行政院國家科學委員會專題研究計畫成果報告

企業網路資訊系統之工作流程管理及中文全文及區塊檢索之研究(III)

Intranet Workflow Management and Full-text and Passage-based Information Retrieval (III)

計畫編號 : NSC 89-2416-H-002-056
執行期限: 88 **年** 8 **月** 1 **日** ~ 89 **年** 7 **月** 31 **日**
主持人:**曹承礎教授　國立台灣大學資訊管理系**

中文摘要

在目前的資訊檢索系統之下，使用者希望系統傳回的是更精確的檢索範圍，如段落、表列等。為了解決分段檢索這個問題，本研究實作一個文件區塊檢索系統，利用 LSI (Latent Semantics Indexing) **進行文件的概念檢索。同時也對** LSI **這類檢索方式在分段檢索的情況下所表現出來的的性質做一探討。這些研究方向中包含了最佳的查詢條件長度、最佳的斷詞方式、分段方式對** LSI **的影響、**LSI **新增文件時的影響、以及** relevance feedback **是否能對本系統產生幫助。**

關鍵字: 資訊檢索、分段檢索、隱藏語意索引

Abstract

Using current information retrieval system, users expect more precise result like paragraphs, lists, etc.. In order to solve the problem of passage retrieval, a passage retrieval system is implemented by using LSI (Latent Semantics Indexing). At the same time the properties of LSI under passage retrieval is investigated. These properties includes optimal query length, optimal word segmentation, optimal document segmentation, impact when appending new documents, and the benefit of relevance feed-back.

Keywords: information retrieval, passage retrieval, Latent Semantics Indexing.

一、緒論

隨著資訊科技的進步與網路的發展，資訊的量隨著逐漸增加，而使用者所面對的資訊也就越來越多。為了解決這個問題，產生了資訊檢索 (information retrieval) **這個技術。本研究希望能將資訊檢索技術的範圍，由「全文」拓展到文件的某一部份。使得搜尋的動作可以找出文件中關係最大的一個「區塊」**(block)**，也就是提供解析度較高** (fine-grained) **的搜尋方式。這種文件區塊檢索的方式，可以成為「資訊萃取** (information extraction)**」的基礎。**

由以上的說明，可以瞭解區塊式的檢索方式是比較接近使用者對於全文檢索上的需求的，因此，本研究希望進行對區塊式全文檢索的研究。
研究的目的包括：

1. **如何利用索引典或類似技術，改進向量空間模型檢索方式。**
2. **決定如何分割文件成較小的文件「區塊」，以及對區塊產生索引以幫助檢索。**
3. **文件區塊及區塊索引的儲存管理。**
4. **研究** LSI **在分段文件擷取上的優缺點。若有缺點，應如何改進。**

二、資訊檢索 (information retrieval) 相關技術

資訊檢索系統 (information retrieval system) **可以定義為儲存、展示、組織、存取資訊的系統** [Gerard Salton ,1983]。**在這個定義之中，並沒有提及所要處理的資訊是以如何的形式存在的。因此，資訊檢索系統可以直接在原始資料中利用** pattern matching **的方式進行比對，或是對文件的內容進行分析與索引，以幫助資訊檢索的進行。這些分析與索引的工作主要可以分成以下幾個檢索模型：**

1. **向量空間模型** (vector space model, VSM)
2. **機率模型** (probability model)

這些模型主要的根據是基於文件中某些統計上的資訊對文件加以索引，但問題是：統計上的資訊並不能完全代表文件內的概念。使用者的期望是資訊檢索系統應該基於文件所代表的概念，而非統計上的資訊。

基於對使用者需求的認知，發展出一些不同的檢索方式。如 fuzzy-search [Jeng-Yih Wang]**，使用索引典以增進查詢條件本身所代表的意義** [1]**，或是本研究中所使用的** Latent Semantics Indexing**，都屬於這一類的檢索方式。以下將對這些檢索模型做簡短的說明。**

2.1 機率模型 (probabilistic model)

