行政院國家科學委員會專題研究計畫　成果報告

電子圖書知識分類與再製之研究(2/2)

計畫類別：個別型計畫
計畫編號：NSC91-2213-E-009-077-
執行期間：91 年 08 月 01 日至 92 年 07 月 31 日
執行單位：國立交通大學資訊科學學系

計畫主持人：曾憲雄

報告類型：完整報告

處理方式：本計畫可公開查詢

中　華　民　國 92 年 10 月 27 日

行政院國家科學委員會補助專題研究計畫成果報告

※※※※※※※※※※※※※※※※※※※※※※※※
※ ※
※ 電子圖書知識分類與再製之研究(2/2) ※
※ ※
※※※※※※※※※※※※※※※※※※※※※※※※

計畫類別：■個別型計畫 □整合型計畫
計畫編號：**NSC 91—2213—E009—077—**
執行期間：**90** 年 **8** 月 **1** 日至 **92** 年 **7** 月 **31** 日

計畫主持人：曾憲雄教授

本成果報告包括以下應繳交之附件：

□赴國外出差或研習心得報告一份
□赴大陸地區出差或研習心得報告一份
□出席國際學術會議心得報告及發表之論文各一份
□國際合作研究計畫國外研究報告書一份

執行單位：國立交通大學資訊科學系

中 華 民 國 年 月 日

中文摘要

近年來由於資訊與網路技術的進步，大量的電子文件隨之產生，促成了電子文件的普及並逐漸改變讀者閱讀書籍的習慣。然而大量電子書籍的產生也使得讀者在選讀或檢索時，常常不知從何著手以獲得真正需要的電子書資料。加上現存之電子書標準多著重於電子書的表層資訊及文章的呈現，並沒有考量到內容層級資訊在電子書的應用與管理上所帶來的效益，以致於一般查詢系統只能透過簡單的關鍵字與表層資訊查詢來回應使用者的需求。此外，一般電子書籍管理系統缺乏彈性且無個人化的服務、無法提供個人化與群組化的書籍推薦。有鑑於此，本計畫運用資料檢索、資料探勘等相關技術，發展智慧型電子書資訊管理系統，輔助讀者在查詢或檢索相關資料時能快速且準確地得到真正想要的結果。

本計劃分為二階段進行研究，第一階段為分析案例資訊及特徵蒐集，參考現存的電子文件標準，加入內容層級標籤，制訂一套電子書標準，簡稱 CEBS。並利用資訊擷取及機器學習的方法分析電子文件的資訊和特徵，輔以階層式文件分群法為電子文件進行分群，並設計友善的查詢介面，以利讀者在查詢相關資料時能快速地獲得想要的結果。第二階段為分析使用者選讀習性以改善系統效能，利用資料探勘技術中的關聯式法則演算法找出使用者使用系統之特定模式，以期達到個人化並增進電子書資訊管理系統之效能。

關鍵詞：資料探勘、資訊擷取、文件分群、可擴展標記語言(XML)、電子書。

英文摘要(Abstract)

As the information explodes and network technologies progress, electronic documents have received increasing attentions and popularized surrounding our life. However, the problem of how to select *appropriate* electronic documents from the electronic book database becomes more important and difficult. The major reason is that the existing electronic book standards only emphasis on surface information and presentation and do not consider the information of content level when managing massive electronic books, so general systems usually just fulfill users by simple keywords and surface information. Moreover, the related backgrounds and preferences of user profile are not considered in most of electronic book management systems; the responding documents are thus often demoralized and dispirited. As the results, we attempt to develop an intelligent Electronic Book Information System (EBIS) by using techniques of data mining and information retrieval to assist users in searching and retrieving interesting electronic documents.

Two phases are proceeded in this project. In Phase 1, we first propose an XML-based standard, called Content-based Electronic Book Standard (CEBS), used to store the meta-data such as general information and content-based information of an electronic document. Next, by CEBS, a machine-learning mechanism is proposed to obtain more significant features and concepts to represent documents. After that, a novel *level-wise document clustering* is proposed to manage and search these electronic documents efficiently and flexibly. Finally, an Electronic Book Information System (EBIS) is constructed to provide end-user to obtain interesting electronic document. In Phase 2, we further analyze users' behavior by using data mining technique to improve the system performance and achieve the goal of personalization.

Keywords: Data Mining, Information Retrieval, Document Clustering, XML, Electronic Books.

目　　錄

1、前言(Introduction)

近年來，由於資訊技術與網路技術的進步，大量的電子文件隨之產生，促成了電子文件的普及；其中電子書(Electronic book)不但複製容易且兼顧環保的概念，因此逐漸改變了讀者閱讀書籍的習慣。不過，大量電子書籍的產生也使得讀者在選讀或檢索真正感興趣且符合需求的電子書籍時，常常不知從何著手或無法獲得真正需要的電子書資料。

而目前現存之電子書標準多著重於電子書的表層資訊(Surface information)，例如：作者、出版商、版權、出版日期……等等，以及文章的呈現(Presentation)，並沒有考量到內容層級(Content level)資訊對未來電子書應用和管理所帶來的效益，故對於電子書組織管理與再利用將無法提供直接的幫助，以致於一般查詢系統只能透過簡單的關鍵字(Keyword)與表層資訊查詢來回應使用者的需求。另外，一般電子書籍管理系統缺乏彈性且無個人化的服務，例如：無法混合多本書籍的各章節、無法提供個人化與群組化的書籍推薦、無法由簡單的書籍名稱明確地定義出該書籍的難易度、無法決定該書籍的重要性等等。有鑑於此，為輔助讀者在查詢或檢索相關資料時能快速且準確地得到真正想要的結果，本計畫運用資料檢索(Information retrieval)、資料探勘(Data mining)等相關技術，發展智慧型電子書資訊管理系統，以便推廣電子書的流行，改變讀者閱讀習慣，進而達到節省紙張與加速獲得所需知識的目的。

因此本計畫針對電子書的內容與階層架構進行研究，以具有描述能力的XML(eXtend Markup Language) [29]制訂一著重內容層級之電子書標準。根據此電子書標準，實作一完整管理系統架構，讓使用者可以方便、快速地使用電子書資源。本計劃分為二階段進行研究，第一階段為分析案例資訊及特徵蒐集，先蒐集一定數量的電子文件，利用資訊擷取(Information retrieval)的方法分析其資訊和特徵，並以本計畫所提出之階層式文件分群法(Level-wise document clustering)為電子文件進行分群，以利讀者在查詢相關資料時能快速地獲得想要的結果。第二階段為分析使用者選讀習性以改善系統效能，利用資料探勘(Data mining)技術找出使用者使用系統之特定模式，進而改善系統以期達到個人化並增進電子書資訊管理系統之效能。以下歸納出本計畫的成果與貢獻：

1. 參考 MPEG-7[30]多媒體描述介面的架構和其他 DocBook[27]、Open Electronic Book (OEB)[31]、Dublin Code (DC)[28]等國際標準，根據書籍本身階層架構並加入一些內容層級標籤(Content level tag)，制訂一套電子書標準，簡稱為 CEBS (Content-based Electronic Book Standard)，以便有效了解和記錄電子書的內容。
2. 利用機器學習(Machine learning)技術和書籍的領域(Domain)特性，將關鍵字向量(Keyword vector)轉換為關鍵字對各領域的關聯度向量(Domain-Keyword vector)，以挑選出文件中適當且具代表性的關鍵字。
3. 透過所提出之階層式文件分群法(Level-wise document clustering)，不但保留

了傳統文件分群(Document clustering)的優點，更提供電子書籍中書、章、節和段落彼此關聯特性，讓使用者可以快速地獲得滿意的需求。

4. 透過系統之查詢介面、文章比對，快速尋找想要的文章內容，充份利用過去累積的電子圖書資源，加以編輯、整合和再製，可製作出全新的電子書籍，有效縮短書籍編撰的時程，提供高品質的電子圖書。
5. 利用資料探勘(Data mining)中關聯式法則探勘(Association rule mining)技術，對於使用者查詢的記錄(Log)作分析。將使用者在本系統中查詢且選取的電子文件資訊儲存在使用者查詢記錄檔中(Log file)，配合典型的階層式關連式法則分析演算法，找出特定使用者或使用者族群在選讀電子文件時的選讀習性。
6. 提出一包含使用者介面、轉換模組、知識擷取模組、關鍵字領域學習模組、相似度計算模組等所構成的電子書資訊管理系統(Electronic Book Information System, EBIS)架構。

2、研究背景(Background and Related Work)

本章節將針對本計畫相關之研究背景，包括現存之電子文件標準、文件分群與資料探勘等相關技術，在以下各節中分別介紹。

2.1 現存之電子文件標準(Electronic Document Standards)

為有效管理日益增多的電子文件，以協助文件檢索，國際上制定許多標準予以依循，以便電子文件的流通，包括 DocBook、Open Electronic Book(OEB)、Dublin Code 等，以下均予以詳細介紹。

2.1.1 DocBook

DocBook 乃是依照 XML 所規範的一種應用語言，其本身便是一種標準，用來定義一些技術文件所要用的標記(tag)。DocBook 目前是由 OASIS 的 DocBook Techinical committee 作為維護者，主要是以書本編輯為目的，與 ISO12083 相似；尤其是應用在軟體文件方面，其包含了許多軟體文件的元件，現已被許多大的電腦公司所支持。其 DTD 中特別包含了『XML EXCHANGE TABLE MODEL CLARATIONMODULE』，利於表格資料的交換。

以下選擇關於 DocBook 標準中具代表性的 tag 其意義與用途稍作介紹：

1. Abstract: 在這個 tag 所包含的 paragraph 是對於之前的文章所做的 summary (有可能是 chapter)。
2. Appendix: 一本書的 appendix。
3. Appendixinfo: 在 4.0 之後才出現的 tag，記錄有關 appendix 的一些資訊(作者、編輯者、版權、日期、出版商等)，在這些 tag 當中，若是結尾有 info 的都有如上的類似特徵，所以在次出現類似的 tag 時便不在贅述。

4. Author: 作者。
5. Authorblurb: 對於作者的簡短敘述。
6. Authorgroup: 若是作者有兩人以上則要用此 tag 將作者的 tags 包起來。
7. Beginpage: 對於某個 document 而言，若是要將其 print 出來開始頁是位於 document 的哪處開始。
8. Bibliography: A bibliography。
9. Bibliographyinfo: 類似 appendixinfo。
10. Blockquote: 對於某部分的文章而言，引證某名人的說過的觀點。
11. Book: 定義一本書的 tag。
12. Bookinfo: 類似 appendixinfo。
13. Callout: 對於程式片段或者是些資料的片段，也許是一行，在其後面標上如 的標記然後在此片段之後再做解釋的 tag。
14. Calloutlist: 若是 callout 不只一個，則將一群包在一起（觀念上有一些類似 authorgroup）。
15. Caption: 對於 mediaObject 做解釋(text)，例如某圖片是何作者、日期。
16. Caution: 一段警告文字的 note。
17. Chapter: A chapter, as of a book。
18. Chapterinfo: 類似 appendixinfo。
19. Co: 在文字當中直接插入 callout 的標記。
20. Computeroutput: 標示出某段文字是 computer 的 output。
21. Copyright: 對於 document 的版權宣告。
22. Date: 對於這份 document 日期和版本。
23. Dedication: 這本書奉獻給某人。
24. Edition: 這個 document 第幾版。
25. Editor: 編輯者。
26. Equation : 公式。
27. Example: 對於某項 example 的 title。
28. Figure: 圖片，有 title、內容。
29. Footnote: 腳註。
30. Foreignphrase: 相異於 document 基本語言的外來語。
31. Formalpara: 有 title 的 paragraph。
32. Glossary: 術語。
33. Glossaryinfo: 類似 appendixinfo。
34. Highlights: 在 document 中所討論的 main point 的 summary（以一句或是少許文字來說明）。
35. Important: 將重點特別標示出來。
36. Index: 索引。
37. Isbn: The International Standard Book Number of a document。

