適合大量中文文件全文檢索的索引及資料壓縮技術

簡立峰[1]， 古鴻炎[2]

1.中央研究院資訊科學研究所
E-mail: lfchien@iis.sinica.edu.tw
2. 國立台灣工業技術學院電機工程學系
E-mail: root@guhy.ee.ntit.edu.tw

摘要

本文主要是提出適合大量中文文件全文檢索的索引及資料壓縮技術。在全文索引部份其特色是利用先前發展特殊設計的特徵檔(Signature File)索引配合自動分析技術，文件所需之索引空間因而可隨不同需求而調整，一般而言索引只佔文件大小的 15% ~ 30% 左右，如此不僅索引建置速度極快同時也可獲致極佳檢索效率。另外在文件內容方面，我們以LZ77方法為基礎再經適當改進得到的良好的壓縮率(47.3% ~ 53.0%)同時也具備文件密碼化甚至加快檢索速度的功能。目前這些技術皆已經成功應用在一高效率的中文文件檢索系統－"尋易"(CSmart)。

壹、緒論

隨著網路的佈建及電子文件的成長，文件全文檢索(Full-text Searching)的需求日殷[1]。在英文文件搜尋方面已有許多檢索系統如 WAIS及BRS/Search成功的發展出來。另外考慮中英二者語言的差距，有關中文文件全文檢索技術的研發也日受重視並有不錯的進展[2,3,4,5]。然而過去的研究較少討論大量中文文件全文檢索的索引及資料壓縮技術，由於索引及資料壓縮對於儲存空間節省、光碟製作、文件密碼化、資料傳輸等關係密切，因此本文即以適合大量中文文件全文檢索的索引及資料壓縮技術為題，提出一套已經成功應用在"尋易"(CSmart)中文文件檢索系統的高效率方法，並說明資料壓縮技術對全文檢索的重要性。

以一般全文檢索系統而言，文件及文件索引是系統中最佔儲存空間的兩類資訊。通常索引空間需求大概是文件量的50%~300%左右，由於電腦儲存媒體容量大增且價格日益低廉，研究者對空間節省較不以為意，因此全文檢索中有關文件壓縮與索引壓縮的研究較少。但是就發展實用系統的角度來看，空間的節省仍然有其必要性。因為空間的節省也就代表經費的降低。以製作一個含300MB的文件及700MB索引的全文光碟資料庫為例，所需即是一1GB的光碟，但如果文件及索引可以適當壓縮，則也許只須使用500MB光碟即可，對資料量大且發行數目較多的全文光碟而言空間的節省仍有其價值。此外透過資料壓縮技術將所需空間減低同時也具備資料密碼化的效果，有助於降低非法拷貝的機會，另外壓縮後的資料傳輸較快也有助於加快檢索速度。

為了降低索引的空間需求並維持極佳檢索效率，我們曾發展出一項特殊設計的特徵檔(Signature File)索引方法〔5~8〕，其所需之索引空間可隨不同需求而調整，一般而言索引只佔文件大小的 15% ~ 30% 左右。此外最近我們也發展出自動測試技術，利用這項技術，系統可以自動產生測試查詢以檢測特徵檔的效率，自動決定索引大小及索引配置。由於這項技術使我們瞭解到系統除了提供索引壓縮功能，更重要是必須同時能分析壓縮後的檢索效率，而傳統檢索系統多缺乏類似功能。

另一方面，為了降低文件本身的空間需求並且將文件密碼化，我們以LZ77方法〔9〕為基礎再經適當改進得到的良好的壓縮率(47.3% ~ 53.0%)同時也具備極佳的文件密碼化甚至加快檢索速度的功能。而且根據我們實驗發現文件壓縮後可以減少磁碟存取次數可以進一步加快檢索速度。以上述技術為基礎我們發展出尋易(CSmart)中文文件檢索系統。這個系統的可提供多元化檢索功能且查詢速度極快，一般查詢皆可即時(Real-time)反應。此外，透過與 WWW (World-Wide Web)及MOZAIC的結合，可非常容易製作各種多媒體文件資料庫並提供網路查詢。利用此系統我們並開始製作新聞資料庫、電子詞典等在臺灣學術網路使用。

尋易系統的設計基本上是成功的整合我們發展的許多技術，部份技術我們已發表一些文章，包括整個尋易系統設計〔5〕，近似自然語言技術〔6〕，特徵檔技術〔7〕，以及語音檢索〔8〕，本文主要是著重資料壓縮技術，這是藉助本文第二位作者過去技術的基礎〔11〕，也是尋易系統最近的技術成果。以下各節，第貳節是有關尋易系統架構簡介，第參節是索引技術之介紹，第肆節說明文件壓縮技術，第伍節是結論。