大部份的資訊檢索方法是屬於 perfect model **的。所謂的** perfect model**，這種模型的目標是將「排名在第一個不相關文件之前的所**

有文件」找出來。相反的，機率模型的目標考慮到了資訊檢索過程中產生的實際限制，而在這些限制之下，找出最佳的查詢結果 [Norbert Fuhr, 1989]。機率模型的基礎是機率排名原則 (probability ranking principle) [Robertson S. E., 1977]。這個原則認為，資訊檢索系統的任務，是依查詢條件和文件相關的機率來加以排序，依照這個順序找出最相關的文件。是否要取出某一文件的決策可以利用以下的數學式表示：

R: 表示查詢條件 f_l 及文件 d_m 所形成的序對 (f_l, d_m) 被使用者認定為相關的事件的機率

C_1: 檢索出不相關的文件所花的成本

C_2: 檢索出相關的文件所花的成本

若：

$$C_2 \cdot P(R|f_l,d_m) + C_1 \cdot (1 - P(R|f_l,d_m)) \le C_2 \cdot P(R|f_l,d_j) + C_1 \cdot (1 - P(R|f_l,d_j))$$

則 d_m 應該比 d_j 排名要高（總成本較低），也就是，則 d_m 應該比 d_j 先被檢索出。

依照這個原則，系統可以使用文件中某些「證據 (evidence)」來決定查詢條件和文件之間的機率來進行查詢。而「證據」的來源通常是由原始文件中詞彙統計上的分佈而來。而就因為如此，系統只知道詞彙的統計數字，而忽略了這些詞彙所代表的意義。

2.2 向量空間模型 (vector space model)

G. Salton 認為，在資訊檢索的過程中，必須對資訊本身進行分析以幫助檢索之進行。這個分析的過程稱為「建立索引 (indexing)」。基本上，建立索引的方式為針對系統中所有的文件所成之集合 D，找一組「屬性 (attributes)」 $A_1 \Lambda\ A_k$，使得在 D 中的某一文件 D_i，有一組屬性值 $(a_{i1}, a_{i2}, \Lambda\ , a_{ik})$ 具有足夠的資訊用以代表 D_i。該組屬性值 $(a_{i1}, a_{i2}, \Lambda\ , a_{ik})$ 就稱為 D_i 的索引向量。

既然，文件是由許多的詞彙 (terms) 所組成，因此可以利用文件中有意義的詞彙組合成文件向量。文件向量可以代表一個文件所代表的基本意義。在這個向量之中，每一個維度上的值代表文件在這個維度（也就是某一個詞彙）上強調的程度。這個值稱為「索引詞彙的顯著值(term significances)」，可以由文件的語文統計資料而得到，如索引詞彙出現的頻率等。因此，文件的索引其實是所有文件的向量結合起來成為矩陣的形式。而意義相近的文件，所用的詞彙可能相同，因此表示成向量空間的向量時，這些文件的向量會比較接近在一起。

查詢條件本身，也可以表示成向量的形式。利用向量內積，可以算出查詢條件向量與文件向量之間的相關程度。因此，文件檢索的過程成為找出距離查詢條件所代表的向量最近的向量所代表的文件並依據相關程度加以排序的。因為這種方式和機率模型一樣，可以對結果加以排名，找出最符合查詢條件的文件。

三、Latent Semantics Indexing

向量模型的觀念雖然廣為使用，但卻存在一些缺點。由於向量也是由詞彙所產生的，因此也有這類的問題，如對於一個相同的概念，可能有許多不同文字上的表示方式，如英文中的同義字 (synonymy)。在中文中，這樣的問題更為複雜。使用者需要的是能依文件所代表的「概念」而搜尋的系統。

Latent Semantics Indexing (LSI) [S. Deerwester, 1990] 是為了解決這樣的問題所產生的檢索方式。LSI 的基本假設是當文件所表達的概念相同的時候，由於文件的作者可以選擇的字彙有限，因此概念相同的文章通常會有相似的字彙出現。基本上，LSI 也是產生出文件的向量加以檢索，但最大的不同是 LSI 利用數學上的 singular value decomposition (SVD) 將文件向量的維度縮小，因而去除某些因為字彙有限而產生的詞彙與詞彙之間的相關性。

