全球資訊網之資訊萃取及自動採擷方法之研究(II)

The Design of An Information System for Hypertext Retrieval and Automatic Discovery on World Wide Web (II)

計畫編號 : NSC-89-2213-E002-022

執行期限 : 88 年 8 月 1 日至 89 年 7 月 31 日

主持人: 許清琦　台灣大學資訊工程學系暨研究所教授

中文摘要（關鍵詞: 資訊檢索 全球資訊網 超鏈結）

全球資訊網的成功在於簡易的使用介面，使得資料的取得與散播較之前的網路應用程式更為大眾所接受。據相關研究報告指出，一般使用者上網最主要活動就是找資料。在本研究中，我們的焦點在於超鏈結搜尋機制的探討。我們將討論如何利用鏈入值、鏈出值來幫助超鏈結搜尋，進而歸納出合適的超鏈結搜尋機制。如此一來，便能在全球資訊網上有效的發掘出相關的文件。我們使用了多種不同的航行演算法來比較他們的優劣。

英文摘要(Keywords: Information Retrieval, WWW, Hyperlink)

The advent of WWW makes information gathering and dissemination much easier than before. According to related survey, Internet users spend most of their time searching for information. Our goal is to build an automatic information discovery system based on hyperlink structures to facilitate information finding tasks. We implements a information discovery system embedded with 5 algorithms and compare them for their effectiveness.

二. 計劃緣由與目的

近年來，在 Internet 上的資料量迅速增加。因此，當一個使用者想要在全球資訊網上找尋想要的資料時，往往會感到費時而徒勞無功。很明顯的，這對使用者來說是一個嚴重的問題。

通常，我們可以用兩種方法來解決這個問題。第一種方法，就是經由網路上的『搜尋引擎』來找出想要的資料。所謂『搜尋引擎』，其實就是一個全球資訊網上的索引資料庫。在平時，這些搜尋引擎就試著去探索資訊網上的網頁，而試著對他們去作索引。所以當有使用者對搜尋引擎發出詢問時，這些搜尋引擎便可以利用已經索引好的資料庫去找出相關的文件，並傳回給使用者。這時，使用者便可以試著去瀏覽這些結果，看看是否有想要的文件。

另一方面，由於全球資訊網是基於超鏈結所形成的架構，我們也可以試著去直接利用超鏈結來搜尋相關的文件，以找出想要的資料。

對於搜尋的種類，我們也可以概分為兩種。第一種，我們稱之為線上搜尋。在這種情況下，使用者是立即就想知道搜尋的結果；使用者會希望所詢問的結果能立即傳回。要解決這種搜尋問題的方法是利用搜尋引擎。由於搜尋引擎在平時便去索引整個資訊網上的資料，所以他能在短時間內便將結果傳回給用者。

另一種情況，我們稱之為離線搜尋。在這種情況下，使用者不需要立即就知道搜尋的結果。一般來說，使用者會對某些主題有持續性的興趣。像是一個使用者若今天對籃球有興趣，那麼他不可能到了明天就對這個主題喪失了興趣。所以，我們可以試著去設計一些離線搜尋的工具，在背景持續性的去替使用者發掘出相關的文件。

在這種情況下，我們便可以利用超鏈結去發掘出相關的文件。由於使用者不會需要馬上就知道搜尋結果，所以一個背景搜尋的系統可以利用超鏈結去航行於全球資訊網中，去發掘出使用者所感興趣的資料。

在本研究中，我們的焦點在於超鏈結搜尋機制的探討。我們將討論如何利用鏈入值、鏈出值來幫助超鏈結搜尋，進而歸納出合適的超鏈結搜尋機制。如此一來，便能在全球資訊網上有效的發掘出相關的文件。

三. 研究方法與成果

3-1 資訊擷取中常用技術

一般來說，我們利用向量空間模型來計算一篇文章和詢問間的相似度。而在計算前，我們必須先對文章去除 stop word 和作 stemming。

所謂 stop word，就是在英文中出現太頻繁的文字。這些文字並不能幫助我們粹取出一篇文章的特性。所以我們必須先把這些文字濾掉，而留下一些比較具有鑑別力的文字。

