國 立 交 通 大 學

資訊科學與工程研究所

碩 士 論 文

部落格分群

Blog Clustering

研 究 生：陳佑州

指導教授：李嘉晃 教授

中 華 民 國 九 十 九 年 六 月

部落格分群

Blog Clustering

研 究 生：陳佑州　　Student：You-Chou Chen

指導教授：李嘉晃　　Advisor：Chia-Hoang Lee

國 立 交 通 大 學

資 訊 科 學 與 工 程 研 究 所

碩 士 論 文

A Thesis
Submitted to Institute Computer Science and Engineering
College of Computer Science
National Chiao Tung University
in partial Fulfillment of the Requirements
for the Degree of
Master

in
Computer Science
Jun 2010

Hsinchu, Taiwan, Republic of China

中華民國九十九年六月

部落格分群

學生 ： 陳佑州　　指導教授：李嘉晃 教授

國立交通大學資訊學院 資訊科學與工程研究所碩士班

摘要

網際網路能夠如此快速的發展成為現代人生活中不可或缺的一部分，搜尋引擎的出現功不可沒，但是現今的搜尋技術幾乎都是以關鍵字來查找網頁，也就是當使用者輸入關鍵字之後，搜尋引擎幫忙找出含有這個關鍵字的網頁。假設你是一位部落格作者，擁有自己的部落格，搜尋引擎目前並無法根據你撰寫文章的主題來自動找出你可能有興趣閱讀的部落格，因為目前搜尋引擎無法根據部落格的特徵自動分群。因此，在本篇論文中，我們將研究如何將網路上的部落格根據其主題分群，藉此找出有相同興趣的作者。我們提出利用部落格的標籤雲來代表部落格的概念。標籤雲就是一個部落格中所有文章之標籤的集合，部落格上的標籤是由人工所標記的，很適合用來代表一篇文章的概念或主題，所以本系統直接以標籤雲來代表部落格而不是藉由分析每一篇文章來找出部落格的主題。得到部落格的表示法後，就可以計算部落格與部落格之間的相似度，接著再使用不同的分群演算法將部落格分群，比較其結果。根據實驗結果可知，我們幾乎可以很準確的把相同主題的部落格分在同一個群中，這代表同一個群中之部落格的作者都有著相同的興趣或喜好。

Blog Clustering

Student：You-Chou Chen　Advisor：Prof. Chia-Hoang Lee

Department of Computer and Information Science

National Chiao Tung University

ABSTRACT

Discovering social interests from user's blog content or social tags is one of the interesting and challenging problems in social network research. We tackle this problem using blog clustering based on the tags of blogs. In blog representation, we employ the tags of a blog to represent the blogger's interests and discover user's common interests using blog clustering. In this paper, we propose two kinds of approaches to tackle this problem. In the first approach, we employ spectral clustering to cluster the blogs in the concept vector space. The construction of concept vector representation is similar to dimensionality reduction. First, we regard the Web as system corpus to measure the relevance of two tags based on the hits returned from the search engine. Second, a balanced hierarchical agglomerative clustering, which takes into account the size of the clusters, is proposed to aggregate the tags that are relevant. Finally, the original tag vector representation can be transformed into its corresponding concept vector representation. The experimental results show that the F1 value can be improved a lot as compared with the clustering in the tag vector space. In the second approach, we propose to employ multidimensional scaling technique to perform dimensionality reduction and then apply K-means clustering in the reduced coordinates. The experimental results show that our approaches can effectively cluster the blogs with similar interests and it can be applied to other social network clustering easily.

誌謝

本論文可以順利完成，首先要感謝的就是我的指導教授李嘉晃教授。有了教授的指引，我在研究的過程中才不會手足無措；也謝謝教授的耐心指導，讓我對自然語言處理這個領域有更深的認識。我從教授的身上學到了做研究的方法，未來將成為我工作上的助力。接著要感謝三位辛苦的口試委員，王勝德教授、張道行教授與李漢銘教授，謝謝教授們的建議，讓本論文的內容可以更加完整。

再來要特別感謝我的學長，在我課業上有問題時，你們總是細心的指導我，讓我獲益良多。還要謝謝同屆的同學們，紀孝承、鍾喻安與楊瑞敏。不管在研究上或是課業上，我都從你們的身上學到很多東西；也因為有你們的陪伴，讓辛苦的碩士生涯增添了許多歡樂。謝謝實驗室的學弟們，因為你們在實驗上的幫助，讓我可以更快完成論文。謝謝交通大學提供一個這麼好的環境，讓我可以在這學習且有所成長。

最後，謝謝我的家人，你們總是一直支持我、鼓勵我。謝謝所有幫助過我的朋友們，我以此篇文章表達我誠摯的謝意。

目錄

圖目錄

表目錄

第一章、緒論

1.1 研究動機

網際網路能夠如此快速的發展成為現代人生活中不可或缺的一部分，搜尋引擎的出現功不可沒，因為「網路」幾乎已經成為全世界最大的資料庫。由於網路上的資料量實在過於龐大，且每天都在持續增加中，如果沒有一個方法能讓使用者快速的找到所需的資料，使用者必定得花費大量的時間在搜尋上。但是現今的搜尋技術都是以關鍵字來查找網頁，也就是當使用者輸入關鍵字之後，搜尋引擎幫忙找出含有這個關鍵字的網頁。但是假設今天有一個部落格作者想找到與自己興趣相同的其他作者時，使用搜尋引擎就顯得不太方便。因此本篇論文希望發展一套系統，能夠找出哪些部落格所描述的主題是相似的，以減少使用者的搜尋時間。

1.2 研究目的

本論文希望能分析出在網際網路上的許多部落格作者中，哪些作者有相同的興趣，本論文假設若某兩篇部落格中的文章所描述的主題都相近，則這兩篇部落格的作者有相同的興趣。

在自然語言處理的領域中，常常是用統計的方法來找出一些重要詞彙來當成文章的標籤，當文章數量越多的時候，計算所花費的時間也會越多。由於許多部落格作者在撰寫文章時都會替文章標上標籤，本系統將不考慮部落格中每篇文章的內容，而直接以標籤雲裡的標籤來代表整個部落格，將所有部落格分群，然後檢視是否同一個群內的部落格都描述相同的主題。根據最後的實驗結果顯示，系統確實可以將大部份的部落格分到正確的群中，也間接表示以標籤雲來代表整個部落格是非常合適的。

1.3 論文架構

第一章：緒論，描述本論文之動機與目的。

第二章：相關研究，描述本論文會用到的演算法及技術原理。

第三章：系統設計，將系統的整體架構做一個完整的介紹。

第四章：實驗過程與結果討論，分析實驗結果。

第五章：結論與展望，將本論文做個總結並討論系統未來走向。

第二章、相關研究

2.1 標籤雲(Tag Cloud)

標籤雲[1]主要的功能就是關鍵詞的視覺化描述，它是由許多用戶生成的標籤聚集而成。通常標籤雲中的標籤按照字母的順序排列，而且每一個標籤都是一個獨立的詞彙。標籤可以透過改變字體大小和顏色來表示不同的意義。大多數的標籤本身就是一個超連結，可以直接指向與標籤相關聯的資訊。

標籤雲的其中一個普遍的應用就是應用在部落格中，部落格中通常有一區塊用來容納部落格作者產生的標籤。每當部落格作者發表文章時，可以將文章與一個或數個標籤連結，標籤雲中的每一個標籤就會根據與它連結的文章數的多寡而呈現不同的字體大小。當部落格的閱讀者來訪時，可以根據標籤雲中每一個標籤字面上的意義得知這個部落格所談論的主題。舉例來說：假設某一部落格上的標籤雲裡的標籤有「comedy, DVD, review, …」，則可以推測這個部落格的內容可能是關於電影或電視劇。每一個標籤的字體大小可以更進一步告訴閱讀者與這個標籤連結的文章數的多寡。通常部落格中的標籤為超連結，閱讀者可以直接點擊標籤閱讀與這個標籤相關的文章。所以標籤雲在部落格中所扮演的角色就像是書中的目錄或是索引，通常人們可以根據目錄來找到所需的資訊，也可以根據目錄大致上了解這本書的內容。

2.2 Multidimensional Scaling(MDS)

先考慮下面這個例子：

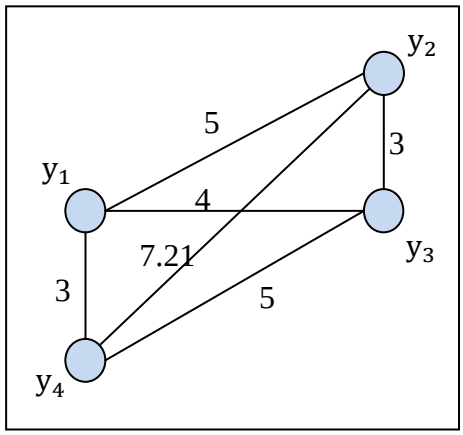


圖 2-1 Multidimensional Scaling 範例

在許多應用中，常常可以得到如圖 2-1 這樣的圖形。每一個節點代表一個物件，而每一個邊上的值代表物件之間的距離，距離可以代表相似度，距離越近代表相似度越高，距離越遠代表相似度越低。而在某些應用中，可能需要更進一步知道每一個節點的座標。Multidimensional scaling 就是用來賦予每一個節點一個合適的座標，使得節點之間利用此座標計算出來的距離盡可能的接近原本節點之間的距離。以下說明 Multidimensional scaling 的原理[2]：

假設 $\mathbf{a}$ 與 $\mathbf{b}$ 是兩個向量，則 $\mathbf{a}$ 與 $\mathbf{b}$ 之間的距離的平方為：

$$d^2(\mathbf{a},\mathbf{b}) = (\mathbf{a}-\mathbf{b})^{\mathrm{T}}(\mathbf{a}-\mathbf{b}) = \mathbf{a}^{\mathrm{T}}\mathbf{a} + \mathbf{b}^{\mathrm{T}}\mathbf{b} - 2\times(\mathbf{a}^{\mathrm{T}}\mathbf{b}) \qquad (2.1)$$

假設矩陣 $\mathbf{X}$ 的行向量就是最後賦予每一個節點的座標：

$$\mathbf{X} = \begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{bmatrix}$$

利用矩陣 $\mathbf{X}$ 可以求得 gram matrix $\mathbf{G}$：

$$\mathbf{G} = \mathbf{X}^{\mathrm{T}}\mathbf{X} = \begin{bmatrix} \mathbf{x}^{\mathrm{T}}\mathbf{x} & \mathbf{x}^{\mathrm{T}}\mathbf{y} & \mathbf{x}^{\mathrm{T}}\mathbf{z} \\ \mathbf{y}^{\mathrm{T}}\mathbf{x} & \mathbf{y}^{\mathrm{T}}\mathbf{y} & \mathbf{y}^{\mathrm{T}}\mathbf{z} \\ \mathbf{z}^{\mathrm{T}}\mathbf{x} & \mathbf{z}^{\mathrm{T}}\mathbf{y} & \mathbf{z}^{\mathrm{T}}\mathbf{z} \end{bmatrix} \tag{2.2}$$

接著再利用矩陣 **G** 求得矩陣 **D**：

$$\mathbf{D} = \mathbf{g} \times \mathbf{1}^{\mathrm{T}} + \mathbf{1} \times \mathbf{g}^{\mathrm{T}} - 2 \times \mathbf{G} \tag{2.3}$$

$$= \begin{bmatrix} \mathbf{x}^{\mathrm{T}}\mathbf{x} & \mathbf{x}^{\mathrm{T}}\mathbf{x} & \mathbf{x}^{\mathrm{T}}\mathbf{x} \\ \mathbf{y}^{\mathrm{T}}\mathbf{y} & \mathbf{y}^{\mathrm{T}}\mathbf{y} & \mathbf{y}^{\mathrm{T}}\mathbf{y} \\ \mathbf{z}^{\mathrm{T}}\mathbf{z} & \mathbf{z}^{\mathrm{T}}\mathbf{z} & \mathbf{z}^{\mathrm{T}}\mathbf{z} \end{bmatrix} + \begin{bmatrix} \mathbf{x}^{\mathrm{T}}\mathbf{x} & \mathbf{y}^{\mathrm{T}}\mathbf{y} & \mathbf{z}^{\mathrm{T}}\mathbf{z} \\ \mathbf{x}^{\mathrm{T}}\mathbf{x} & \mathbf{y}^{\mathrm{T}}\mathbf{y} & \mathbf{z}^{\mathrm{T}}\mathbf{z} \\ \mathbf{x}^{\mathrm{T}}\mathbf{x} & \mathbf{y}^{\mathrm{T}}\mathbf{y} & \mathbf{z}^{\mathrm{T}}\mathbf{z} \end{bmatrix} - 2 \times \begin{bmatrix} \mathbf{x}^{\mathrm{T}}\mathbf{x} & \mathbf{x}^{\mathrm{T}}\mathbf{y} & \mathbf{x}^{\mathrm{T}}\mathbf{z} \\ \mathbf{y}^{\mathrm{T}}\mathbf{x} & \mathbf{y}^{\mathrm{T}}\mathbf{y} & \mathbf{y}^{\mathrm{T}}\mathbf{z} \\ \mathbf{z}^{\mathrm{T}}\mathbf{x} & \mathbf{z}^{\mathrm{T}}\mathbf{y} & \mathbf{z}^{\mathrm{T}}\mathbf{z} \end{bmatrix}$$

