國立政治大學資訊科學系
Department of Computer Science
National Chengchi University

碩士論文

Master's Thesis

在高度分散式環境下進行 Top-*k* 相似文件檢索

Similar Top-*k* Documents Retrieval in Highly Distributed Environments

研 究 生：王俊閎

指導教授：陳良弼

中華民國一〇一年十一月

November 2012

在高度分散式環境下進行 Top-*k* 相似文件檢索

Similar Top-*k* Documents Retrieval in Highly Distributed Environments

研 究 生：王俊閎　　Student：Chun-Hung, Wang

指導教授：陳良弼　　Advisor：Arbee L.P. Chen

國立政治大學
資訊科學系
碩士論文

A Thesis
submitted to Department of Computer Science
National Chengchi University
in partial fulfillment of the Requirements
for the degree of
Master
in
Computer Science

中華民國一〇一年十一月

November 2012

在高度分散式環境下進行 Top-k 相似文件檢索

摘要

在文件資料庫的查詢處理上，Top-k 相似文件查詢主要是協助使用者可以從龐大的文件集合中，檢索出和查詢文件具有高度相關性的文件集合。將資料庫內的文件依據和查詢文件之相似度程度，選擇出相似度最高的前 k 篇文件回傳給使用者。然而過去集中式資料庫，因其覆蓋性和可擴充性的不足，使得這種排名傾向的文件查詢處理，需耗費大量時間及運算成本。近年來，使用端對端(Peer-to-peer, P2P)架構解決相關的文件檢索問題已成為一種趨勢，但在高度分散式環境下，支援排名傾向的相似文件查詢是困難的，因為缺乏全域資訊和適當的系統協調者。

在本研究中，我們先針對各節點資料庫作分群前處理，並提出一個利用區域切割的作法[1]，將 P2P 環境劃分成數個子區塊後，建立特徵索引表。因此在查詢處理時，可透過索引表加快挑選出 Top-k 相似群集的速度，並且確保有適當數量的回傳結果。最後在實驗中，我們提出的方法會與傳統集中式搜尋引擎以及 SON-based [1] 做比較，在高度分散式環境下，我們的方法在執行 Top-k 相似文件查詢時，會比上述兩種作法有較為優異的表現。

Similar Top-*k* Documents Retrieval in Highly Distributed Environments

Abstract

On query processing in a large database, similar top-*k* documents query is an important mechanism to retrieve the highly correlated document collection with query for users. It ranks documents with a similarity ranking function and reports the *k* documents with highest similarity. However, the former approach in web searching, i.e., centralized search engines, rises some issues such as lack of coverage and scalability, impact provides rank-based query become a costly operation. Recently, using Peer-to-peer (P2P) architectures to tackle above issues has emerged as a trend of solution, but due to the shortage of global knowledge and some appropriate central coordinators, support rank-based query in highly distributed environment has been difficulty.

In this paper, we proposed a framework to solve these problems. First, we performed the local cluster pre-processing on each peer, followed by the zone creation process, forming sub-zones over P2P network, and then constructing the feature index table to improve the performance of selecting similar top-k cluster results. The experiments show that our approach performs similar top-*k* documents query outperforms than SON-based approach in highly distributed environment.

誌謝

這兩年的研究所生涯之中，我有太多的人需要感謝了。兩年時光飛逝，不僅充滿許許多多愉悅的回憶，也充實了學問與知識，更為我的人生增添了精彩的一頁！

首先，感謝我的指導教授陳良弼老師，在這兩年的碩士生涯裡，讓我從老師身上學習到做學問時應有的態度，不只是思考問題的邏輯模式，更包含了對任何疑問追根究底的研究精神。在我的研究過程中，老師總能適時的給予我學術指導與建議方向，更能在學生犯錯時給予當頭棒喝，將我們引導至正確的方向，謝謝老師這兩年來的諄諄教誨。

再來也感謝資料庫實驗室的學長姐、學弟妹們，每次的報告完都能給我完整的建議，不僅如此，也陪伴我渡過了這兩年快樂的研究生活。很高興當初選擇加入資料庫實驗室遇見了大家，一路走來我們相互扶持與鼓勵，進而共同成長。謝謝你們讓我的碩士生涯留下了如此美好的回憶。

最後，由衷的感謝我的家人這一路走來的支持與鼓勵，特別是媽媽總能在我研究不順利時，給予我溫暖的鼓勵，讓我能無後顧之憂的專心於學業上。謝謝我最摯愛的家人們，今日終於能在這裡與你們分享我的喜悅，我愛你們。

我要再一次地謝謝所有師長、家人、朋友們，過去的日子因為有你們的陪伴，才能成就現在的我，謝謝各位。謹以此篇文章表達我心中的感謝。

目錄

表目錄

圖目錄

第一章 導論及研究動機

在文件資料庫的查詢處理上，Top-k相似文件查詢主要是協助使用者可以從龐大的文件集合中，檢索出和查詢文件具有高度相關性的文件集合。將資料庫內的文件依據和查詢文件之相似度程度，選擇出相似度最高的前 k 篇文件回傳給使用者。然而過去集中式資料庫，因其覆蓋性和可擴充性的不足，使得這種排名傾向的文件查詢處理，需耗費大量時間及運算成本。近年來，使用端對端(Peer-to-peer, P2P)架構解決相關的文件檢索問題已成為一種趨勢，但在高度分散式環境下，支援排名傾向的相似文件查詢是困難的，因為缺乏全域資訊和適當的系統協調者。

因此本研究中，我們首先針對各資料庫作分群前處理，並提出一個利用區域切割的作法，將 P2P 環境劃分成數個子區塊後，建立特徵索引表。因此在查詢處理時，可透過索引表加快挑選出 Top-k相似群集的速度，並且確保有適當數量的回傳結果。最後在實驗中，我們提出的方法會與傳統集中式搜尋引擎以及 SON-based [1] 做比較，在高度分散式環境下，我們的方法在執行 Top-k相似文件查詢時，會比上述兩種作法有較為優異的表現。

近年來，由於網路環境中資料量的成長速度急遽上升，過去傳統的集中式資料庫處理相似文件檢索的方法，已不再適用於高度分散式的環境上，因為各資料庫中，皆擁有不同的資料分佈及運算能力的差別。因此，利用 P2P 網路架構 (Peer-to-Peer Network)，解決在分散式資料庫中的文件檢索問題，以及網路文件搜尋領域上，已成為一種主流趨勢。相較於傳統典型的集中式資料庫 (Centralized system)或是一般主從式架構 (Client-Server architectures)，P2P 環境內節點的新增刪除、資料的儲存分享皆有一套自主的控管的方法，可避免了部分節點不希望將全部資料公開，或是成為集中式索引的問題。因此如何在這樣動態且高度分散式的環境中，提供一個有效率且可擴展性

(Scalability) 的搜尋機制，而不需將實際資料做搬移或備份，便成了需探討的重要議題，也是本篇研究想要解決的問題。

現今已提出了許多在 P2P 環境中，利用鍵值 (key) 做相似文件搜尋的系統，這些系統大多以分散式雜湊表 (Distributed Hash Table, DHT) 實作出的，諸如：Chord [2]、CAN [3]、Pastry [4] 等等，而部分架構則是利用層疊式網路 (Overlay Network) [5] 的概念去建立節點架構。然而部分架構在解決相似文件搜尋時，無法有效的提供整體環境的全域資訊，而造成查詢處理時，須耗費大量運算時間與成本在做比對，尤其當資料形態屬於超高維度(文件、影音等等)時，更為明顯。因此，Christos. Doulkeridis 所提出的分散及分佈式語義層疊式網路之建構方法 (SON-based) [1]，主要是先利用傳統資訊檢索技術，針對各資料中文件做特徵抽取 (Feature extract)，並將文件表達成特徵向量 (Feature vector) 後，計算各文件之間的相似度，依此做為分群依據。接著，依照各資料庫的群集資訊，將具有相似文件內容的資料庫再做一次分群篩選，形成階層式語意層疊式網路。因此搜尋時，需將查詢文件路由至層疊式網路最上層的資料庫搜尋，但因為在資料庫彼此之間的分群步驟，容易因篩選掉過多的群集文件或是歸屬到不合適的語意層疊式網路，而造成路由目標的資料庫文件可能是與查詢較為不相關。

因此在本篇論文中，我們提出了一個架構流程，來解決在 P2P 環境上進行 Top-k 相似文件檢索的問題。主要是先以分群演算法對各文件資料庫做前處理，得到各資料庫內的文件群集資訊。並藉由區域分割 [1] 的方式，挑選出啟動節點後，彙整各區域內的群集資訊，用以建立特徵索引表。在使用者下達查詢文件時，先針對該文件做一次前處理程序得到特徵向量，再將此向量路由至區域內的啟動節點，比對特徵索引表內與查詢文件有共同特徵的群集，列出相似度高的 Top-k 群集。接著，啟動節點會再將查詢路由到這些 Top-k 群集各自所在的節點資料庫中，進行相似文件搜尋並得到一組結果列表，最後再將查詢路由至各啟動節點平行處理，將所有結果合併後，便可回傳給使用者參考。

第二章 相關研究

在此章節中將會介紹與本研究相關的文獻及知識。由於我們的研究是建立在超級節點網路上,首先會介紹關於端對端環境,及其延伸發展出的超級節點網路架構與層疊式網路。接著會描述在此環境下其他相關研究如何進行所謂的相似文件搜尋的方法。

2.1 端對端網路的相關研究

端對端網路架構近年來在高度分散式環境下,對於內文搜尋領域已成為一種相當具有吸引力的解決辦法。所謂的端對端網路架構,是一種合作式型態的網路技術,相較於集中式資料庫系統和單純主從式系統的侷限性,端對端網路依賴的是網路中參與節點的計算能力與頻寬,而非僅靠單一伺服器處理運算,且任意節點在網路中皆可以作為客戶端(Clients)和伺服器端(Severs)。當節點為客戶端時,主要是發出查詢在整個 P2P 環境中搜尋;而節點為伺服器端時,則是負責提供資訊內容,且可將查詢路由至其他的伺服器端。像是過去典型的檔案分享 P2P 系統:BitTorrent [6] 和 eMule [7],當某一節點從其他節點下載檔案時,同時本身也分享著檔案給其他節點。

而在 P2P 環境下的查詢種類一般可區分為:一般查詢(Queries)、路由查詢(Query Routing)、檢索(Retrieval)三大種類。而本篇研究主要是針對檢索類型中的相似度查詢,將其延伸至高度分散式環境下,做 Top-k 相似文件搜尋,故以下我們會先介紹所謂的高度分散式環境。

2.1.1 超級節點網路

超級節點網路(Super-peer Networks)[8] 運作與一般的純 P2P 網路相似,只是在網路中選擇了特定節點作為超級節點(Super-peer),這個特定節點在某些客戶端節點集