而後，我們要做的是 stemming。所謂的 stemming 就是 term conflation。我們把意思相同而形式不同的關鍵字做合併。

在上面兩個步驟都做完之後，我們將詢問和文章都做成向量的形式。在向量空間中，文章和詢問都是以向量的形式來呈現。

$$\overrightarrow{Q} = (q_1, q_2, \ldots, q_i)$$

$$\overrightarrow{D} = (d_1, d_2, \ldots, d_j)$$

其中向量Q是詢問，向量D是文章。

在這兩個向量中，每一個元素就是關鍵字的比重。至於我們要如何去計算出每一個關鍵字的比重呢？我們可以應用 *tf*idf* 的觀念來計算。所謂 *tf* 便是 term frequency，而 *idf* 便是 inverse document frequence。我們用一個關鍵字在某篇文章中出現的數量除以有多少篇文章擁有此關鍵字的數量來計算出此值。一般來說，這些向量都會標準化成長度為一的向量，以便計算。

至於我們要如何來計算一篇文章和詢問間的相似度呢？答案就是計算這兩個向量間的餘弦值。要是向量都已經過標準化，那麼我們就可以直接利用向量的內積來計算餘弦值。所以，詢問和文章間的相似度可用以下的公式來計算。

3-2 效能的估定

在傳統的資訊擷取領域中，我們常用 precision 和 recall 兩個值來估定一種方法的好壞。所謂 precision 就是指在取出的資訊中，有多少比例是相關的部分；而 recall 就是指在相關的資訊中，我們取回了多少比例。

一般來說，我們會希望我們的方法能有高的 precision 值和好的 recall 值。因為一個方法若是只能擁有高的 precision 值，那就意味著他只取出了相關資訊中的一小部分。對於使用者來說，越多的相關資訊是越有用的。因此，好的 recall 值也是一個好方法所不可或缺的。

3-3 全球資訊網上的航行演算法

我們使用了多種不同的航行演算法來比較他們的優劣。前三種方法稱之為基本模式。

第一種是廣度優先法(Breadth-First)。這種演算法是很簡單而直接的，只是把在文件中的超鏈結以廣度優先的順序來加以探索。由於沒有任何機制來決定一篇文章是否應該收錄到資料庫中，所以這種演算法的 precision 會不高。

第二種是以文件內容為基準的航行方式(Text-Based)。在 Arachnid 中，他也是利用這種方式來航行於全球資訊網上。當此方法探索到一篇文章時，他會去計算出此篇文章和詢問間的相似度，並以此相似度來當作決定值。要是決定值高於我們所定的門檻時，我們便把他收錄到資料庫之中；反之，要是決定值低於門檻，此篇文章就會被捨棄。

在這種方法中，會試著去利用文章的內容來決定是否該收錄。因此，這種方法的 precision 應該是會較高。但由於此方法會以相似度來判斷一篇文章是否值得收錄，所以此方法會捨棄掉部分探索到的文章。如此一來，不僅某些網路的頻寬會被浪費掉，所執行的時間也會比較長。

第三種方法(CPR)，則一種是以文章的內容再加入了超鏈結資訊的新方法。這是我們所提出來的新嘗試。我們試著去利用一篇文章的本文，有多少鏈結指到它(鏈入值)，以及它擁有多少鏈結(鏈出值)來決定此篇文章的決定值。同樣的，在用這種方法時，我們也訂出一個門檻值。要是所計算出來的決定值高於此門檻，我們則收錄此篇文章；反之，則丟棄。

要獲得一篇文章的鏈出值並不困難，我們只要剖析文章的內容便可知道。但是，我們要如何得知鏈入值呢？解決的方法是去詢問搜尋引擎。由於搜尋引擎可說是整個全球資訊網上的索引資料庫，因此這項資訊對它來說並不是一件難事。在系統實作時，我們是經由詢問 Hotbot 來得知此項資訊。

當然，因為又加上了詢問 Hotbot 的時間，所以此方法更是耗時。

以下稱之為高級模式。在高級模式中，每一種方法都是以前述的第三種方法為基底而加以強化。

在第四種方法(VCPR)中，我們採用了調整式門檻。通常，由於不同詢問間的一般性不盡相同，所以我們不能對於不同的詢問設下相同的門檻值。在此方法中，我們試著對不同的詢問動態去訂出適當的門檻值。好讓靠近起始點的文件不會過早填滿資料庫，同時也不會因門檻值過高值造成沒有結果傳回的窘境。

在第五種方法(EVCPR)中，我們嘗試利用機率的選擇來決定航行的路徑。若是我們把廣度優先的方法視為門檻值為 0 的話，我們可說在前幾種方法裡，位於路徑上的每一篇文章其決定值均大於門檻值。換句話說，若是在路徑上有某篇文章的決定值低於了門檻值，那麼不管往後的文章是不是使用者想要的，都不會有機會被系統擷取到。為了想要克服這種情況，我們加入了機率的考量。讓一篇文章的決定值只是決定它被收到資料庫中的機率大小。