貳、尋易系統架構

尋易檢索系統是針對中文文件特性所設計的的檢索程式，其角色與Internet上著名的WAIS系統接近，使用者可以透過WWW server直接存取儲存在尋易系統內的中文文件。尋易系統與WWW的結合方式可以如圖1所示。另外尋易檢索系統的整個架構主要是以特徵檔索引為基礎，其主要結構如圖 2 所示。

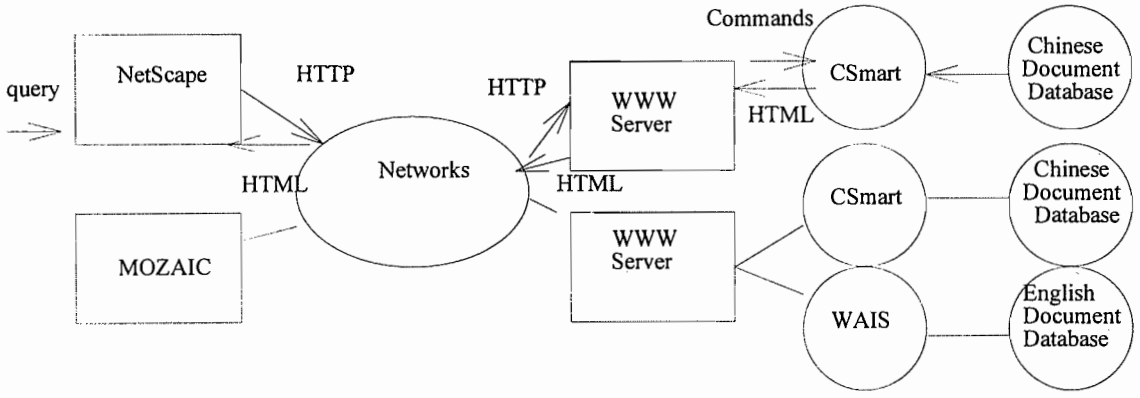

圖 1. 尋易檢索系統與WWW的整合應用方式

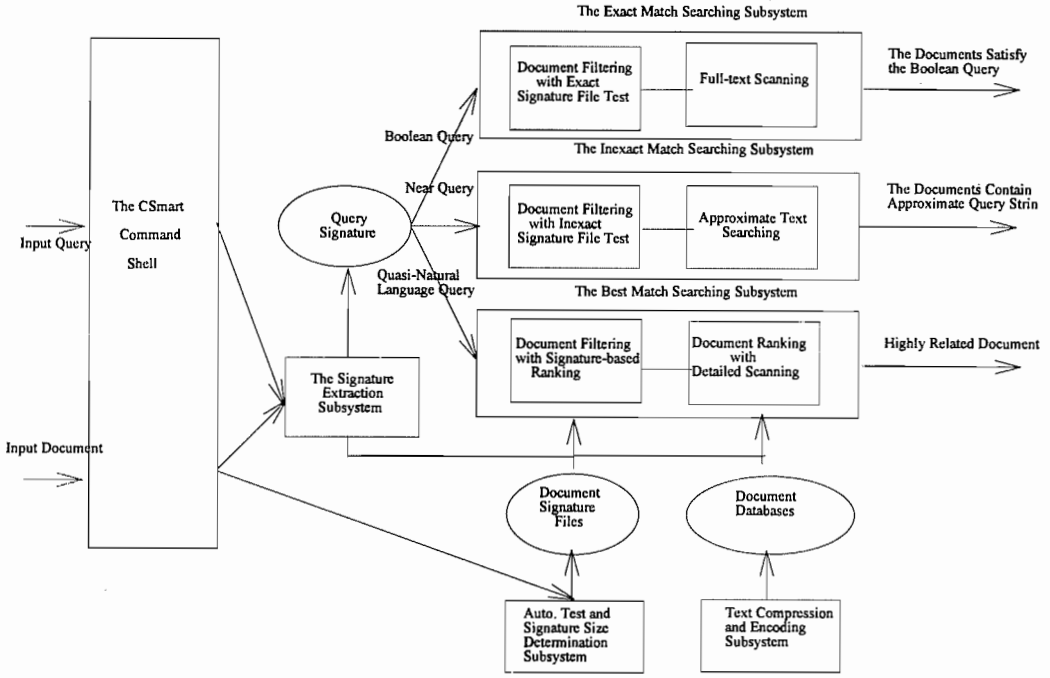

圖 2 尋易系統的系統架構

其中包括輸出入介面(Command Shell)、特徵檔產生(Signature Extraction)、精確比對搜尋(Exact Match Searching)、近似比對搜尋(Inexact Match Searching)、最佳比對搜尋(Best Match Searching)、自動測試及索引大小決定(Auto Test and Signature Size Determination) 以及文件壓縮解碼 (Text Compression and Encoding)等七個主要程序。

