陳林志、林育任（2013），『個人化的網頁摘要文件分群系統』，
資訊管理學報，第二十卷，第一期，頁97-130。

個人化的網頁摘要文件分群系統

陳林志*
國立東華大學資訊管理學系

林育任
國立東華大學資訊管理學系

摘要

本論文發展了一套具有分群能力之個人化系統，Personalization Web-Snippet Clustering System（PWSC），此系統是基於元搜尋技術。此系統的第一階段根據使用者所輸入之查詢，針對不同搜尋引擎匯集相關網頁摘要文件。第二階段，透過Mean Reciprocal Rank（MRR）計算模型重新排列網頁摘要文件。第三階段，將收集到的網頁摘要文件，經由 N 字詞語言模型產生分群標籤。第四階段，依據分群標籤建構出階層式分群。最後階段為建立個人化系統，其能依據使用者所選擇的標籤及運算，產生不同的搜尋結果，這樣將能幫助使用者快速尋找想要的資訊。根據實驗結果，本系統的性能優於商業和學術系統。

關鍵詞：網頁摘要文件分群、個人化搜尋引擎、階層式分群、分群標籤、元搜尋技術

* 本文通訊作者。電子郵件信箱：lcchen@mail.ndhu.edu.tw
2011/09/11 投稿；2011/12/11 第一次修訂；2012/06/15 第二次修訂；2012/09/19 接受

Chen, L.C. and Lin, Y.R. (2003), 'A Personal Search System with the Clustering Ability',
Journal of Information Management, Vol. 20, No. 1, pp. 97-130.

A Personal Search System with the Clustering Ability

Lin-Chih Chen*
Department of Information Management, National Dong Hwa University

Yu-Ren Lin
Department of Information Management, National Dong Hwa University

Abstract

In this paper, we develop a personal search system with the clustering ability, called Personalization Web-Snippet Clustering System (PWSC) that is based on a Metasearch technique. The first stage of the system is to collect the relevant snippets from different search engines based on the user's query. The second stage is to rearrange the weight of the collected snippets based on a Mean Reciprocal Rank (MRR) measure. The third stage is to use word N-gram for language model to generate the clustering labels from our collected snippets. The fourth stage is to build a hierarchical tree based on all clustering labels. The final stage is to build a personal search system by the user to select some of the most interesting labels and operations to help the user quickly locate information of interest. According to all experiment results, the performance of our system is superior to the commercial and academic systems.

Keywords: Web-Snippet Clustering, Personal Search Engine, Hierarchical Clustering, Clustering Label, Metasearch Technique

* Corresponding author. Email: lcchen@mail.ndhu.edu.tw
2011/09/11 received; 2011/12/11 1st revised; 2012/06/15 2nd revised; 2012/09/19 accepted

壹、導論

隨著科技日新月異的變化，資訊產業進入了一個前所未有的蓬勃發展，World Wide Web 已經成為一個全世界共同分享與交流的虛擬世界，而隨著電腦與網路的迅速發展，當今社會已經走向數位化，人們找尋資料的習慣已不再是透過實體紙本或圖書館中取得，而是越來越習慣於透過網路這個豐富且實用的平台來獲取自己需要的資訊。現今在網際網路快速發展下，全世界的網頁已無法盡數，因此要在這樣龐大的網路世界中找尋使用者所關注的資訊，作為現代資料檢索（Information Retrieval, IR）技術的主要應用，搜尋引擎（Search Engine）變成不可或缺的工具。

搜尋引擎是利用使用者輸入查詢字串來搜索網路上相關的網站，目前較為人熟知的有 Google、Yahoo 和 Bing 等等。根據調研機構 comScore 公司所公佈的數據，2011 年 5 月份 Google 在美國搜索的市占率為 65.5%、Yahoo 占 15.9%、Bing 占 14.1%（comScore 2011）。針對使用者而言，搜尋引擎雖然在一定程度上滿足了人們的搜尋要求，但是回傳的大量資訊也同時在考驗著人們的耐性。如何使搜尋引擎理解使用者的查詢需求，以便讓搜尋引擎回傳的結果更準確，同時不包含太多的無用資訊，是現今搜尋引擎新的課題及挑戰（Maxymuk 2008）。其解決問題的根本辦法在於如何使用自然語言處理（Natural Language Processing, NLP）技術，使搜尋引擎瞭解使用者真正的搜尋問題（Horowitz & Kamvar 2010），因此本研究也使用了某些 NLP 以便讓搜尋結果更符合使用者需求。

搜尋引擎技術源自於 IR 領域，是目前應用最廣泛的資訊技術，搜尋引擎的出現部分地解決了 Web 資訊檢索所帶來的問題。一般而言，使用者使用搜尋引擎查詢 Web 資訊時，返回的查詢結果一般是包含一組網頁摘要文件（Web-Snippet），所謂的網頁摘要文件，通常包含網頁的標題文字（Title）、精簡的網頁介紹或網站簡介（Snippet）、以及網頁超連結位址（URL），使用者可藉由此類的摘要式文件，簡略地了解該網頁的內容（Ferragina & Guli 2008）。

但是，在我們使用搜尋引擎時會發現，輸入一個查詢，搜尋出來的網頁動輒上千萬筆，使用者想經由這些網頁之中，透過瀏覽的方式尋找所需資訊，也不是一件容易的事，更有研究顯示有 77%的使用者在使用搜尋引擎時，只會輸入一個英文單字，同時有 85%的使用者只會觀看第一頁的搜尋結果（Google 2012）。隨著 Web 網頁的迅速增加，現有的搜尋引擎越來越不能滿足人們的需要，搜尋引擎的改進已成為近年來網際網路領域研究的新焦點。

網際網路之中最重要的搜尋工具就是搜尋引擎，雖然經過數十年的發展，搜尋引擎有了明顯的成長，但目前仍因諸多原因尚未改善，導致搜尋引擎尚有大幅

改進的空間。傳統搜尋引擎具有以下幾個常見的問題（Carpineto et al. 2009b）：

1. 回傳資料量大但相關度低：搜尋引擎回傳結果通常有上千萬筆，這容易讓使用者淹沒在過多的資訊中。
2. 列表呈現搜尋結果：搜尋引擎往往只將查詢相關的網頁以列表方式呈現，然而列表的結果並無進行任何後續分類，這樣讓使用者在判讀上容易造成困難。
3. 搜尋結果種類繁多但無統整性：各種類型的網頁文件混雜在同一個搜尋頁面，容易造成使用者不易閱讀及判斷。
4. 缺乏個人化特性：使用者具有不同的背景、興趣及使用目的，依照使用者的需要對搜尋結果進行最佳化安排是一件很重要的工作。然而傳統搜尋引擎並不能針對個別使用者做出不同的搜尋結果。

一個解決傳統搜尋引擎問題的方法為應用分群技術於搜尋結果（Carpineto et al. 2009b），其利用相關文件分群技術對搜尋結果進行適當分群，將搜尋結果按照文件內容分成不同類別，並貼上分群標籤，以方便用戶的搜索，此種系統一般稱之為網頁摘要文件分群系統（Web-Snippet Clustering System），網頁摘要文件分群系統主要是基於多個搜尋引擎結果的後續利用（即元搜尋技術），此技術主要就是要改善使用單一搜尋引擎可能產生之資訊檢索效能問題（Weiss & Stefanowski 2003）。

分群的主要目的是將一個複雜的資料集分成不同的群集，而群集內的相似度要越高越好，群集間的相異度要越高越好（Huang et al. 2005）。現今常用的分群方法主要分成階層式（Hierarchical）與分割式（Partitional）。階層式分群還可再分成凝聚式（Agglomerative）跟分裂式（Divisive）兩種類型（Garai & Chaudhuri 2004），凝聚式分群主要是由下而上（Bottom-Up），從底層開始將相類似的群集合併，直到全部資料都已經形成群集，而分裂式分群主要是由上而下（Top-Down），將全部的資料先視為同一群集，接著將全部資料由上而下一層一層進行分裂，直到全部資料都已經分入群集。分割式分群主要是藉由切割空間的觀念來歸類資料，將資料空間分割成數個大小不一的子空間，在同一個子空間內的物件則被視為同一群集，接著再找尋各群集之形心點（Centroid），利用各資料間距離的遠近達成分群效果，其中最有名的分割式分群方法就是 K-means（MacQueen 1967）。

網頁摘要文件分群系統架構在元搜尋技術之上，根據使用者輸入之查詢，向多個搜尋引擎發出查詢需求，進而收集網頁摘要文件，接下來再根據文件內所包含的語意經由適當的 NLP 處理後，產生各個分群群集，並由群集中最具代表性文字顯示該群集，進而提供給使用者選擇之用，其中比較知名的系統如：Carrot2 及 Vivisimo 等。

個人化搜尋，也是近幾年熱門的資訊檢索方法，一般有下列兩種處理方式：(1) 收集使用者在網路上所進行的查詢和點閱紀錄，以供使用者未來的查詢和建議，

但此種方式需要儲存大量個人化查詢紀錄，同時對於個人隱私也是一個很大的問題，例如：Google 個人化搜尋就是按照此方法處理；(2)不需儲存個人化查詢紀錄，採用適當的 NLP 技術對使用者查詢進行處理，並提供與查詢相關之主題給使用者進行勾選，經由使由者組合勾選後，即產生不同的搜尋結果，以此達成個人化搜尋需求，此種方式可以避免儲存大量個人化查詢紀錄及個人隱私問題，例如：SnakeT 個人化搜尋就是按照此方法處理。

本研究目的在於設計出個人化的階層式分群演算法，對網頁摘要文件進行分群，並以階層式的方式呈現。階層之標籤將從網頁摘要文件中應用 N 字詞（N-gram）語言模式獲取，進而提高標籤擷取的彈性和提高階層的精確度。最後，我們使用勾選標籤的方式並組合不同的二元運算，讓使用者選取出自己感興趣的個人化主題，以此用來協助使用者更快速地搜尋資訊。

本論文具有下列三點主要貢獻：多重分群、快速建立階層樹、快速的運行不同二元運算。(1)多重分群：以文件觀點而言，文件可能包含數個主題，因此好的分群方法必須能夠將文件適當分配到數個主題之中；另一方面，以群集觀點而言，子群集可能被包含在數個父群集之中，良好的分群方法必須能夠處理文件及群集間所存在的多重分群特性。(2)快速建立階層樹：一個好的分群方法所需執行的時間必須與文件數目成線性關係。(3)快速的運行不同二元運算：由於我們儲存每個主題所擁有的文件是採用二元編碼的形式所儲存，因此我們可以透過不同的位元運算達成不同的二元運算，亦即使用者可以透過二元運算達成不同的個人化搜尋需求。相關的說明，請參照第肆節。