$$= \begin{bmatrix} \mathbf{x}^{\mathrm{T}}\mathbf{x} + \mathbf{x}^{\mathrm{T}}\mathbf{x} - 2 \times (\mathbf{x}^{\mathrm{T}}\mathbf{x}) & \mathbf{x}^{\mathrm{T}}\mathbf{x} + \mathbf{y}^{\mathrm{T}}\mathbf{y} - 2 \times (\mathbf{x}^{\mathrm{T}}\mathbf{y}) & \mathbf{x}^{\mathrm{T}}\mathbf{x} + \mathbf{z}^{\mathrm{T}}\mathbf{z} - 2 \times (\mathbf{x}^{\mathrm{T}}\mathbf{z}) \\ \mathbf{y}^{\mathrm{T}}\mathbf{y} + \mathbf{x}^{\mathrm{T}}\mathbf{x} - 2 \times (\mathbf{y}^{\mathrm{T}}\mathbf{x}) & \mathbf{y}^{\mathrm{T}}\mathbf{y} + \mathbf{y}^{\mathrm{T}}\mathbf{y} - 2 \times (\mathbf{y}^{\mathrm{T}}\mathbf{y}) & \mathbf{y}^{\mathrm{T}}\mathbf{y} + \mathbf{z}^{\mathrm{T}}\mathbf{z} - 2 \times (\mathbf{y}^{\mathrm{T}}\mathbf{z}) \\ \mathbf{z}^{\mathrm{T}}\mathbf{z} + \mathbf{x}^{\mathrm{T}}\mathbf{x} - 2 \times (\mathbf{z}^{\mathrm{T}}\mathbf{x}) & \mathbf{z}^{\mathrm{T}}\mathbf{z} + \mathbf{y}^{\mathrm{T}}\mathbf{y} - 2 \times (\mathbf{z}^{\mathrm{T}}\mathbf{y}) & \mathbf{z}^{\mathrm{T}}\mathbf{z} + \mathbf{z}^{\mathrm{T}}\mathbf{z} - 2 \times (\mathbf{z}^{\mathrm{T}}\mathbf{z}) \end{bmatrix} \tag{2.4}$$

其中，**g** 為 **G** 之對角線所形成之行向量 $\begin{bmatrix} \mathbf{x}^{\mathrm{T}}\mathbf{x} \\ \mathbf{y}^{\mathrm{T}}\mathbf{y} \\ \mathbf{z}^{\mathrm{T}}\mathbf{z} \end{bmatrix}$，$\mathbf{1} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$。

根據(2.1)與(2.4)可知，矩陣 **D** 中的每一個元素 $\mathbf{D}_{ij}$ 代表節點 i 與節點 j 之間之距離的平方。

經過以上的推導，可以看到從矩陣 **X** 求得矩陣 **D** 的過程。Multidimensional scaling 的步驟恰好與上述相反，Multidimensional scaling 可以從已知的矩陣 **D** 來求得座標矩陣 **X**：

假設矩陣 **D** 為 n×n 的矩陣。首先，求一個向量 **m**，**m** 為一個 n×1 的向量，**m** 中所有元素皆為 1/n，所以可以得到下列結果:

$$\mathbf{m}^{\mathrm{T}} \times \mathbf{1} = 1 \tag{2.5}$$

接下來，利用向量 **m** 可以求得矩陣 **M**：

$$\mathbf{M} = \mathbf{I} - \mathbf{1} \times \mathbf{m}^{\mathrm{T}} \tag{2.6}$$

求出矩陣 $\mathbf{M}$ 後，利用矩陣 $\mathbf{D}$ 與矩陣 $\mathbf{M}$ 就可以求出矩陣 $\mathbf{G}$：

$$\mathbf{G} = -\frac{1}{2}\mathbf{M} \times \mathbf{D} \times \mathbf{M}^{\mathrm{T}} \tag{2.7}$$

(2.7)留到這一個小節的最後再證明。這裡繼續說明如何從矩陣 $\mathbf{G}$ 求得矩陣 $\mathbf{X}$，先假設矩陣 $\mathbf{X}$ 存在，且將矩陣 $\mathbf{X}$ 做 SVD 分解得到 $\mathbf{X} = \mathbf{U\Sigma V}^{\mathrm{T}}$，根據(2.2)將 $\mathbf{X} = \mathbf{U\Sigma V}^{\mathrm{T}}$代入:

$$\begin{aligned}
\mathbf{G} &= (\mathbf{U\Sigma V}^{\mathrm{T}})^{\mathrm{T}}(\mathbf{U\Sigma V}^{\mathrm{T}}) \\
&= \mathbf{V\Sigma}^{\mathrm{T}}\mathbf{U}^{\mathrm{T}}\mathbf{U\Sigma V}^{\mathbf{T}} \\
&= \mathbf{V\Sigma}^{\mathrm{T}}\mathbf{\Sigma V}^{\mathbf{T}} \\
&= \mathbf{V\Lambda V}^{\mathrm{T}} \qquad (2.8) \\
&= \mathbf{V}\sqrt{\mathbf{\Lambda}}\sqrt{\mathbf{\Lambda}}\mathbf{V}^{\mathbf{T}} \\
&= (\sqrt{\mathbf{\Lambda}}\mathbf{V}^{\mathrm{T}})^{\mathrm{T}}(\sqrt{\mathbf{\Lambda}}\mathbf{V}^{\mathrm{T}}) \qquad (2.9)
\end{aligned}$$

(2.8)為矩陣 $\mathbf{G}$ 的 eigen-decomposition。$\mathbf{\Lambda} = \begin{bmatrix} \lambda_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_n \end{bmatrix}$，其中 $\lambda_1, \lambda_2, \ldots, \lambda_n$為矩陣 $\mathbf{G}$ 之 eigenvalue 且 $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq 0$，因為矩陣 $\mathbf{G}$ 是 positive semi-definite，所以矩陣 $\mathbf{G}$ 的所有 eigenvalue 皆大於等於 0，矩陣 $\mathbf{V}$ 之行向量 $\mathbf{v}_i$ 為矩陣 $\mathbf{G}$ 相對於 λ_i 且長度為 1 之 eigenvector。根據(2.9)可知，取 $\mathbf{X} = \sqrt{\mathbf{\Lambda}}\mathbf{V}^{\mathrm{T}}$ 就能得到矩陣 $\mathbf{X}$，但是當我們使用 Multidimensional scaling 來求得圖形中每一個節點的座標時，我們通常會更進一步地希望能將圖形畫出來以利於觀察，所以通常都會將維度降至 1、2 或 3 維。在降低維度的同時，一定會失去資訊，所以每次都刪去影響最小的部分，一直刪除到目標的維度為止。為了達到這個目的，將矩陣 $\mathbf{G}$ 寫成另外一個形式:

$$\mathbf{G} = \mathbf{V\Lambda V}^{\mathrm{T}} = \lambda_1\mathbf{v}_1\mathbf{v}_1{}^{\mathrm{T}} + \lambda_2\mathbf{v}_2\mathbf{v}_2{}^{\mathrm{T}} + \cdots + \lambda_n\mathbf{v}_n\mathbf{v}_n{}^{\mathrm{T}} \tag{2.10}$$

從觀察式子(2.10)發現，可以將矩陣 **G** 看成是很多矩陣組合而成，因為 $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$，所以從後面的項開始刪除對矩陣 **G** 的影響最小。假設我們希望最後要顯示的座標維度是 p，則取 $\mathbf{X} = \mathbf{I}_{p\times n}\sqrt{\mathbf{\Lambda}}\mathbf{V}^T$ 可以使得 $\|\mathbf{G} - \mathbf{X}^T\mathbf{X}\|$ 最小。

回到圖 2-1，可以求得矩陣 **D** 為:

$$\mathbf{D} = \begin{bmatrix} 0 & 25.00 & 16.00 & 9.00 \\ 25.00 & 0 & 9.00 & 51.98 \\ 16.00 & 9.00 & 0 & 25.00 \\ 9.00 & 51.98 & 25.00 & 0 \end{bmatrix}$$

矩陣 **M** 如下:

$$\mathbf{M} = \begin{bmatrix} 0.75 & -0.25 & -0.25 & -0.25 \\ -0.25 & 0.75 & -0.25 & -0.25 \\ -0.25 & -0.25 & 0.75 & -0.25 \\ -0.25 & -0.25 & -0.25 & 0.75 \end{bmatrix}$$

利用矩陣 **D** 與矩陣 **M** 來計算矩陣 **G**:

$$\mathbf{G} = \begin{bmatrix} 4 & -4 & -4 & 4 \\ -4 & 13 & 4 & -13 \\ -4 & 4 & 4 & -4 \\ 4 & -13 & -4 & 13 \end{bmatrix}$$

將矩陣 **G** 分解成 $\mathbf{V\Lambda V}^T$ 可得:

$$\mathbf{\Lambda} = \begin{bmatrix} 29.035 & 0 & 0 & 0 \\ 0 & 4.957 & 0 & 0 \\ 0 & 0 & 0.004 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \mathbf{V} = \begin{bmatrix} 0.251 & 0.661 & -0.5 & 0.5 \\ -0.661 & 0.251 & 0.5 & 0.5 \\ -0.251 & -0.661 & -0.5 & 0.5 \\ 0.661 & -0.251 & 0.5 & 0.5 \end{bmatrix}$$

為了將y_1~y_4標示在平面上，所以取 $\mathbf{I} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$，則可計算 **X**:

$$\mathbf{X} = \mathbf{I} \times \sqrt{\mathbf{\Lambda}} \times \mathbf{V} = \begin{bmatrix} 1.353 & -3.562 & -1.353 & 3.562 \\ 1.472 & 0.559 & -1.472 & -0.559 \end{bmatrix}$$

計算出矩陣 **X** 之後，**X** 的行向量 $\mathbf{x}_i$ 就是節點 y_i 的座標，將y_1~y_4標示在圖 2-2:

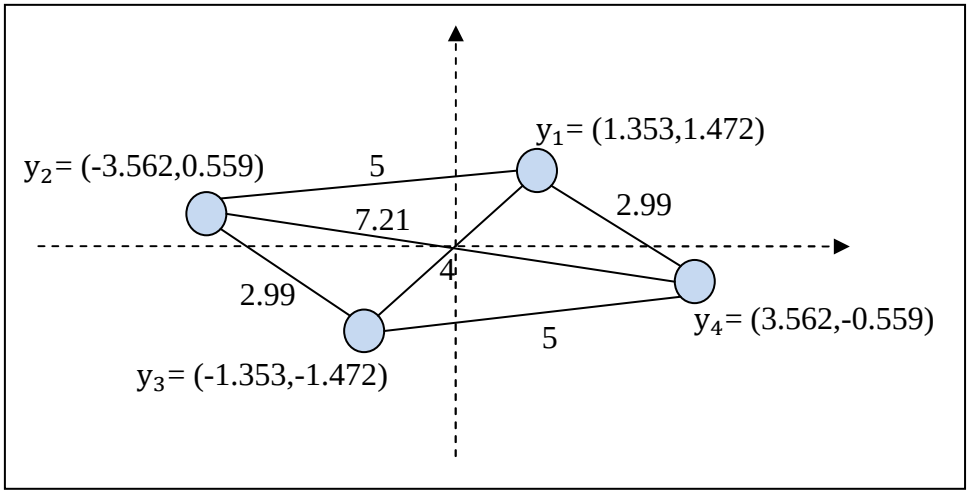


圖 2-2 賦予每一個節點座標

從圖 2-2 中可以觀察到，在賦予節點y_1~y_4座標之後，每一個節點間的距離幾乎與原本節點之間的關係一模一樣。

最後證明(2.7):

首先，從圖 2-2 中可以看到，若將此圖往任意方向平移一段距離，將獲得另一組新的座標，而這組座標之間的距離與原本相同，所以說有無限多組解可以來解決這個問題。為了固定一組解，可以限制此組解的中心點必須為原點，假設中心點為原點可以得到以下式子:

$$\mathbf{Xm} = \mathbf{0} \qquad (2.11)$$

由(2.3)可以知道 $\mathbf{D} = \mathbf{g1}^{\mathrm{T}} + \mathbf{1g}^{\mathrm{T}} - 2\mathbf{G}$ ，將它代入 $-\frac{1}{2}\mathbf{MDM}^{\mathrm{T}}$:

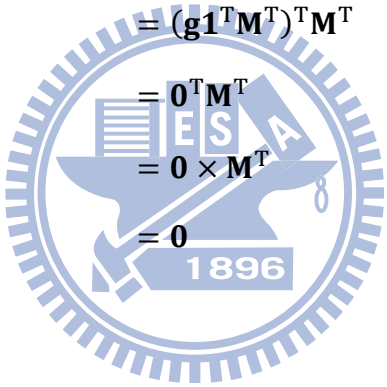
$$-\frac{1}{2}\mathbf{MDM}^{\mathrm{T}} = -\frac{1}{2}\mathbf{Mg1}^{\mathrm{T}}\mathbf{M}^{\mathrm{T}} - \frac{1}{2}\mathbf{M1g}^{\mathrm{T}}\mathbf{M}^{\mathrm{T}} + \mathbf{MGM}^{\mathrm{T}} \qquad (2.12)$$