38. Issn: The International Standard Serial Number of a periodical。
39. Itemizedlist: 用 bullet or other dingbat 標示出來的 list。
40. Keyword: 描述 document 的 word。
41. Keywordset:包含很多的 keyword。
42. Legalnotice: 一些合法行為的宣告。
43. Link: A hypertext link。
44. Lot: A list of the titles of formal objects (as tables or figures) in a document。
45. Manvolnum: 參考文件的卷或冊數的號碼。
46. Mediaobject: A displayed media object (video, audio, image, etc.) 。
47. Msg: message。
48. Msgset: message set，通常是 error message。
49. Note: 對於主要文章之外所留下來的 message。
50. Objectinfo: 對於 object 的一些資訊。
51. Orderedlist: 有順序（編號或是字母）的 list。
52. Para: A paragraph。
53. Part: 分解 book 的另外一種方法。
54. Partinfo: 類似 appendixinfo。
55. Partintro: 對於一個 part 所做的 introduction。
56. Preface: 對於一本書的某章之前所做的 introduction。
57. Prefaceinfo: 類似 appendixinfo。
58. Primary: 在 index 中要拿做 sort 的部分。
59. Programlisting: 對於程式的部分用 text 的方式 list 出。
60. Pubdate: 發行日。
61. Publisher: 出版商。
62. Reference: 類似在 unix 下的參考文件。
63. Referenceinfo: 類似 appendixinfo。
64. Revision: 描述者校訂的歷史。
65. Screen: 在原始文件中如何呈現則在 browser 時便如何呈現。
66. Section: A recursive section。
67. Sectioninfo: 類似 appendixinfo。
68. Set: 在 docBook 中以 set 為最上層，其下才是許多書(book)。Set 所表示的是某類書。
69. Setinfo: 類似 appendixinfo。
70. Sgmltag: A component of SGML markup。
71. Subtitle: 副標題。
72. Table: A formal table in a document。
73. Title: 標題(用於 book chapter section 等) 。
74. Trademark: 商標。

75. Ulink: link 的目標是 URL。

76. Warning: 特別警示讀者的一段文字。

2.1.2 Dublin Code

1995 年 3 月，國際圖書館電腦中心(OCLC)與 National Center for Supercomputing Applications (NCSA)共同研討與制定出 Dublin Core（都柏林核心集）。目的在於希望建立一套描述網路上電子文件特色的方法，以協助資訊之檢索。因此其將詮釋資料(Metadata)定義為資源述(Resource Description)，Dublin Core 所處理之對象將祇限於「類文件物件」(Document-Lick Objects，簡稱 DLO)，DLO 之定義為【可用類似描述傳統印刷文字媒體方式，加以描述的電子檔案】。

其主要目的在幫助使用者查檢網路資源，而不是在詳細描述該資源，因為都柏林核心集之目標是在發展一個簡單有彈性，且各種專業人員皆可輕易了解與使用之資料描述格式，所以其只規範了在大部份情況下，必須提及的資料特性。都柏林核心集其規範了描述網路資源時所最低須具備的 15 個項目而這 15 個項目都是可選擇、可重覆及可延伸的。分別是：

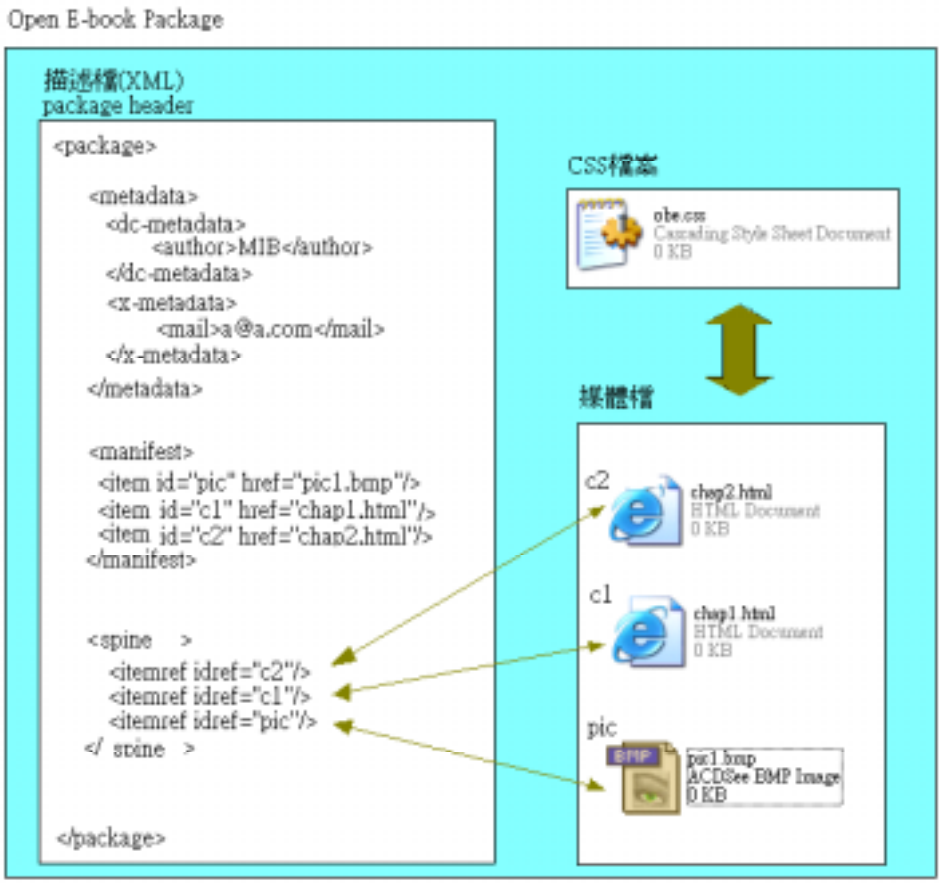
1. Title: 資源名稱。
2. Format: 資源格式，例如：書、錄音帶、.html 檔…。
3. Author: 主要作者。
4. Resource Identifier: 資源識別碼。
5. Other Contributors: 其他作者。
6. Source: 來源。
7. Subject or Keywords: 主題或關鍵詞。
8. Language: 語言。
9. Description: 資源描述。
10. Relation: 關連性。
11. Publisher: 出版者。
12. Coverage: 時空範圍。
13. Date: 日期。
14. Rights Management: 著作權管理。
15. Type: 資源性質，例如：散文、論文…。

2.1.3 Open Electronic Book

Open eBook Publication Structure Specification 1.0 由 Open eBook Authoring Group 於 1998 年冬天到 1999 年夏天制訂。Open eBook Authoring Group 是由 NIST（the National Institution of Standards and Technology）召集 25 個各種有關出版事業及出版 ebook 的組織所組成。

Open E-book 的 package 採用一個描述檔、多個媒體檔、一個 CSS 的架構，來制定一本書的格式及架構。

1. 描述檔(or package header)：是一個 XML 檔案，裡面利用 Dublin Core 的規定說明了一本書的基本資料，例如：作者、出版日期、ISBN 等；另外又利用<manifest>標籤說明 package 中有哪些檔案，這些檔案的類型；利用<spine>標籤說明 package 中的檔案在 E-book Reader 中呈現的順序。
2. 媒體檔：媒體檔是呈現內容的檔案，將文章的內文放入媒體檔，媒體檔通常是一個 HTML 或是 XHTML 檔案，。
3. CSS：Open E-book 的 package 需要一個 CSS 去裝飾書籍的外觀，字型大小，顏色等。如果沒有 CSS 存在 package 中，Open E-book 將呈現純文字的畫面。



圖一：Open E-book package 架構圖

以下簡單介紹 Open E-book 的 tag：

1. Package: 指出這個 package 有哪些 documents、images、或是其他任何形式的 object。
2. Metadata: 標記這本書的基本資料，又將基本的資料分成兩種
3. Dc-metadata: 儲存 Dublin Core 所規定的規格，例如：contributor、出版日期、語言、版權。
4. X-metadata: 儲存 Dublin Core 沒有規定的東西，例如：價格、e-mail。

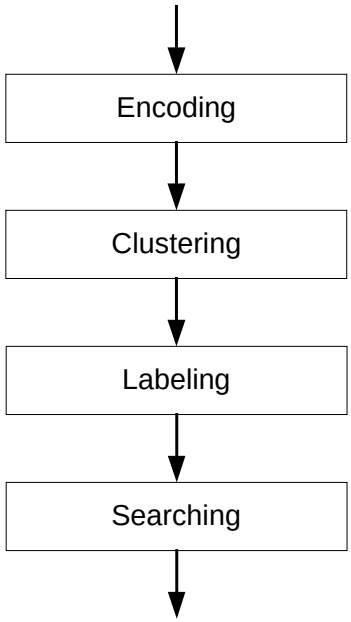
5. Manifest: 條列式的表示出這個 publication 有哪些的元件，元件的類型，預設的讀取程式及後備讀取程式。
6. Spine: 形容這個 publication 應有的基本架構，閱讀順序。
7. Tour: 給予讀者以不同的方法閱讀。
8. Guide: 提供書裡面各個部分(index、bibliography、reference)的 link，讓 reader 方便提供 link 給讀者。

2.2 文件分群暨檢索(Document Clustering and Indexing)

為管理大量的電子文件，並輔助使用者快速檢索有關資料，在文件探勘(text mining)、資訊擷取(information retrieval)等領域有不少學者正研究各種有效的方法，而文件分群法為最佳選擇之一 [7][13][14][21]。

一般電子文件的內容適合人閱讀，但對電腦而言，卻無法處理複雜的文件本文。因此為了方便電腦進行處理，文件分群法遂先針對大量電子文件進行編碼工作(Document encoding/Document representation)，亦即將電子文件以特定、適合電腦分辨的形式表達出來，以便電腦進一步處理。接下來便以特定之文件分群演算法(Document clustering)替文件分群，將內容相似之電子文件分為同一群，每個文件群都有一標示(Document labeling)，該標示可輔助使用者的查詢而進行檢索工作(Document searching)。

即如以上所述，文件分群和文件檢索包含四大步驟，其流程圖如圖二所示。



圖二：文件分群暨檢索流程圖

2.2.1 文件編碼(Document Encoding)

在文件編碼步驟中，最常見的是將電子文件以適當的關鍵字(Keywords)表示

[16][19]。這些「適當的關鍵字」被視為是最具描述力的特徵(Feature)，並以向量形式(Vector)表達，也就是所謂的 Vector space model method [8]。

電子文件向量(Document vector)的每個特徵都有一個權重值，最普遍的權重值計算法即根據該關鍵字在同一文件中出現的次數(Term Frequency, TF)結合其在各個文件中出現的比率(Inverse Document Frequency, IDF)，稱為 TF-IDF 計算法[6][11]。一個電子文件向量可被表示為 $d_{tfidf} = (tf_1 * idf_1, tf_2 * idf_2, ..., tf_n * idf_n)$，其中 tf_i 是第 i 個關鍵字在文件 d 中出現的次數；idf_i 是第 i 個關鍵字在各個文件中出現的比率，可由 $\log(n/df)$ 計算而得（n 是文件總數，而 df 是有出現該關鍵字的文件數）。

這樣的計算其設計概念為，文件中某一關鍵字出現次數越多，表示該關鍵字是作者欲意強調的重點字，也是該文件的重要概念，是可以用來表示該文件內容的特徵；而某一關鍵字若在每一個文件內容都有出現，表示此關鍵字太頻繁出現，其關鍵字缺乏鑑別力，，在文件分群時會使得文件向量沒有區分性故以 IDF 形式來降低該關鍵字的權重。

2.2.2 文件分群(Document Clustering)

