應用 Google N-gram 與 PTFICF 的自動文件分類系統

林俊良
國立高雄第一科技大學 資訊管理研究所
u9924812@nkfust.edu.tw

黃文楨
國立高雄第一科技大學 資訊管理研究所
wenh@nkfust.edu.tw

摘要

文件自動分類是將性質相近的文件放在相同或靠近的位置，以便搜尋者能有效率且迅速地得到所需資料。本研究提出以 N-gram 的方法來實作自動文件分類系統，本實驗結合刪除停用詞(Stopwords)、還原字根(Stemming)與 Google N-gram 語料庫，建立一套 N-gram 的文件斷詞演算法(N-gram Segmentation Algorithm, NSA)。本實驗資料使用 Classic4 資料集，透過 NSA 擷取文件中的關鍵字詞，並使用機率式詞頻與反類別頻率(Probability based Term Frequency and Inverse Category Frequency, PTFICF)將特徵值加權，最後使用 SVM 訓練萃取出特徵值。實驗結果平均準確率達到 96.5%，F_1 value 達到 96.4%，分類效果優於傳統提出的方法。

關鍵詞：Google N-gram、SVM、文件分類

壹、前言

隨著網路技術不斷地進步，有用的資訊也相對地大量成長中。雖然網路上舉手可得的資訊方便人們取得與傳遞資訊，但是當網路資訊量愈來愈大時，如何有效、且快速地取得有用的資訊，便成為非常重要的事情。此時，文件分類(text categorization)技術，即透過演算法分析一電子文件後，將其分配(assign)給一或多個類別(categories)，便扮演著其中重要的角色。

文件分類，需要瞭解文件的主題大意，才能給定類別，因此是相當高階的知識處理工作。要將文件分類自動化， 必須先整理出分類時的規則，電腦才能據以執行。 然而，多數的分類工作，其分類規則通常難以用人工分析歸納獲得。 因此，機器在做自動分類之前，還必須加以訓練， 使其自動學習出人工分類的經驗與知識。

現今自然語言理解的技術，還無法讓電腦瞭解任意的自由文句。因此機器在做文件分類時，常將文件分解成一個個語意較小的單位， 通常為文件的關鍵詞彙，或稱「特徵詞彙」， 再從這些詞彙與類別中找出對應的關係。 有時分類的問題，簡單到只要文件的某個欄位中出現什麼特徵詞， 就分到什麼類別去。但大部分的情況都沒那麼簡單。

例如，「科技」這個類別，如何界定哪些詞彙一定是屬於這個類別，哪些不是？顯然某些詞彙對這個類別較重要（比較有鑑別力），其他的則較不重要（比較不具鑑別力）。如何決定這些詞彙在每個類別的權重，正是機器學習(Machine Learning) 可以派上用場的地方。

過去文件分群方法可分為非監督式與監督式兩種作法，非監督式是透過詞頻分析選出最適當的詞來代表文章類別，但由於單純出現在文章的詞並不一定產生對人類有足夠意義的標籤，例如以一篇籃球為主的新聞，卻不一定會出現「basketball」，反而是「Linsanity」關鍵字，。而監督式是使用分類器進行訓練，同樣經過詞頻分析後，能夠有效將文件關聯到正確的類別，因此我們選擇監督式的作法。本研究提出的文件分類方式能夠應用在文件或網誌分類，能夠減少人工的介入，提升便利性，增加使用者的使用意願。

本研究建立一套 N-gram 斷詞演算法(N-gram Segmentation Algorithm, NSA)，結合刪除停用詞(Stopwords)、還原字根(Stemming)等方法，將文章進行斷詞和斷句，最後再搭配 Google N-gram Database 伍、[1]當作我們 N-gram 的語詞庫，以檢驗斷詞結果是否相符合。本研究採用監督式作法，文件透過 NSA 得出的 N-gram 特徵值套入支持向量機(Support Vector Machine, SVM)來分析詞頻進行自動文件分類，許多過去的文件並未分類，若以人工分類的方式，明顯會效率不足，但若使用 NSA 文分類系統，將能改善傳統分類效率上的問題。

貳、文獻探討

本章節將介紹有關文件分類的相關作法，以了解如何讓機器去辨識文件內容，第一節說明我們所使用的 Google N-gram Dataset 語料庫目前的相關應用，第二節介紹文件前處理所使用的一些相關技術，第三節介紹目前文件分類常使用分類器的作法。

2.1 Google N-gram Dataset

Google 在 2006 年 9 月所公布出一份巨大的 Google 5-gram 語料庫，該語料庫包含了多達 1,024,908,267,229 個詞，語料中整理出 1,176,470,633 個出現過 40 次以上的 5-gram。不論是搜尋引擎，語音識別、拼音輸入還是拼字檢查，想要提高準確率，「分詞」都是最重要的步驟，所謂「分詞」就是讓程式能夠辨識句子中最小語意單位，能夠確地哪些字或詞是不可分開的最小句子單元，類似學語言實的畫分句子成份。在分詞的基礎上，結合各分詞出現的頻率，才能使程式更加準確明白使用者輸入或者意圖。

N-gram 常見的單詞數大小 1：unigram、2：bigram、3：trigram、4：four-gram、5：five-gram 等以此類推，N-gram 是按照固定的單詞數目來進行分詞，單詞數(n)越大，準確性越好，但運算代價也越大。