接下來證明 $\mathbf{Mg1^TM^T}$ 與 $\mathbf{M1g^TM^T}$ 皆為 $\mathbf{0}$ 矩陣:

$$\begin{aligned}\mathbf{Mg1^TM^T} &= \mathbf{Mg1^T(I-1m^T)^T}\\ &= \mathbf{Mg1^T(I-m1^T)}\\ &= \mathbf{M(g1^T-g1^Tm1^T)} \qquad (\mathbf{1^Tm}=1)\\ &= \mathbf{M(g1^T-g1^T)}\\ &= \mathbf{M}\times\mathbf{0}\\ &= \mathbf{0} \qquad (2.13)\end{aligned}$$

且

$$\begin{aligned}\mathbf{M1g^TM^T} &= ((\mathbf{M1g^T})^T)^T\mathbf{M^T}\\ &= (\mathbf{g1^TM^T})^T\mathbf{M^T}\\ &= \mathbf{0^TM^T}\\ &= \mathbf{0}\times\mathbf{M^T}\\ &= \mathbf{0} \qquad (2.14)\end{aligned}$$



然後將$\mathbf{MGM^T}$展開:

$$\begin{aligned}\mathbf{MGM^T} &= \mathbf{(I-1m^T)G(I-1m^T)}\\ &= \mathbf{G-Gm1^T-1m^TG+1m^TGm1^T} \qquad (2.15)\end{aligned}$$

根據(2.2)可知 $\mathbf{G}=\mathbf{X^TX}$，再從(2.11)可知 $\mathbf{Xm}=\mathbf{0}$，我們可以藉由(2.2)與(2.11)來證明 $\mathbf{Gm}$ 與 $\mathbf{m^TG}$ 皆為 $\mathbf{0}$ 向量:

$$\begin{aligned}\mathbf{Gm} &= \mathbf{X^TXm}\\ &= \mathbf{X^T0}\\ &= \mathbf{0} \qquad (2.16)\end{aligned}$$

且

$$\mathbf{m}^{\mathrm{T}}\mathbf{G} = ((\mathbf{m}^{\mathrm{T}}\mathbf{G})^{\mathrm{T}})^{\mathrm{T}}$$
$$= (\mathbf{G}^{\mathrm{T}}\mathbf{m})^{\mathrm{T}}$$
$$= (\mathbf{G}\mathbf{m})^{\mathrm{T}}$$
$$= \mathbf{0}^{\mathrm{T}} \quad (2.17)$$

從(2.15)、(2.16)、(2.17)可知:

$$\mathbf{M}\mathbf{G}\mathbf{M}^{\mathrm{T}} = \mathbf{G} \quad (2.18)$$

最後，從(2.12)、(2.13)、(2.14)與(2.18)得到:$-\frac{1}{2}\mathbf{M}\mathbf{D}\mathbf{M}^{\mathrm{T}} = \mathbf{G}$，得證。

2.3 Spectral Clustering

Spectral clustering 是一個分群演算法，它根據點與點之間的相似度將空間上的一組點分群，使得同一個群內的點與點之間的相似度越高越好，不同群的點與點之間的相似度則越低越好。這一節將簡單說明 spectral clustering 的基本原理[3]。

2.3.1 定義符號

假設有一無向圖(undirected graph) $G = (V,E)$，圖中每條邊(edge)上都有權重(weight)，$V = \{v_1, \ldots, v_n\}$ 是點(vertex)所形成的集合，w_{ij} 為 v_i 與 v_j 相連的邊上的權重。其中，$w_{ij} \geq 0$ 且 $w_{ij} = w_{ji}$ ，若 $w_{ij} = 0$ 代表 v_i 與 v_j 之間無邊相連。

定義weighted adjacency matrix $\boldsymbol{W}$:

$$\boldsymbol{W} = (w_{ij})_{i,j=1,\ldots,n}$$

G 中的每個點 $v_i \in V$ 的degree定義為:

$$d_i = \sum_{j=1}^{n} w_{ij}$$

定義 degree matrix $\boldsymbol{D}$:

$$\boldsymbol{D} = \begin{bmatrix} d_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & d_n \end{bmatrix}$$

$\boldsymbol{D}$ 為對角矩陣且對角線上的值為 $d_1, \ldots, d_n$。

假設有一 V 的子集 $A \subset V$，則 A 的補集(complement)標記為 $\overline{A}$，定義 indicator vector $\boldsymbol{f}_A = (f_1, \ldots, f_n)^T \in \mathbb{R}^n$ 為:

$$\begin{cases} f_i = 1, & \text{if } v_i \in A \\ f_i = 0, & \text{if } v_i \in \overline{A} \end{cases}$$

假設 $A、B \subset V$ 且 $A \cap B = \emptyset$，則定義 A、B 之間的 weight 為:

$$W(A, B) = \sum_{i \in A, j \in B} w_{ij}$$

2.3.2 Laplacian matrix

Laplacian matrix 可以分為 unnormalized 與 normalized。首先說明 unnormalized Laplacian matrix，其定義為:

$$\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W}$$

矩陣 $\mathbf{L}$ 具有下列性質:

(1)對於所有的向量 $\boldsymbol{f} \in \mathbb{R}^n$，下列式子成立:

$$\boldsymbol{f}^T\boldsymbol{L}\boldsymbol{f} = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2$$

證明:

$$\boldsymbol{f}^T\boldsymbol{L}\boldsymbol{f} = \boldsymbol{f}^T\boldsymbol{D}\boldsymbol{f} - \boldsymbol{f}^T\boldsymbol{W}\boldsymbol{f}$$

$$= \sum_{i=1}^{n} d_i {f_i}^2 - \sum_{i,j=1}^{n} f_i f_j w_{ij}$$

$$= \frac{1}{2}\left(\sum_{i=1}^{n} d_i {f_i}^2 - 2\sum_{i,j=1}^{n} f_i f_j w_{ij} + \sum_{j=1}^{n} d_j {f_j}^2\right)$$

$$= \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2$$

(2) $\boldsymbol{L}$ 是 symmetric 且 positive semi-definite。

因為 $\boldsymbol{D}$ 與 $\boldsymbol{W}$ 皆為 symmetric，所以 $\boldsymbol{L}$ 也是 symmetric。且由(1)可知 $\boldsymbol{L}$ 為 semi-definite。

(3) $\boldsymbol{L}$ 最小的 eigenvalue 為 0，且相對於 0 之 eigenvector 為 $\mathbf{1}$。

證明:

$$\boldsymbol{L} \times \mathbf{1} = (\boldsymbol{D} - \boldsymbol{W}) \times \mathbf{1}$$

$$= \boldsymbol{D} \times \mathbf{1} - \boldsymbol{W} \times \mathbf{1}$$

$$= \begin{bmatrix} d_1 \\ \vdots \\ d_n \end{bmatrix} - \begin{bmatrix} d_1 \\ \vdots \\ d_n \end{bmatrix}$$

$$= \mathbf{0} = 0 \times \mathbf{1}$$

由此可知，0 必為 $\boldsymbol{L}$ 之 eigenvalue。且因為 $\boldsymbol{L}$ 為 semi-definite，所以所有的 eigenvalue 皆大於等於 0，由此可知，0 為最小的 eigenvalue。

(4) $\boldsymbol{L}$ 的所有 eigenvalue 皆為實數，且 $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$。

因為 $\boldsymbol{L}$ 是一個對稱矩陣，所以 $\boldsymbol{L}$ 之所有 eigenvalue 皆為實數，且由(3)可知，0 為最小之 eigenvalue。

接下來說明 normalized graph Laplacian。normalized graph Laplacian 分為兩種，其定義分別為:

$$\boldsymbol{L}_{\text{sym}} = \boldsymbol{D}^{-1/2}\boldsymbol{L}\boldsymbol{D}^{-1/2}$$

與

$$\boldsymbol{L}_{\text{rw}} = \boldsymbol{D}^{-1}\boldsymbol{L}$$

矩陣 $\boldsymbol{L}_{\text{sym}}$ 與 $\boldsymbol{L}_{\text{rw}}$ 具有下列性質:

(1) 對於所有的向量 $\boldsymbol{f} \in \mathbb{R}^n$，下列式子成立:

$$\boldsymbol{f}^{\text{T}}\boldsymbol{L}_{\text{sym}}\boldsymbol{f} = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}\left(\frac{f_i}{\sqrt{d_i}} - \frac{f_j}{\sqrt{d_j}}\right)^2$$

證明:

$$\boldsymbol{f}^{\text{T}}\boldsymbol{L}_{\text{sym}}\boldsymbol{f} = \boldsymbol{f}^{\text{T}}\boldsymbol{D}^{-1/2}\boldsymbol{L}\boldsymbol{D}^{-1/2}\boldsymbol{f}$$

$$= \left(\boldsymbol{D}^{-1/2}\boldsymbol{f}\right)^{\text{T}}\boldsymbol{L}\left(\boldsymbol{D}^{-1/2}\boldsymbol{f}\right)$$

$$= \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}\left(\frac{f_i}{\sqrt{d_i}} - \frac{f_j}{\sqrt{d_j}}\right)^2$$

(2)$\boldsymbol{L}_{\mathrm{rw}}\boldsymbol{u} = \lambda\boldsymbol{u} \Leftrightarrow \boldsymbol{L}_{\mathrm{sym}}\boldsymbol{w} = \lambda\boldsymbol{w}$，其中 $\boldsymbol{w} = \boldsymbol{D}^{1/2}\boldsymbol{u}$。

證明:

$$\boldsymbol{L}_{\mathrm{sym}}\boldsymbol{w} = \lambda\boldsymbol{w}$$

$$\Leftrightarrow \boldsymbol{D}^{-1/2}\boldsymbol{L}\boldsymbol{D}^{-1/2}\boldsymbol{w} = \lambda\boldsymbol{w}$$

$$\Leftrightarrow \boldsymbol{D}^{-1/2}\boldsymbol{D}^{-1/2}\boldsymbol{L}\boldsymbol{D}^{-1/2}\boldsymbol{w} = \lambda\boldsymbol{D}^{-1/2}\boldsymbol{w}$$

$$\Leftrightarrow (\boldsymbol{D}^{-1}\boldsymbol{L})(\boldsymbol{D}^{-1/2}\boldsymbol{w}) = \lambda(\boldsymbol{D}^{-1/2}\boldsymbol{w})$$

$$\Leftrightarrow \boldsymbol{L}_{\mathrm{rw}}\boldsymbol{u} = \lambda\boldsymbol{u}$$

(3) $\boldsymbol{L}_{\mathrm{rw}}\boldsymbol{u} = \lambda\boldsymbol{u} \Leftrightarrow \boldsymbol{L}\boldsymbol{u} = \lambda\boldsymbol{D}\boldsymbol{u}$。

證明:

$$\boldsymbol{L}_{\mathrm{rw}}\boldsymbol{u} = \lambda\boldsymbol{u}$$

$$\Leftrightarrow \boldsymbol{D}^{-1}\boldsymbol{L}\boldsymbol{u} = \lambda\boldsymbol{u}$$

$$\Leftrightarrow \boldsymbol{D}\boldsymbol{D}^{-1}\boldsymbol{L}\boldsymbol{u} = \lambda\boldsymbol{D}\boldsymbol{u}$$

$$\Leftrightarrow \boldsymbol{L}\boldsymbol{u} = \lambda\boldsymbol{D}\boldsymbol{u}$$

(4) $\boldsymbol{L}_{\mathrm{sym}}$ 與 $\boldsymbol{L}_{\mathrm{rw}}$ 是 positive semi-definite。

由(1)可知 $\boldsymbol{L}_{\mathrm{sym}}$ 為 semi-definite，接著根據(2)可知，若 λ 是 $\boldsymbol{L}_{\mathrm{sym}}$ 的 eigenvalue，則 λ 也是 $\boldsymbol{L}_{\mathrm{rw}}$ 的 eigenvalue，所以 $\boldsymbol{L}_{\mathrm{rw}}$ 的所有 eigenvalue 皆大於等於 0，由此可知 $\boldsymbol{L}_{\mathrm{rw}}$ 為 semi-definite。

(5) $\boldsymbol{L}_{\mathrm{rw}}$ 最小的 eigenvalue 為 0，且相對於 0 之 eigenvector 為 $\mathbf{1}$。$\boldsymbol{L}_{\mathrm{sym}}$ 最小的 eigenvalue 為 0，且相對於 0 之 eigenvector 為 $\boldsymbol{D}^{1/2}\mathbf{1}$。