所有文件儲存在尋易系統內都須經過壓縮處理，當使用者欲瀏覽文件內文時才會解碼顯示。整個壓縮及解碼處理由文件壓縮解碼程序負責。任何文件欲加入資料庫必須先經特徵產生程序產生該文件的特徵並加入特徵檔(Signature File)。如果系統管理者希望所產生的特徵索引不論檢索速度與儲存空間大小都很有效率，也可執行自動測試及索引大小決定程序。尋易系統會參考現有資料庫內容、儲存空間需求、溢檢率(False Drops)的情形重新決定索引大小及特徵配置方式。

另外，使用者在查詢時可以選擇以一般布林查詢、近似字串查詢、近似自然語言查詢的方式表達查詢主題。不論是那一種查詢都必需先產生查詢特徵(Query Signature)。若是布林或近似查詢即交由精確或近似比對搜尋程式處理，這包括第一階段先將查詢特徵和所有文件特徵比對，未滿足該查詢特徵的文件將被濾掉。未被濾掉的文件內容在第二階段將會被讀出及與查詢仔細比對，眞正滿足該查詢文件才會檢索出。若使用者是以近似自然語言方式查詢文件，則將交由最佳比對搜尋程序處理。在第一階段該程序會將查詢特徵與所有文件特徵一一比對估算出其查詢與文件之初步相似度(Relevance Value)，相似度足夠高的文件才會在第二階段繼續處理。第二階段基本上是將這些文件內文讀出，仔細比對查詢句子中一些關鍵語出現在文件中的頻率及位置，以進一步判斷其相似度，最相似的一些文件才會檢索出。

根據前述說明，特徵檔的產生基本上是整個系統的核心且同時提供布林查詢、近似查詢及近似自然語言查詢之用。另外精確比對搜尋、近似比對搜尋和最佳比對搜尋皆是一種兩段式搜尋概念，其第一階段皆以快速的方式濾去絕大多數不相關的文件，第二階段才仔細比對餘下的文件，這種搜尋技術非常適合處理相當大量的文件。根據我們實驗發現由於我們所發展之特徵檔具極佳之過濾(Filtering)效果，在需快速反應時，往往只要施行第一階段搜尋(即比對特徵檔)即可得到很好的結果。以下第參節將進一步說明特徵檔的產生。

參、特徵檔的產生與自動調整

英文特徵檔技術

從前節介紹可知尋易系統以特徵檔為主要索引。特徵檔的主要是針對布林查詢、近似字串查詢及近似自然語言查詢而設計。這種特徵檔一方面必須將全文訊息儘量完整且清楚的記錄起來，另一方面又必須節省空間且方便存取(Access)。特徵檔的效率是以其在處理布林查詢及近似自然語言查詢時能過濾掉不相關文件的程度來衡量。由於中、西文文件特性大不相同，西文通常是以詞為單位的方式來產生文件特徵[1]。基本上西文系統會以 hash 方式令每一個詞有一固定長的 0/1 字串(其中有若干處設定為 1)為其詞特徵(Word Signature)，而文件特徵即是文件所有詞簽名之重疊處理(Superimpose)。如圖 3 所示。

signatures of words:

document	00100100
information	00100001
retrieval	10000100

signature of a document just contains the above three words:

10100101

圖 3. 西文系統以詞特徵及重疊處理(Superimpose)產生文件特徵舉例

因為英文 word 有豐富的詞尾變化，產生特徵時這些 word 都必須先進行原形化處理。然而即始經過原形化後詞的數目仍很多，考慮索引空間大小及每一個詞都必須有一個唯一的特徵值，勢必造成很多詞的特徵很接近，如只有一個 bit 值不同。因此由重疊後的文件特徵並不易還原出原先眞正組成的詞，所以這類特徵比較適合充當精確比對時判斷有無包含查詢字串的濾波器(Filter)，而不適合充當特徵向量(Feature Vector)，也即不適合應用在相關性估計(Relevance Estimation)。