假設每個電子文件皆已編碼為電腦可處理的形式，在文件分群步驟中，相似的文件向量會被分在同一群，而所謂「相似」的定義即以 *cosine function* 為計算：

$$Similarity = \cos ine(d_1, d_2) = \frac{d_1 \bullet d_2}{|d_1||d_2|},$$

其中，d_1 和 d_2 分別為二個電子文件的文件向量，$\bullet$ 是向量的相乘(dot product)，$|d_1|$ 和 $|d_2|$ 分別為 d_1 和 d_2 向量的長度。若 cosine 值大於所定義的門檻值(threshold)，則此二個電子文件即視為「相似」。

2.2.3 文件標示(Document Labeling)

文件經由分群後，相似的文件被歸於同一群，為方便將來使用者檢索相關資料，須將每個文件群標上特定的標示，如此可根據使用者的要求回應相關的文件群給使用者。常見的標示法可用文件群中最常出現的關鍵字表示或由文件群的中心值(Cluster center)代表該文件群，其中心值計算法可由同一文件群中的所有文件向量做平均動作而得。

2.2.4 文件檢索(Similarity Search)

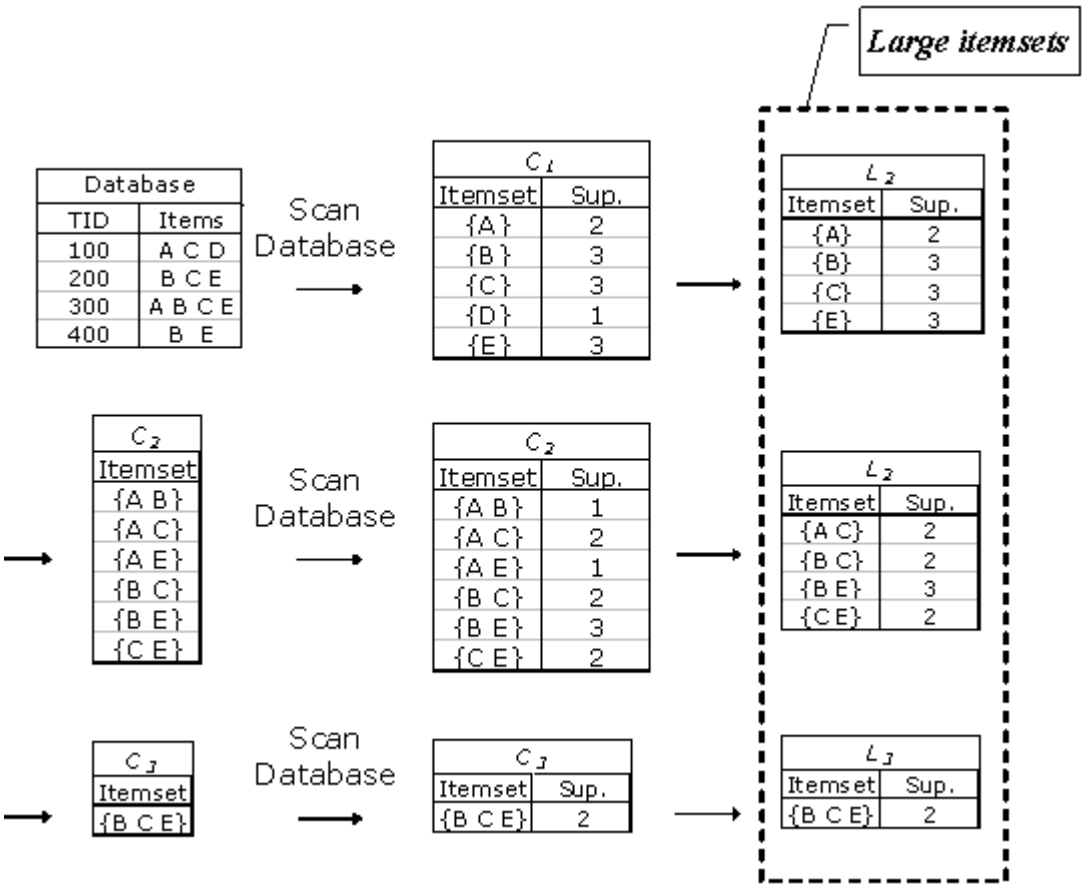
在文件檢索時，由使用者先輸入查詢(Query)，系統再根據使用者所要求的門檻值進行文件群和查詢問句的相似度比對。若文件群與查詢問句的相似值大於所定義的門檻值，該文件群即為使用者所需要的，系統便將該文件群回應給使用者。而此檢索系統的好與壞即由使用者的查詢滿意結果為指標。

2.3 資料探勘與關聯式法則(Data Mining and Association Rules)

資料探勘(Data mining)是近來新興熱門的技術[2]，目前廣泛應用於大型資料庫(Database)與資料倉儲(Data Warehouse)，以幫助企業組織從這些龐大且複雜的資料中挖掘出有用的資訊與知識。其所應用的範圍相當的廣泛，根據不同的資料庫型態，資料探勘可應用到交易資料庫(Transaction databases)、時間序列資料庫(Temporal databases)、關聯式資料庫(Relational databases)、多媒體資料庫(Multimedia database)以及其它不同類型資料庫。另外一方面，以所獲取的知識來區分，萃取出來的資訊包含：關連式法則(Association rules)、分類法則(Classification rules)、族群法則(Clustering rules)、循序項目(Sequential patterns)[4]……等等。其中，以由交易資料庫中所得到的關連式法則最為大家所熟悉[1][2][3][4][5][9][10][17][18]。

概念上來說，關連式法則意指在某筆交易資料中若出現了某些項目集(Itemset）時，則此筆交易資料中也可能伴隨出現其他特定項目集。最典型的例子為：「當某位顧客購買牛奶時，他也可能會購買麵包」。一般來講，獲取關連式法則的程序主要可分為二個步驟：(1) 針對所給的資料集合(dataset)，找出所有滿足使用者定義之最小支持度(Minimum support)的項目集；亦即尋找較頻繁、較常出現的項目集(Frequent itemsets)。(2) 接著再由前一步驟所獲得的頻繁項目集產生符合使用者定義之最小信賴度(Minimum confidence)的法則(Strong rule)；亦即產生關聯式法則(Association rule)。因此步驟(1)主要是獲得在統計上大量具顯著性的項目集，步驟(2)主要是產生關聯性法則。

而在關連式法則探勘的過程中，最大的困難與挑戰是如何在步驟(1)中縮小搜尋空間(Problem space)並降低計算時間。在之前的研究中，Apriori 演算法[1][2][3]利用階層式候選者產生方式(level-wise candidate generation approach)，大量地刪除不滿足使用者定義之最小信心值的項目集，是最具經典而代表性的。Apriori 演算法最重要的概念為：若一項目集未達到使用者定義之最小支援值，則此項目集的所有父集合(supersets)亦不滿足而可以將之刪除。因此只有在本階層為頻繁項目集才需要在下一階層被考慮。這種特性可以大大地減少需要考慮的項目集的數量，進而降低計算時間。也就是 Apriori 演算法一開始會針對項目集中只包含一個項目(Item)進行檢核(亦即 Candidate 1-itemsets)，保留符合條件(滿足使用者定義之最小支持度)的項目集(亦即 Frequent 1-itemsets)；接著由這些符合條件的項目集組合出可能需要考量之包含兩個項目的項目集(亦即 Candidate 2-itemsets)，並同樣進行檢核；以同樣的方式，每次處理完成，即處理比前次程序多一個項目的項目集，直到無法再產生更長且需要考量的項目集為止。以下例子是針對當使用者定義之最小支持度為 2 時，對交易資料庫處理的程序。



3、電子圖書知識分類與再製之研究

本計劃共分二階段，第一階段為分析案例資訊及特徵蒐集，第二階段為探勘使用者選讀習性以增進系統效能，均在以下各節詳細介紹。

3.1 第一階段：分析案例資訊及特徵蒐集

在本階段，為分析案例資訊並蒐集特徵，我們根據現有的國際標準，並考慮能展現內容特性，發展一新的電子書標準，稱為「內容層級之電子書標準(Content-based Electronic Book Standard, CEBS)」。所有蒐集到的電子文件將先依此標準收歸系統案例庫，做為接下來分析案例資訊及特徵蒐集的資料來源。

為能使電腦做進一步處理，我們發展一領域關鍵字學習機制(Domain-keyword learning mechanism)，先將案例分門別類，各類分別為一領域(Domain)，利用此一領域關鍵字學習機制粹取出各領域的特徵，之後利用「階層式文件分群法(Level-wise document clustering)」為文件分群，將相似的文件歸為一個文件群，以供使用者查詢相關文件資料，並且提供編輯內容及重組段落之功能，以達到電子圖書分類與再製之目的。

3.1.1 內容層級之電子書標準(Content-based Electronic Book Standard, CEBS)

圖三為 CEBS 依據電子書各層級結構所訂定之標籤規格，相對應於現有之電子書標準，CEBS 有下述之特點：

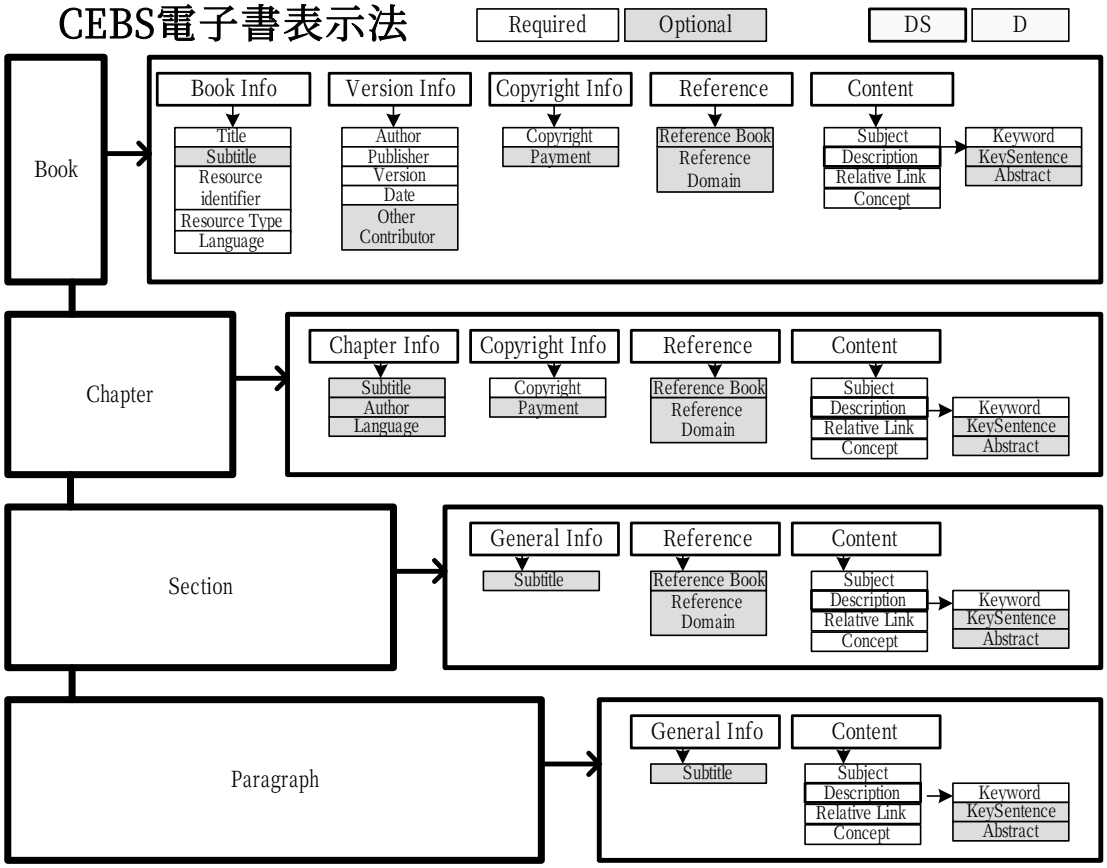
1. 物件概念的 XML 表示法：CEBS 參考 MPEG-7 [30]規格中的 *D*/*DS* 概念對標籤進行描述，其中的 *D* 為基本單元，而 *DS* 為允許引用 *D* 的描述單元，因此，對照 CEBS 內容，我們把相關的標籤組成一個單元 *D*，例如以圖三而言，*Keyword*、*KeySentence*、*Abstract* 三個標籤組成 *Description D*，讓其他標籤可以直接引用形成所謂的 *DS*，透過這樣的一個描述方式能簡化標籤的描述架構。
2. 階層架構式的描述標準：對應於書籍本身的章(Chapter)、節(Section)、段落(Paragraph)，CEBS 依循此階層架構對電子書內容進行描述。整個 CEBS 標準如圖三說明，共分為書、章、節和段落四種階層，每個階層都透過制訂的標籤對內容進行描述，底下為各階層的說明：