因為 $\boldsymbol{L}_{\mathrm{rw}}\mathbf{1} = \boldsymbol{D}^{-1}\boldsymbol{L}\mathbf{1} = \mathbf{0} = 0 \times \mathbf{1}$ 且 $\boldsymbol{L}_{\mathrm{rw}}$ 為 semi-definite，所以 0 為 $\boldsymbol{L}_{\mathrm{rw}}$ 之最小的 eigenvalue。由(2)與(4)可知，0 也是 $\boldsymbol{L}_{\mathrm{sym}}$ 最小的 eigenvalue 且 $\boldsymbol{L}_{\mathrm{sym}}$ 相對於 0 之 eigenvector 為 $\boldsymbol{D}^{1/2}\mathbf{1}$。

(6) $\boldsymbol{L}_{sym}$ 與 $\boldsymbol{L}_{rw}$ 的所有 eigenvalue 皆為實數，且 $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$。

因為 $\boldsymbol{L}_{sym}$ 為對稱矩陣，所以 $\boldsymbol{L}_{sym}$ 之所有 eigenvalue 皆為實數，且由(5)可知，0 為 $\boldsymbol{L}_{sym}$ 最小之 eigenvalue。由(2)可知，$\boldsymbol{L}_{rw}$ 與 $\boldsymbol{L}_{sym}$ 具有相同的 eigenvalue，所以 $\boldsymbol{L}_{rw}$ 亦有此性質。

2.3.3 RatioCut 與 Ncut

給定一 similarity graph $G = (V, E)$，為了將所有點 V 分成任意 k 群，首先定義 cut 如下:

$$\mathrm{cut}(A_1, \ldots A_k) = \frac{1}{2}\sum_{p=1}^{k} W(A_p, \overline{A_p})$$

其中，$A_1 \cup \ldots \cup A_k = V$ 且 $A_1 \cap \ldots \cap A_k = \emptyset$。

spectral clustering 的目標就是找出一組分割(partition) $A_1, \ldots, A_k$，使得 $\mathrm{cut}(A_1, \ldots, A_k)$ 為最小。但是在分群時，有時會希望分群完成之後每一個群內點的數量都差不多，所以必須將群的大小也考慮進去，於是就有 RatioCut 與 Ncut 的出現。分別定義如下:

$$\mathrm{RatioCut}(A_1, \ldots, A_k) = \sum_{p=1}^{k} \frac{\mathrm{cut}(A_p, \overline{A_p})}{|A_p|}$$

$$\mathrm{Ncut}(A_1, \ldots, A_k) = \sum_{p=1}^{k} \frac{\mathrm{cut}(A_p, \overline{A_p})}{\mathrm{vol}(A_p)}$$

其中，$|A|$代表A中點的數量，$\mathrm{vol}(A) = \sum_{i \in A} d_i$。

接下來就可以將重點放在如何找出 RatioCut 與 Ncut 的最小值上面。在這一小節中，只說明 $k = 2$ 之原理，因為 k 為任意值時，其原理與 $k = 2$ 時是一樣的。先說明 RatioCut:

根據之前的敘述，目標是要找出：

$$\min_{A \subset V} \text{RatioCut}(A, \bar{A})$$

首先定義向量 $\boldsymbol{f} = (f_1, \ldots, f_n) \in \mathbb{R}^n$ 為：

$$f_i = \begin{cases} \sqrt{|\bar{A}|/|A|} \ , & \text{if } v_i \in A \\ -\sqrt{|A|/|\bar{A}|} \ , & \text{if } v_i \in \bar{A} \end{cases} \tag{2.19}$$

則 $\boldsymbol{f}$ 具有以下性質：

(1) $\boldsymbol{f} \perp \mathbf{1}$

證明：

$$\boldsymbol{f} \cdot \mathbf{1} = \sum_{i=1}^{n} f_i = \sum_{i \in A} \sqrt{\frac{|\bar{A}|}{|A|}} - \sum_{i \in \bar{A}} \sqrt{\frac{|A|}{|\bar{A}|}}$$

$$= |A| \sqrt{\frac{|\bar{A}|}{|A|}} - |\bar{A}| \sqrt{\frac{|A|}{|\bar{A}|}} = 0$$

(2) $\|\boldsymbol{f}\| = \sqrt{n}$

證明：

$$\|\mathbf{f}\|^2 = \sum_{i=1}^{n} {f_i}^2 = |A| \frac{|\bar{A}|}{|A|} + |\bar{A}| \frac{|A|}{|\bar{A}|}$$

$$= |\bar{A}| + |A| = n$$

(3) $\boldsymbol{f}^{\mathrm{T}}\boldsymbol{L}\boldsymbol{f} = |\mathrm{V}| \cdot \mathrm{RatioCut}(\mathrm{A}, \bar{\mathrm{A}})$

證明:

$$\begin{aligned}
\boldsymbol{f}^{\mathrm{T}}\boldsymbol{L}\boldsymbol{f} &= \frac{1}{2}\sum_{\mathrm{i,j=1}}^{\mathrm{n}} \mathrm{w_{ij}}\left(\mathrm{f_i} - \mathrm{f_j}\right)^2 \\
&= \frac{1}{2}\sum_{\mathrm{i}\in\mathrm{A},\mathrm{j}\in\bar{\mathrm{A}}} \mathrm{w_{ij}}\left(\sqrt{\frac{|\bar{\mathrm{A}}|}{|\mathrm{A}|}} + \sqrt{\frac{|\mathrm{A}|}{|\bar{\mathrm{A}}|}}\right)^2 + \frac{1}{2}\sum_{\mathrm{i}\in\bar{\mathrm{A}},\mathrm{j}\in\mathrm{A}} \mathrm{w_{ij}}\left(-\sqrt{\frac{|\bar{\mathrm{A}}|}{|\mathrm{A}|}} - \sqrt{\frac{|\mathrm{A}|}{|\bar{\mathrm{A}}|}}\right)^2 \\
&= \mathrm{cut}(\mathrm{A}, \bar{\mathrm{A}})\left(\frac{|\bar{\mathrm{A}}|}{|\mathrm{A}|} + \frac{|\mathrm{A}|}{|\bar{\mathrm{A}}|} + 2\right) \\
&= \mathrm{cut}(\mathrm{A}, \bar{\mathrm{A}})\left(\frac{|\mathrm{A}| + |\bar{\mathrm{A}}|}{|\mathrm{A}|} + \frac{|\mathrm{A}| + |\bar{\mathrm{A}}|}{|\bar{\mathrm{A}}|}\right) \\
&= |\mathrm{V}|\left(\frac{\mathrm{cut}(\mathrm{A}, \bar{\mathrm{A}})}{|\mathrm{A}|} + \frac{\mathrm{cut}(\mathrm{A}, \bar{\mathrm{A}})}{|\bar{\mathrm{A}}|}\right) \\
&= |\mathrm{V}| \cdot \mathrm{RatioCut}(\mathrm{A}, \bar{\mathrm{A}})
\end{aligned}$$

因為 $|\mathrm{V}|$ 為定值,所以找出 $\mathrm{RatioCut}(\mathrm{A}, \bar{\mathrm{A}})$ 的最小值相當於找出 $\boldsymbol{f}$ 使得 $\boldsymbol{f}^{\mathrm{T}}\boldsymbol{L}\boldsymbol{f}$ 的值為最小,因此可以將問題重新表示如下:

$$\min_{\mathrm{A}\subset\mathrm{V}} \boldsymbol{f}^{\mathrm{T}}\boldsymbol{L}\boldsymbol{f} \text{ subject to } \boldsymbol{f} \perp \mathbf{1} \text{ 且 } \mathrm{f_i} \text{ 定義於(2.19)}, \|\boldsymbol{f}\| = \sqrt{\mathrm{n}}$$

但是這是一個 NP hard 的問題,沒有辦法有效率的被解決,所以將 $\boldsymbol{f}$ 放寬一些限制,改成 $\mathrm{f_i} \in \mathbb{R}$ 皆可,如此可以將問題簡化成:

$$\min_{\boldsymbol{f}\in\mathbb{R}^{\mathrm{n}}} \boldsymbol{f}^{\mathrm{T}}\boldsymbol{L}\boldsymbol{f} \text{ subject to } \boldsymbol{f} \perp \mathbf{1}, \|\boldsymbol{f}\| = \sqrt{\mathrm{n}}$$

由Rayleigh-Ritz theorem[4]可知:

$$\lambda_1 \leq \frac{\boldsymbol{f}^{\mathrm{T}}\boldsymbol{L}\boldsymbol{f}}{\boldsymbol{f}^{\mathrm{T}}\boldsymbol{f}} \leq \lambda_{\mathrm{n}}$$

且 $\boldsymbol{f}$ 若取 $\boldsymbol{L}$ 之 eigenvector 代入 $\frac{\boldsymbol{f}^{\mathrm{T}}\boldsymbol{L}\boldsymbol{f}}{\boldsymbol{f}^{\mathrm{T}}\boldsymbol{f}}$ 會得到相對應的 eigenvalue,所以取 $\boldsymbol{L}$ 相對於 λ_1 之 eigenvector 即可,但是前面敘述可知, $\boldsymbol{L}$ 相對於 λ_1 之

eigenvector為 $\mathbf{1}$，但是因為 $\boldsymbol{f}$ 被限制要垂直於 $\mathbf{1}$，所以改取相對於 λ_2 之 eigenvector。求得 $\boldsymbol{f}$ 後將 $\boldsymbol{f}$ 當成 indicator vector，就可以藉由 $\boldsymbol{f}$ 得到 A，也就是將 V 分成兩群:

$$\begin{cases} v_i \in A, & \text{if } f_i \geq 0 \\ v_i \in \bar{A}, & \text{if } f_i < 0 \end{cases} \tag{2.20}$$

接下來說明如何求 Ncut 之最小值，其過程與求 RatioCut 之最小值非常類似。首先定義向量 $\boldsymbol{f} = (f_1, \ldots, f_n) \in \mathbb{R}^n$ 為:

$$f_i = \begin{cases} \sqrt{\dfrac{vol(\bar{A})}{vol(A)}}, & \text{if } v_i \in A \\ -\sqrt{\dfrac{vol(A)}{vol(\bar{A})}}, & \text{if } v_i \in \bar{A} \end{cases} \tag{2.21}$$

則可求得:

(1) $(\boldsymbol{Df})^T\mathbf{1} = 0$

證明:

$$(\boldsymbol{Df})^T\mathbf{1} = \begin{bmatrix} d_1 f_1 \\ \vdots \\ d_n f_n \end{bmatrix}^T \times \mathbf{1}$$

$$= d_1 f_1 + \cdots + d_n f_n$$

$$= \left(\sum_{i \in A} d_i\right)\sqrt{\frac{vol(\bar{A})}{vol(A)}} - \left(\sum_{i \in \bar{A}} d_i\right)\sqrt{\frac{vol(A)}{vol(\bar{A})}}$$

$$= vol(A)\sqrt{\frac{vol(\bar{A})}{vol(A)}} - vol(\bar{A})\sqrt{\frac{vol(A)}{vol(\bar{A})}}$$

$$= 0$$

(2)$\boldsymbol{f}^{\mathrm{T}}\boldsymbol{D}\boldsymbol{f} = \mathrm{vol}(\mathrm{V})$

證明:

$$\boldsymbol{f}^{\mathrm{T}}\boldsymbol{D}\boldsymbol{f} = \sum_{i=1}^{n} f_i^{\,2} d_i$$

$$= \mathrm{vol}(A)\frac{\mathrm{vol}(\bar{A})}{\mathrm{vol}(A)} + \mathrm{vol}(\bar{A})\frac{\mathrm{vol}(A)}{\mathrm{vol}(\bar{A})}$$

$$= \mathrm{vol}(\bar{A}) + \mathrm{vol}(A)$$

$$= \mathrm{vol}(V)$$

(3)$\boldsymbol{f}^{\mathrm{T}}\boldsymbol{L}\boldsymbol{f} = \mathrm{vol}(\mathrm{V})\mathrm{Ncut}(A, \bar{A})$

證明:

$$\boldsymbol{f}^{\mathrm{T}}\boldsymbol{L}\boldsymbol{f} = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2$$

$$= \frac{1}{2}\sum_{i\in A, j\in\bar{A}} w_{ij}\left(\sqrt{\frac{\mathrm{vol}(\bar{A})}{\mathrm{vol}(A)}} + \sqrt{\frac{\mathrm{vol}(A)}{\mathrm{vol}(\bar{A})}}\right)^2 + \frac{1}{2}\sum_{i\in\bar{A}, j\in A} w_{ij}\left(-\sqrt{\frac{\mathrm{vol}(\bar{A})}{\mathrm{vol}(A)}} - \sqrt{\frac{\mathrm{vol}(A)}{\mathrm{vol}(\bar{A})}}\right)^2$$

$$= \mathrm{cut}(A, \bar{A})\left(\frac{\mathrm{vol}(\bar{A})}{\mathrm{vol}(A)} + \frac{\mathrm{vol}(A)}{\mathrm{vol}(\bar{A})} + 2\right)$$

$$= \mathrm{cut}(A, \bar{A})\left(\frac{\mathrm{vol}(A) + \mathrm{vol}(\bar{A})}{\mathrm{vol}(A)} + \frac{\mathrm{vol}(A) + \mathrm{vol}(\bar{A})}{\mathrm{vol}(\bar{A})}\right)$$

$$= \mathrm{vol}(V)\left(\frac{\mathrm{cut}(A, \bar{A})}{\mathrm{vol}(A)} + \frac{\mathrm{cut}(A, \bar{A})}{\mathrm{vol}(\bar{A})}\right)$$

$$= \mathrm{vol}(V)\mathrm{Ncut}(A, \bar{A})$$

所以可以將問題表示如下:

$$\min_{A\subset V} \boldsymbol{f}^{\mathrm{T}}\boldsymbol{L}\boldsymbol{f} \text{ subject to } \boldsymbol{D}\boldsymbol{f} \perp \mathbf{1} \text{ 且 } f_i \text{ 定義於(2.21)，} \boldsymbol{f}^{\mathrm{T}}\boldsymbol{D}\boldsymbol{f} = \mathrm{vol}(V)$$