在 Gerard Salton 的書[Gerard, 1968]中已經提到，奇異值分析 (eigenvalue analysis) 可以用以將詞彙加以分類。在最佳的情況之下，可以找到一個最好的分類方式，使得在不同類之下的詞彙完全不相關[Gerard, 1968, p.135]。在這樣的情況之下，我們可以將一類詞彙視為一種「概念」的代表。基本上，我們可以將 LSI 視為一種將詞彙分類的一個方法。最好的情況下，LSI 可以將文件內不同的概念分類出來，因而有助於概念檢索的進行。

3.1 LSI 檢索方式的進行

基本上，LSI 是向量空間模型的一種。因此，在利用 LSI 進行檢索之前，向量模型內所必須的文件向量與代表查詢條件的向量必須先形成。取得這些向量之後，必須經過 SVD 的奇異值分解，取得化簡之後的向量。最後，將這些化簡之後的向量利用向量內積或餘弦值等方式加以比較相似程度，以取得文件與查詢條件的接近程度。LSI 的步驟可以利用以下的數學式表示：

Def 1: LSI 中必須建立一個詞彙-文件 (term-document) 矩陣 $A = \left[a_{ij} \right]$。其中 a_{ij} 表文件 j 中詞彙 i 出現的機率（或頻率）。a_{ij} 可以利用 local/global weighting 加以取得或調整。

Def 2: [singular value decomposition]矩陣 A 是

一個 $m \times n, m \geq n$ 的矩陣，A 的 SVD，記為 SVD(A)，為：

$$A = U\Sigma V^T$$

其中 $\Sigma = \mathrm{diag}(\pm_1, \Lambda, \pm_n)$ 為 A 的特徵值 (eigenvalue) 置於 Σ 的對角線所成的矩陣。$\pm_i > 0$ for $1 \leq i \leq r$, $\pm_j = 0$ for $j \geq r+1$。U，V 為對應於 $\pm_1 \Lambda \pm_n$ 的左右特徵向量 (eigenvector) 所成的矩陣。

若 $\pm_1 \geq \pm_2 \geq \Lambda \geq \pm_r > \pm_{r+1} = \Lambda = \pm_n = 0$，$A$ 的 closest rank-k matrix 可以定義為：

Def 3: [closest rank-k matrix] A 同上，取 $\pm_1 \geq \pm_2 \geq \Lambda \geq \pm_r > \pm_{r+1} = \Lambda = \pm_n = 0$ 所成的 SVD(A)中，取整數 $1 \leq k < r$，取 U 的前 k 行，Σ 的前 k 行乘前 k 列，及 V^T 的前 k 列，可得：$A_k = U_k \Sigma_k V_k^T$ 此處，A_k 是所有 rank-k 的矩陣中最接近 A 的。也就是：

$$\min_{rank(B)=k} \|A - B\|_2 = \|A - A_k\|_2 = \pm_{k+1}$$

而矩陣 V_k 包含的意義則是維度由 n 降為 k 的文件向量所成的矩陣，也就是可以表示文件真正「概念」的向量；也就是前段所說，去除了某些詞彙相關性之後而得的文件向量。

直覺上，由於 SVD 的步驟是利用矩陣的奇異值，將矩陣的列向量中線性相關的部份分離出來。於是，兩篇文章的向量中「線性相關」的部份可以分離出來，LSI 利用這種方式去除因詞彙造成的誤差。在經過此一步驟之後，文件向量可以比較接近文件的真正「概念 (concept)」，因而在利用向量內積比對時，不會因為同義字或是一字多義的詞彙而使某些應該出現的文件沒有查詢到，因而會有比較好的 recall。雖然並沒有理論上的支持，實作上 LSI 確實能產生出比較好的查詢結果 [Gerard,1968][David Hull ,1994]。