在我們論文後續部份，我們先討論與本論文有關之文獻，接下來討論 PWSC 之系統架構，然後探討相關實驗數據，最後以未來研究方向總結本文。

貳、文獻探討

一、網頁摘要文件分群系統概述

網頁摘要文件分群系統架構在元搜尋技術之上，依照使用者輸入之查詢，同時向多個搜尋引擎發出查詢需求，並由回傳之網頁摘要文件進行後續分群，這種分群主要依據查詢建立不同的搜尋主題，使用者在閱讀上較容易集中在某一特定主題。有相關研究指出，網頁摘要文件分群系統，處理一詞多義（Polysemous）、以及品質不佳（Poor）的查詢時特別有效（Frantzi et al. 2000）。

Northern Light 是第一個將網頁文件進行分群的系統，它是在 20 世紀所建立的，它首先設定一些類別，並將搜尋結果按這些類別進行分配，可想而知這樣的做法將分群結果限制在那些預定類別之上，針對未知類別，這樣的做法將是一個很大的問題（Ferragina & Guli 2008）。

目前最有名的學術及商業系統分別為 Carrot2（2011）及 Vivisimo（2011）。Carrot2 有兩個實作系統，分別為 Carrot2-STC 及 Carrot2-Lingo。其中 Carrot2-STC 使用 STC（Suffix Tree Clustering）演算法進行分群，該演算法首先將相關詞彙建立一個 Suffix Tree，經由樹的追蹤建立相關分群標籤，並經由樣式比對（Pattern Match）的方式將文件進行歸類，分群權重則是由 TFIDF 模型進行評估；Carrot2-Lingo 使用 Lingo 演算法進行分群，該演算法採用奇異值分解（Singular Value Decomposition, SVD）產生分群標籤，並再運用向量空間模式（Vector Space Model, VSM）將文件進行歸類（Osinski & Weiss 2005），分群權重同樣採用 TFIDF 模型進行評估。然而，Carrot2 的兩個實作系統都是採用平面式（Flat）分群，而非階層式（Hierarchical）分群。平面式分群必須預先定義相關分群，而階層式分群則不需做此工作。

Vivisimo 採取階層式的方式進行分群，但 Vivisimo 缺點在於，其為商業化系統，一般而言，研究者很難取得其相關技術進行實作，使用者在沒付費的情況下很難使用其提供的服務。不過 Vivisimo 也有提供一個測試網站 Yippy（2011），其採用了簡潔的概念分析，以找出所有符合的相關資訊（Segev et al. 2007）。

其它商業化系統還有 iBoogie（2011）及 WebClust（2011），它們本身都有提供測試網站，但相關商業系統皆無提供個人化搜尋的服務。在網頁摘要文件分群系統的發展上，若能在分群結果加上個人化搜尋的特性，將協助使用者更快速地取得相關資訊。

二、現今網頁摘要文件分群系統整理

網頁摘要文件分群系統依照建立者為區別，可分為商業及學術系統。這兩種建立者所建置的系統在本質上有很大的不同，以商業系統而言，一般都會由一個團隊或公司來經營，由於是營利事業，網站必須保持系統的更新和運作，是故這類系統大多數都還存在；然而，學術系統一般都只存活一段時間，其主要原因在於此類系統非以營利為目的，主要都是為了研究目的或其他非營利之用途，缺乏專人維護。

一般而言，學術系統是採用一個已知的網頁擷取技術（一般稱之為 Web Crawler）蒐集相關資料並進行後續分群，因此研究人員通常可以較簡單的建立系統，而且有些學術系統會有一些開放原始碼供研究人員交流使用並進行改善。然而，商業系統則是採用了封閉原始碼的方法進行分群，故此研究人員無法深入了解相關運作及流程（Chen 2011）。

因為無法確切得知商業系統的細節，故本研究僅整理一些學術系統。一般而言，我們可以依照分群後產生的標籤形式及呈現方式不同，將分群系統分為四大

類，情形如表 1 所示，茲分述如下。

表 1：網頁摘要文件分群學術系統整理

標籤形式 / 呈現方式	單一字詞	一段句子
平面式分群	Scatter/Gather（Dead） WebCat（Dead）	Grouper（Dead） Carrot2-STC Carrot2-Lingo
階層式分群	FIHC（Dead） CREDO	SnakeT Highlight（Dead） PWSC

（一）單一字詞且平面式分群

Scatter/Gather（Hearst & Pedersen 1996）為第一個在學術上所建立的網頁摘要文件分群系統（Benson 1989），其提供相關分群結果以及不同群集間的摘要描述，以便協助使用者瀏覽資訊。它是第一個使用機器進行分群，其依據文件彼此間相似度是否達到一定程度，進而判斷是否需產生新群集（Hearst & Pedersen 1996）。

WebCat（Giannotti et al. 2003）採用 Transactional K-Means 演算法（Giannotti et al. 2002）進行摘要文件分群處理，Transactional K-Means 是以傳統 K-Means 演算法為基礎進行演化，因此，WebCat 屬於分割式（Partitioning）分群演算法。該演算法有兩個主要階段：(1)它計算 k＋1 的群集，根據距離測量後將資料分配到群集，並將未分配到前 k 個群集內的元素放置在第 k＋1 的暫存群集之中；(2)主要目標就是如何管理暫存群集，在這個階段的主要想法是嘗試以遞迴分割暫存群集，直到他們沒有被選擇在前 k 個群集內，當然最後有可能會沒有任何一個資料被劃分出去（Giannotti 2002）。上述兩個系統皆已不存在網路。

（二）一段句子且平面式分群

Grouper 運用 STC 演算法進行分群，其分群過程包含三個步驟：文件的前置處理、辨識基底群集（Base Cluster）、組合基底群集（Zamir & Etzioni 1999）。第一階段主要是將文件做清理的工作，就是去除所有非字標籤（去除如數字、HTML 標籤、以及標點符號）；第二階段主要是依據相關詞彙建立一個 Suffix Tree，並經由樹的追蹤產生基底群集；第三階段主要是透過樣式比對，尋找基底群集之中擁有共同字詞或句子當成群集標籤。Grouper 使用二元相似度度量指標（Binary Similarity Measure）（Zamir & Etzioni 1998）測試兩個群集標籤間彼此重疊程度，並以此當為排序基準。Carrot2-STC（Weiss et al. 2003）是使用 Grouper 的方法實作而成。

Carrot2-Lingo 使用奇異值分解產生分群標籤，並再運用向量空間模式將文件進行歸類（Osinski & Weiss 2005），分群權重採用 TFIDF 模型進行評估。然而這樣的做法存在一個很大的問題，即因為奇異值分解是非常耗時，所以處理的資料量將有限。

（三）單一字詞且階層式分群

FIHC（Fung et al. 2003）應用頻繁項目集（Frequent Itemset-based）的概念執行文件分群。相較於分割為基礎的文件分群方法，以文件為中心，將文件間相似度作為分群的決策依據，FIHC 則以群集為中心，度量群集內聚合度，並以聚合度作為分群依據，聚合度要求在相同主題群集中的文件，比不同主題群集中的文件，應擁有更多相同的項目集。

CREDO（Carpineto et al. 2004）採用了形式概念分析（Formal Concept Analysis），而它是依據單一字詞建立一個分群網路，並以此產生一個導航樹（Navigation Tree）給使用者瀏覽之用。其建立導航樹分成兩個步驟：第一個步驟，根據摘要文件的標題產生最基礎標籤；第二個步驟，以形式概念分析的方法將相似度高的低層基礎標籤進行組合，並持續執行這個步驟，直到導航樹建立完成（Carpineto et al. 2009b）。

（四）一段句子且階層式分群

SnakeT（Ferragina & Guli 2008）為一個整合型的個人化分群系統。其個人化的特點在於透過不儲存個人化查詢紀錄，利用使用者點擊不同分群並依據分群標籤間之 TFIDF 權重值產生排序依據，據此達到個人化的特性。其擷取階層標籤的方式，是依據文件中出現的斷裂句（Gapped Sentences）所產生，斷裂句是藉由不斷的合併鄰近文字所產生。藉由此方式所產生的標籤，擁有較高的彈性。每一個斷裂句的排名順序，依所有組成文字配對所產生排名順序決定而成，並留下較佳的斷裂句當成分群標籤。

針對常見分群標籤，它採用了 DMOZ（2011）知識庫進行排序。DMOZ（也稱為開放式目錄）是全世界最大的開放化網頁目錄編輯系統，其透過人工編輯的方式將相關目錄依層次編排，而每個網站或網頁都可以被添加到相關目錄。SnakeT 使用由下而上的方式建立階層樹。

Highlight（Wu & Chen 2003）首先使用概念術語（一些出現在文件之中的名詞片語）產生分群標籤；接下來，它使用一個 PCA（Probability of Co-occurrence Analysis）（Wu et al. 2002）技術分析標籤之間存在之高低層的關係，進而產生階層式分群。其特點在於其使用機率的形式描述標籤之間的關係，而非傳統的詞頻的方式，這樣的好處在於標籤間的關聯程度可以更清楚的表達。

本研究所提出的 PWSC 系統在標籤形式屬於一段句子，在分群呈現方式採用

階層式分群。我們首先採用 N 字詞語言模式產生具有句子形式之分群標籤；接下來，我們發展一套由上而下的階層式分群技術，將分群標籤以階層樹的形式呈現。在第三章中，我們將詳細描述我們的系統。

三、個人化搜尋相關研究

依據眾多學者（Chen 2011; Ferragina & Guli 2008; Jeh & Widom 2003; Speretta & Gauch 2005; Sun et al. 2005; Wu et al. 2003）在個人化搜尋所做的相關研究，此研究可以採用存入及不存入使用者網路瀏覽記錄兩種處理方式，表 2 說明此兩種處理方法的比較。傳統上，個人化搜尋的處理方式大都是以存入使用者的一些網路瀏覽記錄，並以此記錄進行相關建議搜尋，如 Google（2012）及 Yahoo（2012）允許使用者瀏覽自己過去的搜尋記錄，同時儲存每一筆搜尋記錄及其所對應的搜索結果；使用者可以根據這些搜尋結果進行個人化調整，並推薦使用者其他建議搜尋結果。

此種存入使用者瀏覽記錄依採用分析的方法可以區分為非自動化及自動化分析（Speretta & Gauch 2005; Sun et al. 2005）。非自動化分析需要使用者自行輸入相關資料才能建檔。例如在早期，一些網站在會員註冊時，會要求使用者輸入許多個人相關資訊（如興趣或專長等等），這些舉動都會造成使用者額外的負荷，而且有可能使用者本身並無法很清楚的描述自己的需求。現行商業化搜尋引擎，如 Google 及 Yahoo 是依據自動化的方式分析使用者所存入的瀏覽記錄，其它如 Sun 等（2005）學者也是採用自動化分析的方法，該方法主要依據使用者歷史查詢及使用者點擊網頁進行分析，分析方法採用奇異值分解（Singular Value Decomposition, SVD）找出使用者可能尋找之潛在網頁。然而，此種方式的缺點在於當歷史記錄及點擊網頁過多時，奇異值分解所需的時間將是非常耗時，因此非常不利於應用具有即時回應需求之搜尋引擎（Ferragina & Guli 2008）之上。