合（Subsets of clients）中，運作方式和集中式伺服器相似。比起普通節點，超級節點一般都擁有較好的處理能力、儲存空間和網路頻寬等，且每一個超級節點負責連結著一組客戶端節點，而每一個客戶端節點只可連結單一超級節點。如下圖表示在超級節點網路可能存在的網路拓撲圖：

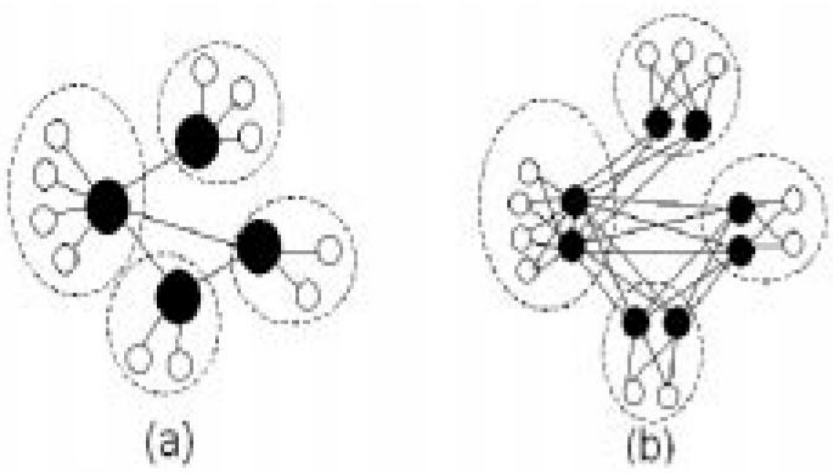

圖 2.1：超級節點網路（引自［8］圖 1）

(a)為無冗餘超級節點網路

(b)有 2-冗餘超級節點網路

在圖 2.1(a)中，黑點表示一個超級節點，而每個超級節點連結的客戶端節點集合可形成一個節點群集（Peer Cluster）。當超級節點接收查詢時，會將查詢傳送至底下所連結的客戶端節點集合，同時也會保留著這些客戶端節點集合的索引方便查詢。而這些索引必須提供將來所有查詢時，可能會用到的資訊。舉例來說：若有一檔案分享 P2P 環境，使用者下達的查詢皆是針對檔案標題的關鍵字搜尋，因此在這樣的環境下，超級節點必須建立一張列表，內容包含底下所有連結的客戶端節點的檔案標題。當找到其中一個節點存在著查詢結果時，會回傳一組回覆訊息，當中包含著結果和結果所在客戶端節點的網路位址。超級節點為了維護這些索引資訊，當有任意的節點作為客戶端加入此超級節點時，此節點資料庫會將本身的元資料（Meta-data）傳送給超級節點，並依據此資料新增索引。若有客戶端節點離開時，超級節點也會將該資料從索引中移除。當客戶

端節點想提交查詢至整體 P2P 環境上，只需將查詢傳送所屬的超級節點即可，而超級節點會再將此查詢路由至其他相鄰的超級節點鄰居，再由這些超級節點進行查詢。

為了提供節點群集的可靠性以及降低超級節點的負載，在設計超級節點網路時，引入了 k-冗餘 (k-redundant)。所謂的 k-冗餘，就是以 k 個節點去分擔超級節點的負載，形成單一虛擬的超級節點 (Virtual Super-peer)，如圖 2.1(b)所示：在每一個虛擬超級節點中，每個節點互相都是合作夥伴擁有相同的負載，皆連結著群集中所有的客戶端節點和所有的節點索引。因此，有 k-冗餘特性的超級節點網路在可靠性及分享性，是優於無冗餘超級節點網路，但是其建置成本與搜尋時間較高。

目前已有相當多的 P2P 檔案分享系統，諸如：Gnutella [9] 和 KaZaA [10]，皆是採用超級節點的網路拓撲做為其架構，本研究一樣是採用超級節點網路架構做為基礎。

2.1.2 層疊式網路

所謂層疊式網路 (Overlay Networks) 是指我們刻意忽略網路的實體架構，將網路上的每台電腦都視為一個單獨的節點，並且假設他們可以自由的互相連接。將這些單獨的節點採用某種特殊的結構來建置一個網路，一條層疊式網路上的連線可能是由數條實體網路的連線所構成，透過互相連接的網路節點，使其能夠達到某種我們期望的效果，也就是透過虛擬環境所提供的系統機制來建立另一種網路系統。

層疊式網路是架構在實體網路之上的另一層網路，所以實體網路的連接關係與變動絕對性的影響了層疊式網路的穩定性，而任何的端對端網路都有節點會隨時加入及離線的特性，相當於在其上運作的系統必需有適應高度變化的能力。因此，目前已發展出的路由機制，可允許非網路位址來當訊息，來傳送至目的地：例如分散式雜湊表，將訊息路由到一個特定的邏輯地址，但其網路位址是事前未知的。

2.2 內文檢索的相關研究

所謂的內容檢索是透過支援簡單的關鍵字搜尋，比對查詢文件的關鍵字與資料庫中文件所擁有的關鍵字是否符合，只有符合查詢關鍵字的文件才會被檢索出來，並回傳給使用者。但是應用在 P2P 這樣高度分散式的環境時，因資料內容廣泛的分散在網路環境中，系統無法有效的提供一個全域資訊作為參考，因此往往必須實際移動資料庫中內容，才可進行跨資料庫的複雜內容檢索。因此，目前在 P2P 環境中，已發展出數套系統架構用來解決較複雜的內容檢索問題，諸如：Christos. Doulkeridis 所提出建構在分散及分佈式語義層疊式網路下，執行相似文件搜尋的架構（SON-based）。此方法主要是先利用傳統資訊檢索技術，針對各資料中文件做特徵抽取（Feature extraction），並將文件表達成特徵向量（Feature vector）後，計算各文件之間的相似度，依此做為分群依據。接著，依照各資料庫的群集資訊，將具有相似文件內容的資料庫再做一次分群合併，形成階層式語意層疊式網路。因此當查詢產生時，該查詢文件會計算其特徵向量後，並根據其向量與各資料庫內群集資訊，將查詢文件路由至層疊式網路最上層的資料庫搜尋，最後將各語意層疊式網路的查詢結果進行合併後回傳給使用者。以下我們會詳細介紹此方法中，如何進行相似文件搜尋。

2.2.1 向量空間模型

向量空間模型（Vector Space Model）[11] 主要是用來表示文件在空間上的關係。在此模型下，一份文件通常可以表示成為一個字詞向量 [12]，而每一向量內的元素代表著其對應的字詞在文件中的重要性。各元素的計算是依據 TF/IDF 的加權技術處理，其中 TF（Term Frequency）代表字詞在文件中的頻率，而 IDF（Inverse Document Frequency）表示擁有該字詞的文件在資料庫中的頻率。此加權技術背後的想法，是若某一字詞在文件中出現的越頻繁越可以作為這份文件的代表字詞。任何查詢在此空間中

一樣表示成向量，且查詢與文件之間的相似度是採用餘弦相似度做計算，以下是在 n 維向量空間中，存在著兩特徵向量 A 與 B 做餘弦相似度的計算公式：

$$\textbf{Cosin Similarity} = \frac{\boldsymbol{A}\cdot\boldsymbol{B}}{\|\boldsymbol{A}\|\|\boldsymbol{B}\|} = \frac{\sum_{i=1}^{n}\boldsymbol{A_i}\times\boldsymbol{B_i}}{\sqrt{\sum_{i=1}^{n}(\boldsymbol{A_i})^2}\times\sqrt{\sum_{i=1}^{n}(\boldsymbol{B_i})^2}}$$

2.2.2 語義層疊式網路之內文檢索方法

在[1]的研究中，主要是透過語義層疊式網路的建構（Semantic Overlay Network Construction, SON）來解決在端對端網路環境下，進行內文檢索的問題。該研究所提出建構在分散及分佈式語義層疊式網路下，執行相似文件搜尋的架構（SON-based），主要是先利用傳統資訊檢索技術，針對各節點資料庫中文件做特徵抽取，並將文件表達成特徵向量後，計算各文件之間的相似度，依此做為分群依據。接著，依照各節點資料庫的群集資訊，將具有相似文件內容的節點資料庫再做一次分群合併，形成階層式語意層疊式網路。因此當查詢產生時，該查詢文件會計算其特徵向量後，並根據其向量與各資料庫內群集資訊，將查詢文件路由至層疊式網路最上層的資料庫搜尋，最後將各語意層疊式網路的查詢結果進行合併後回傳給使用者。

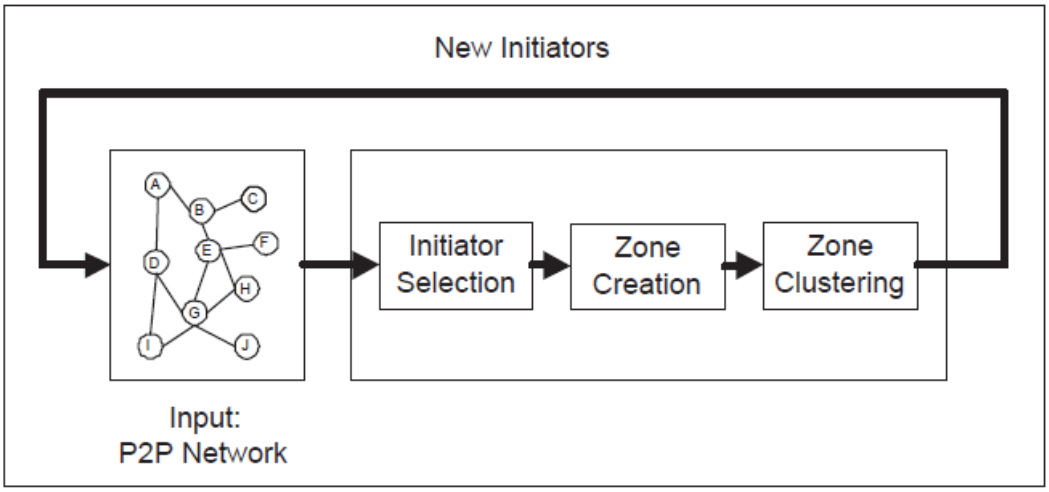

圖 2.2：分散及分佈式語義層疊式網路架構流程（引自［1］圖 1）

而語義層疊式網路的建構，是針對特定節點集合遞迴運行的一種多階段分散式程序。如上圖 2.2 所示，假設各個節點(例如網站)皆儲存著文件集合，且節點間存在連線，當有一非結構化的端對端網路節點輸入作為起始來源時，首先，會對其各節點資料庫作文章前處理，並以隨機的方式去選取一些特定節點當作為啟動節點，而所謂的啟動節點主要是負責建立區域並且管理區域內的節點資訊，此步驟即是啟動節點選取（Initiator Selection Phase)。接著，啟動節點會對其鄰居節點建立局部性的拓撲區域，此為區域建立步驟(Zone Creation Phase)。此時，每個啟動節點會去收集其組成的各個區域內，節點集合所擁有的群集資訊，並且執行群集合併演算法（Cluster Merging Algorithm)，如表 2.1 所示，去建構出語義層疊式網路，此為區域分群步驟（Zone Clustering Phase)。

表 2.1：群集合併演算法(引自［1］第 3.2.3 節)

Algorithm 1 Cluster merging.