LSI 最大的問題在於進行資料庫更新時（加入文件或刪除文件），必須針對所有的文件加以重新計算由文件向量所組成的索引矩陣，經由 SVD 重新將這些文件的概念重新找出來。這一整個步驟是很費時的。目前，已經有研究是針對如何更新 LSI 資料庫的問題找出解答 [Gavin W. O'Brien, 1994]，但這些方式主要也牽涉到計算時間與索引矩陣的精確度之間的 trade-off，因此仍有一些問題需要解決。

3.2 LSI 索引矩陣的更新

由上一段可以發現，LSI 雖然有很好的檢索效果，但在更新文件資料庫時，卻也要比其他檢索方式花費更多計算時間。尤其在文件分段的檢索環境之下，加入新文件需要加入許多文章的段落。一般來說，有三種方式進行 LSI 索引矩陣的更新。

第一種方式是重新計算每個文件的向量，再經由 SVD 的過程得到新的索引矩陣。演算法經常都需要針對維度數以萬計的的矩陣進行 SVD，時間上以及空間上的複雜度是無法忍受的。

第二種方式稱為 Folding-In，Folding -In 可以利用比較少的時間和空間進行索引矩陣的更新，但卻可能造成索引矩陣失去精確度的效果，Folding -In 最大的問題在於它的假設。通常，加入新的文件，會影響這些文件和原有文件之間的關係。文件在經過 SVD 之後，會造成原有文件向量的移動。但 Folding-In 完全忽略這方面的影響，因此，對查詢的正確性來說，可能會有不良的影響。

在 [Gavin W. O'Brien, 1994] 中提到，為了解決 Folding-In 的缺點，作者提出了 SVD-Updating 的方法。這個方式最大的不同點在於它提供了一個方式更新原有文件的向量。相比於 Folding-In，SVD -Updating 事實上也產生比較好的結果。SVD-Updating 的步驟如下：

狀況一：新增 d 個文件

設 D 是 d 個代表新文件的向量所成的 $m \times d$ 矩陣。因為 A_k 是 A 的 rank-k approximation matrix，可以將 D 附加於 A_k 之後，重新計算 SVD 程序：

令 $B = (A_k | D)$，$\mathrm{SVD}(B) = U_B \Sigma_B V_B^T$。因 $A_k = U_k \Sigma_k V_k^T$，

則 $U_k^T B \begin{pmatrix} V_k & \\ & I_d \end{pmatrix} = (\Sigma_k | U_k^T D)$

令 $F = (\Sigma_k | U_k^T D)$，$\mathrm{SVD}(F) = U_F \Sigma_F V_F^T$，經過一番計算，可得：

$$U_B = U_k U_F$$

$$V_B = \begin{pmatrix} V_k & \\ & I_d \end{pmatrix} V_F$$

$$\Sigma_B = \Sigma_F = (U_F U_k)^T B \begin{pmatrix} V_k & \\ & I_d \end{pmatrix}$$

狀況二：新增 t 個詞彙

令 T 是由 t 個代表新詞彙所成的 $t \times n$ 矩陣，令 $B = \begin{pmatrix} A_k \\ T \end{pmatrix}$，利用狀況一相同的步驟，

可令 $H = \begin{pmatrix} \Sigma_k \\ TV_k \end{pmatrix}$，$\mathrm{SVD}(H) = U_H \Sigma_H V_H^T$，

因此：

$$U_B = \begin{pmatrix} U_k & \\ & I_t \end{pmatrix} U_H$$

$$V_B = V_k V_H$$

$$\Sigma_B = \Sigma_H = U_H^T \begin{pmatrix} U_k^T & \\ & I_t \end{pmatrix} B V_k V_H$$

利用這個方式，在計算 A 的 SVD 的時候，可以直接利用 A_k 計算，因為矩陣的維度降低了許多（可以視為常數），所需的時間及空間複雜度也就相對降低許多。這種方式對於分段文件的更新幫助很大，因為文件分段使得檢索的單位增加許多，計算上的複雜度也就隨之增加。如果可以利用 A_k 代替 A 進行計算，由於 k 可以視為常數，其複雜度自然低許多。