以統計特性為基礎之特徵檔產生方式

為了發展適合中文特性的特徵檔並且同時兼具精確比對時的濾波器及最佳比對之特徵向量雙重效果，我們提出以統計特性為基礎之特徵檔產生方式[6,7]。我們以單字和雙字做為特徵檔產生之基礎，因為中文有分詞上的困難，我們捨去以詞為單位的處理方式，取而代之是以單字和相連雙字為單位。在台灣地區中文常用字在5,000 字左右，也就是對每一文件只要 5,000 bits 令每一bit分別代表一中文字之存在與否，如此一來即可清楚記錄這些字是否出現在文件內。這種方式的問題在於文件內許多字序及相鄰訊息(Positional Information)，如"日本"或"本日"無法藉此表達，因此如果能夠把雙字的特性同時考慮進來則可大為加強位置訊息，因此我們首先將特徵分成兩段，第一段代表常用單字出現的訊息，第二段代表雙字及罕用字出現的訊息，這每一段的長度是可以調整的。雖然常用字有 5,000 字，但許多字很少在一起聯用，因此我們提供一套統計分群方法。利用一些訓練語料(Corpus)，將統計特性差距大的一些字合為一群，在第一段特徵裡令其共用一個 bit 以記錄其出現之訊息。這個分群的群數自然和第一段特徵長度有關，如果第一段長度有 1,000 bits，則將 5,000 字分群 1,000 群。再者，因為中文罕用字及雙字數目太多，我們並不採用統計分群技術，而是利用 hash 方式令某些罕用字或雙字共用一個 bit，如此一來，文件特徵之產生即如下圖 4所示。

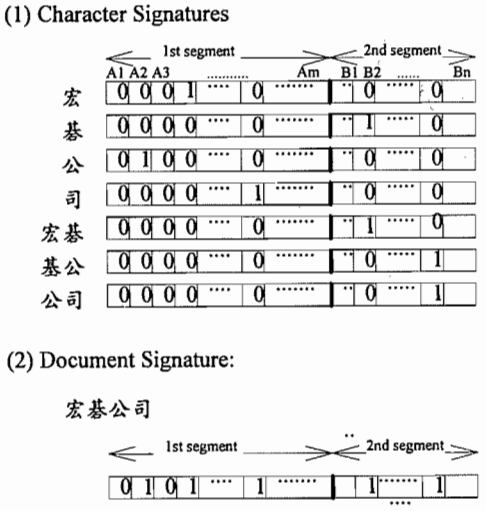

圖 4 尋易系統文件特徵(Signature)產生舉例

若文件內容是如(2)所示之"宏碁公司"之字串，且"宏"、"公"、"司"等三個常用字及"碁"這罕用字及"宏碁"、"碁公"、"公司"這三個雙字之特徵如(1)所示，則其最後文件特徵將如(2)之結果。根據我們的研究這樣的特徵結構比英文特徵檔基本上記錄很豐富的訊息不佔太多空間又易存取。此外特徵長度及每一個字或雙字的特徵可以隨文件特性加以調整。其進一步的說明可參閱[6,7]。

實驗結果

尋易系統目前已初步完成且基本上做了相當多的測試並陸續發表不少成果〔5~8〕，包括在精確比對方面與傳統逐字反檔(Character Inversion File)方法比較，其不論索引空間、索引時間、及查詢速度皆遠優於該方法。另外在溢檢率降低方面也有不少改進及分析。由於這些結果可以在前述文獻上參閱得到，本節只略列一些實驗結果供做參考，包括在新聞資料庫應用環境之索引時間、索引空間、查詢速度等。

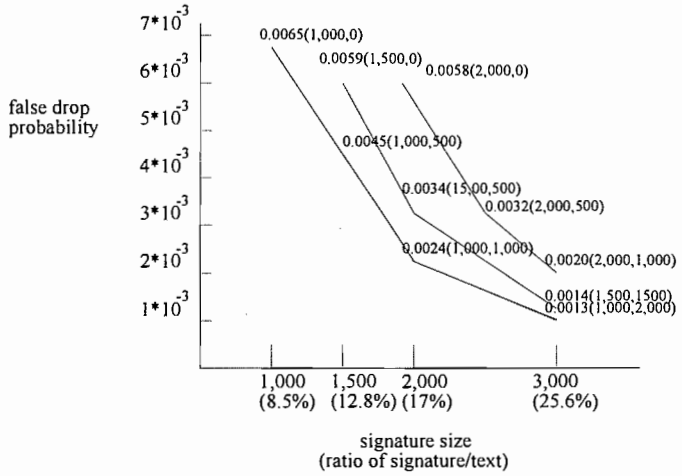

圖 5. 溢檢率與索引大小的關係，圖中括弧內第一個數字代表第一段特徵長度，而第二個數字代表第二段特徵長度。

首先在溢檢率與索引大小關係方面，由圖5所示可發現利用前述特徵檔方法可以得到不錯的溢檢率，特別是當第二段特徵長度較長時。舉例，當第一、二段特徵長皆為1,000 bits 時索引大小只佔文件的17%，而溢檢率只有0.0024，也即對任一查詢平均每1000件無關文件只有2.4件會被誤認，但這些誤認文件可利用進一步比對文件內文而濾掉。另外根據表1，我們可以發現當索引文件比維持25%時皆可獲致極佳的結果，不論工作站或PC，索引時間，平均字串搜尋時間。表1索引時間並不包括以下第肆節將介紹的壓縮時間。