另一方面，存入使用者記錄除了需要儲存大量的使用者搜尋記錄及點擊記錄等資訊；同時，系統需要持續的進行分析才能找出使用者所感興趣之潛在網頁，當記錄龐大時，所需的時間將非常可觀；最後，使用者需要先透過註冊的方式確立其身分，並且要做登錄動作後，瀏覽記錄才會開始被儲存，這些動作對使用者而言都是具有一定程度的不便利性。最重要的，上述儲存的所有個人化資訊，都具有一定程度的違反隱私權（Ferragina & Guli 2008）。

為了解決傳統採用存入使用者瀏覽記錄所產生的一系列問題，相關學者（Chen 2011; Ferragina & Guli 2008; Wu et al. 2003）建議採用不存入使用者瀏覽記錄的方式達成個人化搜尋，該方式主要採用分群的方式達成個人化搜尋。使用者可以依據分群所產生之不同主題進行選取，同時系統會依據使用者所選取之主題顯示相

關的網頁。現行學術界最有名的系統是 SnakeT（Ferragina & Guli 2008），其主要先依所收集之網頁摘要文件進行適當分群，進而產生一序列之主題，並對每個主題產生一個核取方框，以便使用者進行不同主題之間的勾選，SnakeT 與其它採用分群方法（如 WSC（Chen 2011）與 Highlight（Wu et al. 2003））比較，其所建立之個人化搜尋系統具有同時選取多個主題的能力，亦即使用者可以針對不同主題進行 OR 運算。然而，除了多主題之間的 OR 運算，實務上使用者可能針對不同主題進行其它的二元運算。透過我們所提供的不同二元運算之個人化選項，系統可自動幫使用者進行篩選。理論上，我們可以提供任何可能的二元運算。

表 2：個人化搜尋相關整理

使用者瀏覽記錄	不存入	存入
優點	無須註冊 無隱私權的問題 結果為動態產生 無紀錄所以無資料龐大問題	結果較貼近使用者
缺點	結果可能為使用者較不感興趣之內容	需先註冊 有隱私權的問題 紀錄的資料龐大 分析的時間過長 無法動態產生
例如	SnakeT，WSC，Highlight，PWSC	Google，Yahoo，Sun

參、系統架構

本研究建立了一套具有分群能力之個人化系統，稱之為 PWSC，以達成本研究實際測試與獲得實驗數據之目的，圖 1 為 PWSC 整個系統的架構圖，其主要包含下列幾個主要處理工作：(1)前置處理與部份資訊瀏覽、(2)自然語言處理以及建立候選分群標籤、(3)使用二元編碼（Binary Code）建立階層式分群、(4)個人化建置。

在我們系統的第一個步驟主要是使用 Web Crawler 技術（Chen & Luh 2005; Jansen et al. 2007）收集 HTML 文件，並透過適當的規則文法（Hazel 2012）取出網頁摘要文件，最後經由 MRR 的計算模式（Baeza-Yates & Ribeiro-Neto 1999）產生網頁彙總結果。第二個步驟針對網頁摘要文件首先使用一系列的 NLP 處理（Liddy 2001）產生乾淨詞語（Clean Term）；接著，我們使用 N 字詞（Manning & Schuetze 1999）產生不同形式的樣式標籤；最後，我們針對樣式標籤中，採用樣式

比對及門檻值設定的方式產生候選分群標籤。第三個步驟的處理，我們首先使用 Inverted matrix（Rijsbergen 1979）的儲存結構，依據候選分群標籤及其對應之網頁摘要文件所存在之關係建立二元編碼；接著，我們針對所建立之二元編碼，使用集合觀念判斷並配合由上而下之分群法，用以安排所有候選分群標籤分配於階層樹之上。第四個步驟，主要針對所產生之候選分群標籤，產生對應之核取方框，並運用不同二元運算，可以方便使用者組合使用，由於候選分群標籤是採用二元編碼，因此理論上，我們可以處理所有可能二元運算。

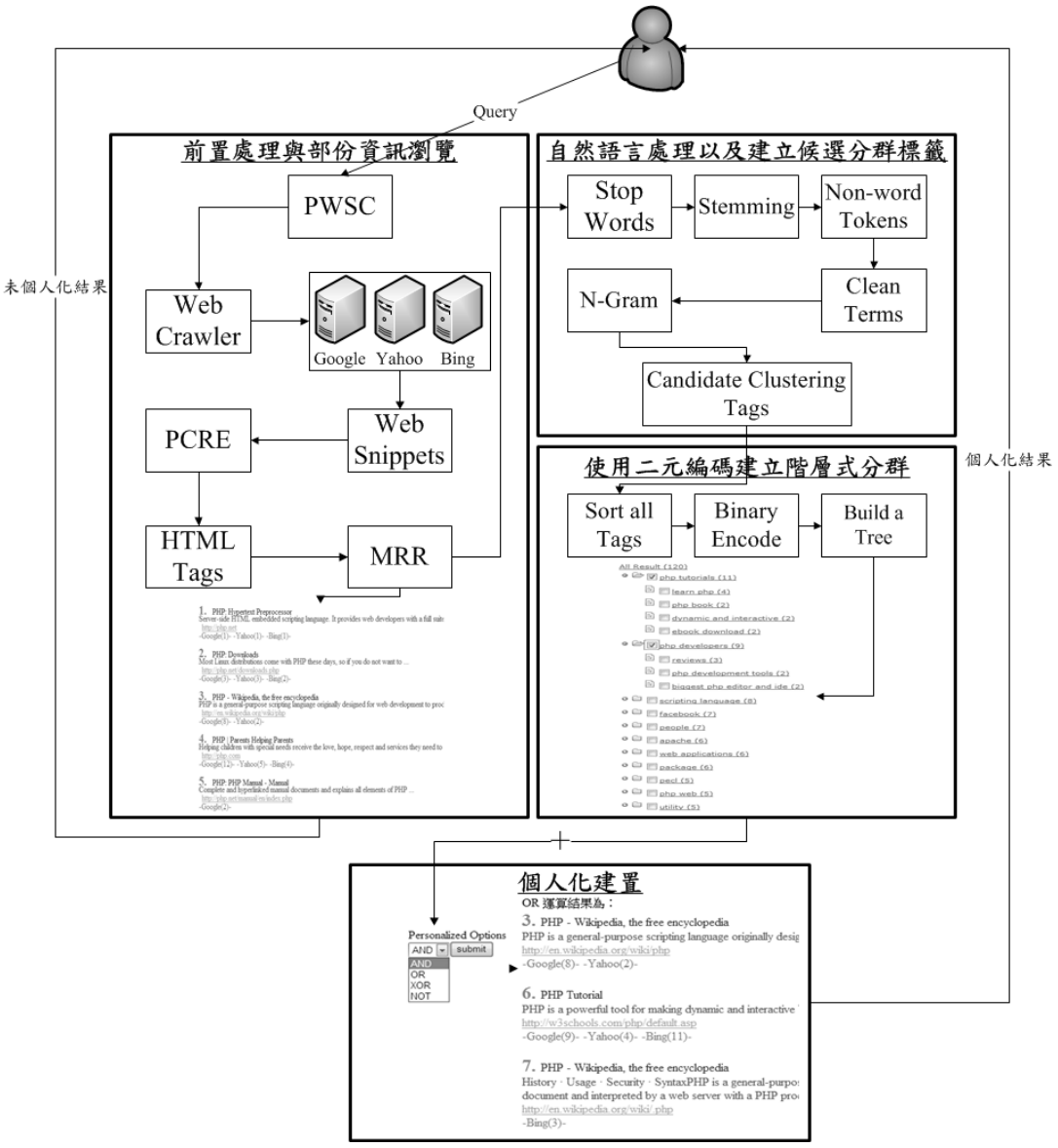

圖 1：PWSC 系統架構

一、前置處理與部份資訊瀏覽

PWSC 首先透過我們自行發展之網頁爬蟲（Web Crawler）技術取得網頁摘要文件，而爬蟲的資料來源為 Google、Yahoo、Bing 三大搜尋引擎，我們對每個搜尋引擎，取回 50 個網頁摘要文件當成我們的處理來源。由於搜尋引擎回傳的結果為 HTML 文件，此文件格式為非結構化文件（Hashemi et al. 2002），因此我們必須將 HTML 文件轉換成可後續處理的資料格式，本研究利用 Perl Compatible Regular Expressions（PCRE）語言表示式（Hazel 2012）達成轉換工作，它是由一組函數執行規則文法來達成 HTML 文件的分析動作，以便擷取 HTML 文件中的 Title、URL 和 Snippet。

接下來，我們將 PCRE 處理後的結果去除掉多餘的 HTML 標籤，因為這些標籤只是純粹的標籤符號，對使用者閱讀而言是無意義且也會對後續的 NLP 處理造成誤判。上述步驟完成後，我們接下來採用 Mean Reciprocal Rank（MRR）計算模式（Baeza-Yates & Ribeiro-Neto 1999）達成彙總工作。最後，即可依照彙總後結果，將部份結果呈現給使用者瀏覽，這是為了讓使用者減少等待最後分群結果，而且有研究顯示，使用者會等待頁面回應的時間為 2 秒（Nah 2004），所以先給使用者一些資訊觀看後，讓系統繼續執行後續分群的動作，比較會讓使用者較無等待的感覺。

採用 MRR 的原因就是要將多個搜尋引擎回傳之網頁摘要文件產生一致性且經由投票後的結果給使用者瀏覽，這樣的做法不會將搜尋結果偏向（Bias）至任一搜尋引擎。MRR 的精神在於計算網頁 w 在不同搜尋引擎的平均排名，其公式如下所示，其中 SE 為我們使用的搜尋引擎個數（在現行系統上，我們的 SE 為 3），而 $rank_{w,i}$ 為網頁 w 在搜尋引擎 i 的排名：

$$MRR_w = \frac{1}{SE}\sum_{i=1}^{SE}\frac{1}{rank_{w,i}} \tag{1}$$

接下來就是將 MRR 的結果由大到小排序，並產生適當的頁面結果（如圖 1 之 MRR 輸出）。

二、自然語言處理以及建立候選分群標籤

此階段主要是產生候選分群標籤，首先我們將擷取回來之所有網頁摘要文件經由適當的 NLP 處理後產生乾淨詞語（Clean Term）。接下來，再運用 N 字詞的方法產生不同樣式（*Patterns*）。最後，我們運用樣式比對及門檻值設定的方式產生候選分群標籤。

本研究採用的 NLP 處理包含下列三項工作：停用字處理（Stop Words）、字根

處理（Stemming）、非字標籤處理（Non-word Tokens）。停用字處理主要是去掉所有停用字，因為這些停用字對整個分群處理是沒有意義，我們採用的停用字主要是以 Fox（1989）所提出的 421 組停用字為基礎，並自行擴充產生足夠之停用字字庫。字根處理主要是將屬於同一字根的單字歸為同一類，比方：move、moved、moving 皆為同一字根，若視這三個單字為不同單字，則對分群處理是沒意義，我們採用的字根處理是依照 Porter 與 Boulton（2007）字根演算法進行處理。非字標籤處理（例如：純數字、HTML 標籤和標點符號）主要是將不屬於單字的符號去除，以產生較有意義的乾淨詞語。