四、部份文件的擷取 (document passage retrieval)

在目前的全文檢索環境中，許多儲存在全文檢索系統中的文件是相當長的，常常在同一篇文章中包含了許多不同的主題。在這樣的情況下，搜尋一整篇文章變的沒有意義。因此，將大文章分開成不同的段落，再加以索引更能接近使用者真正的需求 [Gerard Salton, 1993]。這種方式的優點有兩點。第一點，文件的利用率可以提高。使用者不用去面對一整篇文章之中不相關的資訊，而能專注於真正相關的部份。第二，檢索本身的效率也能夠提高。原因是較短的文件存取容易，越長的文件，其中所包含的主題也就越多。

4.1 分段的方式

目前一般的分段方式可以歸類為三種：依文件的架構 (discourse)，依文件的語意流向 (semantics)，以及取固定字數段落 (window) [James P. Callan, ,1992]。

依文件架構分段 (discourse passages)

依文件原有的架構，如段、節等，一篇文章可以很自然的被分割成不同的部份。直覺上來說，這種方式是最有效率的方式。實際上，文件的概念在段與段之間是否能保持概念上的一致性，和作者寫文章的方式有很大的關係。如果作者將相同的概念分在許多的段落 ("shift" in content)，或是將許多不同的概念加以整理，集合在同一段中 (summarize)，對查詢的效率都有負面的影響。一般來說，這種方式在文件具有高度結構性時效果最好，或是經過編輯以確保概念完整。

依文章語意分段 (semantics passage)

第二個方式是依文章的語意或主題，將文章分為概念不同的段落，如 [Marti A. Hearst, 1993] 中的 Text Tiling。Text Tiling 的原理是利用文件中用詞的相關性 (lexical connectivity) 來將文件分為不同的部份，每一部份的相關性都很高。如在此處前後兩段出現數次 Text Tiling，但是這篇文章的其他部份卻沒有這麼密集。利用數量化的方式，文件的不同部份可以找出不同的 tiles，也就是表達概念相同的段落。

本研究提出的作法分為兩個步驟。首先，文件被分為每 3-5 個語句所組成的一個區塊 (block)。相鄰的區塊之間先互相比較其相似性（如用向量模型之中所用的向量內積方式），並記錄相鄰區塊之間的相似性大小。求出段落間的相似性以後，可以依照文章的流向和相似性的大小畫出圖形，圖形的高峰和谷底代表相似性的大小。其中兩個谷底之間代表語意相似的一段。因此，文件的分段方式可以決定出來。

以固定字數分段 (window passages)

依前面所述的兩種分段的方法中，都假設文件中存在一個「唯一的」、「符合查詢的」資訊架構。但對不同的查詢，分段的方式可以不同。因此，以上的兩種方式並不能符合所有的查詢條件。

這種問題的解決方式是在文件上開一個固定大小的查詢視窗 (fixed-length window)，將分段的大小固定。並移動視窗的位置進行查詢。在[James P. Callan, ,1992]中所提出的方式是先搜尋到文件中第一個符合查詢條件的字。從那個字開始，設視窗的大小為 n，每 n/2 字分出一個長度為 n 的段落。如第一個符合查詢條件的字位於第 62 個字處，而視窗的大小是 100 個字，則文件會被分為第 62-161 字，第 112-211 字…依此類推。視窗之間的相互重疊降低了相關文字被分至不同段落的機會。