Input: Clusters C=$\{c_1 ... c_N\}$, Limit $L < N, k$
Output: Clusters C'=$\{c'_1 ... c'_L\}$

while $sizeOf(C) < L$ **do**
  getMostSimilarClusters(C, c_i, c_j)
  Cluster $c_{new} \leftarrow c_j$
  for $(t_i \in c_i)$ **do**
    if $(t_i \in c_j)$ **then**
      $w_{new} \leftarrow \frac{tf_i+tf_j}{\sum_{c_i,c_j} tf} \times log(\frac{1+D_i+D_j}{df_i+df_j})$
      c_{new}.update($t_i, w_{new}, tf_i + tf_j, df_i + df_j, D_i + D_j$)
    else
      $w_{new} \leftarrow \frac{tf_i}{\sum_{c_i} tf} \times log(\frac{1+D_i+D_j}{df_i})$
      c_{new}.add($t_i, w_{new}, tf_i, df_i, D_i$)
    end if
  end for
  c_{new}.keepTopFeatures(k)
  C.delete(c_i)
  C.delete(c_j)
  C.add(c_{new})
end while
$C' \leftarrow C$
return C'

表 2.1 的群集合併演算法主要是依據各個節點內的資料內容進行合併，挑選出兩個具有相似內容的群集後，接著判斷該相似兩群集是否具有相同字詞特徵，再透過權重值的重新定義，去保留固定數量的字詞特徵後形成新的群集，便稱作一個語義層疊式網路(SON)。藉由此步驟遞迴的執行，直到所屬區域內的相似群集皆合併完成，形成複數個 SON 便終止合併，因此各區域內便形成了多個 SON。

在區域分群步驟中，主要是藉由該分群演算法遞迴構成語義層疊式網路。因此，在進行群集合併演算法的過程中，必須去考慮到當不同節點內的相似群集在做合併時，所產生的一個語義層疊式網路（SON）當中，舊有的群集內，各自的文件內容如何以新的群集資訊去描述。藉由該演算法遞迴的執行，直到各區域內的相似群集皆做完合併後，且群集數量達到特定數目時，便可以此表達區域內新的群集資訊。如下圖 2.3 所示：

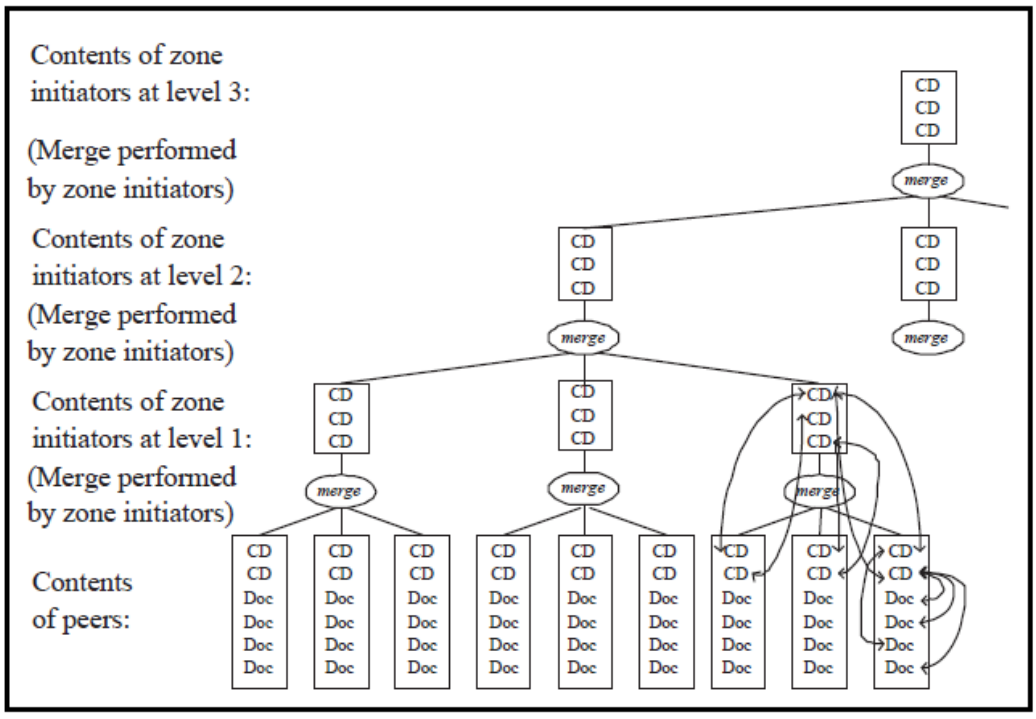

圖 2.3：遞迴群集合併的示意圖（引自［1］圖 2）

而區域分群步驟中又包含了兩類型的分群處理：一種是區域內分群(Intra-zone Clustering)，如同上述的群集合併步驟是發生在區域內的一種合併行為；另一種則是跨區域分群(Inter-zone Clustering)，將各區域間相似的語義群集，再做一次合併動

作，並以數個虛擬的啟動節點作為代表，建構出階層式的區域與啟動節點，以此作為最後的語義層疊式網路，如下圖 2.4 所示：

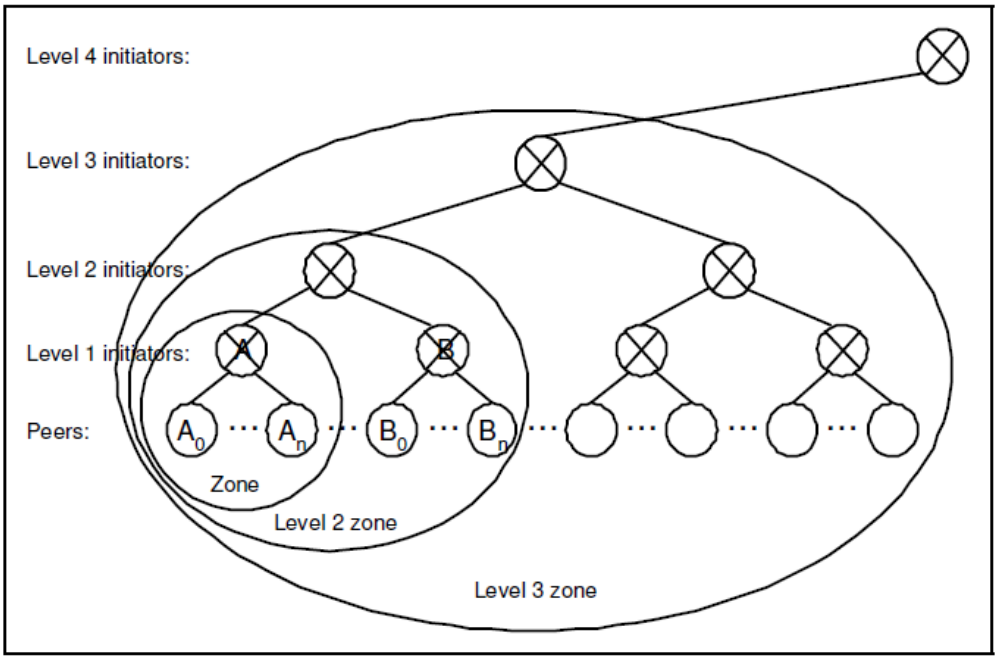

圖 2.4：階層式區域與啟動節點的構成示意圖（引自 [13] 圖 1）

當查詢下達時，便可至與查詢相似的語義群集內做搜尋，並回傳出相似結果文件給使用者作為參考。但是因為語義層疊式網路的方法在群集合併時，所建立的新群集資訊中，其包含的群集特徵向量的數目有所限制，容易造成我們在未來查詢處理時必須接觸大量節點資料庫進行搜尋，使得整體查詢成本過於昂貴。因此本研究希望透過改良式架構去解決此問題，並提供一個有效率且可擴展性的搜尋機制，使得我們在進行 Top-k 相似文件搜尋時不需將實際資料做搬移或備份。

我們的方法是採用建立特徵索引表的方式，來解決此類問題，並在前處理步驟中，重新定義各特徵字詞所採用的權重計算方式，而建立特徵索引表的過程中，亦會以特定的方法加速建立此特徵索引表。以下章節會詳述本研究的方法細節與架構流程。

第三章 方法描述與實作

在這一章節中，我們將完整介紹演算法的設計流程。本研究的目的希望可以在高度分散式環境下，使得 Top-k 相似文件的查詢可以有效率的處理，並且達到最大的準確率(Recall)。由兩種前處理的方法組成與特徵索引表的建立，以達到上述預期效果。

我們的流程分成三個步驟,首先,會利用分群演算法先針對各節點資料庫作前處理，並且透過區域切割和建立特徵索引表，最後達到減少查詢時間的效果。在過去的分散式環境上，當需要進行 Top-k 相似文件檢索時，我們無法單純的將每個節點資料庫中的 Top-k 結果做合併後，輸出為查詢結果，因為這樣產生的數量過於龐大且耗時，而是希望藉由少部份的資訊提供者即可回傳結果。因此我們提出的架構中，主要概念是希望藉由前處理事先建立的特徵索引表來加速查詢處理，在此章節我們會詳細的介紹整體架構流程。

3.1 文章前處理

首先，在前處理步驟中，我們會先針對各文件資料庫內的文件作特徵抽取，包括三階段的處理，利用傳統資訊檢索常用的技術，諸如斷詞切字 (Tokenization)將語句拆解為單字，包含空白及標點符號等（表 3.1)、去除停用字 (Stop-word removal)、詞幹還原(Stemming) 等步驟，並且根據各資料庫內的每一份文件內容計算其詞頻 (Term Frequency, tfi) 做為權重後，得到數個特徵向量做為該文件的壓縮資訊描述。

去除停用字是透過 Frequency-based Indexing Method 的方法，先將文件內的字詞分類為功能字詞 (Function words) 和內容字詞 (Content words)兩大類，前者常用於表示文法結構，像是 the, and, or…等常出現的介詞；後者則是實際給予文件內容涵義的字詞。舉例來說：「This lazy dog is sleeping on the floor.」中，即包含了四個

功能字詞（This, is, on, the）與四個內容字詞（lazy, dog, sleeping, floor），分別用來表示文章結構及內容。接著採用一停用字列表（Stoplist)(表 3.2）將功能字詞包含於其中，這些字詞雖然經常在文件中出現，但對於文件內容意義貢獻不高，且不具備鑑別力，因此在未來的查詢處理上是必須被忽略的，如此一來也可降低文件在特徵空間之維度，以便加速未來在分群演算法上的速度。

而詞幹還原則是還原字根，主要是為了統一詞性與時態上的變化，因為若沒有執行此步驟，會使得同義詞或相同字詞的變化時態擁有不同的特徵向量而產生錯誤，例如：automate, automatic, automation 屬於同義字，必須表示為同一字。因此我們採用波特詞幹還原演算法（Porter Stemming Algorithm）進行還原字根的處理。