同樣可以將 $\boldsymbol{f}$ 放寬限制將問題變成:

$$\min_{\boldsymbol{f}\in\mathbb{R}^n} \boldsymbol{f}^{\mathrm{T}}\boldsymbol{L}\boldsymbol{f} \text{ subject to } \boldsymbol{D}\boldsymbol{f} \perp \mathbf{1}，\boldsymbol{f}^{\mathrm{T}}\boldsymbol{D}\boldsymbol{f} = \mathrm{vol}(V)$$

最後將 $\boldsymbol{f} = \boldsymbol{D}^{-1/2}\boldsymbol{g}$ 代入:

$$\min_{\boldsymbol{g}\in\mathbb{R}^n} \boldsymbol{g}^{\mathrm{T}}\boldsymbol{D}^{-1/2}\boldsymbol{L}\boldsymbol{D}^{-1/2}\boldsymbol{g} \text{ subject to } \boldsymbol{g} \perp \boldsymbol{D}^{1/2}\mathbf{1}，\|\boldsymbol{g}\|^2 = \mathrm{vol}(V)$$

其中，$\boldsymbol{D}^{-1/2}\boldsymbol{L}\boldsymbol{D}^{-1/2} = \boldsymbol{L}_{\mathrm{sym}}$，且如前所述 $\boldsymbol{L}_{\mathrm{sym}}$ 相對於 λ_1 之 eigenvector 為 $\boldsymbol{D}^{1/2}\mathbf{1}$，所以根據 Rayleigh-Ritz theorem，取 $\boldsymbol{g}$ 為 $\boldsymbol{L}_{\mathrm{sym}}$ 相對於 λ_2 之 eigenvector，則 $\boldsymbol{f} = \boldsymbol{D}^{-1/2}\boldsymbol{g}$。最後如同(2.20)，可以利用 $\boldsymbol{f}$ 將 V 分成 A 與 $\overline{\mathrm{A}}$。

2.3.4 Spectral clustering 演算法

這一小節將介紹兩個較有名的 spectral clustering 演算法。分別為 Shi and Malik 在 2000 年所提出的演算法[5]，以下為其虛擬碼：

Spectral clustering according to Shi and Malik
Input: A set of point $X = \{x_1, \ldots x_n\}$, number k of clusters to construct .
Output: Clusters $A_1, \ldots, A_k$ with $A_i = \{j \mid y_j \in C_i\}$.

1. Construct $\boldsymbol{W}$ and $\boldsymbol{D}$, $\mathbf{W}_{ij} = \exp\left(-\left\|x_i - x_j\right\|^2 / 2\sigma^2\right)$ if $i \neq j$, and $\mathbf{W}_{ii} = 0$. $\boldsymbol{D}$ is a diagonal matrix and $\boldsymbol{D}_{ii}$ is the sum of $\boldsymbol{W}'s$ i-th row .
2. Compute $\boldsymbol{L}_{rw} = \boldsymbol{D}^{-1}\boldsymbol{L}$, $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W}$.
3. Compute the first k eigenvectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k$ of $\boldsymbol{L}_{rw}$.
4. Let $\boldsymbol{U} \in \mathbb{R}^{n \times k}$ be the matrix containing the vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k$ as columns .
5. For $i = 1, \ldots, n$, let $\boldsymbol{y}_i \in \mathbb{R}^k$ be the vector corresponding to the i-th row of $\boldsymbol{U}$.
6. Cluster the point $(\boldsymbol{y}_i)_{i=1,\ldots,n}$ in $\mathbb{R}^k$ with the k-means algorithm into clusters $C_1, \ldots, C_k$.

這個演算法非常的簡單，首先，必須先決定要將這些點分成多少群，標記為 k。利用 Gaussian function 計算每兩點之間的相似度，建構出矩陣 $\boldsymbol{W}$。利用矩陣 $\boldsymbol{W}$ 就可以計算 $\boldsymbol{L}_{rw}$，得到 $\boldsymbol{L}_{rw}$ 之後就可以計算 $\boldsymbol{L}_{rw}$ 之 eigenvalue 與 eigenvector。接著取前 k 小的 eigenvalue 所對應的 eigenvector $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k$ 形成矩陣 $\boldsymbol{U}$，其中，矩陣 $\boldsymbol{U} = [\boldsymbol{v}_1 \quad \ldots \quad \boldsymbol{v}_k]$。矩陣 $\boldsymbol{U}$ 是一個 n×k 矩陣，我們用矩陣 $\boldsymbol{U}$ 的第 i 列來代表點 x_i 的座標，因此我們可以賦予每個點一個新的座標，接著對此新的座標執行 k-means 所得到的分群結果即為最後的分群結果。

Ng, Jordan, and Weiss 在 2002 年提出了另一個演算法[6]，這個演算法與 Shi and Malik 所提出的有些許不同，其虛擬碼如下:

Spectral clustering according to Ng, Jordan, and Weiss
Input: A set of point $X = \{x_1, \ldots x_n\}$, number k of clusters to construct .
Output: Clusters $A_1, \ldots, A_k$ with $A_i = \{j \mid y_j \in C_i\}$.

1. Construct $\boldsymbol{W}$ and $\boldsymbol{D}, \boldsymbol{W}_{ij} = \exp\left(-\|x_i - x_j\|^2 / 2\sigma^2\right)$ if $i \neq j$, and $\boldsymbol{W}_{ii} = 0$. $\boldsymbol{D}$ is a diagonal matrix and $\boldsymbol{D}_{ii}$ is the sum of $\boldsymbol{W}'$s i-th row .
2. Compute $\boldsymbol{L}_{sym}$.
3. Compute the first k eigenvectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k$ of $\boldsymbol{L}_{sym}$.
4. Let $\boldsymbol{U} \in \mathbb{R}^{n \times k}$ be the matrix containing the vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k$ as columns .
5. Form the matrix $\boldsymbol{T} \in \mathbb{R}^{n \times k}$ by normalizing the rows to norm 1 .
6. For $i = 1, \ldots, n$, let $\boldsymbol{y}_i \in \mathbb{R}^k$ be the vector corresponding to the i-th row of $\boldsymbol{T}$.
7. Cluster the point $(\boldsymbol{y}_i)_{i=1,\ldots,n}$ in $\mathbb{R}^k$ with the k-means algorithm into clusters $C_1, \ldots, C_k$.

這個演算法不同的地方在於，它將 $\boldsymbol{L}_{rw}$ 換成 $\boldsymbol{L}_{sym}$。且在計算完矩陣 $\boldsymbol{U}$ 之後，必須將矩陣 $\boldsymbol{U}$ 的每一個列向量正規化成長度為1，形成矩陣 $\boldsymbol{T}$ 。然後用矩陣 $\boldsymbol{T}$ 的第i列來代表點 x_i 的座標，再利用 k-means 對這些座標分群。

第三章、系統設計

3.1 概念

在本篇論文中，假設若兩個部落格裡的文章其描述的主題都相近，則猜測部落格的作者有相同的興趣。本論文利用一個部落格的標籤雲來代表整個部落格，這樣的做法各有其優缺點。優點是可以避免處理大量的文章資料，通常在自然語言處理中利用計算每一個詞的頻率來找出重要的詞彙，這個方法必須統計所有文章中每一個詞出現的次數，這樣的做法需要花費較多的時間。而標籤雲裡的標籤是作者在撰寫文章之後標記在文章上，可以視為文章之概念或標題，也因為它是由人工所產生，所以更具代表性也更精確。但是，其缺點也是因為它是由人工自由產生，所以造成在描述相同之概念時卻有可能使用不同的詞彙。舉例來說:兩個部落格的主題可能都是關於旅遊，大部分部落格文章都是關於出遊的遊記或是旅遊景點的介紹，但是這兩個部落格的標籤雲中卻可能完全沒有重複的標籤；假設這兩個部落格都沒有重複的標籤，在這個情況下，如果以兩個部落格的標籤重複的程度來代表它們之間的相似程度，則這兩個都是以旅遊為主題的部落格相似度將為零，因此，本論文將針對該問題提出適當的演算法以克服這個困難，然後再嘗試各種不同的分群演算法，試著找出相同主題的部落格。

3.2 系統架構

本系統分成幾個步驟來完成：首先，收集部落格之標籤雲，接著將收集來的標籤做前置處理，在這個步驟中將刪除一些不合適的標籤。本論文提出，利用搜尋引擎的幫助來計算標籤與標籤之間的相似度。在獲得相似度之後，本論文使用兩個不同的方式來表示部落格。最後，分別使用不同的分群法來將這些部落格分群，然後觀察是否在相同群中的部落格都討論相同的主題，來判斷分群結果的好壞。圖 3-1 為系統流程圖。

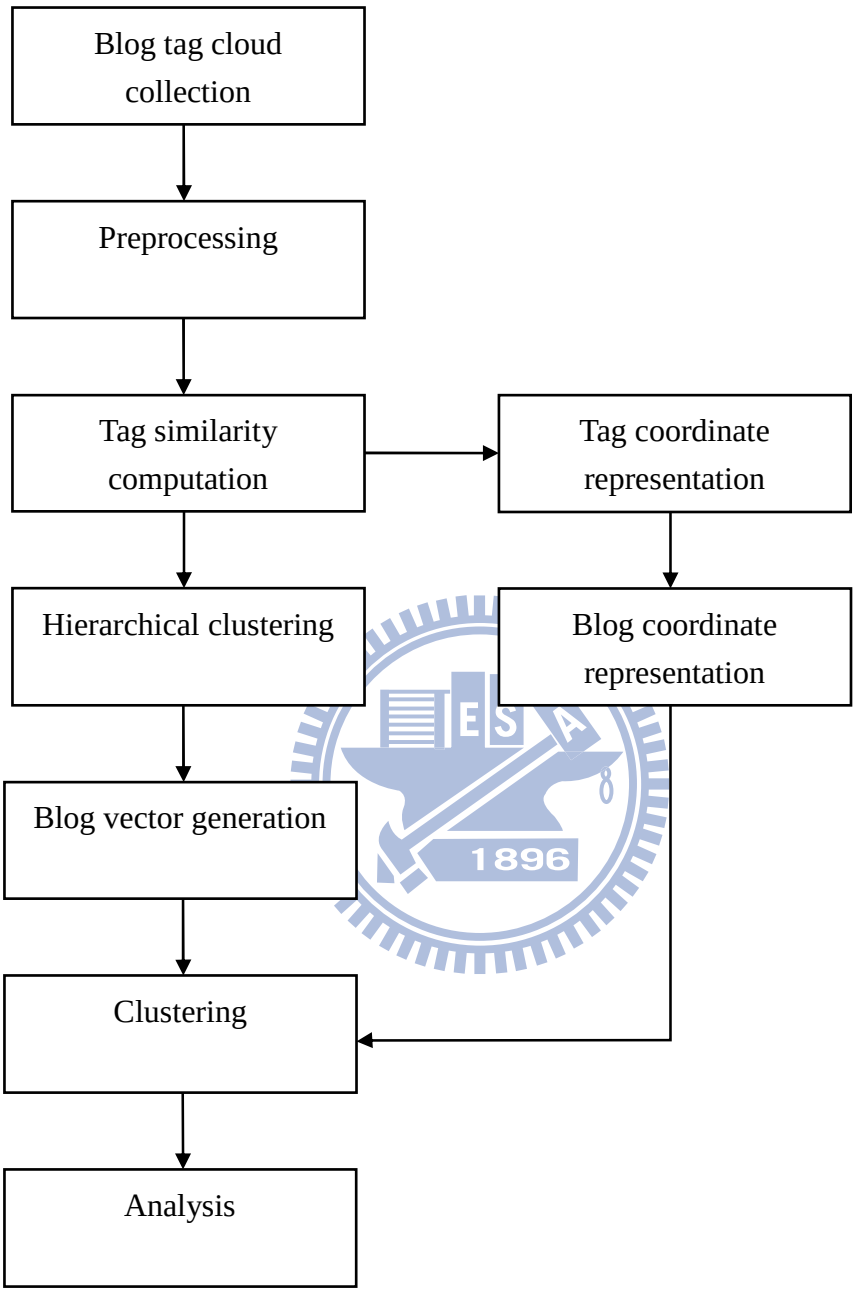

Blog tag cloud collection
Preprocessing
Tag similarity computation
Tag coordinate representation
Hierarchical clustering
Blog coordinate representation
Blog vector generation
Clustering
Analysis