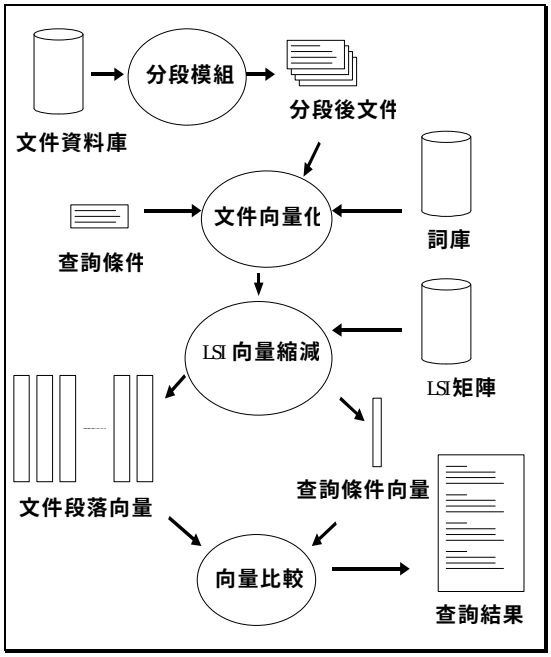
實驗結果顯示，利用這種方式查詢，精確度都有一定程度的進步。在某些實驗中，甚至精確度提昇了 20.7%。使用這個方法必須先決定視窗的大小 (window size)。在 [James P. Callan, 1992] 中，對四種不同的文件集合，實驗最佳的視窗大小的結果是，在不同種類的文件下，查詢結果最好時的視窗大小也不一樣。對一般的文件來說，介於 200-400 個英文字之間，大多都可以得到一個滿意的結果。

五、系統實作

本研究文件分段檢索系統乃利用 LSI 文件檢索的結果，利用其所產生的奇異值矩陣及經 SVD 過程所得到的縮減後的 (reduced) 詞彙向量，經由前面所介紹的 Folding-in 的方式，計算各個文件段落的向量，並將這些向量與查詢條件所得向量加以比較。

5.1 發展架構

本研究文件分段擷取系統的內部架構可以用以下的圖表示之：



在圖中，分段模組負責的工作為將文件依據某些方式將文件分成不同的段落。在最好的情況下，這些段落會具有獨立的意義，並且與前後段落有明顯意義上的差別。在本研究中所實作出來的分段方法有：依文件的長度分段、依文件原有的段落分段、以及利用 HTML 本身所具有的結構進行分段。在文件向量化模組中，主要的工作為將文件依照詞彙的出現頻率將文件轉換成向量的形式。

在取得文件的向量之後，下一步是利用 LSI 將原始的文件向量加以縮減。對 LSI 矩陣而言，分段之後的文件可說是新文件，因為這些文件原來並沒有被統計到矩陣當中。因此，必須利用 Folding-in 的方法，將新文件向量加入到矩陣當中。將向量經 LSI 矩陣縮減之後，最後本系統利用向量內積的方式計算向量相近的程度，並加以比較不同文件段落與查詢條件之間的相近程度。最後將結果輸出給使用者。

六、實驗結果

根據本研究實驗結果的數據發現

1. 利用文件本身結構進行分段比較能夠接近作者對文件意義的流向。因此，利用文件本身結構進行分段效果較好。
2. 在斷字方式上，本研究認為取長字串比較適合。取長字串可以產生更精確的意義，使得檢索的精確度提高。
3. 在查詢條件的長度上，在單一句子的部份檢索效果最好。這代表了 LSI 的檢索方式需要數個關鍵詞以幫助檢索。
4. 在 Folding-in 對查詢效能的影響上，本研究發現前述假設：利用文件向量矩陣將段落向量 Folding-in 並縮減這個動作並不會影響檢索的效果。並且，使用文件向量矩陣進行分段檢索的效果比直接利用段落向量矩陣查詢要好。
5. 在利用 Folding-in 進行新增方面，我們發現在本研究的環境下，可以忍受利用 Folding-in 新增 20%的文件。這代表了在新增 20%文件以前，可以不用重新計算 LSI 索引矩陣。而利用 LSI 相關技術進行索引的系統必須在新增文件比率大於 20%之後，重新計算索引矩陣。
6. 在本系統中, relevance feedback 提高了本系統的查詢效能。我們認為，為了降低使用者與系統對同一索引詞的認知差距，以及幫助使用者擴充查詢條件，relevance feedback 是必要的。