Host	Main Memory	Document Size	No. of Documents	Index Space	Indexing Time	Average Search Time
PC486	8M	1GB	1.25M	250MB	55min	about 10sec
PC486	8M	100MB	0.12M	25MB	4min	1sec below
SPARC10	32M	1GB	1.25M	250MB	25min	3~5sec
SPARC10	32M	100MB	0.12M	25MB	2min	1sec below

表 1 實驗結果舉例

特徵檔自動分析方法

綜合上述實驗，我們發現利用前述特殊設計的特徵檔索引技術已經將索引空間降到最低，為了存取效率方便，如非必要索引不應再進一步壓縮。取而代之，因為所發展的特徵檔索引為可調式，我們決定發展自動分析技術。允許系統管理者自行決定所需索引文件比，尋易系統利用先前類似圖5的數據決定若干可能的索引大小，之後參考使用者已鍵入的查詢及資料庫內容，重新對每一種索引組合配置特徵，另外隨機產生一些測試查詢。進而，系統會自我分析，決定最適合系統管理者期望的索引文件比以及最低的溢檢率。根據我們初步實驗發現，由此方式產生的溢檢率可以降得更低。而且對系統管理者而言，管理上非常方便。利用此方式，我們創造了更有智慧的可調式索引，也使得索引壓縮可以配合檢索成效一併考慮，這是傳統索引壓縮不易做到的。

肆.中文文件壓縮方法

對大資料量(如 100mega bytes 以上)的中文文句(Chinese text)作全文搜尋(full-text searching)的應用裡，資料只能儲存在速度較慢的次要儲存設備(如硬碟、光碟等)裡，為了加快從硬(光)碟到主記憶體的傳輸速度、及減低記憶空間的需求，我們可先把原始資料作壓縮處理，然後才將壓縮後的資料存入硬(光)碟裡。不過，在全文搜尋及類似的資訊檢索應用裡，只考慮降低壓縮率(壓縮後資料長度除以原始資料長度)是不夠的，因為從硬(光)碟傳回的資料需先經過解壓縮處理，還原成原始資料後，才能去作搜尋，如果解壓縮的速度太慢，而資料量又很大，就會使整個全文搜尋系統的反應變慢到無法忍受。

因此，在這裡我們提出一種適用於大資料量中文全文搜尋應用的資料壓縮方法，事實上它是以 LZ77 方法[9,10]為基礎再經過修正而得到的，我們提出的主要修正是：

(1)擴大字符集(alphabet)

在 256 個 ASCII 字元之外，再將五大(Big-5)中文碼裡的 5401 個常用中文字，及我們定義的 32 個特殊字元(表示兩字串間成功比對的長度)含蓋進來，根據我們過去的經驗[11]，把表示一個中文字的兩個 bytes 當作一個字符集字元來處理，要比當成兩個分開的 ASCII 字元來處理，會得到更低 10% 以上的壓縮率。

(2)調適分群(adaptive grouping)

在把字符集裡的字元分成若干群後，隨著壓縮處理之進行，讓常出現的字元轉移到一個以較短的碼來編碼的字元群去，而將不常出現的字元排擠到一個以較長的碼來編碼的字元群去。這裡提出的調適分群想法，和以前被提出的固定式分群想法[12,13]是很不一樣的。

所以要將字符集裡的字元加以分群的原因是，當我們對前述之大字符集編一個變長碼(variable-length code)，再配合使用移前(move-to-front)[10,14]調適策略時，會發現字符集太大使得移前調適的處理速度變得更慢，這樣就無法達成快速解壓縮的目標，而經由分群再配合我們提出的一種調適方法，就可得到很高的解壓縮速度，詳細情形以下將進一步說明。

資料壓縮前段處理:字串比對與字串編碼

由於我們提出的修正作法是以 LZ77 壓縮方法[9]為基礎的，所以先回顧原始的 LZ77 壓縮方法，它的處理程序可配合圖6來作說明，圖6裡 P 指標所指示的前看區(lookahead buffer)用來儲存 32 個等待作壓縮的輸入字元，而P指標之前的已編碼區則是用來儲存最近 4684 個已做過壓縮處理的輸入字元，原始的 LZ77 壓縮方法就是，到已編碼區中去尋找一個可以和前看區字元串比對成功的最長字元串，然後將此字元串的位置(和指標 P 間的差距) d，長度 e，及前看區中未能繼續比對成功的字元 u 等三項訊息，以一個三重組(triple) <d, e, u> 記載下來(例如當圖6 裡Q指標所指的是比對成功的最長字元串時，就會得到 <3, 2, c>)，接著，把指標 P 往右移 e 個位置(相當於滑動窗裡的字元往左移動)，再反覆做前述的動作，如此就可獲得一序列的三重組 $<d_1, e_1, u_1>$, $<d_2, e_2, u_2>$, ...。至於對應的解壓縮器，它不必去做字串比對的動作，只需依據接收到的三重組來把對應的原始資料輕鬆地恢復回來，所以解壓縮的速度會比壓縮的快很多。