   Book 階層：包含 *Book Info*、*Version Info*、*Copyright Info* 三個 *DS* 記錄書籍有關書名、型態、語言、作者、出版社、版本、版權和付費…等一般性資訊。此外利用 *Reference* 和 *Content* 兩 *DS* 分別記錄參考書籍、參考領域和內容層級資訊，其中內容層級資訊包含關鍵字、關鍵句、摘要、領域、概念等訊息。

   Chapter 階層：同樣包含了 *Copyright Info*、*Reference* 和 *Content* 三個 *DS* 用以紀錄 Chapter 階層的資訊。此外利用 *Chapter Info* 此 *DS* 記錄了關於本章標題、作者和語言的說明，允許共同著作的個別作者指定。

   Section 階層：同樣使用 *Reference* 和 *Content* 兩個 *DS* 記錄 Section 階層的資訊，除此，還新增了 *General Info* 此 *DS* 紀錄 Section 的標題資訊。

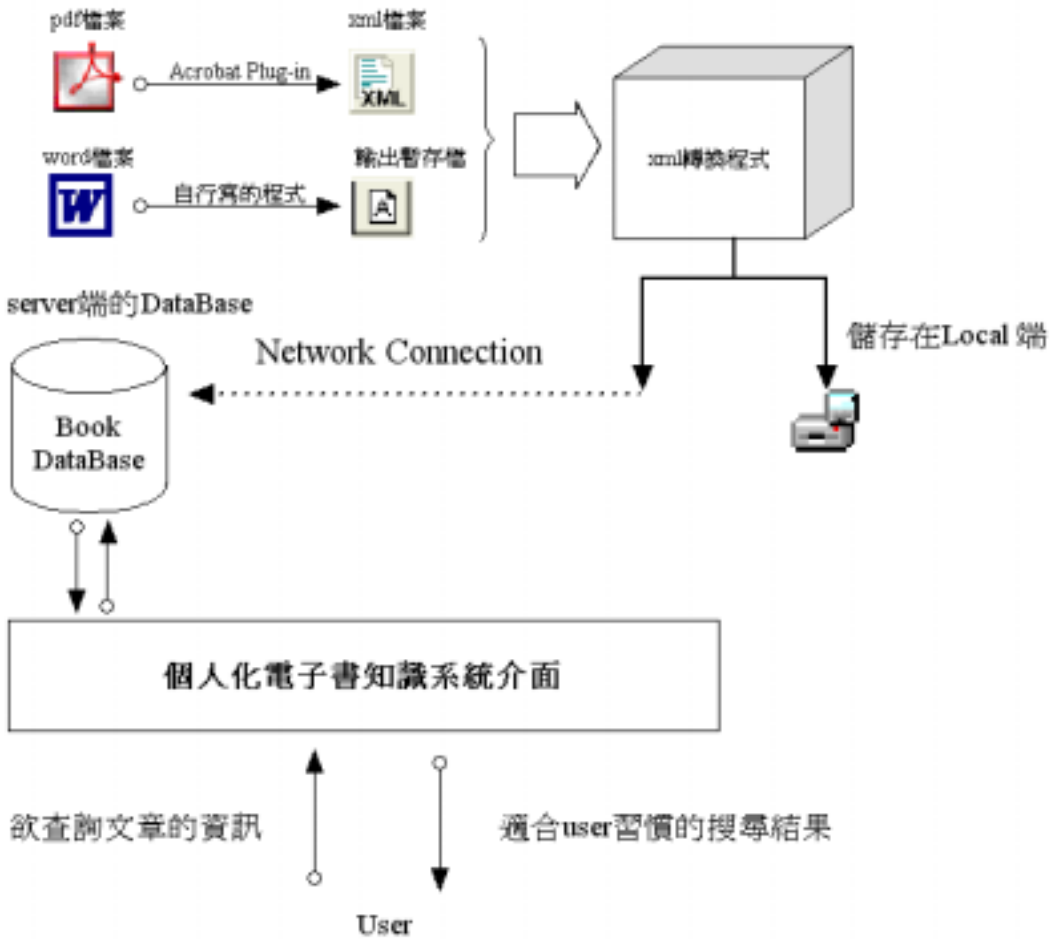
   Paragraph 階層：在 CEBS 中為表示書籍內容的基本單位，由於內含的資訊有限，因此僅使用 *General Info* 和 *Content* 兩個 *DS* 來記錄每個 Paragraph 的訊息。
3. 內容層級的描述標準：有別於其他電子書規格只有類似於 CEBS 中 *Book Info*、*Version Info*、*Copyright Info* 三個 *DS* 中一般性資訊的記錄，CEBS 標準著重在對內容意義的描述，藉由定義多元化內容相關的標籤(e.g., *Reference* & *Content DS*s)，紀錄書籍中內容層級的資訊，因此，更能有效表達作者所要傳達的概念和書籍所陳述的內容。
4. 內容描述和內容分離的電子書標準：在 CEBS 標準中僅對內容描述進行紀錄，也就是文章背後的意義，對於文章內容僅透過超連結(Hyperlink)的方式連結到對應的文章片段，如此內容和內容描述分離的架構能減少 CEBS 檔案的大小、簡化 CEBS 的複雜度進而增加 CEBS 檔案的處理效能，且更有助於對內容描述的應用。



圖三：CEBS 電子書標準

3.1.2 案例轉換工具(Document Transformation)

為能得到格式一致的電子文件做為後續研究的資料來源，我們發展了一個案例轉換工具，將使用者透過一般文書編輯軟體所產生之 MS Word、PDF、HTML 等類型的檔案，依據內容層級之電子書標準(CEBS)轉換成格式一致的 XML 檔案。透過 XML 文件的標準格式和階層化架構可以讓後續的關鍵字領域學習機制及階層式文件分群法清楚了解每份 XML 文件內的書籍內容和階層架構，進而有效的取得各階層下的資訊，這些階層包括標題、章節和段落。



圖四：案例轉換流程圖

3.1.3 領域關鍵字學習機制(Domain-Keyword Learning Mechanism)

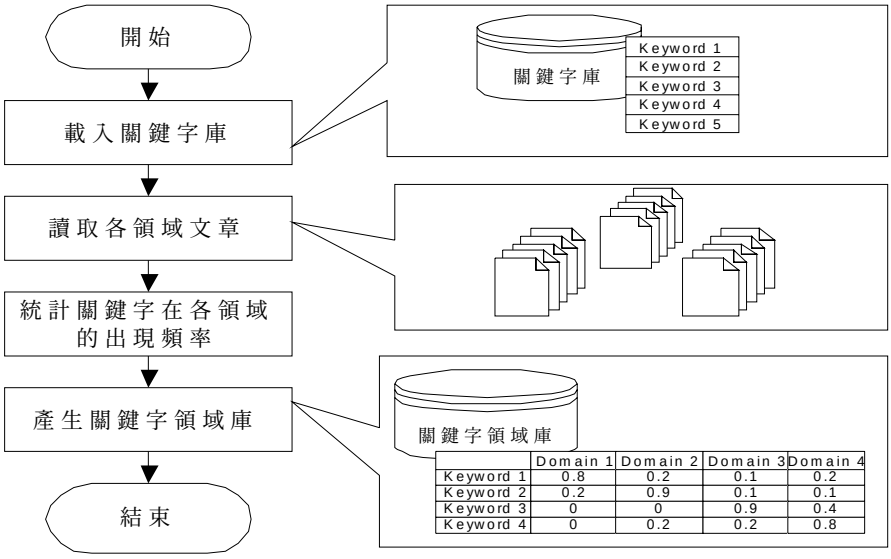
我們在看一篇文章的時候，可以由內容知道其中的意義，並且可以和其他文章做比較，了解文章所要表達之概念；但是對於電腦而言，它並無法像人類一樣去看一篇文章並且知道意義，所以我們必須要用一些方法去讓電腦去模擬人類「看」的動作，我們所選擇的方法是將文章數值化，因為數值化之後才可以讓電腦做類似的動作。

就像人類一樣，若是我們看過了許多文章之後，便可以得到某類文章的特徵，我們也想模擬這種動作，所以我們會先將許多的文章丟給電腦來當做訓練，並且找到一些特徵，例如對於某個領域的文章，常常都會出現一些的關鍵字，那麼若是下次在新的文章中，我們也找到類似的關鍵字，那我們也就可以說哪篇新文章也可能是屬於該領域的。

基於上述想法，我們訂出一個領域關鍵字學習機制(Domain-keyword learning mechanism)，讓電腦可以學習出某特定領域(Domain)該有的關鍵字(Keyword)/特徵(Feature)。在領域關鍵字學習機制中，首要決定的是什麼可以用來表達文件或領域的「特徵」？關鍵字為表示一份文件內容最簡單的方式，而透過關鍵字詞庫

的使用能限定其範圍在一些較為重要的關鍵字上。但是如何挑選適當且具代表性的關鍵字，一直是文件分群(Document clustering)、資訊擷取(Information retrieval)和特徵挑選(Feature selection)等領域的重點研究方向。在本研究中，我們利用機器學習(Machine learning)技術，學習各領域(Domain)中具代表性的關鍵字，接著將所獲得的關鍵字作簡單的整理，即可歸納出關鍵字與領域的關係對照表，很容易由表中了解某一關鍵字對於哪些領域較具代表性。圖五為關鍵字領域學習模組的學習流程。

圖五：關鍵字領域學習模組的學習流程



若單純以關鍵字庫中之關鍵字為向量(Vector)來代表段落、節、章和書，不但造成向量維度過大的缺點，而且可能因關鍵字不具代表性以致無法充份代表文件內容。因此，我們利用書籍的領域(Domain)特性，將關鍵字向量轉換為關鍵字對各領域的關連度向量。例如：以電腦書籍而言，區分為文書處理領域、程式語言領域、……，接著透過最基本的關鍵字詞庫去統計每個關鍵字在各領域的出現次數，進而決定每個關鍵字和各領域的關連度並產生了關鍵字領域庫。其中，對於一些沒有鑑別性(在多數的領域中都有出現)和出現頻率過少的關鍵字，降低其影響的關連度，往後一篇文章或一本書根據其出現頻率較高的關鍵字便可初步判斷該文章或書的可能領域並提供知識擷取模組判斷的依據。

在領域關鍵字學習機制中，最終結果是要得到一個關鍵字對於領域的權重值表(Weighted table)，建立此權重值表總共分作三個階段；首先，利用學習階段(Training Phase)先建立了基礎的權重值表，然後利用鑑別度調整階段(Discrimination Phase)調整關鍵字的鑑別能力，最後根據未來新輸入的文章利用微調階段(Tuning Phase)來做微調動作；以期自動學習產生新的關鍵字權重值，且因為關鍵字的分佈並不是穩定的，還是會改變的，所以希望能夠在有改變的時候做一些相關的調整。此三階段分別詳細如下：