接下來，我們再對乾淨詞語進行 N-字詞處理，據此產生不同樣式。N-字詞處理是為了讓分群標籤產生的更加精確的結果，因為描述詳細的一個句子會比單一個字詞來的更讓使用者了解（Brown et al. 1992）。經由 N-字詞處理所產生之不同樣式結果，為我們所有可能候選分群標籤，最後再針對不同樣式結果經由門檻值篩選（如後述）的方式，以決定最終使用之分群標籤。當我們原始文件為“divisive hierarchical clustering algorithm”，經由上述 NLP 處理後，所得到的乾淨詞語為“divis hierarch cluster algorithm”，表 3 顯示所有 N-字詞的不同樣式結果（N=1~4）。

表 3：N-字詞的不同樣式結果

N	結果
1	divis、hierarch、cluster、algorithm
2	divis hierarch、hierarch cluster、cluster algorithm
3	divis hierarch cluster、hierarch cluster algorithm
4	divis hierarch cluster algorithm

最後再使用樣式比對及門檻值設定的方式，找出不同樣式中最常出現的幾個樣式當成候選分群標籤。圖 2 列出 N 字詞處理及產生候選分群標籤的演算法。

```
1   Algorithm GenClusterLabels {
2   Input: {CleanTerm} （乾淨詞語） / Output: {NewPatterns}（候選分群標籤）
4       Use N-gram language model to generate different patterns {Patterns} for
        {CleanTerm};
14      //Pattern match
15      Sort {Patterns} by the number of words containing each pattern
        descending, and store the sorted results in {NewPatterns};
16      Foreach ({NewPatterns} as k) {
17        Foreach ({NewPatterns-k} as l) {
18          If (words for l is subset of words of k) {
19            #ChildNodes = the number of documents in k;
20            #ParentNodes = the number of documents in l;
21            Rate = #ChildNodes / #ParentNodes;
22            If (Rate ≥ Threshold)
23          Remove l from {NewPatterns};
24        } End of If;
25      } End of Foreach;
26    } End of Foreach;
27  } End of Algorithm;
```

圖 2：N 字詞處理及產生候選分群標籤的演算法

樣式比對的方式如下：假設我們蒐集的網頁摘要文件其中三個包含了下列 *Patterns*：“A B C”、 “B C”、“B C A”。經由 N-字詞處理後，我們可能產生的候選分群標籤為：A、B、C、A B、B C、A B C（B C A 我們視其為 A B C 的不同文字排列，因此我們只會保留 A B C 的結果）。若不做後續處理，我們將有可觀的候選分群標籤，為了減少分群標籤的數量，我們使用門檻值繼續進行篩選。設置門檻值的目的是為減少分群標籤的個數，以避免使用者資料過載的情形，門檻值是用來判斷兩個分群標籤 A B 及 A B C（其中 A、B、C 皆為英文字詞）同時存在且皆為候選分群標籤時，系統判斷是否只需保留 A B C 即可，其主要原因在於使用者對於長字詞會比較感興趣，然而我們不能單單只判斷兩個標籤的字詞長度，還需考慮長字詞標籤（即 A B C）的文件個數是否佔了短字詞標籤（即 A B）的一定比例（此即為門檻值），若是則只需保留長字詞標籤。本系統的文件交集率公式如下所示，其中 *#ParentNodes* 為父節點（即短字詞標籤）的文件個數，*#ChildNodes* 為子節點（即長字詞標籤）的文件個數：

$$Rate = \frac{\#ChildNodes}{\#ParentNodes} \tag{2}$$

假設我們蒐集的網頁文件有十個出現ＡＢ樣式（或其不同文字排列），八個出現ＡＢＣ樣式（或其不同文字排列），依上述公式計算後，文件交集率為 0.8。假設我們所設定的門檻值為 0.8 時，且子節點擁有較長且相似的字詞，因此我們只保留ＡＢＣ的分群標籤。

三、使用二元編碼建立階層式分群

本研究提出一個新的階層式分群，本分群方法運用二元編碼建立一個由上而下的階層樹，其演算法如下圖所示：

```
1   Algorithm HierarchicalClustering {
2   Input: {NewPatterns} (候選分群標籤) / Output: {Tree}（階層樹）
4       Sort all patterns in {NewPatterns} by the number of snippets containing each
        pattern descending, and store the sorted results in {SortTags};
5       Build a 2D matrix BinaryCode where BinaryCode[j,k] = 1 if tag j appears in
        snippet k, otherwise BinaryCode[j,k] = 0;
6       //Build a Tree
7       Foreach ({SortTags} as l) {
8           Foreach ({SortTags-l} as m) {
9               If((BinaryCode for m is subset of BinaryCode for l) and (BinaryCode
                for m is not subset of BinaryCode for other l's children))
10                  Mark m as l's child in {Tree};
11          } End of Foreach;
12      } End of Foreach;
13  } End of Algorithm;
```

圖 3：使用二元編碼方式建立階層樹之演算法

上述演算法，主要有三個重要步驟：(1)Sort all Tags：排序所有的候選分群標籤（以下簡稱標籤）；(2)Binary Encode：轉換已排序之標籤為二元編碼型態；(3)Build a Tree：尋找每個標籤之子分群。接下來，我們以一個例子說明整個建立流程。

第一個步驟的主要工作是根據每個標籤所擁有之文件個數進行遞減排序。例如以表 4 為例，我們假設標籤 Tag_4 有下列文件{T1, T2, T4, T5}，因此這個標籤擁

有四份文件，其餘標籤也在這個表格一併顯示。在這一個步驟之中，我們針對所有標籤按照其所擁有的文件個數進行遞減排序，其結果如表 5 所示。

表 4：排序前的文件情形

標籤	包含文件	文件個數
Tag_1	{T1, T2, T3, T4, T5}	5
Tag_2	{T1, T5}	2
Tag_3	{T1, T3, T5}	3
Tag_4	{T1, T2, T4, T5}	4
Tag_5	{T1, T3, T4, T5}	4
Tag_6	{T2, T5}	2
Tag_7	{T3, T4, T5}	3

表 5：排序後的文件情形及二元編碼

標籤	包含文件	文件個數	二元編碼
Tag_1	{T1, T2, T3, T4, T5}	5	11111
Tag_4	{T1, T2, T4, T5}	4	11011
Tag_5	{T1, T3, T4, T5}	4	10111
Tag_3	{T1, T3, T5}	3	10101
Tag_7	{T3, T4, T5}	3	00111
Tag_2	{T1, T5}	2	10001
Tag_6	{T2, T5}	2	01001

第二個步驟的主要工作是將標籤所擁有的文件轉換為二元編碼型態。我們定義二元編碼中，第 i 個位元為 1 時，該標籤必須包含文件 Ti。例如標籤 Tag_2 包含文件 T1 及 T5，則第 1 個位元（最左邊的位元）及第 5 個位元被編碼為 1，因此 Tag_2 的二元編碼為 10001。其餘標籤之二元編碼結果顯示在表 5 之中。

第三個步驟的主要工作是尋找每個標籤的子分群。我們定義當父分群 Tag_j 有兩個子分群 Tag_k 及 Tag_l 若且唯若 Tag_k 所包含的文件是 Tag_j 的子集合且 Tag_k 所包含的文件不是 Tag_l 的子集合。例如在圖 4 的第一回合之中，Tag_1 有 Tag_4 及 Tag_5 子分群，其原因為 Tag_4（{T1, T2, T4, T5}）及 Tag_5（{T1, T3, T4, T5}）皆為 Tag_1（{T1, T2, T3, T4, T5}）之子集合，並且 Tag_4 並非其兄弟分群 Tag_5 之子集合。雖然 Tag_3（{T1, T3, T5}）是 Tag_1 的子集合，但是 Tag_3 是其兄弟分群 Tag_5 之子集合，因此

Tag_3 不能當成 Tag_1 之子分群。詳細階層式分群建立情形請參考圖 4。

子集合運算可以簡單的經由位元 AND 運算完成。亦即當我們說 Tag_k 是 Tag_j 的子集合，代表下列情形成立：“（Tag_k 的二元編碼）AND（Tag_j 的二元編碼）”等於 Tag_k 的二元編碼。例如 Tag_4 是 Tag_1 的子集合，因為“（Tag_4 的二元編碼 11011）AND （Tag_1 的二元編碼 11111）”等於 Tag_4 的二元編碼。

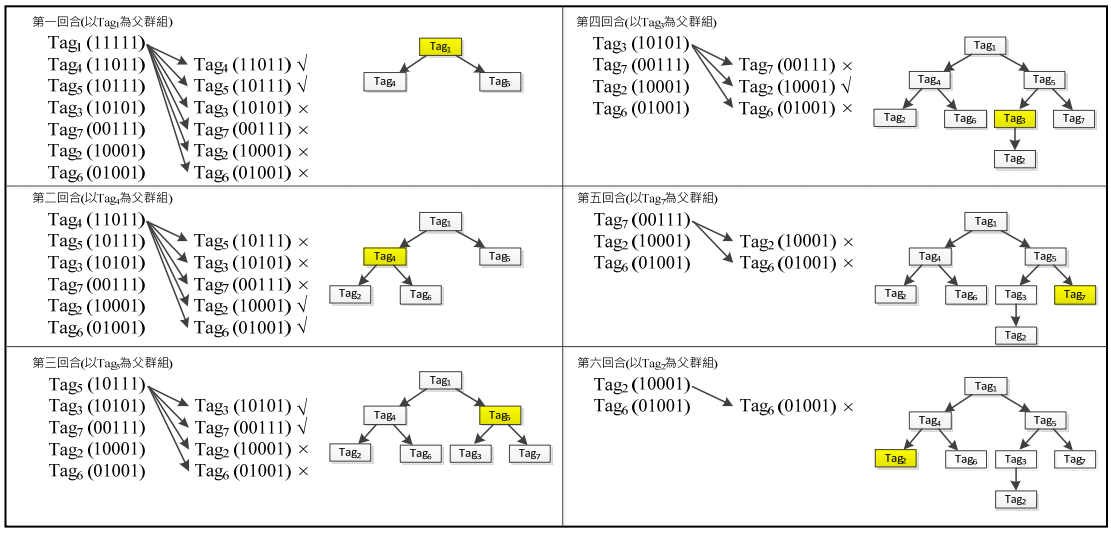

圖 4：階層式分群的建立情形

四、個人化建置

本研究個人化搜尋主要就是透過不儲存使用者瀏覽紀錄的方法來完成，我們會在每個分群標籤前面加上勾選選項，讓使用者可以選取自己所需要的分群標籤，這樣的做法與 SnakeT 相似，然而 SnakeT 只能針對相關選項進行聯集運算，亦即當使用者選取 manual 及 php tutorials 兩個主題（分群標籤），它會將這兩個主題所包含的文件進行聯集（OR）顯示，即在這兩個主題其一出現的相關文件。然而現實上，使用者可能需要其它運算，例如：使用者想要知道有那些文件同時包含上述兩個主題，亦即使用者可能想知道有什麼手冊（manual）是在講 php 教學（php tutorials），如果只是透過 SnakeT 的方式，使用者可能需花許多額外時間進行解讀，然而透過 PWSC 所提供的個人化選項，系統可自動幫使用者進行篩選。理論上，我們可以提供任何可能的二元運算，目前我們提供四種最常見的二元運算給使用者選擇：（1）AND（交集）、（2）OR（聯集）、（3）XOR（差集）、（4）NOT（補集）。我們舉幾個例子說明，上述二元運算的實際運用情形。使用者想要查詢跟相機有關的分群，當使用者輸入 camera 查詢時，系統可能回傳 nikon 和 canon 分群標籤，則上述二元運算的實際個人化搜尋意義分別代表如下：