最後一種方式(FEVCPR)，則是為了克服第五種方法的缺點。在第五種方法中，有時系統會落到決定值曲線的谷底而無法再行爬起。為了解決這個問題，我們加入了強行前進的機制。我們強迫系統去盡量取到足夠的篇數後方才終止。

3-4 系統架構

本系統由以下部分所組成

(1) Configuration Manager：這個部分負責系統參數的設定。

(2) Preprocessing Manager：這個部分負責系統的初始化作業。

(3) Database Manager：此部分負責--切有關資料庫的運作。

(4) Lock Manager：這部分是負責系統中有關同步的問題。

(5) Content Manager：此部分負責有關文章內容的處理。

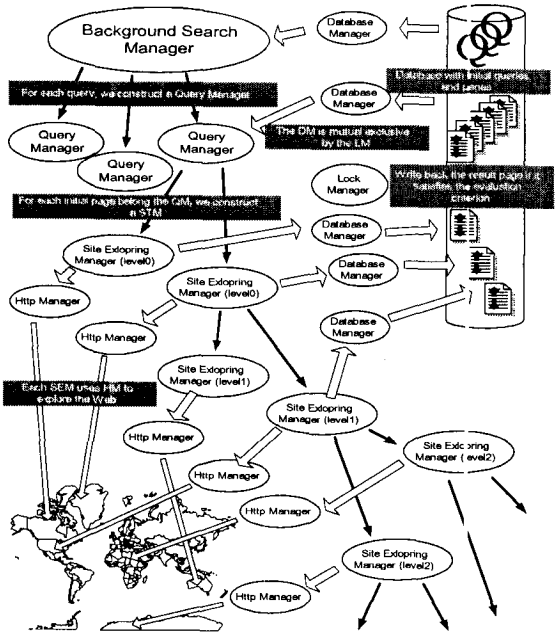
(6) Http Manager：此部分是有關網路運作的處理。

(7) Background Search Manager：這是系統中最上層的模組。對於每個詢問，會建立出一個 Query Manager。

(8) Query Manager：對每個詢問，會有一個 Query Manager 來負責。這個模組的責任是對每個超鏈結建立出一個 Site Exploring Manager。

(9) Site Exploring Manager：這個模組負責去探索給定的 URL 位址。

下圖為系統架構組織圖：



圖一. 系統架構圖

四. 實驗與結論

4.1 實驗環境架構

在 Arachnid 中，使用了大英百科全書的資料庫來當作模擬的環境。然而，這有可爭議之處。因為鏈結的強健性在真實的全球資訊網上和大英百科全書中不盡相同。因此，我們的實驗是真正在全球資訊網上進行。在基本模式中，我們請十位同學分別給予詢問，總共十三個。而每一組結果也讓原來的詢問者給予 precision 的估定值。而高級模式的實驗值，是由五個詢問所實驗而來。

4.2 評估標準

在評定這些演算法的效能時，我們採用了 precision 和 traversing depth 兩個值來做估定指標。Precision 是從傳統的資訊擷取估定方式所沿襲而來；而 traversing depth 則是新的估定方法。Precision 的重要性不用多說，而 traversing depth 是為了制衡有投機的現象產生。試想要是某種演算法均傳回使用者想要的資料，可是這些資料都存在於起使文件鄰近的鏈結中，那麼這些資料對使用者而言是有意義的嗎？很明顯的，使用者可以輕易的使用瀏覽器去獲得這些鄰近的資料，所以這些資料對使用者的重要性並不大。我們希望一種演算法不僅能傳回使用者想要的資料，而這些資料的分佈也能盡量遠離起使文件。如此一來，這些結果才有意義。

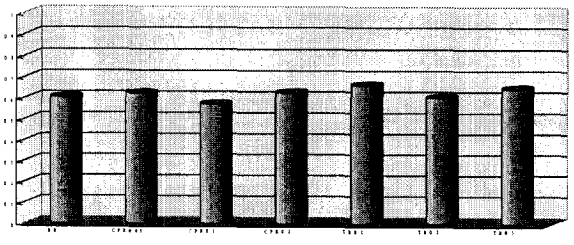
4.3 實驗結果

4.3.1 基本模式的結果比較

對於每組數據，我們只取篇數足夠的詢問結果加以平均，而不足者略去。

4.3.1.1 Precision

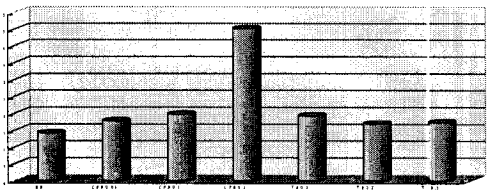
以下是前十篇、前二十篇、和前三十篇的 precision 平均狀況。縱軸座標是 precision 值，而橫軸座標是所使用的方法。方法後的數字是門檻值。