表 3.1：斷詞切字的範例

輸入：This lazy dog is sleeping on the floor.
輸出：<This, lazy, dog, is, sleeping, on, the, floor>

表 3.2：停用字列表的部分範例

a	anther	being	c
about	Any	below	can
above	anyhow	beside	co
across	anything	both	could
after	are	but	cannot
all	at	beyond	down
also	an	behind	during
among	and	been	eg
…	…	…	…

接著用剩餘有意義的字詞，計算在文件內的詞頻權重，並且進行排序（Ranking）動作，最後每份文件只保留權重高的 Top-k 字詞作為特徵向量。以下是權重計算公式 [19] 以及相關變數說明：

$\boldsymbol{W_t}$：字詞 t 的權重　　$\boldsymbol{tf_{t,d}}$：字詞 t 在文件 d 內的詞頻

$\boldsymbol{N}$：此節點資料庫內的總文件數　　$\boldsymbol{d_t}$：擁有此字詞 t 的文件數量

$$\boldsymbol{W_t} = \begin{cases} \dfrac{\mathbf{1}}{\sqrt{\sum_{j=1}^{d}\left(tf_{t,d} \times \log\left(\dfrac{N}{d_t}\right)\right)^2}}, & if\ tf_{t,d} > 0 \\ \mathbf{0}, & otherwise \end{cases}$$

字詞 t 在文件 d 內的詞頻，需要再做正規化動作，使該字詞的權重合理化。透過權重計算後，每份文件皆有 k 個字詞的特徵向量（Feature vector, Fi），而每個特徵向量都包含該字詞特徵（Term Feature, fi）、權重(Weight, wi)，可表示為：$F_i = \{f_i, w_i\}$，使得每份文件皆可對應至特徵向量空間中。

3.1.1 分群的概念

另外，為了使未來在高度分散式環境下的查詢處理可以有效率的進行，並且減少計算時間，我們需再針對各節點資料庫內的文件進行分群動作。分群演算法是一個影響我們整體方法的準確率很重要的因素，透過精確的分群演算法我們可以有效的提升查詢處理時的準確率，因此在本節中，我們將介紹為什麼需要使用分群演算法以及採用的分群演算法細節。

一般在高度分散式的環境之下，進行相似文件搜尋時，往往因為欠缺一個完整的全域資訊，使得在搜尋過程中，必須去擷取到各個節點資料庫內的所有文件詳細內容，使用者才可正確得到整體環境內的 Top-k 相似文件結果，而這樣的步驟既費時又不具效率的。因此我們希望透過分群的概念，事先針對各節點資料庫內的文件進行分群處理，將相似的文件分至同一群集，並以數個特徵向量表達此群集內的文件分佈，使得各資料庫

可以較少量的群集特徵向量來表達所擁有的大量文件資訊。

3.1.2 分群演算法

在分群演算法的挑選上，我們使用的是階層式聚集分群演算法（Hierarchical Agglomerative Clustering, HAC）（表 3.3），對同一資料庫下的文件集合做分群。該方法的分群流程，首先是在特徵空間中，將每一份文件轉換為特徵向量，因此在分群過程中，我們會將每份文件視為一個各自獨立的群集，然後每次迭代皆選取距離最近的兩個群集做合併，形成新的群集，反覆遞迴直到分群數目達到系統要求為止。最後依此邏輯逐序將單一資料庫下的所有文件，建立成一階層群集樹（Hierarchical Clustering Tree, HCT）（圖 3.1），而兩群集之間在特徵空間中的距離是以餘弦相似度（Cosine Similarity）做為計算方式。以下定義我們在分群演算法中，所使用到的變數說明：

N_C：系統要求的分群數量　　C：目前系統內的群集數量

α、β：目前系統內分群的子群集　　γ：合併後產生的新群集

表 3.3：階層式聚集分群演算法

ALGORITHM : Hierarchical Agglomerative Clustering(*NC*, *C*)

```
1:  Let each document d be in a singleton cluster {d}
2:  Let C be the set of clusters
3:  while |C| > NC do
4:      choose α, β from C, according to cosine similarity cos(α, β)
5:      remove α and β from C
6:      Let γ = α ∪ β
7:      insert γ into C
8:  end while
9:  end ALGORITHM Hierarchical Agglomerative Clustering
```

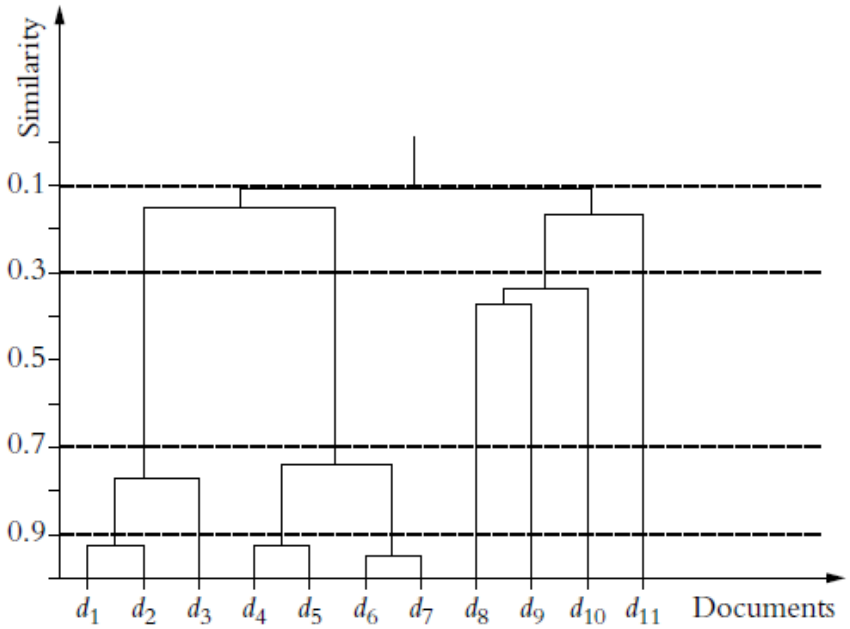

圖 3.1：階層群集樹示意圖

如圖 3.1 所示，在分群的過程中，各個節點資料庫必須設定一群集相似度門檻值(Cluster Similarity Threshold)，做為文件彼此之間的分群依據。一般來說，在階層群集樹較低層的兩群集，若需做合併處理，必須擁有較高的相似度才可執行，而此門檻值會隨著群集合併的規模變大而調降。因此，系統便可依據相似度門檻值大小的調整，建構出各個節點資料庫合適的群集數量。

而在分群演算法的過程中，當同一節點資料庫中的兩群集透過餘弦相似度計算後，大於事先設定的群集相似度門檻值時，則兩群集的特徵向量必須做合併，當中的字詞特徵也需做聯集（Union）運算。因此，各個字詞特徵的權重也須重新計算：

$\mathbf{W_t}'$：字詞 t 的新權重

$\mathbf{w_{t,C_i}}$：字詞 t 在群集 C_i 內的詞頻

$\mathbf{D_{C_i,C_j}}$：在群集 C_i 和 C_j 內，擁有字詞 t 的文件數量

$$W_t{}' = \begin{cases} \log \dfrac{\sum_{C_i,C_j} w_{t,C_i} + w_{t,C_j}}{D_{C_i,C_j}}, & if\ w_{t,C_i}, w_{t,C_j} > 0 \\ 0, & otherwise \end{cases}$$

最後，每份文件群集會根據新的權重，對群集內的特徵向量做排序，保留住 Top-k 的特徵向量，作為該群集的壓縮資訊描述，如表 3.4。上述的前處理步驟是必須在所有的節點資料庫中區域性執行，藉由兩步驟的前處理，各個資料庫已先擁有系統性的整合資訊，方便未來查詢處理時利用。

表 3.4：節點資料庫中各群集 Top-k 字詞特徵的存放圖

Peer$_i$	Cluster$_{C1}$		Cluster$_{C2}$		Cluster$_{C3}$	
Top-1 feature	feature$_{f1}$	0.87	feature$_{f4}$	0.74	feature$_{f7}$	0.92
Top-2 feature	feature$_{f2}$	0.75	feature$_{f5}$	0.7	feature$_{f8}$	0.85
Top-3 feature	feature$_{f3}$	0.72	feature$_{f6}$	0.64	feature$_{f9}$	0.76

3.2 特徵索引表建立

在各個資料庫做完前處理動作後，我們必須選擇出數個啟動節點（Initiator Peer）對整體的高度分散式環境做區域切割（Zone Creation）[1]。為何要做區域切割：主要是希望可將龐大的 P2P 網路，在邏輯上切割成數個子區塊（Sub-Group），而所謂的子區塊即是區域（Zone）。而每個區域由單一啟動節點負責各別運算，並收集該區域內所有節點擁有的群集特徵向量後，建立屬於各區域內的特徵索引表（Feature Index Table）。

當未來有查詢產生時，我們可透過特徵索引表，搜索出與查詢相關的 Top-k 群集所在的節點位置，並且進一步搜尋。因此，藉由區域切割與特徵索引表建立的方式，在後續的查詢處理過程中可有效減少運算時間，且確保回傳結果的數量，進而達到負載平衡(Load-Balancing)，接下來我們會描述如何挑選啟動節點以及區域切割的方法，最後則是特徵索引表建立的過程。

3.2.1 挑選啟動節點的想法

首先，所謂的啟動節點是在整體高度分散式環境中，具有特定工作的節點，負責決定各節點資料庫在前處理步驟中的分群演算法的執行周期，以及收集該資料庫下的所有群集之特徵向量，在功能上類似於超級節點。而啟動節點的數量及位置，也會影響後續的特徵索引表建立的完整性。因此，我們在挑選啟動節點上，盡可能滿足以下原則，並透過下列計算方法，使得我們的挑選過程盡可能合理化，以下是啟動節點的挑選公式、原則以及相關說明：

$\boldsymbol{IP_{P_i}}$：節點P_i的網路位置

$\boldsymbol{T}$：分群演算法的執行周期

$\boldsymbol{S_z}$：預設可建立的啟動節點數量

$$\left(\boldsymbol{IP_{P_i}} + \boldsymbol{T}\right) \text{MOD } \boldsymbol{S_z} = \boldsymbol{0}$$

當節點各項因素滿足上述公式時，才可被選擇作為啟動節點。所以須先假定整體高度分散式環境中，我們可建立的啟動節點數量（*Sz*），而公式中納入時間周期（*T*）為考量，是為了避免在高度分散式環境下的因為資料變動，而選擇到不適合的啟動節點，從而影響區域分割的可靠性。