1. AND 運算：在找尋 camera 資訊時，使用者希望的結果是要同時有 nikon 和 canon 這兩者同時出現的文件。
2. OR 運算：在找尋 camera 資訊時，使用者希望的結果是要有 nikon 和 canon 這兩者任一種出現即可的文件。
3. XOR 運算：在找尋 camera 資訊時，使用者希望的結果是只要有 nikon 而不要出現 canon 的文件或只要有 canon 而不要出現 nikon 的文件，也就是去除 nikon 和 canon 同時出現的文件。
4. NOT 運算：在找尋 camera 資訊時，使用者希望出現的結果是不要有 nikon 也不要有 canon 的文件，亦即代表，使用者想找尋除了 nikon 和 canon 以外的文件。

肆、實驗結果與討論

本實驗的目的是為了驗證分群系統所產生的分群標籤是否和使用者所輸入的查詢具有關聯，亦即計算兩個關鍵字（查詢與分群標籤）的語意關係，傳統的做法都是使用人工判讀。Cilibrasi 與 Vitanyi（2007）提出一個語意評估指標，稱之為正規化 Google 距離（Normalized Google Distance, NGD），計算關鍵字之間所具有的語意關聯程度。這個指標利用 Google 搜尋引擎所回傳的筆數，計算兩個關鍵字的距離，亦即彼此之間存在的語意關係。例如：在 Google 搜尋引擎輸入 adobe 與 flash player 兩個關鍵字，約可以搜尋到 1,260,000,000 筆結果，不過若輸入 adobe 與 word 兩個關鍵字，則約僅有 227,000,000 筆結果，這樣就可以很明顯的看出 adobe 與 flash player 之間存在的語意關係比 adobe 與 word 來得強。為了能有一個良好的評估指標，用來衡量兩個關鍵字的語意關係，我們採用 NGD 進行評比，其公式如下：

$$NGD(x,y)=\frac{\max\{\log f(x),\log f(y)\}-\log f(x,y)}{\log M-\min\{\log f(x),\log f(y)\}} \tag{3}$$

其中 M 為 Google 總索引的頁面；x 和 y 為輸入關鍵字；$f(x)$和 $f(y)$分別是搜尋引擎回傳 x 和 y 的筆數；而 $f(x,y)$為同時包含 x 和 y 的筆數。NGD 計算出來的數值介於 0 至 1 之間，數值越低，代表兩個關鍵字之間的距離越近，亦即彼此間存在愈高的語意關係；反之，代表兩個關鍵字之間的距離越遠，亦即彼此間存在愈低的語意關係。上述參數除了 M 以外，其餘皆可透過 Google API 呼叫取得相關數值。然而 M 的數值除非 Google 有公布，不然很難取得，所幸 Google 在其官方部落格公布其目前索引了一兆個（即 10 的 12 次方）不重複的頁面（Alpert & Hajaj 2008）。

我們以上述之 adobe、flash player、word 的例子進行 NGD 計算。假設想找出

adobe 與 flash player 或 word 之間何者存在較高的語意關係，透過 Google API 呼叫，adobe(x)和 flash player(y)分別回傳 1,340,000,000($f(x)$)及 209,000,000($f(y)$)筆結果，而 adobe 和 flash player 同時出現(x,y)的筆數為 1,260,000,000($f(x,y)$)，透過 NGD 公式計算後，adobe 與 flash player 的距離為 0.00727；另一方面，word(y)出現的筆數為 1,780,000,000，而 adobe 和 word 同時出現(x,y)的筆數為 227,000,000($f(x,y)$)，透過 NGD 公式計算後，adobe 與 word 的距離為 0.31132。兩相比較後，我們發現 adobe 與 flash player 的距離較短，亦即兩者存在較高的語意關係。

一、Threshold 的設定

本小節主要是決定門檻植（Threshold），我們將門檻值評估範圍從 0 到 1 之間，設定 19 個不同測試數值，我們的目的就是從其中選出一個門檻值。實驗過程如下：(1)首先，我們將測試關鍵字按照不同測試數值產生不同分群標籤（每個測試數值取出前十個文件數目最多的分群標籤）；(2)接下來，針對每個測試數值，我們分別計算測試關鍵字與這十個分群標籤之 NGD 數值，並對該測試數值之所有 NGD 數值進行平均，進而取得該測試數值之 NGD 代表數值。本研究採用的測試關鍵字是採用 2010 年 Google（2011）及 Yahoo（2011）的熱門關鍵字，去除掉非英文語系的關鍵字後，總共獲得 100 個熱門關鍵字，並且再從 Google 及 Yahoo 每日關鍵字排行榜中隨機選出 100 個隨機關鍵字。本研究採用這 200 個測試關鍵字進行門檻值實驗。由於論文篇幅限制，有興趣的讀者可以參考我們的測試關鍵字，網址為 http://cayley.sytes.net/pwsc/q.html。圖 5 為測試關鍵字經過 NGD 計算後不同測試數值之分佈情形。

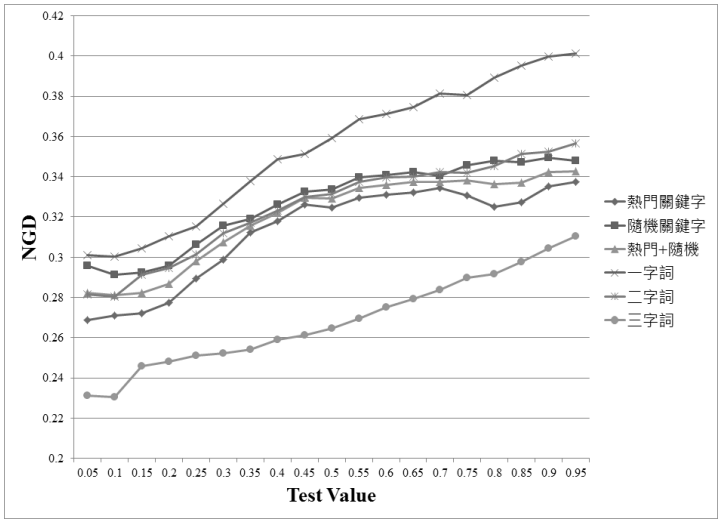

圖 5：測試關鍵字經過 NGD 計算後不同測試數值之分佈情形

根據圖 5 的結果，我們可以發現熱門關鍵字的 NGD 數值都比隨機關鍵字來得低，其原因在於熱門關鍵字是經由搜尋引擎統計整理後所產生的，而且熱門關鍵字中有大部份的字詞都是兩個單字以上所組成，這代表當使用者輸入的查詢字詞越長，搜尋引擎回傳的結果將會與輸入查詢更相關，這與現實情形相符；另一方面，隨機關鍵字大部份的字詞都是一個單字為主，所以普遍上其 NGD 數值會比熱門關鍵字來得高。同樣地，根據圖 5 的結果，我們發現熱門關鍵字在測試數值為 0.05 的時候 NGD 數值最低，其 NGD 數值約為 0.26869；另一方面，隨機關鍵字在測試數值為 0.1 的時候 NGD 數值最低，其 NGD 數值約為 0.29118；最後，在混合熱門及隨機關鍵字時，測試數值為 0.1 的時候 NGD 數值最低，其 NGD 數值約為 0.28111。

接下來，為了驗證長字詞效能優於短字詞，我們進行不同長度字詞之比較。在這個實驗之中，我們統計這 200 個關鍵字中，包含一個字詞的個數為 75 個、二個字詞的個數為 100 個、三個字詞的個數 22 個。由於“the oil disaster in the gulf of mexico”、“the world cup soccer tournament”、“2011 car of the year”所擁有的字詞數個數太少，因此我們不考慮這三個關鍵字，亦即我們只考慮 1~3 字詞的情形。圖 5 也顯示不同字詞之 NGD 數值，根據圖中的結果，我們發現長字詞的 NGD 數值明顯優於短字詞。

另外，為了比較不同門檻值所產生的“子節點取代父節點”，我們接下來進行另一個模擬，其主要計算當不同測試數值下，不同類型的關鍵字所發生的平均取代率。表 6 為實驗結果，表格中的每一個儲存格代表在某一個測試數值下，該類型關鍵字所發生的平均取代率。例如：當測試數值為 0.1 時，熱門關鍵字取代比率為 0.9394。根據表格中的結果，我們發現當測試數值愈高時，NGD 數值也會愈高，其原因在於測試數值愈高，子節點要取代父節點的機會就會愈低；相反的，當測試數值愈低，子節點要取代父節點的機會就會愈高。

表 6：不同類型關鍵字之平均取代率

	熱門關鍵字	隨機關鍵字	熱門＋隨機	一字詞	二字詞	三字詞	平均
0.1	0.9394	0.9212	0.9225	0.9480	0.9025	0.9100	0.9239
0.2	0.8122	0.8215	0.8064	0.8048	0.8232	0.8130	0.8135
0.3	0.7145	0.7198	0.7261	0.7103	0.7449	0.7491	0.7275
0.4	0.6345	0.6466	0.6449	0.6423	0.6229	0.6367	0.6380
0.5	0.5093	0.5375	0.5152	0.5012	0.5370	0.5321	0.5221
0.6	0.4467	0.4176	0.4202	0.4083	0.4132	0.4218	0.4213
0.7	0.3002	0.3283	0.3072	0.3134	0.3467	0.3036	0.3166
0.8	0.2448	0.2341	0.2190	0.2385	0.2060	0.2003	0.2238
0.9	0.1315	0.1071	0.1223	0.1149	0.1357	0.1446	0.1260