七、總結

本研究針對文件分段檢索的問題，利用 LSI 的技術，將可能出現的問題以及解決的方案，做一探討。本研究之研究結果歸納如下：

1. 利用本研究所提出之方法進行文件分段檢索，為一可行之方案。尤其在分段方式為文件本身段落、斷字方式為取長字串、以及查詢條件長度中等情況下，效果最好。
2. 在 Folding-in 新增研究上，我們首先證明本研究中所使用之假設正確。其次，我們發現利用文件向量矩陣進行分段檢索效果最好。而在利用 Folding-in 新增的研究上，在新增文件比例達到 20%以前，不會對查詢結果有重大影響。
3. 在 Relevance Feedback 方面，我們發現 Relevance Feedback 對查詢幫助極大。因此，在未來之分段檢索系統中，仍應該有此部份之功能。

綜合以上結論，本研究證明分段檢索系統為一可行之方案。至少在分段查詢上，確實

能查詢到相關的文件段落。因此，未來在文件的分段檢索上，可以利用 LSI，進行接近概念式的檢索。根據研究成果，本研究的未來研究方向包括：

1. 將索引典的功能加入到系統當中。目前本系統對於不是索引詞的詞彙是無法處理的。這使得查詢的效能受到限制。
2. 對不同的 relevance feedback 方式或參數做一評估或實驗，以找出最佳的 relevance feedback 結果。
3. 提供一部份語意分析的功能。系統能針對查詢的需要，自動判定應該取得的文件段落大小。
4. 提供一個好的方法來作索引詞的挑選，替索引矩陣找出最具有鑑別力的索引詞來增加查詢的效果。

本研究相信，在增加這些功能與研究之後之後，此 LSI 文件段落檢索系統將更趨完備，也更接近使用者對全文檢索的需求。

八、參考文獻

[1]美國資訊協會台北分會、農業科學資料服務中心, 索引典理論與實務研討會, 國立中央圖書館.

[David Hull, 1994]David Hull, Improving Text Retrieval for the Routing Problem using Latent Semantic Indexing, ACM SIGIR Conference, 1994.

[Gavin W. O'Brien ,1994]Gavin W. O'Brien, Information Management Tools for Updating an SVD-Encoded Indexing Scheme, Thesis for the Master Science Degree, University of Tennessee, Knoxville, Dec. 1994.

[Gerard, 1968]Gerard, Salton, Automatic Information Organization and Retrieval, McGraw-Hill, 1968.

[Gerard Salton, 1983]Gerard Salton, Michael J. McGill, Introduction to Modern Information Retrieval, McGraw-Hill, 1983.

[Gerard Salton, 1993] Gerard Salton, J. Allan, C. Buckley, Approaches to Passage Retrieval in Full Text Information Systems, Proceedings of ACM SIGIR, 1993.

[James P. Callan, 1992] James P. Callan, Passage-Level Evidence in Document Retrieval, ACM SIGIR, 1992.

[Jeng-Yih Wang]Jeng-Yih Wang, Dr. Shyi-Ming Chen, A Knowledge-based Method for Fuzzy Information Retrieval, Thesis of Institute of Computer and Information Science, National Chiao-Tung University.

[Marti A. Hearst, 1993] Marti A. Hearst, Christian Plaunt, Subtopic Structuring for Full-Length Document Access, Proceedings of ACM SIGIR, 1993.

[Norbert Fuhr, 1989] Norbert Fuhr, Optimum Polynomial Retrieval Functions Based on the Probability Ranking Principle, ACM Transactions on Information Systems, July 1989.

[Robertson S. E.,1997] Robertson S. E., The Probability Ranking Principle in Information Retrieval, J. DOC. 33(1977).

[S. Deerwester, 1990] S. Deerwester, S. Dumais, G.Furnas, T. Landauer, R. Harshman, Indexing by Latent Semantics Analysis, Journal of the American Society for Information Science, 41(6), 1990.