原則一 ： 挑選出的啟動節點個數必須適量

原則上，挑選出的啟動節點並無數量上的限制，任何滿足上述計算公式的節點皆可做為啟動節點。但是，當端對端網路中擁有過多啟動節點時，表示需要切割相同數量的區域空間，這樣的結果在後續建立特徵索引表的演算法中，並無法大幅降低我們查詢處理時，所必須接觸到的節點數量。因為擁有大量的啟動節點，也代表著各個啟動節點僅能收集較少數節點資料庫中的群集特徵向量，所建立的特徵索引表也較無代表性意義。

因此，必須考量原始端對端網路中總節點數量，並採用相對適當個數的啟動節點運作，使得切割後的區域擁有的節點個數適量，這樣建立出的特徵索引表，所包含之群集特徵向量資訊才具有代表性。

原則二 ： 挑選出的啟動節點需均勻散佈在端對端網路環境中

由於在我們演算法中，啟動節點所負擔的工作量，相較於正常的端對端網路的節點來說，是較為龐大的。不僅要收集所屬區域內所有的群集特徵向量，並對此大量資料做重覆性的比對後，建立出專屬的特徵索引表，更需負責傳遞各區域經過查詢處理後的結果集合，以及未來在節點資料庫的文件內容有所變動時，可有效地管理分群演算法執行的周期時間。

當挑選出的啟動節點彼此之間過於密集且座落在端對端網路環境的末端時，容易造成特定者負擔工作量過於龐大，等同於單一啟動節點所負責處理的工作量，要包含整個端對端網路的所有文件資訊，往往造成未來在查詢處理時的一大瓶頸。如下圖 3.2 所示：

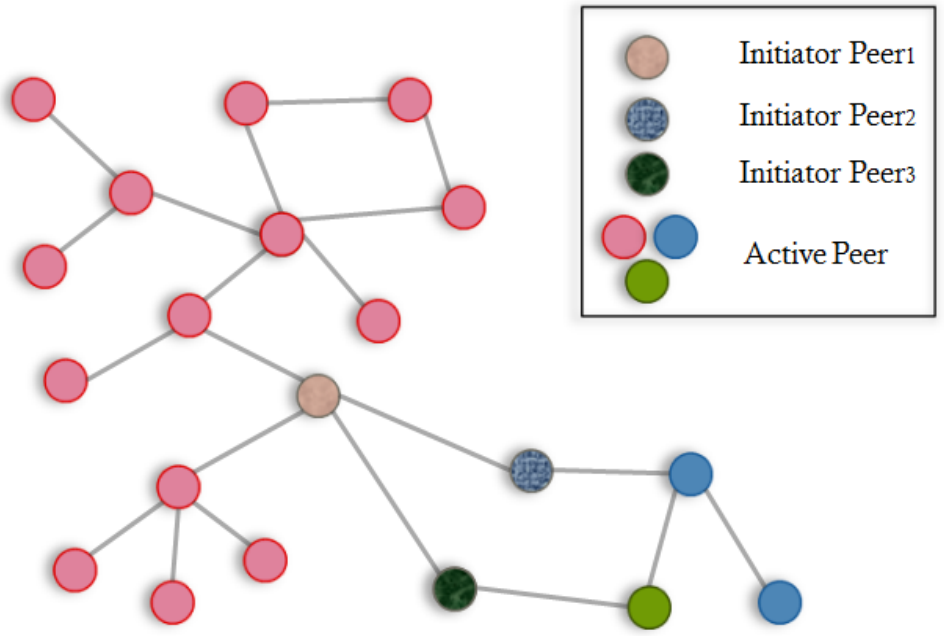

圖 3.2：挑選出的啟動節點過於密集時的示意圖

為了避免上述狀況，因此我們在透過計算方式挑選出數個啟動節點後，必須再檢驗各啟動節點間位置與彼此相對距離，盡可能使挑選出來的啟動節點均勻散佈在整個端對端網路環境中。所謂的「均勻散佈」也就是彼此之間的實體網路位置距離，需約略在一個範圍之內，且分布位置要涵蓋整體端對端網路環境，才可達成分工效應，而完成負載平衡。

3.2.2 執行區域切割的方法

當決定完各個啟動節點之後，每個啟動節點會根據預設的區域節點數量上限，執行訊息擴散演算法（Flooding-based Algorithm），將整體端對端網路環境在邏輯上切割成數個區域。每個啟動節點會先對周遭直接連結的鄰居節點發送一探測訊息（Probe Message），標記該節點成為此區域內的成員節點，如下圖 3.2 所示：

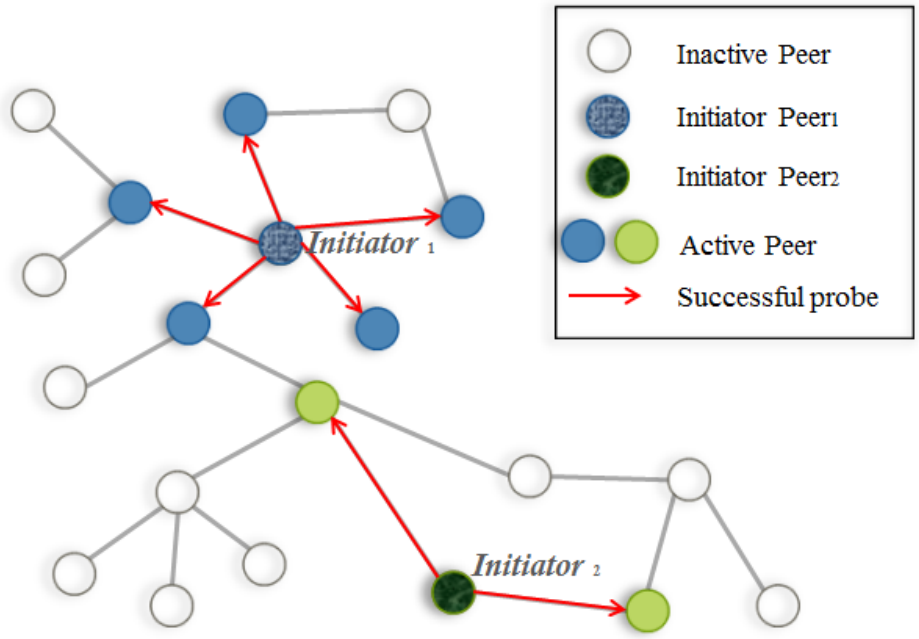

圖 3.3：啟動節點執行區域切割過程的示意圖

而成員節點會再對其鄰居發送相同的探測訊息，最後直到 P2P 環境內的所有節點皆各自屬於某個區域控管。如下圖 3.3 所示：

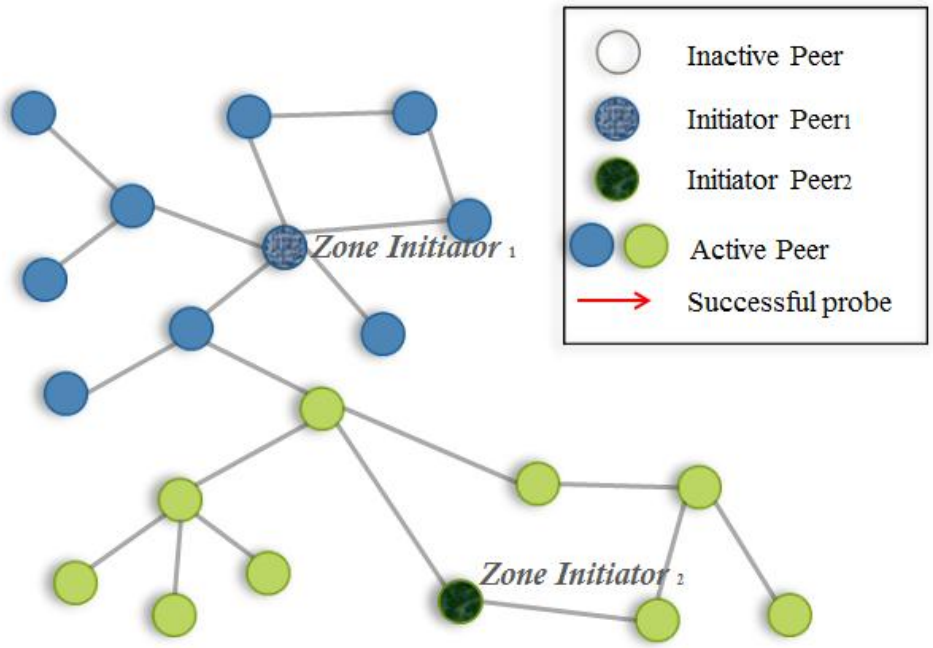

圖 3.4：P2P 環境切割成兩個子區域的示意圖

在擴散的過程中，若成員節點重覆收到不同啟動節點發出的探測訊息時，會對後來抵達的發送方告知已屬於特定區域，並且通知本區域的啟動節點相關資訊。當區域切割的程序結束後，P2P 環境中的每個節點必定會屬於特定區域控管、且每個啟動節點會知道該區域內的節點位置以及本身周遭鄰居的啟動節點資訊。而各個挑選出來的啟動節點，在建構超級節點網路的過程中，我們將其視為超級節點，負責收集區域內的整合資訊。

3.2.3 建立特徵索引表演算法

接著，在此一章節中，我們將介紹整體架構流程所採用的演算法，藉由前述的文章前處理所產生的文件資訊，以及分群處理過後得到的群集特徵描述。我們如何透過這些用來描述原始文件內容的元資料（Meta-data），以及經過啟動節點的挑選與區域切割後，去建立整個 P2P 環境的特徵索引表。

以下定義在我們演算法中，所使用到的變數說明和演算法流程：

N：端對端網路環境上的節點數量

I_N：系統要求的啟動節點數量

T_C：系統設定的分群門檻值

表 3.5：建立特徵索引表演算法

ALGORITHM : Building Feature Index Table(N, I_N,T_C)

1: **for each** peer
2: **for each** document
3: Generate its feature vector f_{di} by ***VSM***
4: **for each** document vector f_{di} vector
5: Compute cosine similarity cluster between document vectors
6: Merge two similar cluster which threshold exceed T by ***HAC***
7: Generate peer's weighted cluster summaries f_{Cj}
8: Pass f_{Cj} to its Initiator
9: Select Initiator to do *Zone Creation*
10. **for each** Initiator
11. Collect each peer's weighted cluster summaries within the *zone*
12. **for each** peer's weighted cluster summaries f_{Cj}
13. Reducing amount of feature vector to Top-m
14. Generate represented weighted cluster summaries f_{RCk}
15. Eliminate redundant feature vector
16. Generate *zone*'s group weighted cluster summaries f_{GCm}
17. Pass f_{GCm} to other Initiators
18. **end ALGORITHM Building Feature Index Table**

如上表 3.5 的演算法所示，首先，對於每一個節點中其所擁有的文件，必須先透過向量空間模型（Vector Space Model, VSM），將其表示特徵向量 f_{d_i}（第 3 行）。接著，針對每一個文件所擁有的特徵向量，去計算文件彼此之間的餘弦相似度值（第 5 行），並且透過階層式聚集分群演算法，將同一節點內的兩個相似群集合併起來後，得到各節