為了測試不同的選用標籤對門檻值的影響，我們針對 200 個測試關鍵字分別比較前 10 及 20 個文件數目最多的分群標籤之門檻值，其結果如圖 6 所示。根據圖中的結果，我們發現取 10 個標籤的 NGD 數值明顯都優於取 20 個標籤，這原因在於前 10 個標籤都是與測試關鍵字高度語意相關，一般而言，這樣算出來的 NGD 數值會較低；相反的，取 20 個標籤的結果除了會包含前 10 個標籤的結果，還包含後 10 個標籤，由於後 10 個標籤與測試關鍵字的語意關係較低，這會造成 NGD 數值的提昇。然而，無論我們以那一種取樣的結果，其只會造成 NGD 整體趨勢為提昇或降低，並不會對最佳門檻值的決定有太大的影響。因此，根據本實驗結果，我們所採用之門檻植為 0.1。

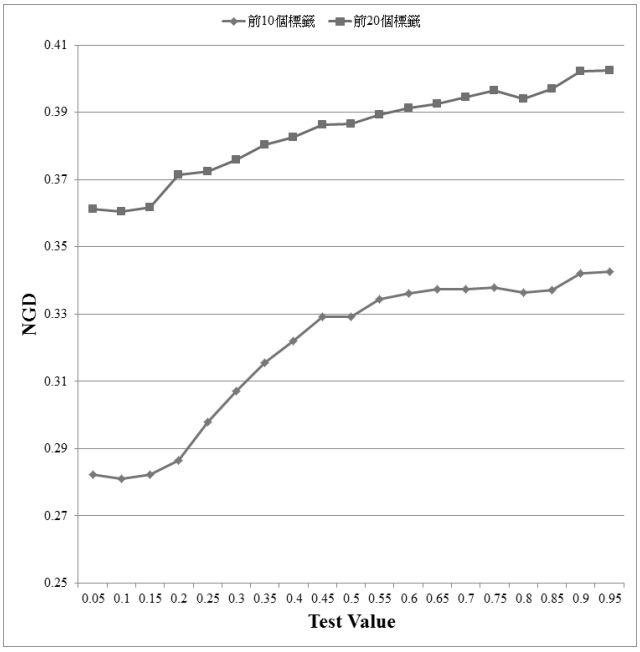

圖 6：不同的選用標籤對門檻值的影響

二、相關網頁摘要文件分群系統的比較

針對不同系統之間的比較，本小節使用兩個實驗進行評比，其分別為 NGD 與傳統評比指標。針對傳統評比指標，我們使用 Precision、Recall 及 F-measure 三項指標進行評比，相關描述請參照後續說明。我們使用線上網頁摘要文件分群系統進行比較，其中包含學術系統（Carrot2-STC、Carrot2-Lingo、CREDO、SnakeT）和商業系統（Yippy、iBoogie、WebClust），相關說明請參照本論文之文獻探討相關說明。

（一）以 NGD 指標進行評比

本評比主要使用 NGD 指標對線上網頁摘要文件分群系統進行評比，測試資料集為熱門及隨機關鍵字，總共 200 個測試關鍵字進行測試，實驗進行方式是將測試關鍵字輸入不同系統中，並計算每個關鍵字與前十筆第一層分群標籤的平均 NGD 數值。

圖 7 為不同系統之 NGD 數值評比，根據圖中的結果，我們發現熱門關鍵字比隨機關鍵字效能來得好，其原因如前面所述，即使用者輸入的查詢字詞越長，搜尋引擎回傳的結果將會與輸入查詢更相關。另一方面，CREDO 的數值在不同測試關鍵字下結果都很平均，其原因在於 CREDO 分群的結果屬於單一字詞，其採用的形式概念分析所產生之分群標籤皆為單一字詞，而其他系統所產生之分群標籤都是屬於一段句子的形式，這也間接證明，分群標籤採用一段句子的形式會比單一字詞效能來的更好。

同樣地，根據圖 7 的結果，我們發現不論測試關鍵字為熱門關鍵字、隨機關鍵字、混合關鍵字時，PWSC 的 NGD 數值皆為最低，其平均 NGD 數值分別為 0.26755、0.29238、0.27996；這說明了無論是在那一種類型的關鍵字下，PWSC 的效能都是優於其他系統。

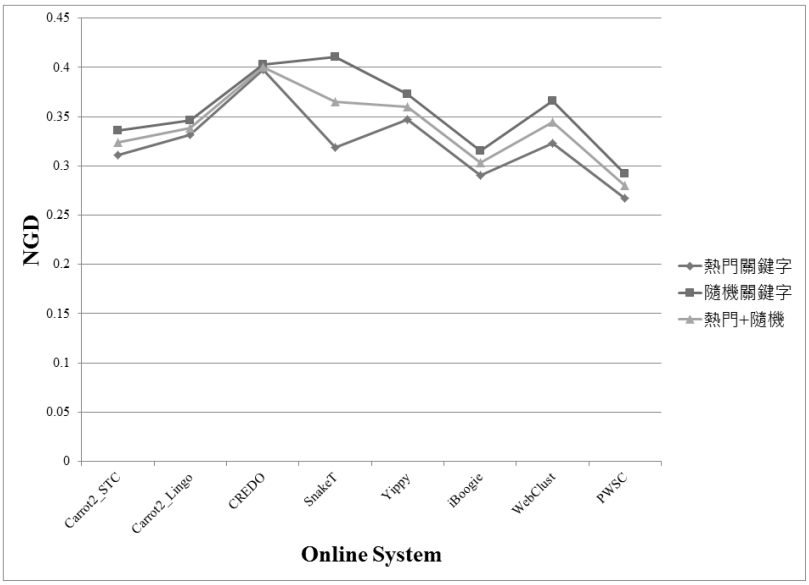

圖 7：使用 NGD 比較不同系統之效能

（二）以傳統評比指標進行評比

本評比主要使用傳統評比指標對上述系統進行評比，此處我們採用 Precision（Baeza-Yates & Ribeiro-Neto 1999）、Recall（Chen 2011）及 F-measure（Wan 2009）

三項傳統評比指標，並以使用者為對象進行評比。參與本實驗的對象共 15 人，其教育程度均在研究所以上，且均為資訊應用領域相關背景並具有英文閱讀能力，使用網路搜尋資料的經驗至少為 5 年。本實驗所使用的測試關鍵字共有 15 個，分別為 camera、iphone、new moon、christmas、japan earthquake、olympics、cupcakes、mcdonalds nutrition、starcraft 2、facebook、megan fox、volcanic eruption、ipad、michael jackson、world cup，其均選取自 NGD 評比之 200 個測試關鍵字。接下來，我們對各個評比指標作一個說明，Precision 的公式如下所示：

$$Precision_i = \frac{|UC_i|}{10} \tag{4}$$

其中 i 為第 i 個系統(i∈{Carrot2-STC, Carrot2-Lingo, CREDO, SnakeT, Yippy, iBoogie, WebClust, PWSC})；｜UC_i｜為前十個分群標籤之中，使用者判斷與測試關鍵字相關之個數；而 $Precision_i$ 為某一系統產生之分群標籤經由使用者判斷後所產生之精確率。Recall 的公式如下所示：

$$Recall_i = \frac{|UC_i|}{|\bigcup_{i=Carrot2-STC}^{PWSC} UC_i|} \tag{5}$$

其中 $|\bigcup_{i=Carrot2-STC}^{PWSC} UC_i|$ 針對所有系統所產生的前十個分群標籤，使用者判斷與測試關鍵字相關之不重複個數；$Recall_i$ 為某一系統產生之分群標籤經由使用者判斷後所產生之召回率。F-measure 的公式如下：

$$F-measure_i = 2 \times \frac{Precision_i \times Recall_i}{Precision_i + Recall_i} \tag{6}$$

F-measure 主要是結合 Precision 及 Recall 所計算出的綜合性度量指標，我們採用的原因是因為單一 Precision 或 Recall 指標都會有誤差的情形(Rijsbergen 1979)。

圖 8 顯示所有傳統評比指標的結果，我們發現 PWSC 在 Precision、Recall、F-measure 分別為 0.531、0.183、0.271，而這些數值均高於其他系統。這代表 PWSC 所產生的分群標籤，對使用者而言比較符合其查詢需求。

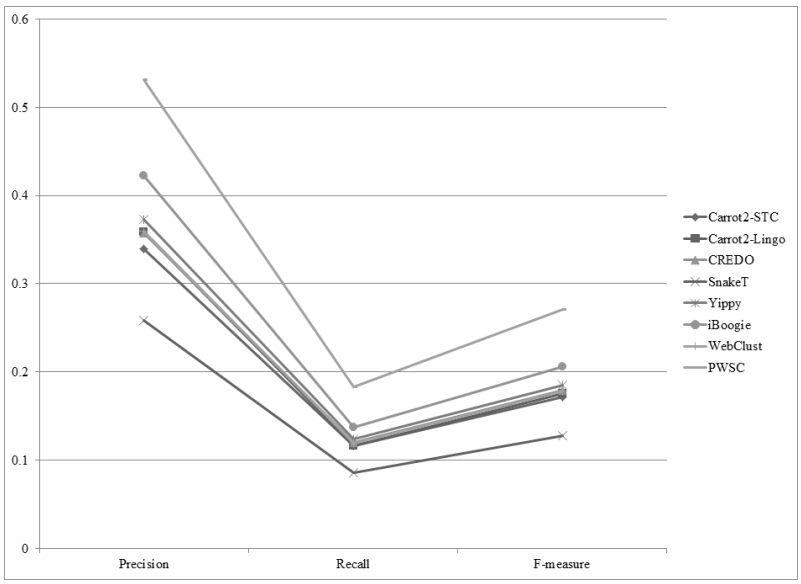

圖 8：使用傳統評比指標比較不同系統之效能

三、個人化搜尋評比

本研究的另一個主題為個人化搜尋，因此，本研究將與 SnakeT 進行評比，我們選擇 SnakeT 的主要原因是因為它也是採行不儲存使用者瀏覽紀錄的個人化搜尋系統。本小節使用兩個實驗進行評比，其分別為主題到達時間(Subtopic Reach Time, SRT）及二元運算工具。在個人化搜尋評比的實驗對象及評比主題如同前一個實驗，其分別為 15 人及 15 個測試關鍵字。

（一）主題到達時間

本實驗使用 SRT 評估分群標籤快速搜尋的效能（Carpineto et al. 2009a）。SRT 這個評估指標原先是用來評定手機上資料的搜尋速度，其原因為手機螢幕並沒有電腦螢幕來的大，本身所能顯示的資訊有限，而在這種情況下，就要將使用者越想知道的資訊放置愈前面的順序，以供使用者快速的進行瀏覽。本實驗就是為了要評定，當使用者在瀏覽本系統與 SnakeT 時，兩個系統所產生的分群標籤，何者較能快速的達到使用者的要求。其計算某一個主題 r 的到達時間為 r 的父分群 c 的順序位置加上 r 在 c 底下的順序位置。對於第 i 個系統之 SRT 數值表示如下，其中 n 為評比的主題個數，j 為與主題 k 相關的順序位置。

$$SRT_i = \frac{\sum_{k=1}^{n} \min_j (c_{k,j} + r_{k,j})}{n} \tag{7}$$

以圖 9 為例，當使用者搜尋 Facebook 時，若其想要尋找的主題為 Social Network，根據圖中的結果，Social Network 由上至下計算，PWSC 及 SnakeT 的 SRT 數值分別為 5 和 1。若其想是要尋找的主題為 Social Network Website，PWSC 及 SnakeT 的 SRT 數值分別為 6 和 3。當測試者在評比過所有測試資料集後，我們會將所有的 SRT 數值進行平均，以求出最終 SRT 評比數值。