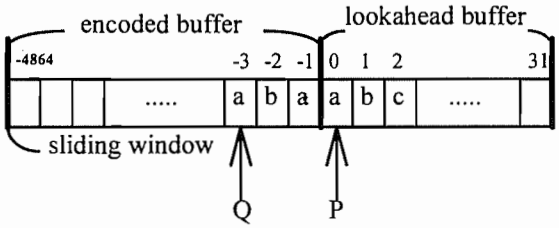

圖6. LZ77壓縮處理之滑動窗

前述的三重組序列，其實也可看成是一個二重組 $<d_i, e_i>$ 和單重組 $<u_i>$ 交替出現的序列，因此，後來 Bell 等人提出一種變形作法稱為 LZSS[10]，將二重組與單重組需交替出現的限制加以取消，但這需使用一個額外的 bit 來告知解壓縮器，接著出現的是一個二重組還是單重組，由它們的實驗顯示，取消限制的確可獲得更低的壓縮率。

研究了 LZ77 及 LZSS 的方法後，我們遂提出如下的修正作法，將 LZSS 裡需額外用一個 bit 去分辨二重組與單重組的作法加以改變，而將二重組的 d 組件與 e 組件的次序對調，再將二重組的 e 組件和單重組的 u 組件看成是來自於同一個字符集的不同字元(如 ASCII 字符集包含了控制字元與英文字母)，如此，就可將原先的特定用途 bit 融入到表示字元的 bits 去，而解壓縮器也可依據目前解得的字元去判斷，接下來要解出字元還是二重組裡的 d 組件。除了將二重組之 e 組件看成是字符集裡的字元，我們也把 Big-5 碼裡的常用中文字放到字符集去，而不常用的中文字則仍然看作是 2 個連接的 ASCII 字元，如此，我們的字符集裡便有6827個字元，使用這樣一個字符集，輸入的資料 bytes 很明顯地就不是直接放入圖6 裡的前看區，而需先經過構字分析(lexical analysis)，把二個 bytes 表示的常用中文字和一個 byte 表示的 ASCII 字元分別切割出來，並轉成字符集裡該字元的代號(2byte整數)，然後才將字元代號放入圖6 的前看區裡。

關於圖6 裡，要從已編碼區中找出一段能和前看區字串比對成功的最長字元串的問題，我們使用如下的解決辦法，當一個字元剛從前看區進入已編碼區時，就將此字元及其後的 31 個字元看成一個 key(可以跨越到前看區去)，然後依據此 key 的前二字元去作赫序(hashing)[15]，以決定它要插入(insert) 到 512棵二元搜尋樹中的那一顆，而當一個字元要從已編碼區的左端離去時，就將此字元及其後 31 個字元所構成的 key，從它所在的搜尋樹中加以刪除(delete)，這樣先將已編碼區中的字元依據赫序值建造成許多棵搜尋樹，而前看區裡 32 個字元構成的 key 也先經過赫序才到對應的搜尋樹去找具有最長共同子字串的 key，就可使字串搜尋的速度提高很多。前面提到的赫序，會使用一個key的最前面兩個字元去計算，這意謂搜尋後若發現最長共同子字串長度為 0，並不表示前看區的第一個字元就絕對未出現於已編碼區中，因此，最長共同子字串長度為 0 或為 1 時，我們都當作 0 值來處理，也就是要送出一個單重組。

資料壓縮後段處理:字元編碼與調適分群

每次當前段處理輸出一個二重組 <e, d> 或單重組 <u> 後，後段處理就接著拿去做字元編碼的動作，然後決定要不要將該字元換群，如果剛剛是對二重組的 e 組件編碼，則緊接著還需對 d 組件編碼。前述的字元編碼其實是分成兩部份去做的，即群編碼部份和元素編碼部份，其中群編碼是要產生一個碼用以表示 e 或 u 所在之字元群，我們使用變長碼(variable-length code)來表示不同的群，而元素編碼是要產生一個碼用以表示 e 或 u 目前被排在所屬字元群的第幾個元素上，我們使用定長碼(fixed-length code)來區分同一群內的各個元素。