點的群集特徵彙整資料 f_{C_j}，並且將此資訊路由至該區域內的啟動節點（第 6-8 行），在選定啟動節點的同時，也同步地執行區域切割程序（Zone Creation）（第 9 行）。

當端對端網路環境已被切割成數個子區域後，透過各個區域內的啟動節點，去收集該區域內所有的群集特徵彙整資料（第 11 行），並將其存放至啟動節點所在的資料庫內。而這些收集來的群集資訊，啟動節點會一一將其原始的 Top-k 向量刪減至 Top-m 向量（第 13 行），並將這些特徵向量做為各群集新的代表性群集特徵彙整資料 f_{RC_k}，並且去除具有重覆性的字詞特徵（第 14-15 行）。最後，便以這些剩餘具有代表意義的字詞，去建構出區域的字詞特徵索引表 f_{GC_m}，並路由給其他的啟動節點（第 16-17 行）。

值得注意的是，在我們的演算法流程中，各個啟動節點會針對該區域內所有節點資料庫，收集其擁有的群集特徵向量。並且為了避免在建立特徵索引表時，將群集內過多與查詢相關程度低的字詞特徵向量納入考慮，因此各個節點資料庫內原始的群集特徵向量，會根據特徵字詞的權重高低做排序後，僅回傳 Top-k 當中的前 m 個字詞特徵所組成的 Top-m 群集特徵向量，將此資訊作為每一群集的代表性字詞特徵向量（Represented feature vectors），以便未來給予啟動節點建立特徵索引表。值得注意的是系統設定的 m 數量大小必定小於 k（m < k）。

此時，各區域內的啟動節點，須將收集來的所有群集特徵向量，再做一次去除重複性（Eliminate redundant）的動作，將相同的字詞特徵過濾掉，得到該區域內所有的字詞特徵集合（Feature Sets）。最後，各個啟動節點以此字詞特徵集合，建立出屬於該區域的字詞特徵索引表（Feature Index Table），內容包含區域內所有節點資料庫中，各群集對於字詞特徵集合內，各個字詞特徵的權重列表，如下表 3.6 所示。

表 3.6：各區域的字詞特徵索引表

Zone*i*	Feature*1*	Feature*2*	…	Feature*i*
Cluster*i1*	0.87	0.95	…	0.41
Cluster*i2*	0.67	0.56	…	0.33
Cluster*i3*	0.54	0.66	…	0.7
…	…	…	…	…

為何要建立特徵索引表：當整體 P2P 環境有任意的查詢產生時，啟動節點可針對各自區域，提供一個全域資訊作為協助，使得查詢不需再路由至所有的節點資料庫中，各自做相似文件搜尋。藉由此列表，我們可迅速得到不同區域內與查詢文件最相似的 Top-*k* 文件群集，僅針對此數個群集再做相似文件比對，避免了大量且重複的運算行為。

另外，由於在高度分散式環境下，節點資料庫的加入與離開、以及資料庫內文件的增刪動作，是動態且頻繁的行為，可能導致查詢結果的品質越來越差，所以特徵索引表的內容必須做更新動作。因此，啟動節點必須管理在前處理中的分群步驟，並有機制地以規律的時間間隔重新再執行，以便取得各節點資料庫的最新文件內容，去反映當下的資料分佈。

3.3 查詢處理

假設使用者從任意節點資料庫中，發出的文件查詢（$\mathbf{Q_k}$），希望從整體端對端網路中，查詢出與該份文件 Top-*k* 相似的文件集合時，根據本研究所提出的架構，該節點資料庫會先針對查詢文件進行文章前處理程序，得到該文件的壓縮資訊描述，也就是數個特徵向量，並且將這些和查詢相關的字詞特徵向量，路由至發出查詢的節點其所屬區域的啟動節點內。

接著，啟動節點會根據上述演算法流程所建立出的字詞特徵索引表，比對所有與查詢文件有共同字詞特徵的群集，並列出此區域內與查詢相似度高的 Top-k 群集集合。啟動節點會再次將查詢文件的特徵向量路由到這些 Top-k 群集，其各自所在的節點資料庫中進行相似文件搜尋，去比對該群集內的所有文件，並得到一組結果列表（Result List），此結果列表包含了與查詢相似的所有文件，且相似度值皆高於系統設定的查詢相似度門檻值（$\mathbf{T_s}$）。

由於採用的是超級節點網路架構，因此每個啟動節點在查詢處理的過程中，是平行處理的且彼此互相連結傳遞資訊。該查詢也會同時路由至其他區域內的啟動節點，進行上述相同程序後，得到多組結果列表。再將這多組結果列表做合併動作（Result Merging）後，回傳至原始查詢的節點資料庫，再執行一次 Top-k 相似文件搜尋，最後根據與查詢文件的相似度值高低，我們僅會回傳 Top-k 的結果文件集合輸出給使用者參考。

當系統回傳的結果列表數量不足，或是所回傳的文件品質不滿足使用者要求的條件時，啟動節點必須將查詢再次擴展路由的群集數量，直到結果滿足使用者需求。倘若相似文件的結果提早滿足使用者需求時，該次查詢處理便可提前結束執行。

在後續的實驗結果中，我們的目標便是希望使得查詢處理所得到的文件集合，盡可能具有較高的準確率，接下來的章節會詳細描述我們的實驗設計與結果。

第四章 實驗方法與驗證

在此章節中我們將針對在高度分散式環境下執行 Top-k 相似文件檢索進行實驗，而實驗所使用的資料，是線上醫療資訊文件資料庫:Ohsumed。該文件資料庫是取自於 [14] 研究，從過去的西元 1987 年至 1991 年五年間彙整而成，共包含了三十五萬份的心血管醫療病例報告，構成目前在文本檢索領域中，經常作為實驗測試的文件資料庫。在 Ohsumed 文件資料庫中，每份病例報告已對該報告摘要內容分析後，預先進行分類，將所有文件分成二十三種不同的類別，因此在我們的實驗中，我們將隨機從這二十三種類別當中，挑選一百篇病例報告文件作為查詢集合。由於我們的實驗方法主要目的是模擬高度分散式環境下，進行 Top-k 相似文件檢索時，回傳結果的整體準確率 (Precision) 與查詢成本 (Cost)。所以為了避免在大量節點網路下的傳輸瓶頸影響查詢處理的效率，我們撰寫程式去模擬擁有大量節點的 P2P 環境，並且產生隨機的端對端網路拓撲架構。另外，也須測試我們的演算法在實際高度分散式網路環境的可擴充性，所以我們模擬出數種不同節點數量的 P2P 環境。最後，在前述的模擬環境下，實作出我們的方法架構與 SON-based 的方法，進行 Top-k 相似文件檢索時的效能差異。

在本研究所提出的問題中，為了幫助我們實驗的驗證，會以傳統集中式搜尋引擎 (Baseline Algorithm) 進行 Top-k 相似文件查詢的結果文件，作為標準答案。也就是去檢索所有文件當中包含查詢文件的所有特徵字詞，以便測試我們的方法架構與 SON-based 的準確率優劣，並以各自查詢過程中，須接觸多少個節點資料庫數量，才可回傳 Top-k 相似文件結果，作為整體查詢成本耗費的比較。

4.1 實驗設計

本實驗設計的整體步驟流程，主要分成五個步驟：首先將測試資料文件隨機分佈在節點中，並紀錄下各文件所在的節點網路位置。接著，對各節點資料庫擁有的文件，透過傳統資訊檢索技術做特徵抽取，將各文件表示成特徵向量，並紀錄每份文件的 Top-k 特徵字詞向量。並且根據這些特徵向量，設定一個門檻值，對各資料庫內文件做階層式聚集分群演算法，再紀錄各群集的代表特徵向量集合。再選定數個啟動節點做區域切割，並向各區域內節點資料庫發出訊息，收集所有的群集向量集合，最後建立各區域的特徵索引表。而我們整體實驗過程皆是在 Intel(R) Core i5-2400 3.10GHz 的核心處理器以及 8GB 的存取記憶體，搭配 Windows 7 64 位元作業系統的電腦規格下進行，並且以 Java 和 Matlab 來實作出本實驗的模擬環境，而實驗當中所有建立的資料表格皆儲存至 MySQL 資料庫，並以 phpMyAdmin 作為該資料庫的管理系統。

首先，我們的實驗必須撰寫程式去模擬出擁有大量節點的端對端網路環境，藉此我們透過喬治亞理工學院所提出的 GT-ITM 軟體 [15] 去模擬大型網路拓撲架構，並產生隨機的端對端網路拓撲架構。

另外，為了測試我們的演算法在實際高度分散式網路環境的可擴充性，我們透過此軟體模擬出兩種不同節點數量的 P2P 環境，並設定四種環境：節點資料庫數量（$\boldsymbol{N_p}$）分別為 100 個、500 個、1000 個與 2000 個。接著，我們採用 Ohsumed 線上醫療資訊文件資料庫，並將完整的三十五萬份文件作為整體端對端網路環境中，節點內容取樣的文件來源。這邊要注意的是節點在文件取樣的過程中，是隨機選取 $\boldsymbol{D}$ 份文件，因此各節點可擁有部份相同文件，而非將三十五萬份文件依照節點資料庫的數量做切割（$\boldsymbol{D}$ / $\boldsymbol{N_p}$），以符合實際 P2P 網路在作內文檢索時的狀況。

在設定好網路環境與資料分佈後，按照我們前述第三章節所採用的文章前處理程序，包含斷詞切字、去除停用字、詞幹還原等三步驟，對所有分佈在節點內的文件作特徵詞

彙抽取，依照其權重值的大小對各文件保留 Top-k 特徵字詞向量，建立其所屬的字詞文件表格（Term-Document Table），並且依各文件的特徵字詞向量，計算彼此在向量空間中的餘弦相似度，依據此相似度值作階層式聚集分群演算法，對各節點內所有文件作分群動作。而分群後各個群集仍需保留文件原有的 Top-k 特徵字詞向量，並且對所有特徵字詞取聯集後，重新計算群集的 Top-k 特徵字詞權重，以作為群集的代表資訊。最後，在選出數個起動節點後，我們會將整體端對端網路切割成數個區域，每個顏色即代表一個區域，如下圖 4.1 所示：