1. 學習階段(Training Phase)：給定某個領域的文章 N 篇，並且每一篇的文

章都有計算關鍵字的個數，然後以亂度(Entropy)的方式來做為該關鍵字在該領域所得到的權重值；然後在經過正規化(Normalize)即可得到該領域的分佈。公式如下：

$$Kweight_{iDj} = -\sum_{d,d\subset Dj}^{Dj} n_d * log\left(\frac{n_d}{N}\right), where\ N_i = \sum_{d,d\subset Dj}^{Dj} n_d$$

$$\text{Weight}_{\text{iDj}} = \text{Kweight}_{\text{iDj}} * \left(\frac{N_i}{\sum_i^I N_i}\right)$$

舉例如下：

DB	文件 1	文件 2	文件 3	文件 4	Count	%	Weight	%*Weight	Normalize
Relation	50	50	50	30	180	0.6	354.74	212.84	1
DBMS	30	30	30	30	120	0.4	240	96	0.451

在 DB Domain 中我們蒐集了 4 篇文件，粹取出兩個 keyword 其在 4 個文件中的所得到的統計如上表所示。其中 Count 的部份便是計算 4 個文件中該關鍵字出現次數的總合，例如 Relation 的出現次數為 180 (180=50+50+50+30)。 %是將現在的出現次數除以總出現次數數，例如 Relation 中為 0.6 ((Current Count)/Σ(Count)=180/(180+120))。然後是 Weight，計算方法是計算亂度(Entropy)的方法，例如 Relation 中為 354.74(-Σ文件的 count log(文件的 count/Current Count))=-(50*log(50/180)+ 50*log(50/180)+ 50*log(50/180)+ 30*log(30/180)))。%*Weight 是將%*Weight 得值計算出來。Normalize 是將(%*Weight)的數值縮放到[0,1]的範圍內。

2. 鑑別度調整階段(Discrimination Phase)：在學習階段時是區域性觀察(local view)，也就是針對各個領域分別蒐集計算關鍵字的權重值，但在學習階段完成時，整體性觀察(Global view)卻發現有些關鍵字在每個領域的權重值都差不多，如此在分群時是較不具鑑別力(Discrimination Power)的關鍵字，我們稱之為「贅字」，於是為了提高分群時的與檢索時的準確度，我們就降低這類關鍵字的權重值，減低它的影響。

我們利用「Gini index value」[22]做為衡量關鍵字鑑別力的標準，先將向量的值正規化(Normalization)，調整成為總合為 1 的數值之後，在將平方和加起來，若超過一個數值便不去修改，否則將向量中的每一個數字乘以剛剛算出來用作判斷的數字，並且儲存回去。

舉例如下：

Keyword	資料庫領域	DM 領域	作業系統領域	文書處理領域	美工排版領域
Database	1	0.9	0	0	0
Data	0.4	0.3	0.1	0.3	0.2
DBMS	0.5	0.2	0	0	0
Mining	0.2	0.4	0	0	0
Clustering	0	0.4	0	0	0

Step1：先將各個 Vector Normalize 成總和為 1

$V_{database}$=[1/1.9, 0.9/1.9, 0/1.9, 0/1.9, 0/1.9] = [0.53, 0.47, 0, 0, 0]

V_{data}=[0.31, 0.23, 0.08, 0.23, 0.15]

V_{DBMS}=[0.72, 0.28, 0, 0, 0]

V_{Mining}=[0.33, 0.67, 0, 0, 0]

$V_{Clustering}$=[0, 1, 0, 0, 0]

Step2：求出各 Keyword Vector 的 gini index value

$P_{database}$= 0.53*0.53+0.47*0.47+0*0+0*0+0*0=0.5018

P_{data}= 0.2308

P_{DBMS}= 0.5968

P_{Mining}= 0.5578

$P_{Clustering}$= 1

Step3：新的 Keyword Vector ＝ 原 Vector * 其 gini index value

$V'_{database}$=[0.53*0.5018, 0.47*0.5018, 0*0.5018, 0*0.5018, 0*0.5018]=[0.27, 0.23, 0, 0, 0]

V'_{data}=[0.07, 0.05, 0.02, 0.05, 0.03]

V'_{DBMS}=[0.43, 0.17, 0, 0, 0]

V'_{Mining}=[0.18, 0.37, 0, 0, 0]

$V'_{Clustering}$=[0, 1, 0, 0, 0]

結果如下：

Keyword	資料庫領域	DM 領域	作業系統領域	文書處理領域	美工排版領域
Database	0.27	0.23	0	0	0
Data	0.07	0.05	0.02	0.05	0.03
DBMS	0.43	0.17	0	0	0
Mining	0.18	0.37	0	0	0
Clustering	0	1	0	0	0

2. 微調階段(Tuning Phase)：當有一篇新文章輸入之後，我們依然可以先

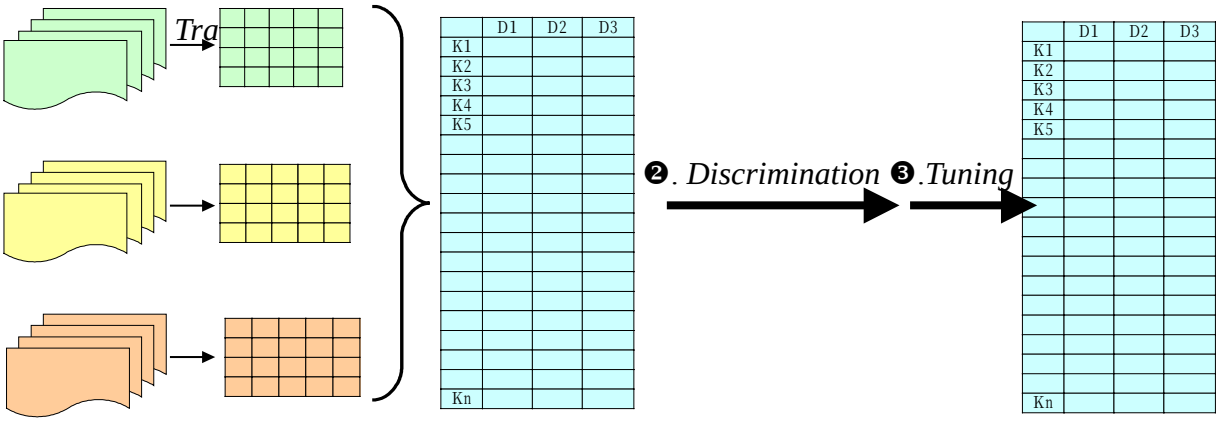
計算該文章的關鍵字分佈，將每一個關鍵字在每一個領域的分佈紀錄下來(若是在學習階段中沒有出現就視為 0)，然後計算該文章在每一個領域中的值，算法是以每一個關鍵字的(次數)*(在每個領域中的值)做總合，然後再除以總次數(所有關鍵字的次數總合)，來當做這一篇文章的向量值。然後對於每一個出現在文章中的關鍵字做修正，修正的方法是將((本來的 vector*N)+(後來的算出的 vector))/(N+1)來做為新的 vector。

舉例如下：

Keyword	#	DB 領域	DM 領域
Database	50	1	0.9
Table	50	0.86	0.3
DBMS	50	0.367	0
Mining	50	0	0.24
Clustering	50	0	0.26
SUM	250	2.183	1.7

我們可以算出來該文件的向量值為[0.4366,0.34]利用該值去修正之前算出來的權重值表中的值，公式為 Vnew=(Vold*N+Vd*1)/(N+1)，其中 N 為學習階段時的文件數目加上已微調過的文章數。所以新向量為：

(50*[1,0.9]+1*[0.4366,0.34])/51=[0.99,0.89]



圖六：三階段示意圖

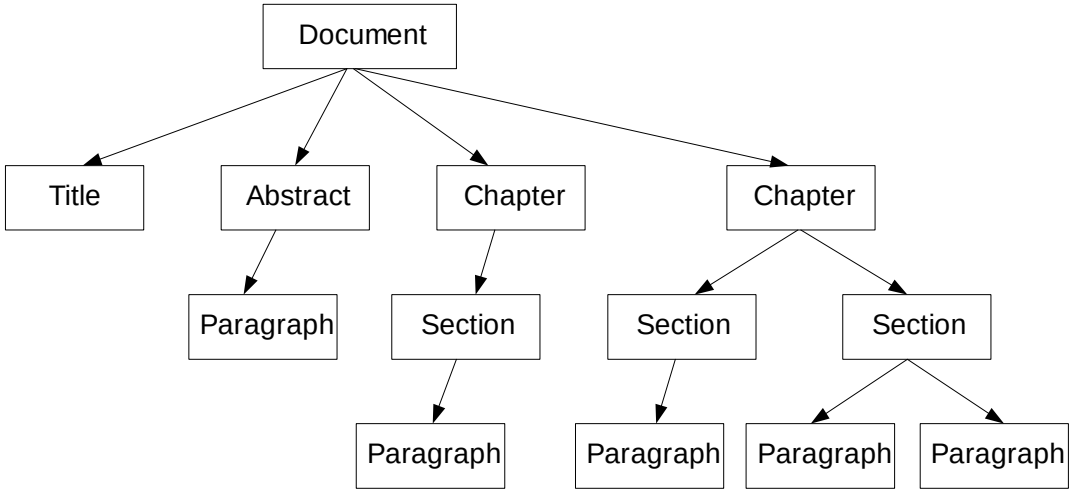
3.1.4 階層式文件分群法(Level-wise Document Clustering)[25]

在過去的研究中[12][20][24][26]，為有效縮短使用者查詢的時間，一般而言，儲存於資料庫中的文件會先經由分群技術(Clustering)將文件作分群分類的前處理。常見的方法為以關鍵字向量(Keyword vector)表示每份文件，並根據每份文件的向量進行分群的動作，並記錄各個群聚的成員與彼此關聯。當使用者透過關鍵字查詢相關文件時，則先轉換該查詢為關鍵字向量後，再利用轉換後的向量找尋最相近的群，不再需要一一比對所有的文件，可以先利用少量但概略的群聚資訊作初步的篩選，再對符合的群聚作進一步的處理。

然而，如何挑選適當的關鍵字來表示一份文件，避免關鍵字過多導致向量維度過大或偏差的缺點，將是一大挑戰；另外，對於電子圖書結構上的書、章、節和段落彼此關聯特性，並無法由傳統文件統一之關鍵字向量來表示，傳統文件分群法甚少針對文件的階層架構進行考慮，導致分群的結果無法真正反應文件的特性。因此，在本計劃中我們經由領域關鍵字學習機制的輔助，挑選具代表性的關鍵字；另外考量電子書籍的階層架構特性，進而提出了階層式文件分群(Level-wise document clustering)的概念。

