第二章　相關文獻探討

2.1　書目自動索引工具

我們利用一篇著作被引用的次數，作為評估其重要性的指標。目前有一些資料庫提供這一類的計算服務，例如 Institute for Scientific Information(ISI)建立維護的數個書目索引資料庫中，其中的 Science Citation Index(SCI)提供了科學類期刊的書目索引。而 West Group 則提供法律類的書目索引，不過上述幾個資料庫都屬於商業化經營，必須付費才可使用。因為他們在書目資料解析工作上，花費了大量的人力以及金錢。因此書目自動索引工具是比要的。

書目自動索引系統，在書目資料查詢的過程中，必須先將書目資料的後設資料解析出來，方可與結構化書目資料庫作比對。目前已有書目自動索引系統被建立，例如 CiteSeer、Open Journals (OJs)、OpCit, …等，其中 CiteSeer[3][4][5][6][7]主要的工作是要建立一個書目資料庫，而 OJs 和 OpCit 的功能比較接近，它們是要將書目資料連結到 WWW 上的 PostScript 檔案或 PDF 檔案，不過對於連結的目標有點差異，OJs 主要著重在出版商所提供的網站，而 OpCit 則是專注於一些公開的典藏上像是 Los Alamos Eprint Archive(LANL)[8]，以下幾節我們將針對一些相關技術作更深入的探討。

2.1.1 OpCit

OpCit 是一個由 Southampton University (UK)和 Cornell University (USA)共同研發的計畫。OpCit 解析 reference 的方法是使用 Citation.pm 程式[9][10]，這是一個由 Perl 語言寫成的程式。而這個程式一開始設定的任務是要幫忙 Los Alamos 和 CERN 處理物理類的典藏。因為它的程式碼是公開的，可以免費下載測試。2002 年 7 月 12 日 OpCit 將

Citebase 系統公開在網路上徵求使用者的評估。在 12 月 20 日公佈由 200 位使用者評估的結果。為什麼我們會在意這個日期呢？因為我們也是在這個期間使用 OpCit 系統之後，覺得無法達到我們的要求而自行加入研究之列。經過測試之後，在 2003 年 2 月 OpCit 又發表了一個新的工具叫做 ParaTools。之所以會提出這個新工具是因為不少參與 Citebase 評估的使用者認為 Citebase 太專注於物理類的書目資料。為此 OpCit 才又提出這個 general purpose solution。這個新工具完成的時間跟我們差不多，甚至連使用的方法都和我們有異曲同工之妙。Mike Jewell 再著作上[11]指出，該系統使用了一個 Template.pm 的檔案來儲存書目資料結構，它的格式如同下面的結構：

$ParaTools::CiteParser::Template::templates=[‘_AUTHORS_, _PUBLICATION_, _YEAR_, _ISSUE_, _SPAGE_-_EPAGE’,

…

];

以這個檔案來儲存格式樣版，然後將書目資料轉成這個格式，例如像‘Jewell, M (2002) Title’就可以轉換成‘_AUTHIRS_ (_YEAR_) _TITLE_’，以這個新得到的格式在 Template.pm 中對應出最相似的樣版。當然對應過程也是將對應到的欄位作量化積分，數值最大的就是最接近的樣版。如此就可以得知每一個欄位的正確位置，進而將後設資料解析出來。

2.1.2 CERN

CERN Document Server 儲存了 220,000 多筆電子文件。為了方便物理學家使用這些電子文件，它們提出以自動化的方法從書目資料中取出 Internet addresses、report numbers 以及 journal/periodical titles 的資料[12]。 辨認的方法如下：

- Recognizing Internet addresses

Internet addresses 一般都是以“http://”或“ftp://”為開頭因此可以很容易辨認出來。

- Recognizing report numbers

辨識 report numbers 的方法就是找尋有多個連字號(-)的字串。

- Recognizing journal/periodical titles

這是最困難的部分，因為任何格式都可能被使用，唯一的方法就使盡可能收集可能使用的格式。

2.2 基因序列比對工具

由於本系統將書目資料的結構當成蛋白質序列看待。我們把書目資料字串轉成蛋白質序列格式，再以序列比對工具作比對。自從 1970 年代起，科學家開始提出解決序列比對的方法。Needleman 與 Wunsch[13]以 dynamic programming 的方法來分析氨基酸序列的相似度，開啟了基因序列比對的序幕。至今這麼多基因比對的工具，絕大部分都是以它們的方法為基礎。現今較受歡迎的基因序列比對工具有三種，第一種是 Smith-Waterman[14]在 1981 年提出的方法，它可以精確的算出最佳比對。但是因為速度較慢不適合作大量的基因比對運算，所以才有第二種工具 FASTA 的提出。為了彌補 Smith-Waterman 的方法在速度上的不足，Pearson 提出 FASTA 的方法[15]。FASTA 先是以概略的方法找出有興趣的區域，再針對這些區域使用 Smith-Waterman 的方法作細微的比對。所以速度上比 Smith-Waterman 快。而第三種工具是目前最廣為使用的 BLAST（Basic Local Alignment Search Tool）[16]。這是 Altschul 等人在 1990 年提出的方法，不僅提供快速基因序列比對，而且兼顧比對準確度，同時也對結果的品質作

了評估。Shpaer 等人在 1996 年發表的“Sensitivity and Selectivity in Protein Similarity Searches: Comparison of Smith-Waterman in Hardware”[17]一文提出，上述三種比對工具的準確度差異在 5%內，但是 FASTA 在速度上不如 BLAST 快。如果以 BLAST 搜尋基因必須花一分鐘的時間，那麼以 FASTA 來作搜尋的話可能必須花三十分鐘到一個小時，甚至更多。站在時間因素的考量 BLAST 較能符合要求，站在準確率方面考量 FASTA 表現比較好。最後經過取捨我們決定以 BLAST 作為本系統蛋白質序列比對的工具。由於本系統所使用的基因序列長度平均有 14 個 bps 左右，相較於生物學上經常使用的一千多個 bps 實在是小巫見大巫。平均跑一筆資料只需花費三秒鐘，如果不以時間為最重要的考量，而是要求高準確率時，則可將 BLAST 程式裡面“word”參數值設定成 1，作最原始的 dynamic programming。

BLAST 是在 dynamic programming 的程式架構下加速比對的速度。它之所以可以達到這個目的，是因為它在比對之前就先行去除了不可能的序列再針對剩餘的部分作處理。可將 BLAST 的演算方法分成三個步驟。第一步：建立高積分的 word list，所謂 word list 就是依據 word length 將 query sequence 切成固定長度的字。如果 word length 為 w，就將 query sequence 切成 n-w+1 個 w-mer 的字，並且只留下高積分的部分作為 word list。第二步：從資料庫中找出可以和 word list 有數個完全配對的基因序列，我們稱它為 hit。在尋找 hit 的方法可以使用 lookup table，或者是使用 finite state machine。第三步：將 hit 延伸使它成為 MSP，而且積分至少為 S，所謂 MSP(Maximal Segment Pair)就是兩個序列中，積分最高的相同長度片段。所以整個演算法的時間複雜度為：

$$aW + bN + cNW / 20^{\omega}$$

其中 W 為 word list 的大小，N 為資料庫中 residue 的個數，a、b、c

為常數 w 為 word length。