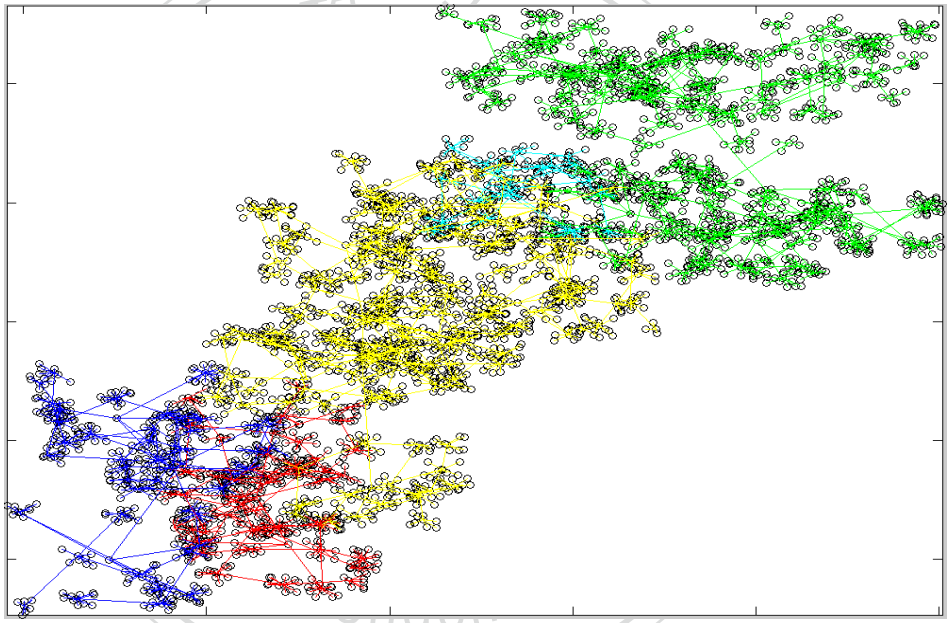

圖 4.1：區域切割後的端對端網路

當 P2P 網路內的區域切割完畢後，各啟動節點便會去蒐集各自區域內的特徵字詞向量，作為未來建立特徵索引表的依據。在演算法的比較部分，我們會與 SON-based Framework 做比較，SON-based Framework 也是在端對端網路環境下，找出與查詢文件 Top-k 相似的文件集合。使用的方法主要是透過語義層疊式網路的建構，來對具有相似內容的文件群集做合併，並且再針對具有相似內容的節點再做一次合併，形成跨區域的階層式語義層疊式網路，算是一種很直覺式的方法。

4.1.1 分群演算法實驗

由於分群演算法所產生出的分群結果，很明顯會影響我們的方法在查詢處理時的結果與搜尋速度。因此在實驗過程中，我們會針對不同節點數量的端對端網路環境下，去測試其各節點資料庫內群集之間的相似度平均值，該數值的大小也代表群集之間的分離程度：當群集之間的相似度平均值較大，代表分群結果是較差的，群集彼此所擁有的文件過於相似；若平均值低，則表示我們所採用的分群演算法，可有效的將文件依特徵字詞向量做分群。並且額外記錄各群集所擁有的 Top-k 特徵字詞向量個數，以及在該網路環境下的總群集數量。以上是我們的第一個實驗。

4.1.2 準確率與查詢成本比較實驗

在本研究所提出的問題中，為了幫助我們實驗的驗證，會以傳統集中式搜尋引擎 (Baseline Algorithm) 進行 Top-k 相似文件查詢的結果文件，作為標準答案。也就是去檢索所有文件當中包含查詢文件的所有特徵字詞，以便測試我們的方法架構與 SON-based 的準確率優劣，並以各自查詢過程中，須接觸多少個節點資料庫數量，才可回傳 Top-k 相似文件結果，作為整體查詢成本耗費的比較，而 SON-based 演算法在相關研究章節中已有做詳細的介紹了。

因此在查詢處理的實驗上，我們首先會針對 Ohsumed 文件資料庫的三十五萬分文件當中，隨機從這二十三種類別當中，挑選一百篇病例報告文件作為查詢集合。再從這一百份文件當中，挑選一篇文件作為查詢文件，並且計算其查詢文件與經過我們方法所得出的結果文件集合之餘弦相似度，再透過設定的相似度門檻值來決定有哪些文件是與查詢文件相似的，因此我們可以定義出檢索後得到文件集合的準確率。

準確率的公式定義如下：

$$\text{PRECISION} = \frac{R_q}{D_q}$$

相關變數說明如下：

R_q：回傳出的文件集合且與查詢相關的文件數量

D_q：所有與回傳出的文件總數量

查詢成本的公式定義如下：

$$\text{COST} = \sum N_{cp}$$

相關變數說明如下：

N_{cp}：在任一回合查詢處理的過程中，接觸的節點數量

另外要注意的是 SON-based 演算法在前處理步驟中，所採用的字詞權重計算方式與本研究所使用的權重差異，SON-based 在字詞權重計算是以［12］的計算公式定義如下：

$$W_t = \begin{cases} 1 + \log\big(n(d,t)\big) \times \log\dfrac{1+|D|}{|D_t|}, & if\ n(d,t) > 0 \\ 0, & otherwise \end{cases}$$

相關變數說明如下：

$n(d,t)$：字詞 t 在文件 d 出現之次數　　$|D|$：文件總集合

$|D_t|$：包含字詞 t 的文件集合

因此在實作上，並不會影響整體比較實驗的結果差異。由於 SON-based 的研究目的與我的研究目的相同，所以此篇是我的比較重點對象，我們期望透過我們的方法所回傳的結果文件，不論是在準確率或是查詢處理時的查詢成本上，我們都可以優於［1］所提出的演算法架構。

4.2 實驗結果

在此一章節中，我們首先比較我們設計的演算法與 SON-based 演算法的整體準確率以及查詢成本的大小。

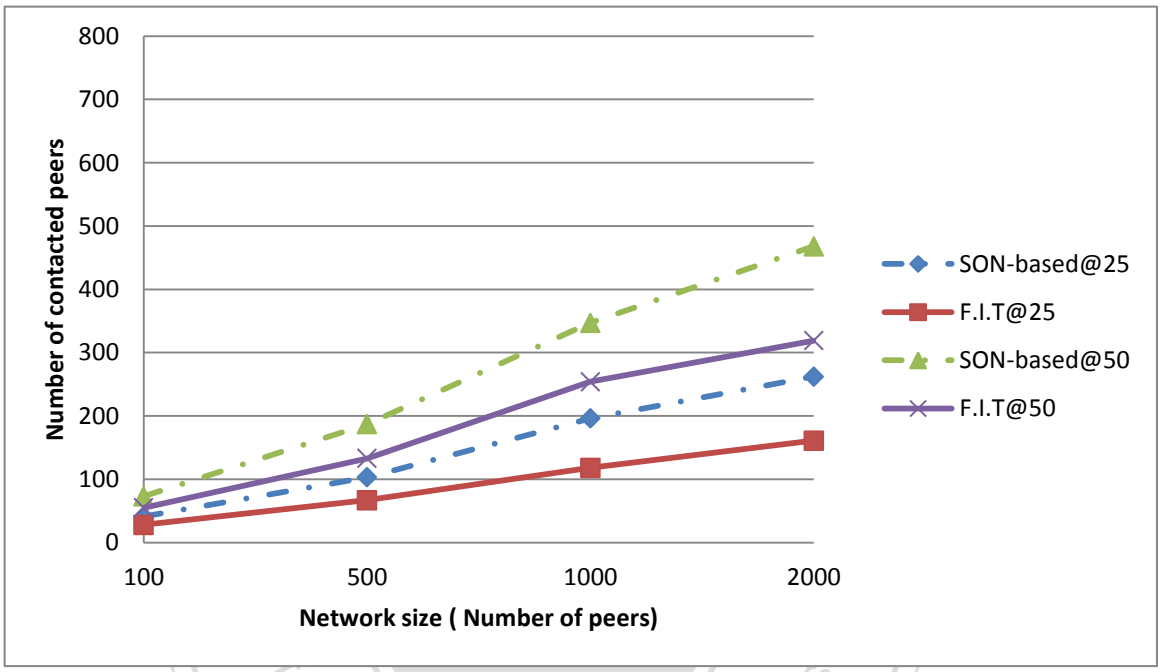

圖 4.2：比較兩種演算法在不同節點數量時的查詢成本比較圖

由上圖所見，下方的 100 到 2000 指的是端對端網路上的節點數料，左方的 0 到 800 指的是在整體查詢處理的過程中，所必須接觸的總節點數量，而兩種演算法皆測試在不同的 Top-k 查詢之下（Top-25 和 Top-50），平均執行 20 次的查詢成本平均值。我們可以看到的是 SON-based 演算法的查詢成本，隨著端對端網路環境中的節點數量增加，而呈現大幅增加查詢成本；而我們所提出的演算法在查詢成本上明顯低於 SON-based 演算法。原因是因為在建立特徵索引表時，我們記錄的是各區域內的群集代表資訊，在查詢處理的過程中，當查詢文件從任意節點發出請求時，我們會優先對於節點所在區域內的啟動節點查詢，並且針對特徵索引表內與查詢文件最相關的群集優先查詢，再到該群集所在的節點做進一步相似文件比對。若回傳結果文件數量不足或是不滿足使用者需求，

才依次到第二相關的群集檢索。

但 SON-based 演算法在查詢處理時，因前處理步驟中需多執行群集合併演算法，所建立的新群集資訊中，其包含的群集特徵向量的數目有所限制，且跨區域間仍然會再做一次群集合併，因此在查詢過程中，進行相同的 Top-k 相似文件搜尋時，該演算法必須探訪較多節點的群集才可回傳出結果文件。這也是我們的演算法在相同的 Top-k 查詢之下，可以減少查詢成本的主要原因。

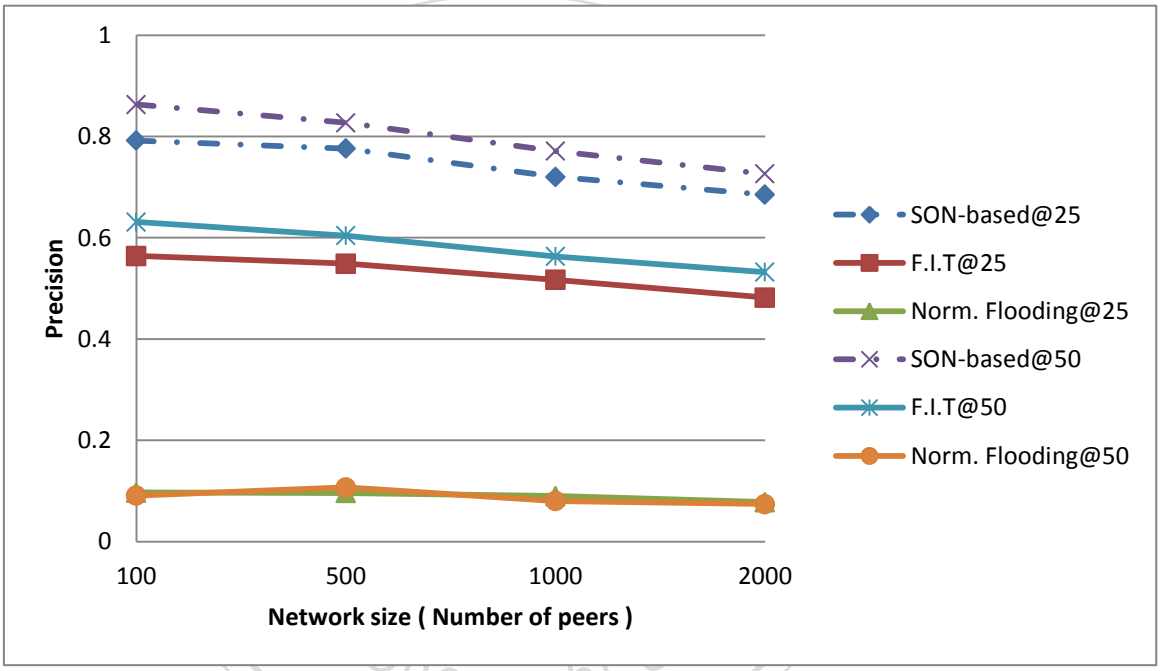

圖 4.3：比較兩種演算法在不同節點數量時的準確率比較圖

如上圖所見，SON-based 演算法在不同節點數量的端對端網路環境下，並測試在不同的 Top-k 查詢之下，其準確率仍然是最好的。而 Normalized Flooding Algorithm [18] 是一種傳統在端對端網路下執行文件檢索方法：Naïve Flooding Algorithm 的改良版，在過去的文件檢索領域，通常用來作為基礎比較對象。該方法主要是限制每個節點在路由查詢時可傳輸的鄰居個數，而非傳統方式是路由至每個節點的所有鄰居。透過實驗結果顯示，我們的演算法在不同的節點數量下以及不同的 Top-k 查詢之下，得到的準確率並沒有預期的好，所回傳出的結果文件其準確率都低於 SON-based 演算法。原因是因為