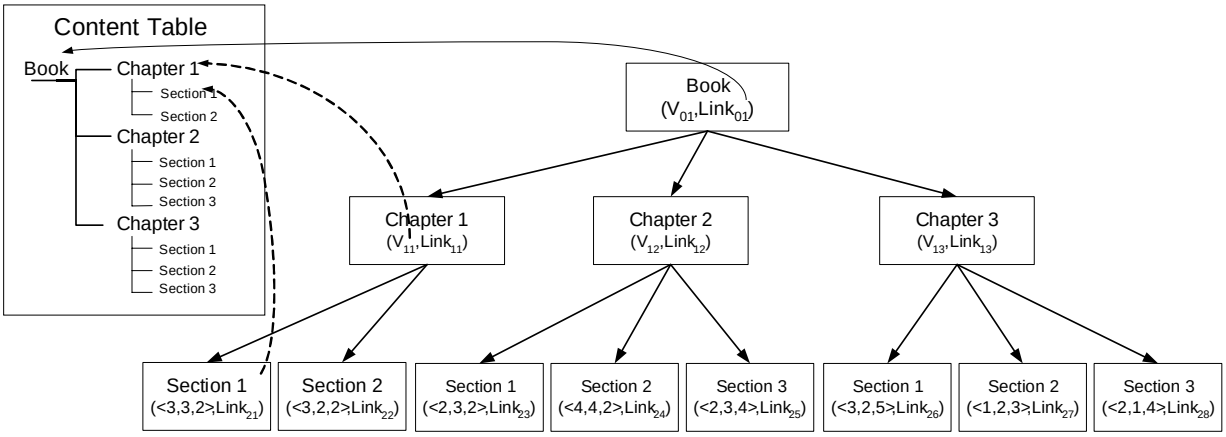
在階層式文件分群法中，有幾個重要步驟，以下分別詳細介紹：

1. **文件編碼階段(Document Encoding Phase)**：一份電子文件其架構類似圖七，可以以樹狀結構表達出來，階層式分群演算法就是透過這樹狀結構的階層特性並利用概念泛化(Concept generalization)的方式，針對不用階層的內容進行分群處理。



圖七：電子文件架構示意圖

因此在文件編碼中，我們先將所有的電子文件依其內容架構表示成樹狀結構，稱為文件樹(Document tree)；在文件樹中的每個節點就稱為文件節點(Document node)；每一個階層稱為文件階層(Document level)，如圖八所示。然後依關鍵字領域學習機制所得的關鍵字對領域的權重值表，將文件樹中最底層的階層的文件節點做編碼(Encoding)動作，即依該節點內容出現的關鍵字(Keywords)比對關鍵字對領域的權重值表而得到一向量(Vector)，該向量代表此節點的內容，用於下一階段的文件分群法中。



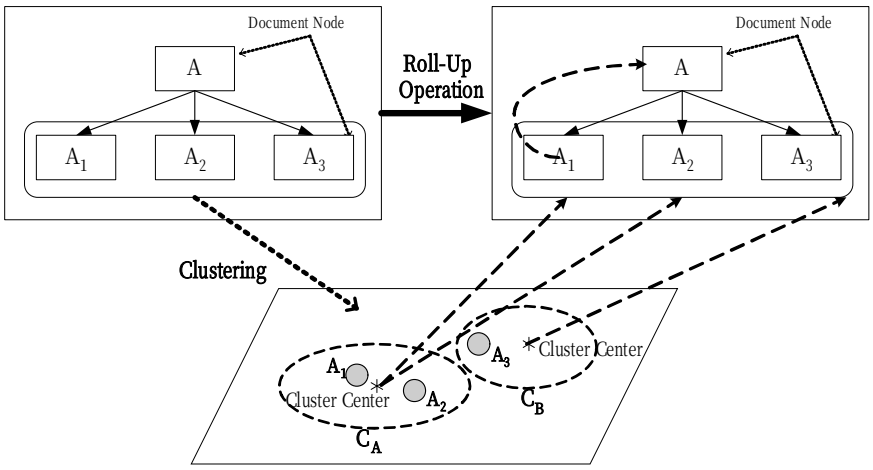
圖八：文件樹範例

2. 階層式文件分群階段(Level-wise Document Clustering Phase)：透過文件編碼階段得到最底層階層的文件節點之向量後，執行一般典型之文件分群演算法，將相似的文件向量歸為同一群，其「相似」的定義為其向量值的 cosine function 值大於門檻值。在獲得底層階層的分群結果後，將經由下一概念泛化階段(Concept Generalization Phase)將底層階層的文件群中心計算出來，從而得到上層階層文件節點的向量值。如此由下而上的文件分群法，即我們所為的階層式文件分群演算法。

3. **概念泛化階段(Concept Generalization Phase)**：此一階段是為了要將底層的分群結果反應到上層，以得到上層文件節點的向量值。如此可表現出電子文件內容與其結構的關聯程度，且節省文件編碼的工作；只要對最底層的節點做編碼工作即可，其餘節點利用概念泛化(Concept generalization) 的方式計算而得。

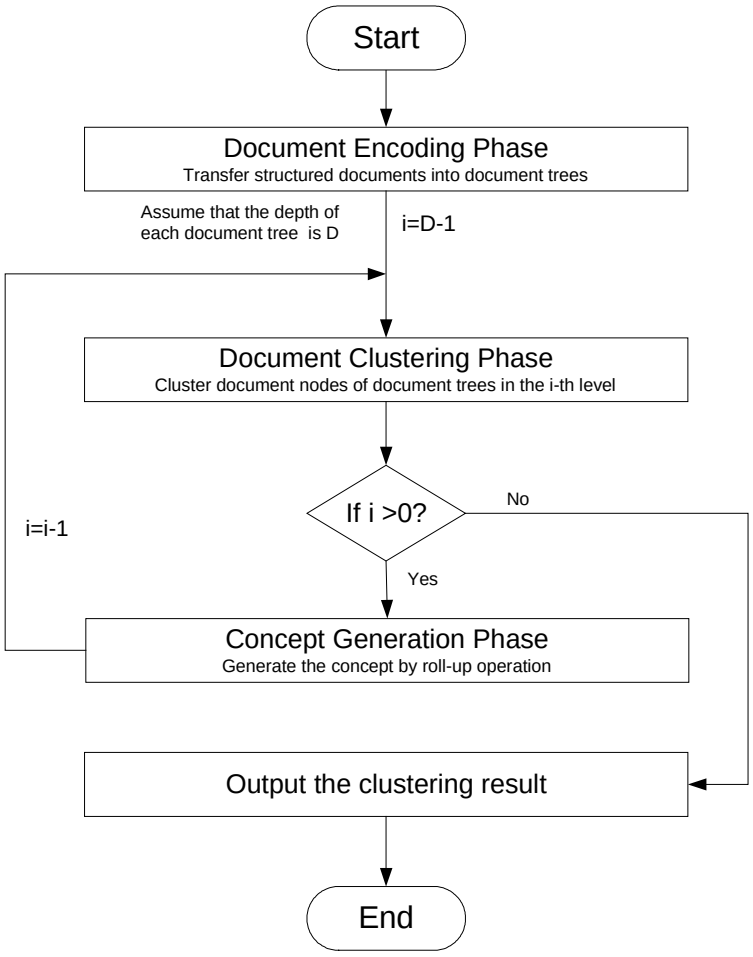
因此我們定義了一個 roll-up operation，即藉由將底層分群結果做平均的計算動作，可得到上層節點之向量值，例如，假設現在有一文件樹 A，包含三個子節點 A_1, A_2 和 A_3，其中 A_1 和 A_2 屬於文件群 C_A 而 A_3 屬於文件群 C_B。若文件群 C_A 的群中心向量值為<3, 3, 2>而文件群 C_B 的群中心向量值為<3, 2, 4>，則文件節點 A 的向量值可由平均此二個文件群群中心的向量值而得，即：

Average (<3, 3, 2>, <3, 3, 2>, <3, 2, 4>) = <3, 8/3, 8/3>



圖九：roll-up operation 例子

經由文件編碼、階層式文件分群和概念泛化後，可得所有的分群結果，這個分群結果將幫助使用者搜尋檢索時更快速、更有效率。這一整個分群步驟之流程示意於圖十：



圖十：階層式文件分群法流程圖

4. **文件檢索階段(Similarity Search Phase)**：針對階層式文件分群演算法，

為幫助使用者快速搜尋檢索，我們提出三個搜尋策略，分別為單層式檢索(Single-level search strategy)、由上而下檢索(Top-down search strategy)、啟發式(Heuristic search strategy)：

單層式檢索(Single-level search strategy): 使用者可決定想搜尋的階層層級，可以是書籍層級、章、節層級，針對同一層級的文件節點做搜尋。若某一文件群與使用者的查詢相似程度(cosine function value)大於門檻值，表示該文件群為使用者有興趣的，系統應將該文件群回應給使用者。

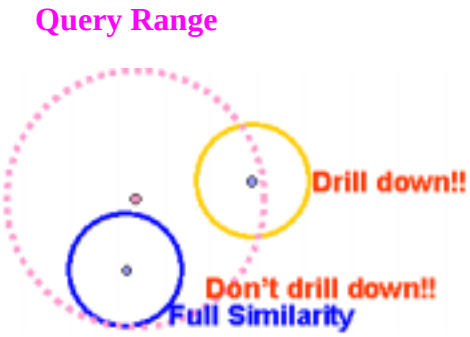
Algorithm: Single-level Search Strategy
Denotation: ClusterSet: a set of clusters
Input: The query vector Q whose dimension is the same as the vector of each document node, the desired destination stage S_{DES} and search threshold S.
Output: The set of similar clusters.
Step 1: ClusterSet=ϕ
Step 2: For each cluster ***N*** in the stage S_{DES}
  Step 2.1: Compute the similarity ***N*** with query Q.
  Step 2.2: If the similarity $\geq$ S then
    ClusterSet=ClusterSet $\cup$ ***N***
Step 3: Return ClusterSet.

由上而下檢索(Top-down search strategy): 使用者下查詢後，系統由文件樹的最上層階層的文件群開始計算其與查詢的相似程度，若其相似程度大於門檻值者，就 drill-down 到下個階層繼續計算相似程度，直到達最底層為止。

Algorithm: Top-down Search Strategy
Denotation:
D: is the number of the depth of document tree
S_0~S_{D-1}: denote the stages of a document tree from the top stage to the lowest stage.
ResultSet, DataSet: the sets of clusters
Input: The query vector Q whose dimension is the same as the vector of each document node, search threshold S and the destination stage S_{DES} where $S_0 \leq S_{DES} \leq S_{D-1}$.
Output: The set of similar clusters
Step 1: Let DataSet be the set of document tree in the stage S_0.
Step 2: ResultSet=ϕ.
  For each cluster ***N*** $\in$ DataSet,
    If the similarity measure with Q$\geq$S then ResultSet=ResultSet$\cup$***N***.
Step 3: If the stage of the node in ResultSet< S_{DES} then
    DataSet=ϕ.
    For each cluster ***N*** $\in$ ResultSet
      DataSet=DataSet$\cup$clusters returned by drill-down operation.
    Go to Step 2.
Step 4: Return ResultSet.

啟發式(Heuristic search strategy): 在由上而下檢索(Top-down Search Strategy)中只要其相似程度大於門檻值，就一直 drill-down 到底層；然而，如果在某階層其中一個文件群其相似程度完全符合使用者的查詢(Full similarity)，這個文件群應該就不用再 drill-down，可以直接回應給使用者。所以在本搜尋策略中，針對這樣子的情況加以設計搜尋方法，可以加速整個搜尋系統的進行且期望能達到和由上而下檢索差不太多的準確程度。

我們於是在階層式文件分群法中定義了所謂的「Full similarity」：



圖十一：full similarity 示意圖

假設在階層式文件分群階段(Level-wise document clustering phase)時的相似度門檻值為 T，而使用者查詢時的相似度門檻值為 S，其中 $S < T$。既然相似程度的計算方式是以 cosine function value 為衡量標準，則門檻值可以角度形式表示：T 的角度是 $\theta_T = \cos^{-1} T$ 而 S 的角度為 $\theta_S = \cos^{-1} S$。當使用者的查詢和文件群的角度小於 $\theta_S - \theta_T$ 時，該文件群即和使用者查詢達到 full similarity。其計算公式如下：

$$\begin{aligned} \text{Full Similarity} &> Cos(\theta_S - \theta_T) \\ &= Cos\theta_S Cos\theta_T + Sin\theta_S Sin\theta_T \\ &= \mathrm{S} * \mathrm{T} + \left(\sqrt{1-S^2}\right)\left(\sqrt{1-T^2}\right) \end{aligned}$$

Algorithm: Heuristic Search Strategy
Denotation:
D: is the number of the depth of document tree
S_0~S_{D-1}: denotes the stage of a document tree from the top stage to the lowest stage.
ResultSet, DataSet, FullSimilaritySet: the sets of clusters
Input: The query vector Q whose dimension is the same as the vector of each document node, search threshold S and the destination stage S_{DES} where $S_0 \le S_{DES} \le S_{D-1}$.
Output: The set of similar clusters
Step 1: Let DataSet be the set of clusters in the stage S_0 and

FullSimilaritySet=ϕ.
Step 2: ResultSet=ϕ.
For each cluster **N** $\in$ DataSet,
If N is full similar with Q then
FullSimilaritySet=FullSimilaritySet $\cup$ **N**.
Else if the similarity measure with Q$\geq$S then
ResultSet=ResultSet $\cup$ **N**.
Step 3: If the stage of the node in ResultSet< S_{DES} then
DataSet=ϕ.
For each cluster **N** $\in$ ResultSet
DataSet=DataSet $\cup$ clusters returned by drill-down operation.
Go to Step 2.
Step 4: Return ResultSet $\cup$ FullSimilaritySet.

