinger， "Hyperlink Analysis for the Web，" IEEE Internet Computing， Jan.-Feb. 2001， pp. 45-50.
5. S. Chakrabarti et al. "Mining the Web's Link Structure，" IEEE Computing， August 1999， pp. 60-67.
6. D. Dreilinger 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.

7.E. Selberg 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

8. Z Z. Nick and P. Themis, "Web Search Using a Genetic Algorithm," IEEE Internet Computing, Mar.-Apr. 2001, pp. 18-26.

9.D E. Goldberg, "Genetic Algorithms in Search, Optimization and Machine Learning," Addison Wesley, 1999.

10.M Sullivan, "An Introduction to Genetic Algorithms", available online at http://www.cs.qub.ac.uk/~M.Sullivan/ga/ga_index.html.

11.M. Obitko, "Introduction to genetic algorithms with Java applets", available online at http://cs.felk.cvut.cz/~xobitko/ga/.

12. D. Sullivan, "What People Search For", INT Media Group©, available online at http://searchenginewatch.com/facts/searches.html.

13. R E. Walpole and R H. Myers, "Probability and Statistics for Engineers and Scientists," Macmillan, 1993.