該演算法在查詢處理的過程中，所接觸到的節點數量是較多的，且檢索出並回傳的文件數量也遠多於我們所提出的演算法，因此在準確率上可明顯提升不少。

由於從我們的實驗結果來看，雖然我們在查詢成本上減少了不少需要接觸的節點數量，可是利用我們設計的演算法所得到的結果文件其準確率並不理想，因此我們再深入探討其準確率不佳的原因，並做了一組實驗，來驗證我們的演算法也可以達到相同的準確率。

首先，我們針對分群演算法做詳細的分析，因為分群演算法所產生出的分群差異，會影響我們的方法在查詢處理時的結果與搜尋速度。藉此我們設計在不同的節點數量之下，群集整體性質的變化，如下表 4.1 所示：

表 4.1：群集整體性質在不同節點數量下比較圖

Network Size	100	1000	2000
Number of Clusters	23	146	281
Avg. Cluster Similarity	0.027	0.031	0.032

由上表可知，我們所採用的分群演算法其分群結果是良好的，因為群集之間的相似度平均值都是很低的，該數值的大小也代表群集之間的分離程度，因此表示我們所採用的分群演算法，可有效的將文件依特徵字詞向量做分群。

接著我們探討在建立特徵索引表時，各啟動節點收集區域內的群集資訊時，所保留的代表性字詞特徵向量的數量，是否影響整體演算法的準確率。因此比較了系統在進行 Top-25 相似文件查詢之下，保留不同數量（m）的代表性字詞特徵向量的狀況下，我們的演算法在準確率的結果比較。如下圖 4.4 所示：

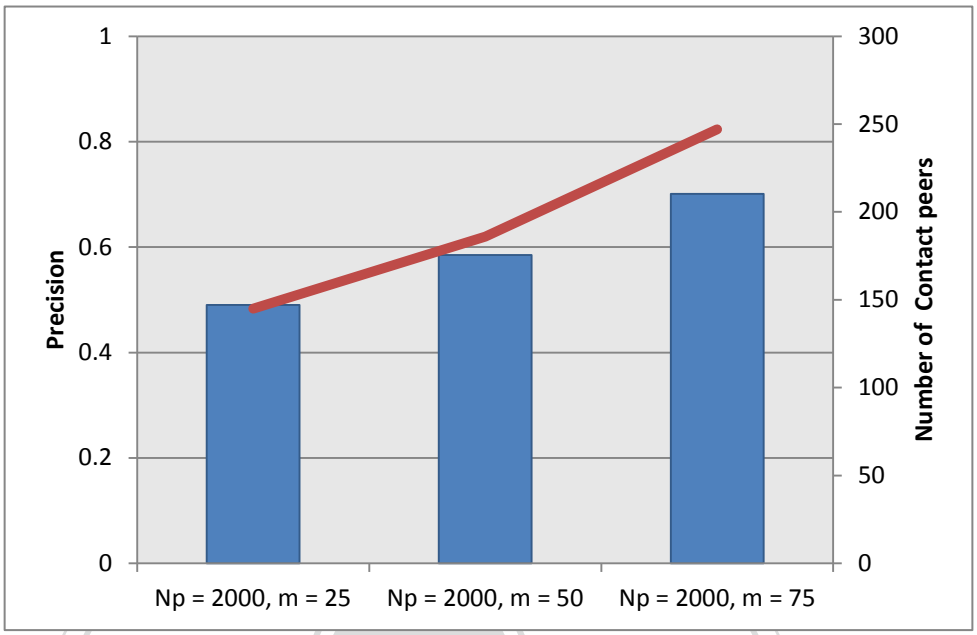

圖 4.4：比較在保留不同數量的代表性字詞特徵向量時的準確率比較圖

由實驗資料顯示，當我們的演算法在增加保留的代表性字詞特徵向量時，我們整體準確率可有效提升。原因是因為當各群集的代表性字詞特徵向量增加保留數量時，各區域內的啟動節點所建立出的特徵索引表，其所包含的群集資訊也會大幅增加，可有效提升未來在查詢處理時的準確率。但整體的查詢成本也因為代表性字詞特徵向量的增加而上升，原因主要是因為當我們增加保留數量時，會造成特徵索引表整體包含的群集資訊過於龐大，使得我們在查詢特徵索引表時，也必須接觸更多的節點才可以檢索出更多的相似文件。因此，此一實驗證明了我們所提出的演算法，在建立特徵索引表的處理機制還有很大的改進空間，在未來我們也會試著逐一去改善完成。

第五章 結論

本研究中提出的方法架構，在各種不同節點數量的 P2P 環境下，執行相同的 Top-k 相似文件搜尋時，我們的方法在查詢成本上是符合我們所預期的成果，但是在整體準確率上並不如預期理想，因此在準確率的部分還可以有很大的改進空間。

在端對端網路環境中，因為缺乏了一個完整的全域資訊與適當的資訊協調者，使得過去在 P2P 的內文檢索問題上，並無法有效率並且準確的查詢出相關文件。因此本研究中，我們針對各資料庫作分群前處理，並提出一個利用區域切割的作法，將 P2P 環境劃分成數個子區塊後，並建立特徵索引表。當在進行查詢處理時，便可透過特徵索引表去加快挑選出 Top-k 相似群集的速度，並且確保有適當數量的回傳結果。雖然準確率並沒有 SON-based 演算法來的優異，但是在查詢成本上卻可以降低不少，很適合做在即時的網路應用程式上。因此，未來將會透過更多實驗，去分析我們的演算法其整體準確率，並且改善建立特徵索引表時的篩選機制，這將會是我們未來繼續努力的目標與方向。

參考文獻

[1] Christos Doulkeridis, Kjetil Nørvåg, Michalis Vazirgiannis. 2008. Peer-to-peer similarity search over widely distributed document collections. *LSDS-IR* 35-42.

[2] Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., Balakrishnan, H. 2001. Chord : A scalable peer-to-peer lookup service for internet applications. *In Proceedings of the ACM SIGCOMM* 149-160.

[3] Ratnasamy, S., Francis, P., Handley, M., Karp, R., Schenker, S. 2001. A scalable contentaddressable network. *In Proceedings of the ACM SIGCOMM* 161-172.

[4] Rowstron, A., Druschel, P. 2001. Pastry : Scalable, distributed object location and routing for large-scale peer-to-peer systems. *In Proceedings of the Middleware*

[5] Chunqiang Tang, Zhichen Xu, Sandhya Dwarkadas. 2003. Peer-to-peer information retrieval using self-organizing semantic overlay networks. *In Proceedings of the ACM SIGCOMM* 175-186.

[6] BitTorrent. <http://bittorrent.com/>.

[7] eMula. <http://www.emula-project.net/>.

[8] Beverly Yang, Hector Garcia-Molina. 2003. Designing a Super-Peer Network. *ICDE* 49-60.

[9] The Gnutella protocol specification v0.6. <http://rfcgnutella.sourceforge.net>.

[10] KaZaA. <http://www.kazaa.com>.

[11] Salton, G., Wong, A., Yang, C.S. 1975. A vector space model for automatic indexing. *Communications of the ACM Volume 18 Issue 11* 613-620.

[12] Bernard J. Jansen, Soumen Chakrabarti. 2006. Mining the Web : Discovering Knowledge from Hypertext Data. *Morgan-Kaufmann Publishers, 352 pp., ISBN: 1-55860-754-4. Inf. Process. Manage. (IPM)* 42(1) 317-318.

[13] Christos. Doulkeridis, Kjetil Nørvåg, and Michalis Vazirgiannis. 2007. DESENT: Decentralized and distributed semantic overlay generation in P2P networks. *IEEE Journal on Selected Areas in Communications (J-SAC)* 25(1) 25–34.

[14] Hersh, W.R., Buckley, C., J.Leone, T., Hickam, D.H. 1994. Ohsumed: An interactive retrieval evaluation and new large test collection for research. *In Proceedings of the ACM SIGIR*. 192–201

[15] GT-ITM : Georgia Tech Internetwork Topology Models. <http://www.cc.gatech.edu/projects/gtitm/>.

[16] Wolf-Tilo Balke, Wolfgang Nejdl, Wolf Siberski, Uwe Thaden. 2005. Progressive Distributed Top k Retrieval in Peer-to-Peer Networks. *ICDE* 174-185.

[17] Wolf-Tilo Balke. 2005. Supporting Information Retrieval in Peer-to-Peer Systems. *Peer-to-Peer Systems and Applications* 337-352.

[18] C. Gkantsidis, M. Mihail, and A. Saberi. 2005. Hybrid search schemes for unstructured peer-to-peer networks. *In Proceedings of INFOCOM.*

[19] Inderjit S. Dhillon, Dharmendra S. Modha. 2001. Concept Decompositions

for Large Sparse Text Data Using Clustering. *Machine Learning 42*(1/2): 143-175.

[20] Akrivi Vlachou, Christos Doulkeridis, Kjetil Nørvåg, Michalis Vazirgiannis. 2008. On efficient top-k query processing in highly distributed environments. *SIGMOD* 753-764.

[21] Shiwei Zhu, Junjie Wu, Hui Xiong, Guoping Xia. 2011. Scaling up top-K cosine similarity search. *Data Knowl. Eng. (DKE)* 70(1) 60-83.

[22] Aoying Zhou, Rong Zhang, Weining Qian, Quang Hieu Vu, Tianming Hu. 2008. Adaptive indexing for content-based search in P2P systems. *Data Knowl. Eng. (DKE)* 67(3) 381-398.