在本研究裡，我們把字符集的 6827 個字元分成如下之 8 群：

A群:元素個數固定為 1,放置最常出現的字元，群碼為 $(010)_2$，不需元素碼；
B群:元素個數固定為 2,聚集次常出現的字元,群碼為 $(100)_2$,元素碼使用1個 bit；
C群:元素個數固定為 8,聚集次次常出現字元,群碼為 $(1110)_2$,元素碼使用3個 bits；
D群:元素個數固定為 32,群碼為 $(101)_2$,元素碼使用5個 bits；
E群:元素個數固定為 128,群碼為 $(011)_2$,元素碼使用7個 bits；
F群:元素個數固定為 512,群碼為 $(00)_2$,元素碼使用9個 bits；
G群:元素個數固定為 2048,群碼為 $(110)_2$,元素碼使用11個 bits；
H群:元素個數固定為 4096,聚集最罕出現字元,群碼為 $(1111)_2$,元素碼用12個 bits；

各群的元素個數是在群數固定為 8 時，經由實驗(此時群碼為固定長度)對不同的組合去量測、比較壓縮率後才決定下來的，然後再依實驗裡各群的出現頻率以霍夫曼(Huffman)編碼方法去決定群碼。

壓縮器(或解壓縮器)一開始動作時，它並不知道那一些字元較常出現或較不常出現，因此它是隨意地將字符集字元分成 8 群，如果能夠隨壓縮處理之進行，把較常出現的字元改分配到較前面(即靠近A群這邊)的字元群去，則可使整體的壓縮率降低。為了達成前述目標，我們嘗試了幾種調適分群的作法，最後發現一種效果最好的，詳細說來是，為字符集的每一個字元 s 設一個出現頻率計數變數 N(s)，並為每一個字元群 X 設一個先進先出指標 P(X)，以將一個字元群當作一個貯列

(queue)來控制群內元素之進出，然後，當一個剛被編完碼的字元 s 目前是字元群 X 的一個元素時，就進行如下的調適動作：

```
delete(X, s);      /* 將字元 s 從字元群 X 中刪除 */
N(s) <= N(s) + 1; /* 字元 s 之出現頻率值加 1 */
Y <= X - 1;        /* 將變數 Y 設定為變數 X 所代表之字元群的前一群 */
while( X ≠ 'A' and N(s) > N( front(Y) ) ) {
    w <= dequeue(Y);      /* 從貯列 Y 的最前端取出一個字元 */
    enqueue(X, w);    /* 將字元 w 插入到貯列 X 的最後端 */
    X <= Y;
    Y <= X - 1;    /* 將變數 Y 設定為變數 X 所代表之字元群的前一群 */
}
enqueue(X, s);    /* 將字元 s 插入到貯列 X 的最後端 */
```

在前述的調適程序(procedure)裡，函數 delete(X, s) 的用途是，把字元 s 從字元群 X 之貯列中強迫去除，不管 s 是否在貯列的最前端；函數 front(Y) 用以查詢貯列 Y 最前面的字元；至於函數 dequeue(Y) 與 enqueue(X, w) 的功用，就如程序裡的註解所說明的。其實，前述程序的想法是取自電腦作業系統裡的一種 CPU 排程(scheduling)的方法(即 multi-level round robin)，再配合我們的問題而修正得到的。

至此還未說明的是二重組裡 d 組件的編碼方法，d 的數值範圍是 0 到 4863 (表示差距 1 到 4864=76*64)，要對 d 編碼，我們先求 d 除以 64 之商數 dq 與餘數 dr，然後以變長碼來對 dq 編碼，而以定長碼來對 dr 編碼。對於 dq 之編碼，我們也是先作分群，然後以群碼結合元素編碼來作為 dq 之編碼，在 dq 的數值範圍(0至75)裡，我們把0至3放在第一群(群碼為00)，4到11放在第二群(群碼為110)，12到27放在第三群(群碼為01)，28到43放在第四群(群碼為111)，44到75放在第五群(群碼為10)，群碼是依據實驗得到之各群出現頻率的數值來訂定的。此外，由實驗結果來看，使用固定分群或調適分群去對 dq 編碼，所得到的壓縮率差異只在 0.05% 的數量級，因此我們就選取固定分群之作法。

實驗結果

我們已將所提出的方法寫成可實際使用之軟體(以C語言寫作)，在經過典型中文文句資料檔的測試後，發現所提方法的壓縮率(47.3% ~ 53.0%)一般說來比著名壓縮軟體 ARJ 或 PKZIP 的壓縮率(52.7% ~ 60.5%)還低 5.3% 以上，至於時間花費，以 486-33 個人電腦來說，我們的軟體的平均壓縮與解壓縮速度分別是 28.2K Bytes/sec 與 112.6K Bytes/sec，解壓縮的速度很明顯地比壓縮的快很多，雖然這樣的速度仍比 ARJ 的 178.6K Bytes/sec 慢，但是， ARJ 是不是以組合語言寫的？有無經過最佳化佈置？我們並不知道。