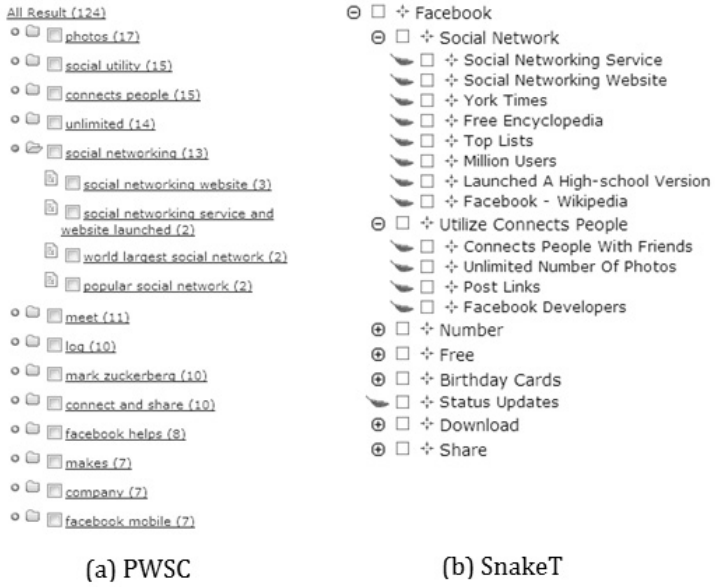

圖 9：SRT 計算範例

為了公平起見，在這個實驗之中，我們使用兩個系統皆有的 OR 二元運算進行 SRT 評比。我們發現 PWSC 及 SnakeT 最終 SRT 數值分別為 10.179 及 13.381，這代表 PWSC 比 SnakeT 更能讓使用者快速找尋到所需資訊。

（二）二元運算工具

本實驗使用 Precision、Recall 和 F-measure 評估相關二元運算工具的效能，我們以二元運算工具（AND、OR、XOR、NOT）篩選後的結果進行相關評比，並比較 PWSC 及 SnakeT 兩者所產生的個人化搜尋結果，是否為使用者所感興趣之內容。實驗過程中，為了公平起見，我們使用前面所提的測試關鍵字，經由測試者評比 PWSC 及 SnakeT 皆出現的分群標籤為評比對象。由於 SnakeT 的個人化搜尋只提供 OR 運算，因此我們會比較 PWSC 及 SnakeT 在進行 OR 運算時的效能差異。另外，本研究提供更多二元運算工具，我們也將一起比較。

實驗過程如下：首先，由測試者輸入測試關鍵字，並點選指定分群標籤及二元運算工具後產生對應之個人化搜尋結果；接下來，由測試者根據個人化搜尋結

果所篩選的前 10 筆網頁進行 Precision 評比，若是測試者有興趣瀏覽，就將該筆網頁的瀏覽次數加 1；最後，將所有測試者所產生的測試資料集平均後，產生之數值當成 Precision 結果。

根據表 7 的結果，我們發現 PWSC 在 OR 二元運算工具下，Precision、Recall、F-measure 分別為 0.807、0.253、0.385，其皆優於 SnakeT 所提供的 OR 運算，這代表在相同的二元運算下，我們的個人化搜尋結果優於 SnakeT。相同的，我們也同樣進行其它二元運算工具的評比，除了 AND 運算在 Recall 數值不理想（其原因在於，當兩個分群無任何交集的情況，其聯集的資料勢必較少），其餘皆在合理範圍，這代表提供其它二元運算工具能符合不同使用者的個人化搜尋習性。亦即 PWSC 所提供的二元搜尋運算工具比 SnakeT 能提供更多元的個人化搜尋能力。

表 7：三種指標評比個人化工具

Average	PWSC				SnakeT（OR）
	AND	OR	XOR	NOT	
Precision	0.595	0.807	0.773	0.748	0.713
Recall	0.047	0.253	0.242	0.234	0.223
F-measure	0.087	0.385	0.369	0.356	0.34

四、PWSC 貢獻之討論

PWSC 具有下列三點主要貢獻：多重分群、快速建立階層樹、快速的運行不同二元運算。茲分述如下：

（一）多重分群

以文件而言，有些文件本身包含了數個主題，因此好的方法必須將文件分配到適當的數個群集之中。以分群而言，有些群集可能包含在數個主題之下。例如：當使用者輸“ipod”時，PWSC 回傳結果如圖 10 所示，我們觀察“apple store”這個群集被包含在“ipod touch”及“ipod nano”群集之中。多重分群符合現實情形，亦即當使用者購買（擁有）上述兩項產品時，他可能想要連上“apple store”購買相關軟體。一個好的分群方法，需要達成多重分群的功能，透過我們所發展的階層式分群演算法，我們可以輕易的達成多重分群。

圖 10：多重分群的例子

（二）快速建立階層樹

PWSC 階層演算法中的第三步驟“Build a Tree”所需的運算只需簡單的透過不同位元運算（Bitwise Operations）即可完成，因此我們建立整個階層樹的時間可以非常快速。然而，不同系統間存在下列幾項差異，因此我們無法針對不同系統間的分群時間進行評比。

- 大部份評比系統的實作演算法未知，因此我們無法 100%模擬出原系統，如果強行進行評比，恐有失準之虞。
- 每個評比系統抓取網頁摘要文件的時間受限於網路存取速度，然而這部份存在著不可預知性。
- 不同評比系統所採用的軟/硬體設備可能存在很大的差異。

根據許多學者（Zamir & Etzioni 1999; Zhao & Karypis 2002）所提出的建議，一個好的分群演算法所需執行時間應與文件數目成線性關係。以下我們進行一個模擬，模擬的對象是使用 Google（2010）最近 50 天（2011/10/10~2011/11/28）所產生的 1000 個查詢關鍵字。我們的模擬過程如下：首先將這 1000 個關鍵字分別使用 Web Crawler 取得所需之網頁摘要文件；接下來，針對所有關鍵字及網頁摘要文件進行相關的「前置處理與部份資訊瀏覽」及「自然語言處理以及建立候選分群標籤」；最後「使用二元編碼建立階層式分群」。由於這個模擬我們關心的是下列議題：使用位元運算是否可以快速建立階層式分群，而且所需執行時間是否與文件數目成線性關係，因此與此議題無關的前置工作（包含擷取網頁文件及相關前置處理產生候選分群），我們不考慮其所需時間。針對模擬所採用的硬體設備，

我們使用現今很常見的設備進行模擬：Intel Core 2 Duro T9600/2GB DDR2。圖 11 是建立階層式分群所需之平均時間，圖中的 X 軸代表抓取回來的文件數目，Y 軸代表所需的平均時間。觀察圖中的結果，我們發現當文件數目為 5000 與 10000 時，所需的執行時間分別為 1.9 與 3.7 秒。這代表我們所需的執行時間與文件數目成線性關係，並不會隨著文件數目增加而快速成長。同樣地，根據圖中的結果，我們發現這收集回來的 110000 文件，其總處理時間為 39.9 秒；亦即每 10000 個文件所需的平均處理時間為 3.63 秒。

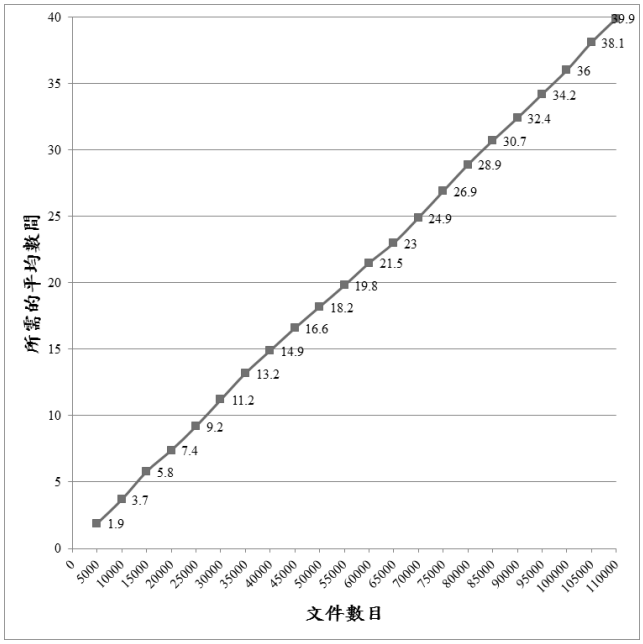

圖 11：PWSC 建立階層樹所需之平均時間

（三）快速的運行不同二元運算

由於我們使用二元編碼的方式建立階層樹，其中每個標籤皆是由二元編碼的方式表示所對應之網頁摘要文件，因此我們可以透過不同的位元運算達成不同的二元運算，亦即使用者可以透過二元運算達成不同的個人化搜尋需求。例如：當我們有兩個標籤 A 及 B，其中二元編碼分別為 1101 及 1001，當使用者執行“AND”二元運算，亦即使用者要找出同時包含這兩個主題的網頁摘要文件時，我們只需回傳第 1 及 4 份網頁摘要文件。與另一個同類型的個人化搜尋引擎 SnakeT 相比，我們除了提供 SnakeT 所提供的 OR 二元運算（SnakeT 也僅提供此二元運算），還可以使用不同的二元運算，其原因在於我們本身即採用二元編碼的資料結構儲存分群標籤，所以只要透過不同二元運算即可達成。SnakeT 使用共享文字（Share

Snippet）儲存分群標籤，由於這個方法是以文字內容判斷兩個標籤是否存在父子關係，其可以經由簡易的判斷兩個標籤是否具有共同存在之 Share Snippet 達成 OR 二元運算，然而其它二元運算卻無法經由此方式達成。

伍、結論與未來研究方向

本研究目的在於設計出一套具有個人化搜尋能力之階層式分群系統。首先，我們收集網頁摘要文件並產生分群標籤；接下來，我們設計一個階層式分群演算法將所有分群標籤建置成一個階層樹的形式；最後，我們使用不儲存使用者瀏覽紀錄的方法完成個人化搜尋，我們會在每個分群標籤前面加上勾選選項，讓使用者可以選取自己所需要的分群標籤，並同時提供多樣的二元運算工具，以便符合不同使用者的個人化搜尋習性。

本研究貢獻之處共兩點：首先，我們提出了一個高性能的分群方法；其次，我們設計了一個具有多種個人化搜尋特性之分群系統。根據不同的實驗結果，我們提出之系統比其它線上網頁摘要文件系統效能都還優良。在個人化搜尋的評比上，我們除了個人化搜尋結果優於 SnakeT 以外，還提供不同二元運算工具以便符合不同使用者的個人化搜尋習性。

未來研究方向總共有三項：(1)進一步的擴充其它語言處理能力，由於每種語言所具有的屬性不同，這代表我們必須為其它語言進行不同的 NLP 處理；(2)將本研究應用至其它不同類型的文件，諸如部落格、學術搜尋、百科全書、購物網站等；(3)目前這種產生分群標籤的方式應該還有改善的空間，未來我們希望繼續尋找更適合標籤產生方式。

致謝

我們感謝二位匿名審稿委員提供寶貴意見，以便改善我們論文的品質。本文接受行政院國家科學委員會專題研究計畫（NSC 100-2410-H-259-010-MY2）之補助研究經費，順利完成此篇著作之研究工作，謹此致謝。

參考文獻

Alpert, J. and Hajaj, N. (2008), 'Official Google blog: we knew the Web was big', available at http://0rz.tw/9TuEV (accessed 11 September 2012).