圖 3-1 系統流程圖

3.3 收集部落格之標籤雲

本論文在網路上收集了 150 個英文部落格的標籤雲，這些部落格所討論的主題分別是電影(movie)、旅遊(travel)與健康(health)，每一個類別各包含 50 個部落格，而這三個類別的部落格所包含的標籤加起來總共有 2447 個。在這些標籤中，有 2068 個標籤只出現一次，只有 379 個標籤重複出現 2 次以上，大約佔所有標籤的 15%，從這個數據可以證明使用者自訂的標籤重複率確實非常的低。在這邊要特別註明的是，系統在執行分群時不會知道每一個部落格的類別，類別的資訊只用在分析分群結果的好壞。

3.4 前置處理

由於每一個標籤都是由部落格作者自由產生，沒有任何限制，所以有些標籤可能無法達到描述部落格主題的功能或是對分群可能沒有任何幫助，必須將這一類的標籤給刪除。舉例來說：在電影的部落格中，許多作者會使用「2008, 2009, 2010, …」這樣的標籤來描述電影上映的年份，但是任何主題的部落格都可以含有這一類的標籤，且標籤本身不含任何語意，所以這一類的標籤將被濾掉。接著，只包含單一個字母的標籤也被過濾掉，因為單一字母標籤無法提供足夠資訊。另外，大於三個詞(word)所組成的標籤也被去除，因為根據實驗結果，過長的標籤與其他標籤的相似度很容易是零，它無法表現出「與某標籤較像」或是「與某標籤較不像」這樣的訊息。

3.5 計算所有標籤之間的相似度

每當在搜尋引擎輸入一個關鍵字之後，搜尋引擎就會找到含有這個關鍵字的網頁並回傳給使用者，這就是搜尋引擎的基本運作原理。本論文使用搜尋引擎 Altavista[7]來幫助計算相似度，假設要計算「標籤 1」與「標籤 2」之相似度時，首先在搜尋引擎中輸入：

標籤 1 NEAR 標籤 2

則搜尋引擎會搜尋所有符合以下條件的網頁：

1.網頁同時包含「標籤 1」與「標籤 2」。

2.「標籤 1」與「標籤 2」在網頁上出現的位置相差在 10 個句子以內。

在 Peter D. Turney[8]的論文中，他使用 NEAR 運算子來計算 PMI。同時，搜尋引擎也會回傳符合以上條件之網頁的數目，將搜尋引擎搜尋到的網頁數目標記為：

hits(標籤 1,標籤 2)

得到 hit 數之後，兩個標籤之相似度就定義為：

Similarity(標籤 1,標籤 2) = log(hits(標籤 1,標籤 2))

相似度不直接使用 hit 數而要取 log 的原因是因為目前網際網路上的網頁數量實在過於龐大，取 log 可以讓這個值縮小，方便之後的計算。

這樣的相似度計算方式可以將它理解成「若兩個標籤一起出現的次數越頻繁，則它們的相似度越高」；這樣的相似度非常適合用在這裡，因為在描述一個主題時，一定會有常用於該主題的詞彙，只要撰寫該主題的文章時，這些詞彙就會常常一起出現。舉例來說，可以試著計算「王建民」與 「伸卡球」和「王建民」與「籃網」之間的相似度，分別為：

Similarity(王建民, 伸卡球) = 4.99

Similarity(王建民, 籃網) = 2.46

從這個例子中可以看到，如果有一個部落格 A 包含「王建民」這個標籤，而另外兩個部落格 B 與 C 分別包含「伸卡球」與「籃網」這兩個標籤，則系統會因為標

籤之間的相似度認為部落格 A 與 B 較相似，事實上，因為「王建民」與 「伸卡球」常常在「棒球」這個主題下出現，所以它們計算出來的相似度也較高。

3.6 將部落格以向量表示

要將部落格表示成向量，首先就是要決定向量的維度，其中最簡單的方式就是每一個標籤都代表一個維度。假設所有的標籤總共有 n 個，則每一個部落格就可以表示為一個 n 維的向量。若一個部落格中包含某個標籤，則其向量表示法中代表該標籤之維度的值就是 1，否則就是 0。這個方式雖然非常的簡單也非常的直覺，但是並不適合用在這裡，原因就如前述，標籤是由人工自由產生，且重複率非常的低。舉例來說：假設所有標籤的集合為{action, animation, comedy, fantasy, airlines, hotels}，且有三個部落格 A、B 與 C，A 與 B 所描述的主題都是電影而 C 所描述的主題是旅遊，其中，部落格 A 包含標籤{action, animation}，部落格 B 包含標籤{comedy, fantasy}，部落格 C 包含標籤{airlines, hotels}，則部落格 A，B 與 C 可以用此向量表示法表示如下：

A = (1, 1, 0, 0, 0, 0)

B = (0, 0, 1, 1, 0, 0)

C = (0, 0, 0, 0, 1, 1)

若使用上述表示法，不管使用 distance 或是使用 cosine similarity 來計算部落格之間的相似度時，部落格 A 跟 B 的相似度與部落格 A 跟 C 的相似度都是一樣的，所以在這種表示法下，沒有辦法顯示出部落格 A 與 B 是比較相近的。所以必須用另一種能顯示部落格之間相似度的表示法。以上述例子來說，首先計算每兩個標籤之間的相似度並將結果紀錄在表 3-1：

	action	animation	comedy	fantasy	airline	hotels
action	--	7.89	7.9	7.48	5.92	7.08
animation		--	7.98	7.09	5.24	5.45
comedy			--	7.44	4.85	6.4
fantasy				--	4.62	7.18
airline					--	7.21
hotels						--

表 3-1 標籤之間的相似度

觀察表 3-1 之後可以發現前四個標籤{action, animation, comedy, fantasy}之間的相似度明顯較高而後兩個標籤{airlines, hotels}之間的相似度也較高，這與前四個標籤所描述之主題是「電影」與後兩個標籤所描述的主題是「旅遊」吻合，所以根據這個結果，本論文嘗試以每個維度來代表不同的「主題」來表示向量，也就是將向量表示為(主題 1, 主題 2, …)。因此，可以將標籤依照相似度分成數群，讓相似度高的標籤聚集在一起，因為根據前面討論的結果，描述同一個主題的標籤相似度較高，所以在分群完成之後可以合理的假設，同一個群集內的標籤都在描述相近的主題。若將上述例子中的標籤根據相似度分成兩群(分群演算法我們將在 3.6.1 討論)，可以到如圖 3-2 之結果。

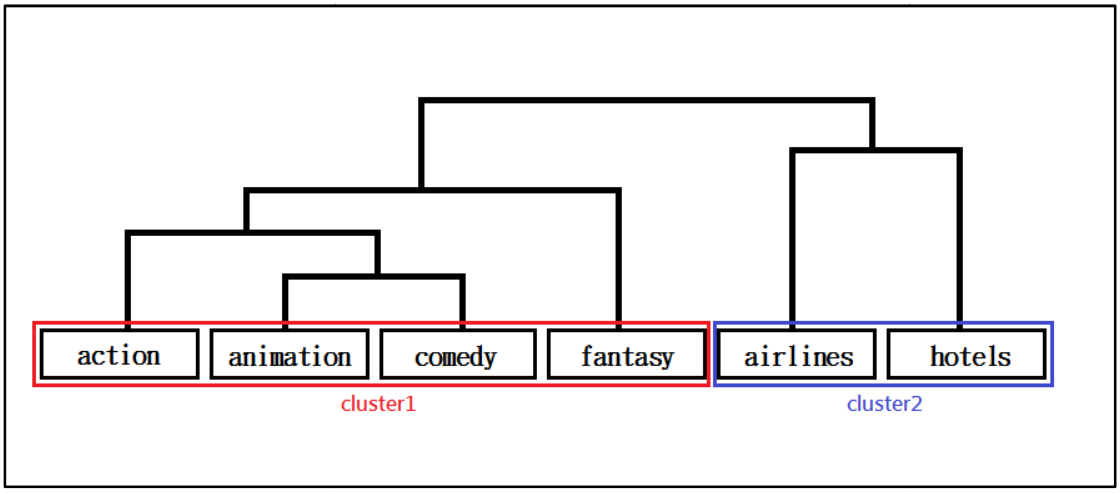


圖 3-2 標籤之分群結果

由圖 3-2 可知，第一群為{action, animation, comedy, fantasy}且第二群為{airlines, hotels}，我們將向量表示法稍微做一點修改，改成以每一個群集代表一個維度，也就是每一個主題代表一個維度，則部落格 A、B 與 C 的向量表示法就變成：

$$A = (2,0)$$

$$B = (2,0)$$

$$C = (0,2)$$

如此一來，若以此向量表示法來計算部落格之間的相似度，可以得到部落格 A 與 B 較像而部落格 A 與 C 較不像的結果，這樣就能正確的表達出部落格 A 與部落格 B 有可能在描述較相似的主題，而部落格 C 是在描述另一個主題。3.6.1 節將描述標籤分群的詳細步驟。

3.6.1 標籤分群

為了找出哪些標籤可能是代表相同的主題，本論文使用 Hierarchical clustering[9]來建立標籤的階層關係。一開始每一個標籤都是一個獨立的群，之後開始每次合併相似度最高的兩個群，一直到剩下一個群為止。

在計算群與群之間的相似度時，主要有三種方法，分別是 Single linkage、Complete linkage 與 Average linkage。本論文使用的是 Average linkage，Average linkage 的優點是較其他方法精確，缺點是計算起來較費時。但是在實際分群的過程中，很容易碰到群與群的數目不平均的情況，考慮圖 3-3 之情況：

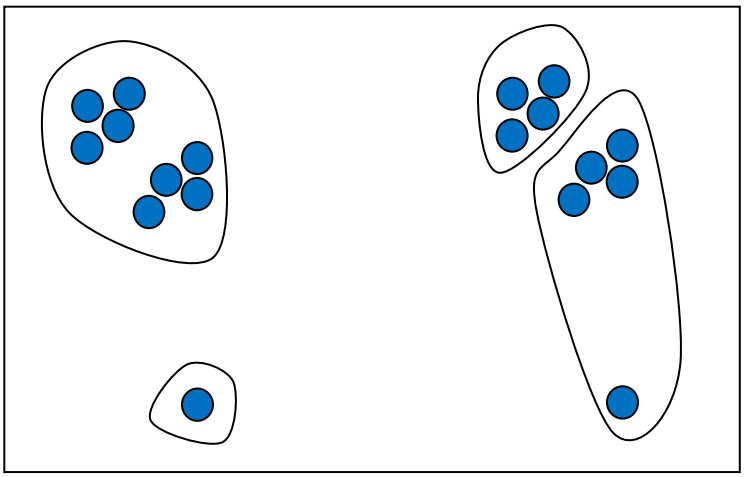


圖 3-3 不同的分群結果

如圖 3-3 左邊的分群結果，因為有一個標籤與其他所有標籤的相似度都很低，所以很容易造成該標籤自成一群而其他所有的標籤形成一個大群，這並不是執行分群演算法時想要看到的結果。較理想的分群結果應該是如圖 3-3 右邊的分群結果，每一個群中的標籤數量較平均，也較符合分群的目的。為了解決這個問題，必須在合併兩個群時考慮是否會造成標籤數目過多之情形，所以要將合併之後的群之標籤數量當成一個參數去調整群與群間的相似度，基本的想法是根據兩個群合併之後的標籤數量來降低這兩個群之間的相似度，若兩個群合併之後的標籤數目越多，則這兩個群的相似度降低越多，此時系統就會選擇其他兩個群來合併。因此將計算相似度的公式稍微調整為：

$$\text{Similarity}(g_i, g_j) = \text{Original Similarity}(g_i, g_j) \times \left(\frac{N - (\text{size}(g_i) + \text{size}(g_j))}{N}\right)^f$$

其中，Original Similarity(g_i, g_j)是原本使用 Average linkage 所計算出來的相似度，N是所有標籤的數量，size(g)代表標記為 g 這個群中的標籤數量，而 f 為一個參數，可以視最後的分群情況來調整，它代表兩個群合併之後的標籤數量對兩個群之間的相似度的影響程度。圖 3-4 為用新的相似度來挑選最相似的兩個群合併之情形：