我們可以看到，事實上幾組數據差的都不多。不過一般來說，使用 Text-Based 的 precision 值會來的較好些。而使用廣度優先的 precision 值並不如原先想像的差，原因是因為我們所選的起使點夠好。

4.3.1.2 Traversing Depth

以下是在前十篇、前二十篇、和前三十篇中使用者感興趣的文章之 traversing depth 平均值。縱軸做標是往下搜尋的深度，而橫軸座標是所使用的方法。同樣的，方法後的數字是門檻值。

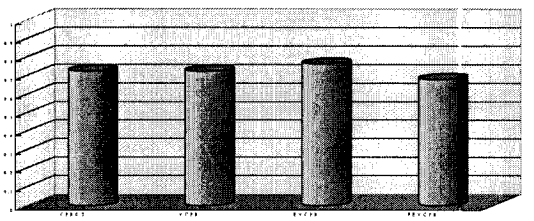


我們可以看到，使用 CPR 的 traversing depth 值是較好的。而門檻值越高，搜尋的深度也越深。雖然廣度優先的 precision 值並不差，但是他的 traversing depth 卻很淺。由此可知，廣度優先並不是個好方法。

4.3.2 高級模式的結果比較

4.3.2.1 Precision

以下是高級模式的 precision 統計圖，圖表的規則和基本模式相同。



我們可以看到，在這張圖中，每一種方法的 precision 值均差不了太多。

4.3.2.2 Traversing Depth

以下是高級模式的 traversing depth 統計圖，圖表的規則亦同。

4.4 結論

我們可以看到，加入調整式門檻值後，系統整體的 traversing depth 微微的下降。不過事實上，他能滿足的詢問數較多；換句話說，我們能夠保證在較多的詢問下，我們的系統可以找回足夠的篇數。其中，FEVCPR 的表現最優；而若是沒有加入強制前進的控制，則使用機率的選擇會讓系統過早於結束在決定值曲線的谷底。所以，EVCPR 的值並不突出。也正是為了要讓 EVCPR 能夠跳脫這種停止於谷底的現象，所以我們才會設計出 FEVCPR。

整體來說，使用 CPR 值來決定前進的方向是合理的。而調整式門檻值能讓較多的詢問傳回足夠的篇數。加入機率的選擇之後，我們又能再度的改進系統。

五.參考文獻

[1] R. Armstrong. Webwatcher: A learning apprentice for the world wide web. In AAAI spring Symposium. 1996

[2] Chia-Hui Chang, Ching-Chi Hsu. Customizable Multi-Engine Search Tool with Clustering. Computer Network and ISDN Systems. Vol. 29 No.8-13 pp1217-1224. 1997

[3] Souman Chakrabarti. Automatic resource compilation by analyzing hyperlink structure and associate text. Proceedings of the 7th International WWW conference.

[4] Heting Chu, Marilyn Resenthal. Search engines for the world wide web: A comparative study and evaluation methodology. http://www.asis.org/annual-96/ElectronicProceedings/chu.html, 1996

[5] dr. P.M.E. De Bra, drs. R.D.J. Post. Information Retrieval in the world-wide web: making client-based searching feasible. Eindhoven University of Technology Dept. of Computing Science.

[6] William B. Frakes, Ricardo Baeza-Yates. Information Retrieval Data structure and Algorithms. Prentice-Hall, 1992

[7] H. Vernon Leighton. Precision among World Wide Web search services(Search Engines): Altavista, Excite, Hotbot, Infoseek, Lycos http://www.winnona.msus.edu/is-f/library-f/webind2/webind2.htm

[8] Alexandros Moukas. Amalthaea: Information Discovery and Filtering using a Multiagent Evolving Ecosystem. MIT Media Lab. http://www-cse.ucsd.edu/users/fil/agents/agents.html, 1997

[9] M. Porter, An Algorithm for suffix striping, 1980

[10] Gerard Salton, Chris Buckley, Term weighting approaches in automatic text retrieval. Dept. of computer science, Cornell university. Ithaca, New York, 1987