Baeza-Yates, R. and Ribeiro-Neto, B. (1999), *Modern Information Retrieval*, Addison Wesley Press, Boston, Massachusetts.

Benson, M. (1989), 'The structure of the collocational dictionary', *International Journal*

of Lexicography, Vol. 2, No. 1, pp. 1-14.

Brown, P.F., deSouza, P.V., Mercer, R.L., Pietra, V.J.D. and Lai, J.C. (1992), 'Class-based N-gram models of natural language', *Computational Linguistics*, Vol. 18, No. 4, pp. 467-479.

Carpineto, C., Mizzaro, S., Romano, G. and Snidero, M. (2009a), 'Mobile information retrieval with search results clustering: prototypes and evaluations', *Journal of the American Society for Information Science and Technology*, Vol. 60, No. 5, pp. 877-895.

Carpineto, C., Osinski, S., Romano, G. and Weiss, D. (2009b), 'A survey of Web clustering engines', *ACM Computing Surveys*, Vol. 41, No. 3, Article 17.

Carpineto, C. and Romano, G. (2004), 'Exploiting the potential of concept lattices for information retrieval with CREDO', *Journal of Universal Computer Science*, Vol. 10, No. 8, pp. 985-1013.

Carrot2 (2011), 'Carrot2 clustering engine', available at http://search.carrot2.org/stable/search (accessed 11 September 2012).

Chen, L.C. (2011), 'Building a Web-snippet clustering system based on a mixed clustering method', *Online Information Review*, Vol. 35, No. 4, pp. 611-635.

Chen, L.C. and Luh, C.-J. (2005), 'Web page prediction from metasearch results', *Internet Research: Electronic Networking Applications and Policy*, Vol. 15, No. 4, pp. 421-446.

Cilibrasi, R.L. and Vit´anyi, P.M.B. (2007), 'The Google similarity distance', *IEEE Transaction on Knowledge and Data Engineering*, Vol. 19, No. 3, pp. 370-383.

comScore (2011), 'comScore releases May 2011 U.S. search engine rankings', available at http://0rz.tw/sPQ6O (accessed 11 September 2012).

DMOZ (2011), 'ODP - open directory project', available at http://www.dmoz.org/ (accessed 11 September 2012).

Ferragina, P. and Guli, A. (2008), 'A personalized search engine based on Web-snippet hierarchical clustering', *Software: Practice and Experience*, Vol. 38, No. 2, pp. 189-225.

Fox, C. (1989), 'A stop list for general text', *ACM SIGIR Forum*, Vol. 24, No. 1-2, pp. 19-35.

Frantzi, K., Ananiadou, S. and Mima, H. (2000), 'Automatic recognition of multi-word terms: the C-value/NC-value method', *International Journal on Digital Libraries*, Vol. 3, No. 2, pp. 115-130.

Fung, B.C.M., Wang, K. and Ester, M. (2003), 'Hierarchical document clustering using

frequent itemsets', *Proceedings of the Third SIAM International Conference on Data Mining*, San Francisco, California, USA, May 1-3, pp. 59-70.

Garai, G. and Chaudhuri, B.B. (2004), 'A novel genetic algorithm for automatic clustering', *Pattern Recognition Letters*, Vol. 25, No. 2, pp. 173-187.

Giannotti, F., Gozzi, C. and Manco, G. (2002), 'Clustering transactional data', *Lecture Notes in Computer Science*, Vol. 2431, No. 2002, pp. 227-239.

Giannotti, F., Nanni, M., Pedreschi, D. and Samaritani, F. (2003), 'WebCat: automatic categorization of Web search results', *Proceedings of the 11th Italian Symposium on Advanced Database Systems*, Cosenza, Italy, June 24-27, pp. 507-518.

Google (2010), 'Google Trends', available at http://www.google.com/trends (accessed 11 September 2012).

Google (2011), 'Google Zeitgeist 2010', available at http://www.google.com/intl/en/press/zeitgeist2010/ (accessed 11 September 2012).

Google (2012), 'Google search history', available at https://www.google.com/history/ (accessed 11 September 2012).

Hashemi, R.R., Ford, C.W., Vamprooyen, T. and R.Talburt, J. (2002), 'Extraction of features with unstructured representation from HTML documents', *Proceedings of the IADIS International Conference WWW/Internet 2002*, Lisbon, Portugal, November 13-15, pp. 47-53.

Hazel, P. (2012), 'PCRE - Perl compatible regular expressions', available at http://www.pcre.org/ (accessed 11 September 2012).

Hearst, M.A. and Pedersen, J.O. (1996), 'Reexamining the cluster hypothesis: scatter/gather on retrieval results', *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Zurich, Switzerland, August 18-22, pp. 76-84.

Horowitz, D. and Kamvar, S.D. (2010), 'The anatomy of a large-scale social search engine', *Proceedings of the 19th International Conference on World Wide Web*, Raleigh, NC, USA, April 26-30, pp. 431-440.

Huang, J.Z., Ng, M.K., Rong, H. and Li, Z. (2005), 'Automated variable weighting in k-means type clustering', *IEEE Transaction on Pattern Analysis and Machine Intelligence*, Vol. 27, No. 5, pp. 657-668.

iBoogie (2011), 'iBoogie - metasearch document clustering engine and personalized search engines directory', available at http://www.iboogie.com/ (accessed 11 September 2012).

Jansen, B.J., Spink, A. and Koshman, S. (2007), 'Web searcher interaction with the

Dogpile.com metasearch engine', *Journal of the American Society for Information Science and Technology*, Vol. 58, No. 8, pp. 744-755.

Jeh, G. and Widom, J. (2003), 'Scaling personalized Web search', *Proceedings of the 12th International Conference on World Wide Web*, Budapest, Hungary, May 20-24, pp. 271-279.

Liddy, E.D. (2001), 'Natural Language Processing', in *Encyclopedia of Library and Information Science*, 2nd Ed., Marcel Decker, New York, USA.

MacQueen, J.B. (1967), 'Some methods for classification and analysis of multivariate observations', *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, University of California, Berkeley, USA, June 21-July 18, 1965 and December 27, 1965-January 7, 1966, pp. 281-297.

Manning, C.D. and Schuetze, H. (1999), *Foundations of Statistical Natural Language Processing*, MIT Press, Massachusetts, USA.

Maxymuk, J. (2008), 'Searching beyond google', *The Bottom Line: Managing Library Finances*, Vol. 21, No. 3, pp. 97-100.

Nah, F.F.H. (2004), 'A study on tolerable waiting time: how long are Web users willing to wait?', *Behaviour and Information Technology*, Vol. 23, No. 3, pp. 153-163.

Osinski, S. and Weiss, D. (2005), 'A concept-driven algorithm for clustering search results', *IEEE Intelligent Systems*, Vol. 20, No. 3, pp. 48-54.

Porter, M. and Boulton, R. (2007), 'Snowball: a language for stemming algorithms', available at http://snowball.tartarus.org/ (accessed 11 September 2012).

Rijsbergen, C.J.V. (1979), *Information Retrieval*, Butterworth-Heinemann, Massachusetts, USA.

Segev, A., Leshno, M. and Zviran, M. (2007), 'Context recognition using internet as a knowledge base', *Journal of Intelligent Information Systems*, Vol. 29, No. 3, pp. 305-327.

Speretta, M. and Gauch, S. (2005), 'Personalized search based on user search histories', *Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence*, Compiegne, France, September 19-22, pp. 622-628.

Sun, J.-T., Zeng, H.-J., Liu, H., Lu, Y. and Chen, Z. (2005), 'CubeSVD: a novel approach to personalized Web search', *Proceedings of the 14th international conference on World Wide Web*, Chiba, Japan, May 10-14, pp. 382-390.

Vivisimo (2011), 'Vivisimo information optimization', available at http://vivisimo.com/ (accessed 11 September 2012).

Wan, X. (2009), 'Combining content and context similarities for image retrieval',

Lecture Notes in Computer Science, Vol. 5478, No. 1, pp. 749-754.

WebClust (2011), 'WebClust - clustering search engine', available at http://www.webclust.com/ (accessed 11 September 2012).

Weiss, D. and Stefanowski, J. (2003), 'Web search results clustering in polish: experimental evaluation of carrot', *Proceedings of the New Trends in Intelligent Information Processing and Web Mining Conference*, Zakopane, Poland, June 2-5, pp. 209-218.

Wu, Y.F. and Chen, X. (2003), 'Extracting features from Web search returned hits for hierarchical classification', *Proceedings of the 2003 International Conference on Information and Knowledge Engineering*, Las Vegas, Nevada, USA, June 23-26, pp. 103-108.

Wu, Y.F.B., Rakthin, C. and Li, C. (2002), 'Summarizing search results with automatic tables of contents', *Proceedings of the 8th Americas Conference on Information Systems*, Texas, United States, August 9-11, pp. 88-92.

Wu, Y.F.B., Shankar, L. and Chen, X. (2003), 'Finding more useful information faster from Web search results', *Proceedings of the 2003 ACM CIKM International Conference on Information and Knowledge Management*, New Orleans, Louisiana, United States, November 2-8, pp. 568-571.

Yahoo (2011), 'Yahoo! 2010 year in review - top 10 searches', available at http://yearinreview.yahoo.com/2010/us_top_10_searches (accessed 11 September 2012).

Yahoo (2012), 'My Yahoo', available at http://my.yahoo.com/ (accessed 11 September 2012).

Yippy (2011), 'Yippy clustering engine', available at http://www.yippy.com/ (accessed 11 September 2012).

Zamir, O. and Etzioni, O. (1998), 'Web document clustering: a feasibility demonstration', *Proceedings of the 21st International ACM SIGIR Conference on Research and Development in Information Retrieval*, Melbourne, Australia, August 24-28, pp. 46-54.

Zamir, O. and Etzioni, O. (1999), 'Grouper: a dynamic clustering interface to Web search results', *Computer Networks*, Vol. 31, No. 11-16, pp. 1361-1374.

Zhao, Y. and Karypis, G. (2002), 'Evaluation of hierarchical clustering algorithms for document datasets', *Proceedings of the 11th International Conference on Information and Knowledge Management*, Graz, Austria, September 7-9, pp. 515-524.