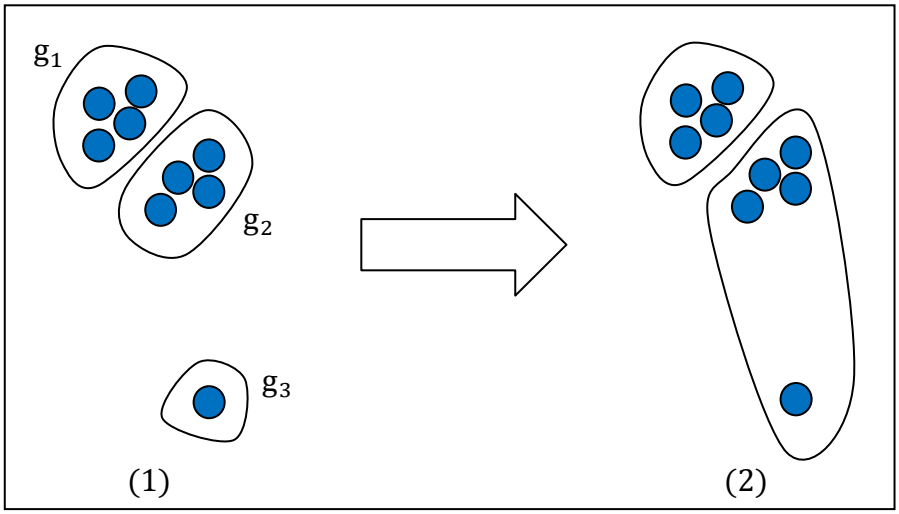


圖 3-4 用新的相似度來挑選最相似的兩個群合併

圖 3-4 的(1)中有三個群，分別為 g_1、g_2 與 g_3。如果使用 Average linkage 來計算群與群之間的相似度時，相似度分別為 s_{12}、s_{13} 與 s_{23}，其中 s_{ij} 代表 g_i 與 g_j 之間的相似度且 $s_{12} > s_{23} > s_{13}$，但是由於 g_1 與 g_2 合併之後的群包含八個標籤而 g_2 與 g_3 合併之後只有五個標籤，假設合併後的標籤數量調整相似度之後造成 $\text{adjust}(s_{23}) > adjust(s_{12})$，其中，$\text{adjust}(s_{ij})$ 就是 g_i 與 g_j 經過調整之後的相似度，所以系統最後選擇合併 g_2 與 g_3，合併之後的結果如圖 3-4 的(2)。

為了證實新的相似度計算方式確實有用，我們以實際的資料分別使用舊的相似度計算方式與新的相似度計算方式來比較:

首先，先觀察使用新的相似度計算方式將所有標籤分為 120 群時的分群狀況，表 3-2 列出前五大的群所包含之標籤之內容。舉例來說:最大的群共有 775 個標籤，其中 69 個標籤曾出現在電影類的部落格中，655 個標籤曾出現在旅遊類的部落格中，67 個標籤曾出現在健康類的部落格中，6 個標籤同時出現在電影與旅遊類，10 個標籤同時出現在旅遊與健康類。藉由表 3-2 可以發現，第一大的群中有 84.5%都是旅遊類的標籤，第二大的群有 84.6%都是電影類的標籤，而第三大的群則有 94.3%都是健康類的標籤。

標籤數	Movie	Travel	Health	M&T	M&H	T&H	M&T&H
775	69	655	67	6	0	10	0
546	462	56	46	12	5	5	4
470	14	26	443	0	0	13	0
455	103	196	203	16	9	29	7
12	0	0	12	0	0	0	0

表 3-2 120 群，新的相似度計算方法

接下來觀察使用舊的相似度計算方式將所有標籤分成 120 群之分群狀況，表 3-3 列出前五大的群所包含之標籤之內容。我們可以看到，第一群就包含了 2236 個標籤，佔了所有標籤的 91.4%，其中 28.9%的標籤曾出現在電影類中，41.5%的標籤曾出現在旅遊類中，33.8%的標籤曾出現在健康類中。第二大的群只包含了 12 個標籤，剩下的群則包含更少標籤。

標籤數	M	T	H	M&T	M&H	T&H	M&T&H
2236	647	928	755	34	14	57	11
12	0	0	12	0	0	0	0
10	10	0	0	0	0	0	0
9	9	0	0	0	0	0	0
7	0	0	7	0	0	0	0

表 3-3 120 群，舊的相似度計算方法

藉由觀察 120 群之情形可以發現，新的相似度計算方式可以將所有的標籤分成較平均的群，且每個群中的標籤大部分都來自同一類的部落格。而使用舊的相似度計算方式，大部分的標籤都落在同一個群中，且這個群中每個類別都大約佔三分之一，也就是沒辦法將不同類別的標籤分開。

表 3-4、3-5 為分別使用這兩種相似度計算方法將所有標籤分為 250、500 與 1000 群之情形，並記錄了前五大群的標籤數。藉由這裡的分析可以證明，不管分成多少群，新的相似度計算方法確實可以使每個群的大小較平均。

群數	1	2	3	4	5
250	415	358	348	332	259
500	237	189	188	142	140
1000	72	68	65	64	58

表 3-4 使用新的相似度計算方法分群時，前五大群的標籤數

群數	1	2	3	4	5
250	1980	14	10	10	7
500	1434	86	19	18	16
1000	718	42	39	32	29

表 3-5 使用舊的相似度計算方法分群時，前五大群的標籤數

3.6.2 產生部落格向量

在標籤分群一節中，使用 Hierarchical clustering 這個分群演算法的其中一個重要原因就是當階層關係建構完成之後，可以根據需求任意的選擇想要的群數，由以下討論可知，若能任意選擇群數就代表可以任意決定向量的維度，如圖 3-5：

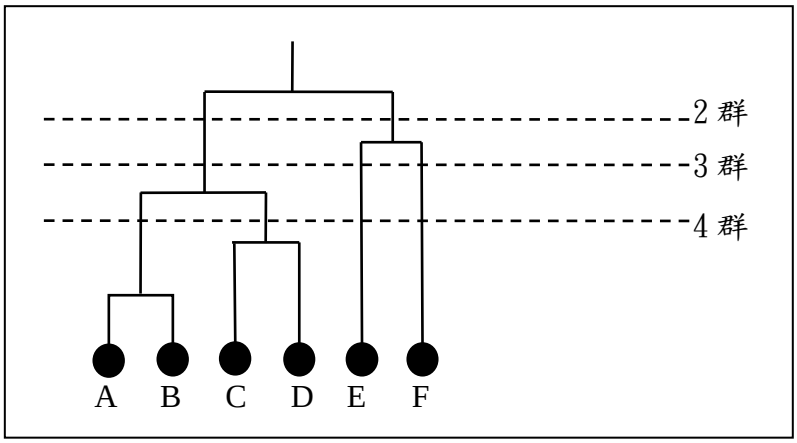


圖 3-5 任意選擇想要的群數

在圖 3-5 中假設標籤的集合為{A, B, C, D, E}，根據相似度建構完階層關係之後，可以任意選擇分成 2 群：{{A, B, C, D}, {E, F}}、3 群：{{ A, B, C, D }, {E}, {F}}或 4 群：{{A, B}, {C, D}, {E}, {F}}，甚至更多。

所以在決定群的數目 $\mathcal{X}$ 之後，就可以根據標籤分群所產生的階層關係將所有的標籤分成 $\mathcal{X}$ 群，如此一來，同一個群內的標籤可以將它們視為都在描述相同的主題，所以共有 $\mathcal{X}$ 個主題。

本論文將每一個部落格都表示成一個 $\mathcal{X}$ 維的向量，假設標籤有 n 個，由上面的敘述可知，$\mathcal{X}$ 的值可以是 1 到 n 之間的任何一個值，也就是說，可以任意決定向量的維度。本論文以一個群代表一個維度，每一個維度的值代表部落格的標籤裡屬於代表該維度之群的個數。舉例來說：假設某一個部落格所包含的標籤為{A, C, F}，若採用圖 3-5 的分群結果，取$\mathcal{X}$ = 3的情況下，此部落格之向量表示法為(2, 0, 1)。若取$\mathcal{X}$ = 4，則此部落格之向量表示法為(1, 1, 0, 1)。

在獲得每一個部落格的向量之後，最後一個步驟就是將向量正規化。要做正規化的原因是因為每一個部落格的標籤數量都不相同，為了避免「標籤數量」這個因素影響最後分群的結果，所以要將向量正規化成長度為 1。舉例來說：有兩個相同主題的部落格 A 與 B，它們各包含 5 個標籤與 100 個標籤，分別為：

$$\{a_1, a_2, \ldots, a_5\}$$

與

$$\{b_1, b_2, \ldots, b_{100}\}$$

且這些標籤都不相同，而另一個不同主題的部落格 C 包含 5 個標籤：

$$\{c_1, c_2, \ldots, c_5\}$$

假設在標籤分群的步驟中，系統將所有標籤分成三群分別為：

$$g_1 = \{\ldots, a_1, a_2, \ldots, a_5, \ldots, b_1, b_2, \ldots, b_{100}, \ldots\}$$

$$g_2 = \{\ldots, c_1, c_2, \ldots, c_5, \ldots\}$$

$$g_3 = \{\ldots\ldots\}$$

可以看到部落格 A 與 B 的 105 個標籤都很「正確」的被分到了第一個群中，而部落格 C 的標籤被分到了第二個群中，所以部落格 A、B 與 C 的向量表示法分別為 A=(5, 0, 0)、B=(100, 0, 0)與 C=(0, 5, 0)。如果接下來在計算部落格之間的相似度時是使用「距離」來代表部落格之間的相似度時，距離越近代表相似度越高，距離越遠代表相似度越低，則部落格 A 與 B 的距離為 95，部落格 A 與 C 的距離大約為 7，因此系統認為部落格 A 與 C 的主題較相近，這與事實不符。因此，為了降低這種因為標籤數量而導致系統誤判的情況，所以需要加入正規化這個步驟。以上述例子來說，經過正規化後 A=(1, 0, 0)、B=(1, 0, 0)、C=(0, 1, 0)，可以很清楚的看到，若同樣以距離來代表相似度時，部落格 A 與 B 為較相似的主題。因此，這個正規化後的向量就是最終部落格的向量表示法。

3.7 以座標表示部落格

另一個表示部落格的方法，就是為每一個部落格計算一個合適的座標。所謂合適的座標代表座標之間的距離可以反映出部落格之間的相似程度。

3.7.1 使用 Multidimensional scaling 產生標籤之座標

在計算完所有標籤與標籤之間的相似度後，就可以使用 Multidimensional scaling 去賦予每一個標籤一個合適的座標。但是這裡有一點要特別注意，因為在本論文所使用的相似度計算方法中，數值越小代表相似度越低，數值越大代表相似度越高。但 Multidimensional scaling 對相似度的定義剛好是相反的，因為在 Multidimensional scaling 中輸入的是節點與節點之間的距離，直覺上，越相似的標籤被賦予的座標應該要越接近，所以距離越小代表相似度越高，而距離越大代表相似度越低。因此在這個步驟中要做的事情就是將相似度轉換成距離，相似度越高要轉換成越小的距離，反之，相似度越低則要轉換成越大的距離。本論文提出下列公式將相似度轉換成距離：

$$\text{distance}(t_1, t_2) = (\text{Max Similarity} + 1) - \text{Similarity}(t_1, t_2)$$

其中，Max Similarity 是指所有標籤中，相似度最高的兩個標籤的相似度。Max Similarity + 1 的原因是避免讓相似度最高的兩個標籤距離是 0，因為只有「完全相同」的兩個標籤距離才是 0，經過這個公式的轉換，就可以將相似度轉換成距離。

3.7.2 計算部落格之座標

在獲得每一個標籤的座標之後，就可以利用標籤的座標去計算部落格的座標。如果利用 Multidimensional scaling 產生之標籤座標的維度是三維，則部落格之座標的維度也是三維。假設有一部落格有 n 個標籤，其標籤的座標分別為：

$$(x_1, y_1, z_1)、(x_2, y_2, z_2)、\ \ldots\ 、(x_n, y_n, z_n)$$

則部落格之座標 (x, y, z) 就是這些標籤之座標的中心點，其計算方法為：

$$x = \frac{x_1 + x_2 + \cdots + x_n}{n}$$

$$y = \frac{y_1 + y_2 + \cdots + y_3}{n}$$

$$z = \frac{z_1 + z_2 + \cdots + z_3}{n}$$

經過計算所得之座標 (x, y, z) 即為代表部落格之座標。

3.8 分群

將標籤表示成向量或者座標之後，就可以利用現有的分群法來將標籤分群。本論文將使用 kmeans 與 spectral clustering 來測試是否能將不同主題的部落格分開。

若使用 spectral clustering 時，會需要計算所有部落格之間的相似度，通常在使用 spectral clustering 時都是用 Gaussian similarity function 來計算所有點與點之間的相似度，其定義為：

$$\text{Similarity}(x_i, x_j) = \exp\left(\frac{-\|x_i - x_j\|^2}{2\sigma^2}\right)$$

3.9 分析結果

本論文採用 F1 cluster evaluation measure[10]來評估最後分群結果的好壞。當系統在收集資料時共收集了三個主題的部落格，分別是電影、旅遊與健康類。系統在執行分群演算法時，並不會知道每一個部落格所屬的類別，它只會根據部落格與部落格之間的相似度將所有的部落格分成三群，然後再將分群的結果與真實的類別比較。比較兩個部落格有下列四種可能:

1. True Positives(TP):系統將兩個部落格分在同一群，而這兩個部落格也是屬於同一個主題。

2. False Positives(FP):系統將兩個部落格分在同一群，但是這兩個部落格是不同主題。

3. True Negatives(TN):系統將兩個部落格分在不同群，而這兩個部落格也是不同主題。

4. False Negatives(FN):系統將兩個部落格分在不同群，但這兩個部落格是同一個主題。

因此，就可以計算 precision、recall 與 F1，分別定義如下:

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{F1} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

第四章、實驗過程與結果討論

4.1 實驗資料

本論文之實驗資料為網路上收集的150個英文部落格之標籤雲，這些部落格所討論的主題分別是電影、旅遊與健康類，每一個類別都包含50個部落格，而這三個類別的部落格所包含的標籤加起來總共有2447 個。本論文將使用類別的資訊當成最後分群結果的答案。

4.2 實驗步驟

本論文將實驗分成數個部分。首先，測試不使用 hierarchical clustering 將相似度高的標籤合併，直接使用 2447 個維度來表示部落格向量，將結果紀錄在表 4-1。接下來測試使用 hierarchical clustering 將相似度高的標籤合併的結果，我們分別使用六個不同的維度來表示部落格，並用三種不同的分群演算法將部落格分群，將結果紀錄在表 4-2、4-3 與 4-4。最後，將使用 Multidimensional scaling 計算出來的部落格座標也使用 kmeans 與 spectral clustering 等分群演算法分群，並將結果紀錄在表 4-5。

4.3 實驗結果

在本表4-2中，(a)代表spectral clustering according to Ng, Jordan, and Weiss，(b)代表 spectral clustering according to Shi and Malik。

演算法	precision	recall	F1
kmeans	0.366	0.671	0.474
(a)	0.497	0.710	0.585
(b)	0.479	0.749	0.584

表 4-1 對 2447 維的部落格向量使用不同分群演算法之結果

維度	precision	recall	F1
250	0.769	0.805	0.787
120	0.817	0.831	0.824
60	0.791	0.807	0.799
30	0.545	0.692	0.610
10	0.553	0.683	0.611
3	0.329	1.000	0.495

表 4-2 各種不同維度的向量使用 kmeans 分群的結果

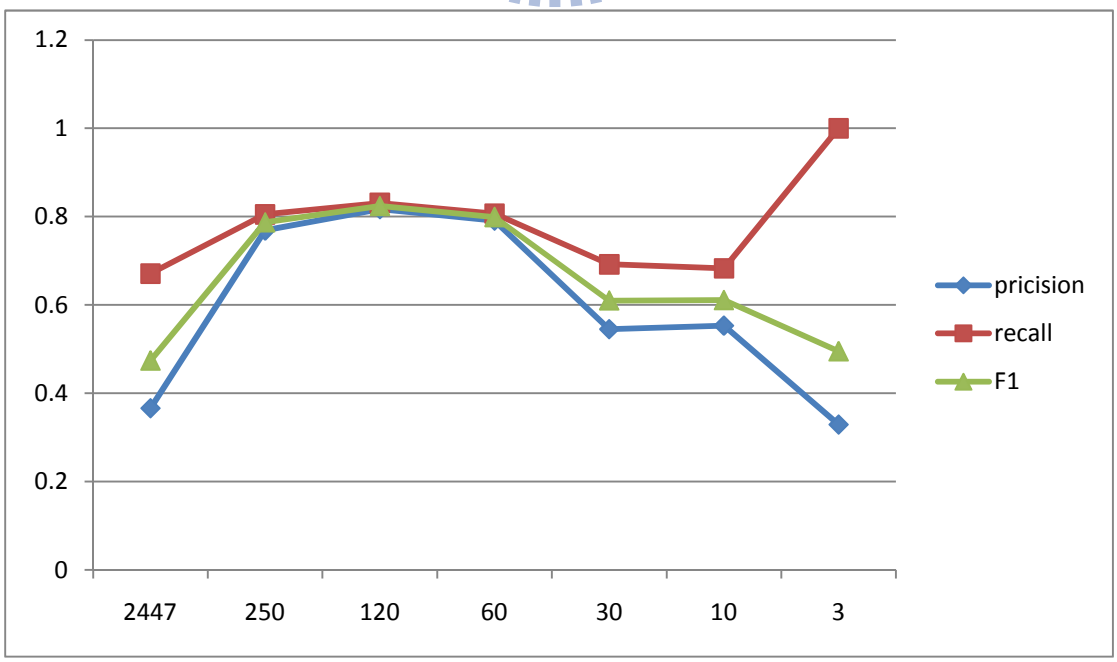


圖 4-1 各種不同維度的向量使用 kmeans 分群之曲線圖

維度	precision	recall	F1
250	0.796	0.824	0.810
120	0.829	0.841	0.835
60	0.791	0.807	0.799
30	0.564	0.642	0.600
10	0.560	0.624	0.591
3	0.329	1.000	0.495

表 4-3 各種不同維度的向量使用演算法(a)分群的結果

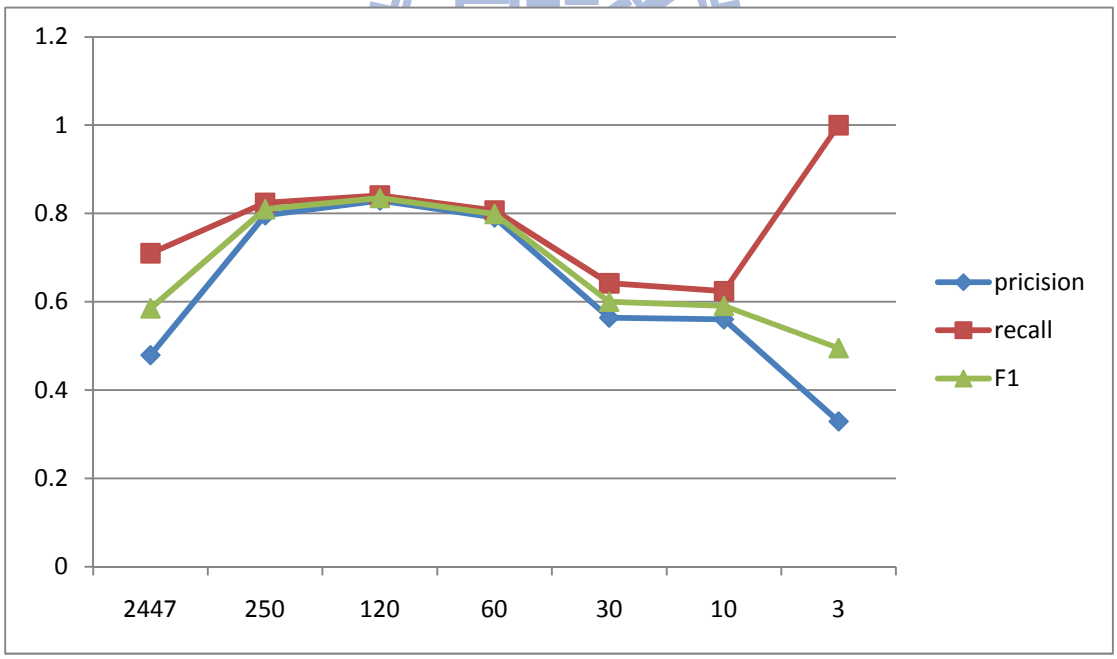


圖 4-2 各種不同維度的向量使用演算法(a)分群之曲線圖

維度	precision	recall	F1
250	0.771	0.807	0.789
120	0.829	0.841	0.835
60	0.791	0.807	0.799
30	0.564	0.642	0.600
10	0.560	0.624	0.591
3	0.329	1.000	0.495

表 4-4 各種不同維度的向量使用演算法(b)分群的結果

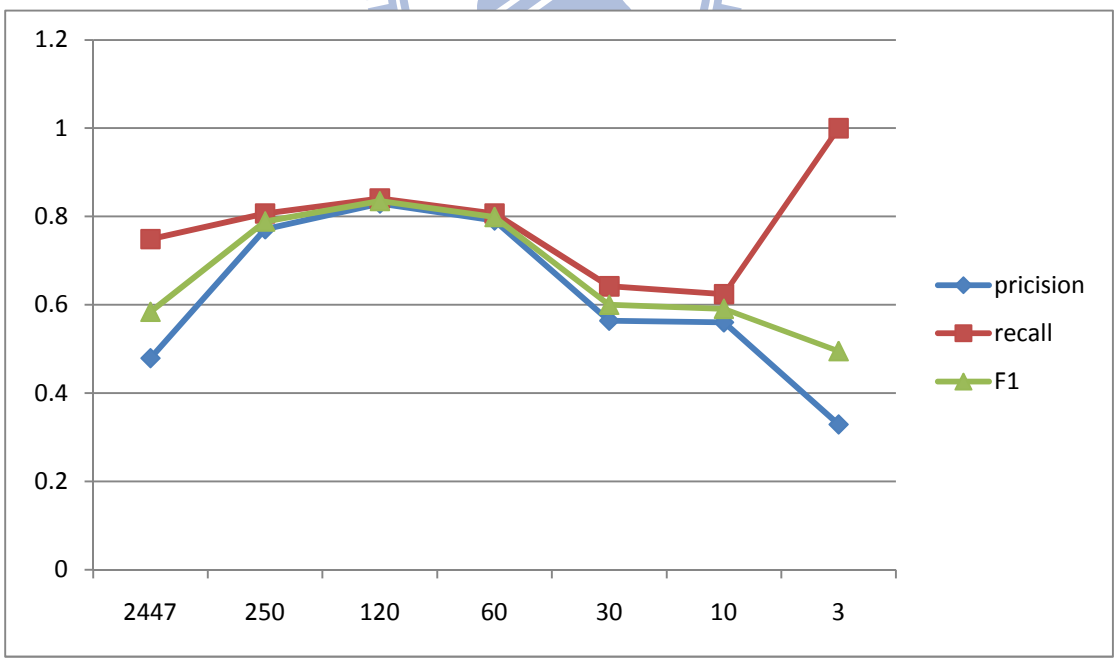


圖 4-3 各種不同維度的向量使用演算法(b)分群之曲線圖

演算法	precision	recall	F1
kmeans	0.870	0.874	0.872
(a)	0.857	0.86	0.859
(b)	0.870	0.873	0.871

表 4-5 對部落格座標分群的結果

4.4 實驗討論

從第一個實驗中可以觀察到，如果將部落格向量設定為每一個標籤都代表一個維度，也就是2447 維，不管使用哪一種分群演算法，其precision、recall與F1的值都非常的低。之後使用hierarchical clustering將相似度高的標籤合併之後，其precision、recall與F1的值就會漸漸上升。部落格向量維度大約在120維時效果最好，120大約是所有標籤數2447的5%。當維度漸漸縮小時，效果就越來越差。效果變差的原因是因為，系統中會存在某些標籤與其他所有的標籤相似度都很低，甚至都是0。在分群的過程中，這些標籤會一直保持自己一群的狀態，此時系統就會選擇合併不同主題的標籤。當有不同主題的標籤被合併成為一群時，產生的部落格向量對於分辨這兩個主題的能力就會降低。最後形成precision與F1的值隨著維度降低的結果。而recall最後會升高的原因是因為在維度是3時，系統將所有的部落格都分到了同一群，導致FN的值為0，所以recall的值為1。當我們使用座標來表示部落格時，並不會有上述問題，所以能得到較好的效果。

第五章、結論與展望

5.1 研究總結

根據實驗結果可以發現，雖然只使用標籤雲來代表部落格，但是系統可以把90%的部落格分配到正確的群中，這代表用標籤雲來代表部落格是非常適合的。雖然本論文使用三個主題的部落格進行實驗，但是因為本系統可以準確的指出哪些部落格是相似的，所以未來在應用時，是不限定任何主題或是類別的，只要給定一個部落格，系統就能找出與它相似的部落格並回傳給使用者。

5.2 未來研究

雖然現在網路上有非常多的部落格，但是並不是每一個部落格作者都有替每一篇部落格文章標記標籤的習慣，未來可以研究如何替一個部落格自動產生標籤雲，如此一來，系統可以根據自動產生的標籤雲在網路上搜尋類似的部落格，如此可以提供使用者更多的選擇。

參考文獻

[1] tag cloud, http://en.wikipedia.org/wiki/Tag_cloud

[2] Hervé Abdi, Metric Multidimensional Scaling(MDS):Analyzing Distance Matrices.

[3] Ulrike von Luxburg, A Tutorial on Spectral Clustering.

[4] Rayleigh-Ritz theorem, http://myyn.org/m/article/rayleigh-ritz-theorem/

[5] Shi,J. and Malik,J., Normalized cuts and image segmentation.IEEE Transactions on Pattern Analysis and Machine Intelligence, pp.888-905, 2000.

[6] Ng,A.,Jordan,M.,and Weiss,Y., On spectral clustering:analysis and an algorithm. In Neural Information Processing Systems 14, pp.849-856, 2002.

[7] Altavista, http://www.altavista.com/

[8] Peter D. Turney, Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews . Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL), Philadelphia, pp.417-424, 2002.

[9] Hierarchical clustering, http://www.resample.com/xlminer/help/HClst/HClst_intro.htm

[10] Daniel Ramage, Paul Heymann, Christopher D.Manning, and Hector Garcia-Molina, Clustering the Tagged Web. In Second ACM International Conference on Web Search and Data Mining(WSDM), 2009.