綜合以上所述，相較於傳統的平面式分群演算法(Flat clustering algorithm)，階層式文件分群法的優點如下：

(1) 完整性(Complete)：因為每份電子文件都以樹狀結構表示，故電子文件之具有結構得以維持，可以藉由樹狀結構更完整地表現其特徵。文件在長度固定的特徵向量下，藉由階層式文件編碼方法，可以減少將文字轉為向量時的資訊流失，且能更完整地表現文件內容。

(2) 代表性(Representative)：所有的文件節點的特徵值皆藉由概念泛化(Concept generalization)階段得以產生，使得文件向量在描述文件內容時更客觀且具代表性。

(3) 彈性(Flexible)：使用階層式文件分群演算法，在檢索時更提供三種搜尋策略供使用者選擇，使用者可以設定回應的搜尋結果是在樹狀結構的最底層或是每一層，相較於其他傳統的文件分群演算法只能回應最底層結果，我們的方法更具彈性。

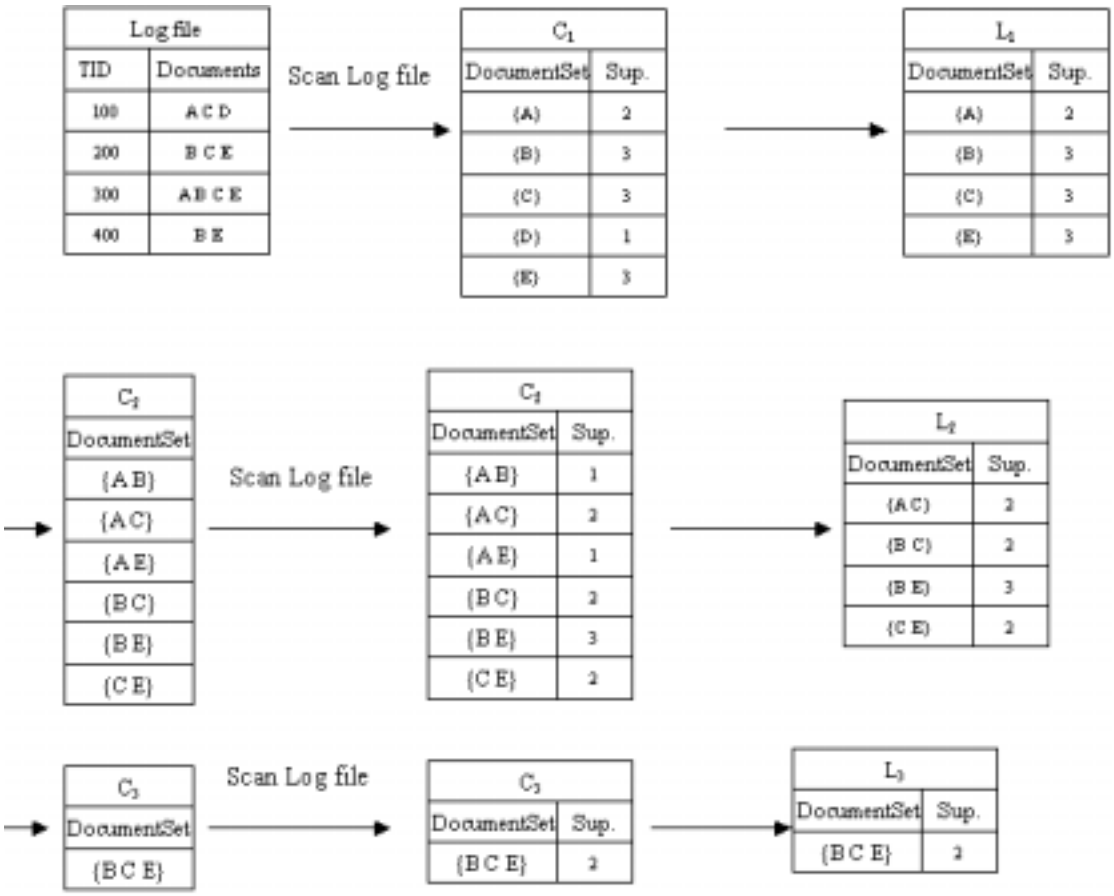
(4) 效率較高(Efficient)： 藉著維持樹狀結構的文件樹，在分群時，各層分群結果皆能有效地儲存下來，使得不管是在做分群或檢索時能夠加快速度，提高效率。

整個電子書知識管理架構在上述各方法機制和案例庫的配合下即能對電子書內容進行有效的管理，進而滿足使用者的需求。

3.2 第二階段：分析使用者選讀習性以改善系統效能

在本階段中，我們首先將利用資料探勘(Data mining)中關聯式法則探勘(Association rule mining)技術，對於使用者查詢的記錄(Log)作進一步的分析。初步的想法是將使用者在本系統中查詢且選取的電子文件資訊儲存在使用者查詢記錄檔中(Log file)，緊接著將使用者基本資料搭配累積一段時間的使用者查詢記錄當成交易資料，並配合典型的階層式關連式法則分析演算法，例如：Apriori 演

算法，找出特定使用者或使用者族群在選讀電子文件時的選讀習性。



舉例來說，若記錄檔中儲存了使用者甲的選讀文件情形，如上圖所示，其第一次瀏覽了文件 A、文件 C、文件 D，第二次瀏覽了文件 B、文件 C、文件 E，以此類推。則我們利用 Apriori 演算法，可得知此使用者最常瀏覽的文件是 B、C、E，是為其較有興趣的文件集合。故當有位與使用者甲背景相似的使用者乙在使用本系統時，系統便可以在適當時候提供使用者甲的選讀資訊供其參考。

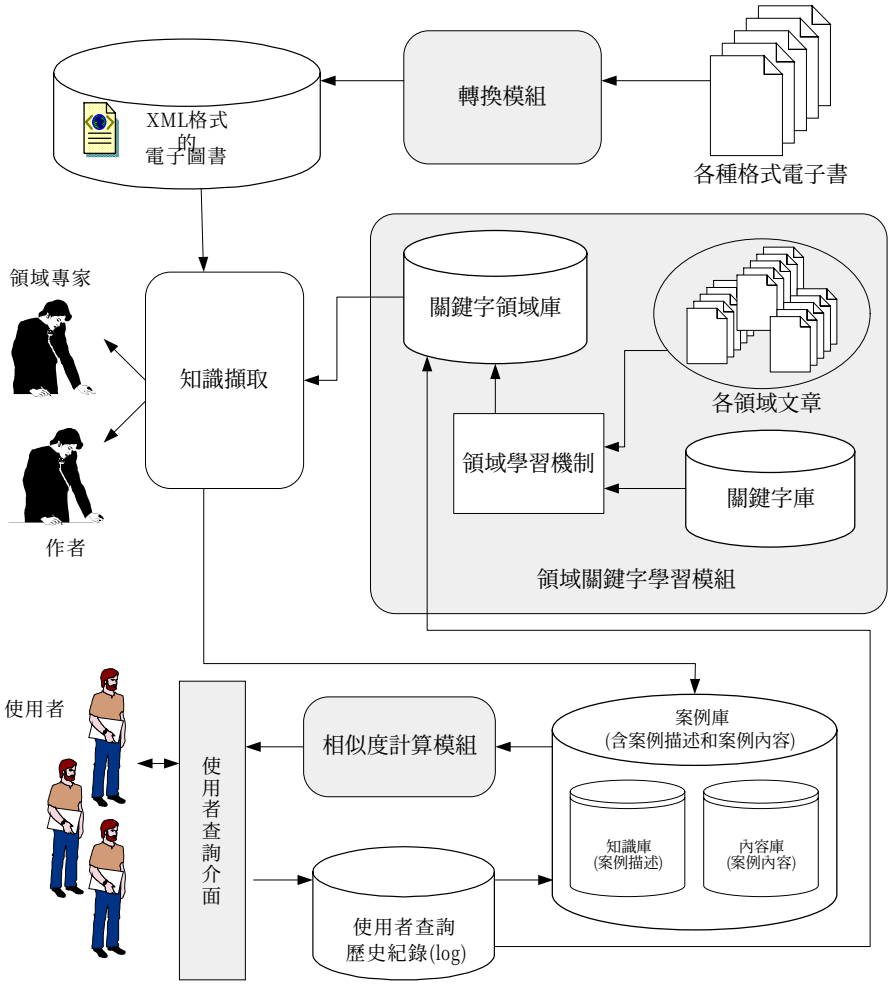
當然除上述的想法外，我們也將對使用者族群以及其他使用者喜好特性做進一步的研究。

4、智慧型電子書資訊管理系統

本計畫發展了一智慧型電子書資訊管理系統，其架構圖如圖十二所示，包含轉換模組、關鍵字領域學習模組、相似度計算模組、案例庫儲存模組、使用者查詢介面。以下分別介紹：

1. 轉換模組：利用 3.1.2 節中所提之案例轉換工具，將所蒐集到的各種不同格式的檔案，依 3.1.1 節之電子書標準(CEBS)轉換成格式一致的 XML 檔案，以供後續步驟使用。

2. 領域關鍵字學習模組：利用 3.1.3 節之領域關鍵字學習機制，找出各領域中關鍵字與領域之權重表，以供相似度計算模組使用。
3. 相似度計算模組：利用 3.1.4 節之階層式文件分群法，將轉換模組得來之案例作階層式的分群，以供使用者查詢相關文件資料。
4. 案例庫儲存模組：儲存所有電子文件案例，加上使用者查詢歷史紀錄，配合資料探勘技術找出各使用者的選讀習性特徵，用以回饋本智慧型電子書資訊管理系統，達到系統個人化之目的。



圖十二：電子書知識管理架構

5. 使用者查詢介面：智慧型電子書資訊管理系統最有創意的部分是查詢功能，使用者除了可以執行基本查詢，即經由輸入查詢問句搜尋想要的電子書，系統另外還提供文章比對查詢的功用。文章比對是當使用者現在手邊有一份文件，想要透過這一份文件找到相似的內容，就可以利用文章比對的功能。此文章比對功能，即及時性的用資訊擷取技術找出這一篇文章的內容所代表的部分候選特徵，然後馬上把這找出來的部分候選特徵與之比對，最後回應相似的內容。