目前這項壓縮技術已應用在尋易系統中。對於尋易系統也產生若干影響，這也是我們將文件壓縮技術應用在中文全文檢索後所獲致的經驗。首先文件壓縮增加了索引建置時間，這幾乎是文件壓縮技術唯一的缺點，幸好多數應用並不要求太快的索引時間。其二，因為解壓縮速度快，加上原先溢檢率已低，因此壓縮技術的應用

並不增加搜尋時間。其實根據我們實驗，如果所使用的硬碟存取不快，利用壓縮技術反而加快硬碟I/O，甚至可加快檢索速度。其三，文件儲存空間可節省一半，就實際應用而言是非常重要的結果。其四，壓縮後造成自然的密碼化，也是所有文件檢索實用系統所關切的。雖然文件既然可瀏覽當然可拷貝，但許多應用裡，我們發現提供資料的單位皆希望辛苦建立的資料不應該太容易就被拷貝。最後，我們希望說明就一檢索系統而言自行發展文件壓縮的技術是絕對有其必要性。因為，任何公共使用的壓縮程式，都會失去密碼化的價值。

伍. 結論

本文主要是提出適合大量中文文件全文檢索的索引及資料壓縮技術。在全文索引部份是利用先前發展特殊設計的特徵檔(Signature File)索引以及本文所提自動分析方法，實驗證實索引已壓縮至極低，並不須進一步壓縮。反倒是，如何提供管理者同時考慮檢索速度與壓縮比才是最重要，這點本文所提方法已具相當不錯的成果。另外在文件內容壓縮方面，我們以LZ77方法為基礎再經適當改進得到的良好的壓縮率(47.3% to 53.0%)，同時也發現壓縮技術有助於文件密碼化甚至加快檢索速度的功能。本文證實了大量中文全文檢索對壓縮技術的需求。目前這些技術皆已經成功應用在一高效率的中文文件檢索系統－"尋易"(CSmart)。

參考資料:

[1]. Salton, G., "Introduction to Modern Information Retrieval", NY, McGraw-Hill, 1983.

[2]. Tseng, S. S., Yang, C. C. and Hsieh, Ching-chun, "On the design of Chinese Textual Database", Computer Processing of Chinese and Oriental Languages, 4, 1989, 240-271.

[3]. Liang, Tyne, Lee, S. Y. and Yang, W. P. "On the Design of Effective Chinese Textual Retrieval Based On the Signature Method", Computer Processing of Chinese and Oriental Languages, Vol. 8, No. 1, June 1994, pp. 87-96.

[4]. Wu, Zimin and Tseng, Gwyneth, "Chinese Text Segmentation for Text Retrieval; Achievements and Problems". JASIS, 44(9), 1994, 532-542.

[5]. Chien, Lee-feng, Huang, Tung-I, Lee, Shi-Bai and Lee, H. C., "尋易 (CSmart) -- A High Performance Chinese Document Retrieval System", submitted to ICCPCOL'95.

[6]. Chien, Lee-feng, "Fast and Quasi-Natural Language Search for Gigabytes of Chinese Texts", to appear on ACM conf. on SIGIR'95.

[7]. Chien, Lee-Feng, "A Model-Based Signature File Approach for Full-text Retrieval of Chinese Document Databases", To appear on Computer Processing of Chinese and Oriental Languages, 1995.

[8]. Lin, Sung-Chien, Chien, Lee-feng and Lee, Lin-Shan, "Unconstrained Speech Retrieval for Chinese Document Databases with very large Vocabulary", The 4th European Conference on Speech Communication and Technology, EuroSpeech"95

[9] Ziv, J. and A. Lempel, "A Universal Algorithm for Sequential Data Compression", IEEE trans. Information Theory, Vol. 23, No. 3, pp. 337-343, May 1977.

[10] Bell, T. C., J. G. Cleary and I. H. Witten, "Text Compression", Prentice-Hall, Inc., 1990.

[11] Gu, H. Y., "Adaptive Chinese Text Compression Using Large Alphabet, Markov Modeling, and Arithmetic Coding", National Computer Symposium(Chia-yi), pp. 568-575, 1993.

[12] Chang, C. C and W. H. Tsai, "A Data Compression Scheme for Chinese-English Characters", Computer Processing of Chinese & Oriental Languages, Vol. 5, No. 2, pp. 154-182, March 1991.

[13] Chang, K. C. and S. H. Chen, "A New Locally Adaptive Data Compression Scheme Using Multilist Structure", The Computer Journal, Vol. 36, No. 6, pp. 570-578, 1993

[14] Bentley, J. L., *et al.*, "A Locally Adaptive Data Compression Scheme", Commun. ACM, pp. 320-330, 1986.

[15] Weiss, M. A., "Data Structure and Algorithm Analysis in C", Benjamin/ Cummings Publishing Company, Inc., 1993.