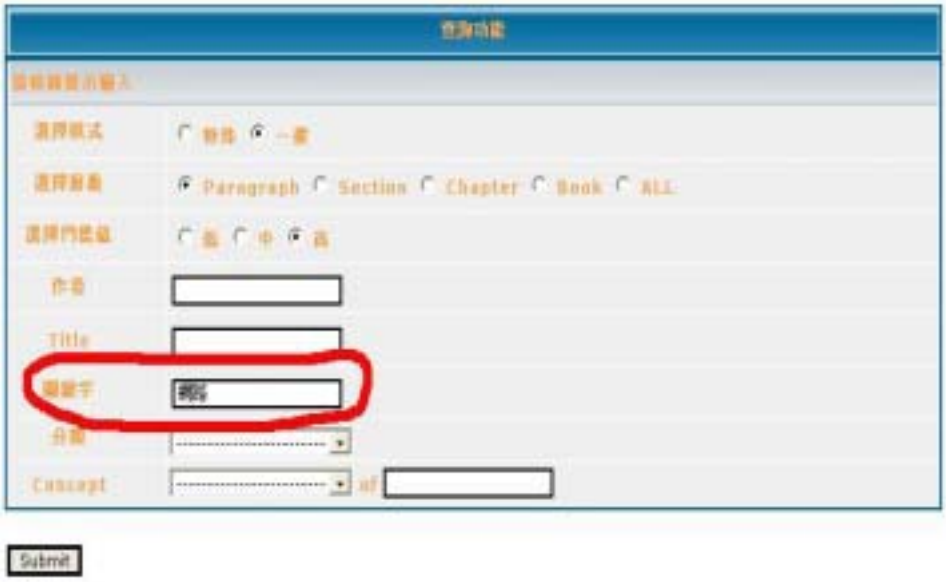
   基本查詢：又分為一般查詢和特殊查詢。在一般查詢中，使用者可以輸

入書籍作者名、書籍名稱(Title)、關鍵字(Keywords)、書籍分類、概念(Concept)等做為查詢指標，並可以選取所查詢的文章層級：Paragraph、Section、Chapter、Book 或選擇查詢所有層級(ALL)。若選擇 ALL，則使用 3.1.4 節的由上而下檢索策略，若是選擇 Paragraph、Section、Chapter 或 Book，則依其點選的層級做 3.1.4 節的單層式檢索策略。若點選特殊查詢，則相似度比對的模式即為 3.1.4 節中所提的 Heuristic Search Strategy，其查詢速度較快而準確度不致下降太多。

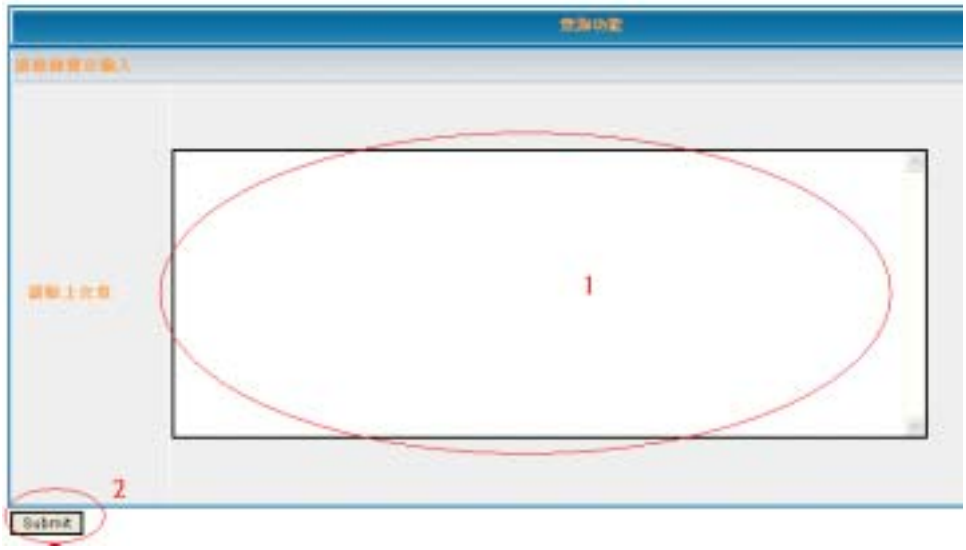
文章比對查詢：使用者輸入一篇短文或長句，系統先根據此內容擷取其特徵，轉化為向量，然後再和案例庫中的電子文件向量做相似度比對。

圖十三：系統封面圖

圖十四：查詢介面



圖十五：基本查詢



圖十六：文章比對查詢 1.使用者輸入 2.確認

5、計畫成果自評

在計劃的第一階段中，我們以 XML 語言制訂了一結構化內容層級之電子書標準 CEBS，並以 CEBS 為基礎建構了一整合電子書查詢系統 EBQS，提供給使用者更彈性、深入與廣泛的查詢與編輯需求。在第二階段，我們利用 Data mining 技術探勘使用者選讀習性，找到影響使用者滿意度的癥結所在，並進而回饋到案例庫中進行修正或是修正領域專家所建立的 Ontology 與提供個人化服務。

同時，我們亦發表了一篇期刊論文[15]、一篇會議論文[25]，並有一篇期刊論文[23]投稿於 Information Science，一位碩士班學生以本計畫內容為碩士畢業論文研究主題，亦有小成。

6、結論

由於資訊與網路技術的進步，促成大量電子文件問世，而如何有效管理眾多的電子文件，減少搜尋檢索的時間，便成為重要課題。但由於現存之電子書標準大多著重在表層資訊，沒有考慮到內容層級與其結構化資訊，故對於電子書組織管理與再利用無法提供直接的幫助；此外，一般電子書籍管理系統並無個人化的服務。有鑑於此，為輔助讀者在查詢或檢索相關資料時能快速且準確地得到真正想要的結果，本計畫運用資料檢索(Information retrieval)、資料探勘(Data mining)等相關技術，發展智慧型電子書資訊管理系統，以便推廣電子書的流行。

本計畫分為二個階段進行。第一階段為分析案例資訊及特徵蒐集，研究發展一智慧型電子書資訊管理系統，參考 DocBook、Open E-Book、Dublin Code 等國際標準，利用具描述能力的 XML 制訂 CEBS 電子書內容儲存標準，加入內容層級標籤。並利用領域關鍵字學習機制與階層式文件分群法，組合而成一電子書知識管理架構，能有效的對電子書內容進行管理，進而提供有別於傳統搜尋引擎的三種搜尋方式，對電子書內容查詢更有效，更快找到符合使用者需求的查詢結果。此外透過系統介面，使用者可充份利用過去累積的電子圖書資源，加以編輯、整合和再製，製作出全新的電子書籍，有效縮短書籍編撰的時程，提供高品質的電子圖書。

第二階段為探勘使用者選讀習性以改善案例表示，利用資料探勘的技術針對使用者進行選讀習性的分析，找出使用者個人使用的愛好與習慣，進而改善系統效能，達到系統個人化的效果。

參考資料(Reference)

[1] R. Agrawal, T. Imielinksi and A. Swami, “Mining association rules between sets of items in large database,“ *The ACM SIGMOD Conference,* Washington DC, USA, 1993.

[2] R. Agrawal, T. Imielinksi and A. Swami, “Database mining: a performance perspective,” *IEEE Transactions on Knowledge and Data Engineering,* Vol. 5, No. 6, pp. 914-925, 1993.

[3] R. Agrawal and R. Srikant, “Fast algorithm for mining association rules,” *The International Conference on Very Large Data Bases*, pp. 487-499, 1994.

[4] R. Agrawal and R. Srikant, ”Mining sequential patterns,” *In 11th IEEE International Conference on Data Engineering,* 1995.

[5] R. Agrawal, R. Srikant and Q. Vu, “Mining association rules with item constraints,” *The 3th International Conference on Knowledge Discovery in Databases and Data Mining,* Newport Beach, California, 1997.

[6] H. Avancini, A. Lavelli, B. Magnini, F. Sebastiani and R. Zanoli, “Expanding

domain-specific lexicons by term categorization," Proc. of ACM Symposium on Applied Computing, pp. 793 -797, 2003.

[7] C. Buckley and A. F. Lewit, "Optimizations of inverted vector searches," SIGIR '85, pp. 97-110, 1985.

[8] D. R. Cutting, D. R. Karger, J. O. Predersen and J. W. Tukey, "Scatter/Gather: A cluster-based approach to browsing large document collections," Proc. of the Fifteenth Interntional Conference on Research and Development in Information Retrieval, 318-329, 1992.

[9] D. W. Cheung, J. Han, V. Ng and C.Y. Wong, "Maintenance of discovered association rules in large databases: An incremental updating approach," *In 12th IEEE International Conference on Data Engineering,* 1996.

[10] D. W. Cheung, S.D. Lee and B. Kao, "A general incremental technique for maintaining discovered association rules," *In Proceedings of database systems for advanced applications, DASFAA'97,* Melbourne, Australia, pp. 185-194, 1997.

[11] F. Debole and F. Sebastiani, "Supervised term weighting for automated text categorization," Proc. of ACM Symposium on Applied Computing, pp. 784 -788, 2003.

[12] E. H. Han and G. Karypis, " Centroid-Based Document Classification: Analysis Experimental Results," Principles of Data Mining and Knowledge Discovery, 2000.

[13] G. Kowalski, Information Retrieval Systems-Theory and Implementation, Kluwer Academic Publishers, 1997.

[14] X. Long and T. Suel, "Optimized query execution in large search engines with global page ordering," Proc. of the 29th VLDB Conference, 2003

[15] Y. T. Lin, S. S. Tseng and C. J. Tsai, "The Design and Implementation of a Computer-Assisted Learning Expert System", Computer Processing of Oriental Languages: An Internal Journal, Vol. 15, No. 1, pp. 33-61, 2002.

[16] Y. K. Lee, S. J. Yoo, K. Yoon and B. Berra, "Index structures for structured documents," Proc. Digital Library, pp. 91-99, 1996.

[17] H. Mannila, H. Toivonen and A. Inkeri Verkamo, "Efficient algorithm for discovering association rules," *Proceeding AAAI Workshop Knowledge Discovery in Databases*, pp. 181-192, 1994.

[18] J. S. Park, M. S. Chen and P. S. Yu "Using a hash-based method with transaction trimming for mining association rules," *IEEE Transactions on Knowledge and Data Engineering,* Vol. 9, No. 5, pp.812-825, 1997.

[19] D. W. Shin, H. C. Jane and H. L. Jin, "BUS: An effective indexing and retrieval scheme in structured documents," Proc. of Digital Libraries, pp. 235-243, 1998.

[20] M. Steinbach, G. Karypis and V. Kumar, “A Comparison of Document Clustering Techniques,” TextMining Workshop, KDD, 2000.

[21] S. Shankar and G. Karyp, “A feature weight adjustment algorithm for document categorization,” Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2000.

[22] S. Shankar and G. Karypis, “Weight adjustment schemes for a centroid based classifier,” Technical report, Dept. of Computer Science and Engineering, University of Minnesota, 2000.

[23] C. J. Tsai, S. S. Tseng, J. R. Cheng and C. T. Chen, “CDM: A Course Directory Manager on e-Learning Systems,” submitted to Information Science: An Internal Journal.

[24] W. C. Wong and W. C. Fu, “Incremental Document Clustering for Web Page Classification,” The IEEE International Conference on Information Society, 2000.

[25] C. Y. Wang, Y. C. Lei, P. C. Cheng and S. S. Tseng, “A Level-wise Clustering Algorithm on Structured Documents,” accepted by National Computer Symposium (NCS), 2003.

[26] Y. Zhao and G. Karypis, “Evaluation of Hierarchical Clustering Algorithms for Document Datasets,” Technical Report #02-22, 2002.

[27] DocBook, DocBook Technical Committee, http://www.oasis-open.org/docbook/.

[28] Dublin Core Metadate Element Set (DEMES), Dublin Core Metadata Initiative, http://dublincore.org/.

[29] Extensible Markup Language (XML), World Web Consortium (W3C), http://ww.w3.org/TR/.

[30] MPEG-7, Moving Picture Experts Group, http://mpeg.telecomitalialab.com/.

[31] Open eBook Publication Structure (OEBPS), Open eBook Forum, http://www.openebook.org/.