2.2 文件前處理

要進行文件分析，一般來說不會載入所有文件內全部的詞進行分析，因為需要分析的文件可能有數千甚至上萬筆，透過電腦程式去運算、分析所有的文件有可能需要花上數十天才能計算完畢，因此大多只會選取符合某些特定特徵的關鍵詞進行分析，目的除了能減少運算所需的時間，也為了提升準確率，一般文件分析常做的的前處理包括刪除停用字(Stopwords)、還原字根(Stemming)以及詞頻統計。

所謂的停用字指的是出現頻率過高、無法代表任何特徵的詞，常出現的如冠詞、代名詞和介詞等等，這部份的詞雖然出現頻率高，但是在許多類形的文件都經常出現，因此即使進行特徵分析，但幫助並不大，所以刪除停用字一般來說可以提升計算效率。英文過去有研究針對停用詞整理出停用字清單(Stopwords List)，因此只需要載入停用字清單即可刪除停用字，部分研究常使用的文件集亦有提供各自的停用字清單。

還原字根主要是針對英文字的形態變化做處理，特別是動詞，例如“have”、“has”和“had”，這三個字的意思相同，但是會根據使用主詞或時態做變化，由於形態的不同而在統計時會被分開，沒有還原字根統計詞頻則容易使的特徵過於分散， 因此透過還原字根將三個字都還原為“have”或“has”，有助於強化特徵，提高分類的準確度。而最常被使用方法是 1980 由 Martin Porter 所提出的 Porter Stemming algorithm 伍、[3]。

目前分析文件應用如文件分類或比較文件相似度的特徵選取大多是透過詞頻統計分析，所謂的詞頻(Term frequency)指的是統計每個詞出現在文件中的次數，由於電腦不能像人類一樣去理解文件的內容，因此文件中出現了什麼詞，出現了幾次便成為分析文件最重要的特徵。

而除了使用一般的詞頻統計分析外，還有另一種經常被使用的加權式詞頻統計稱為詞頻反轉文件頻系數(Term Frequency – Inverse Document Frequency, TFIDF)，其用意是計算某一關鍵詞整體的重要性，若某一詞經常在大部份文件中出現，則對所有文件的辨示度較低，TFIDF 則會調降其評等，而若某一詞只在部份文件出現，則該詞可能為較重要的特徵，而 TFIDF 會提升其評等。

2.3 分類器

而件分類依監督式與非監督式的方法，分別常用支持向量機與 k-means，監督式的優點是可以可以分類文件到有意義的標籤，缺點是事先必須進行資料訓練以建立模型，相對的，非監督式不必是先訓練即可將文件分配到特定標籤下的類別，然而缺點是標籤多半為詞，且不一定能產生有足夠意義的標籤。

機器學習分類器是透過輸入大量的類別特徵資料來進行訓練，產生出預測的模型，常見的有向量支持機(SVM)、KNN 和 naïve Bayes 等等，Yang 和 Liu 的研究[4]測試了

包含 SVM 在內的五種分類器，發現 SVM 和 KNN 在文件分類中的表現最佳， 因此 SVM 和 KNN 為目前許多研究所採用的方法。

k-means 則是另一種常用非監督式的文件分群演算法，Steinbach, Karypis, and Kumar 的研究[5]也比較數種分群方法，k-means 是其中之一準確度最高的方法，假設有 n 個文件，透過比較文件特徵的相似度，不斷的將相似的文件歸納到同一類別，將 n 個文件歸納收斂，直到所有文件群聚到使用者定義 k 個類別便停止。

支持向量機(SVM)是利用統計以及回歸理論所建立的機器學習程式，SVM 將特徵向量建構在同一個超維空間平面上〔圖 1〕，透過 SVM 分割每個類別所屬的特徵，讓每一個類別有最大的邊界，目前在許多分類應用上，如文件和影像，SVM 大多有很好的效果，但是缺點是訓練的時間很長。

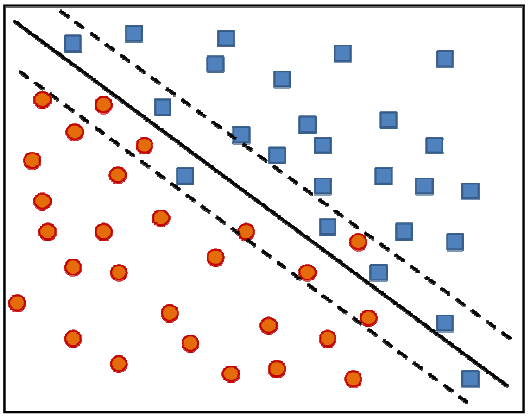

圖 1 SVM 的超維空間平面分割示意圖

參、研究方法

本章分為四節，第一節將介紹文件分類架構，可以分為訓練(Training)和測試(Testing)兩個部分，第二節為本研究的系統環境說明，第三節將介紹本研究提出的 NSA 演算流程，包含刪除停用詞、還原字根與 N-gram 選詞處理，第四節說明我們轉換特徵值的過程與使用 PTFICF 的方法。

3.1 系統架構

本研究系統架構分為訓練(Training)和測試(Testing)兩部分，圖 2 為我們的系統架構。訓練部分分為三個步驟：第一步，我們篩選 Classic4 裡的部分資料集來當作訓練文件樣本；第二步，透過 NSA 處理訓練文件樣本，得出所需的 N-gram 特徵值；第三步，將所有選出的 N-gram 特徵值，以統計和加權的方法轉換成 SVM 能接受的格式以進行訓練，最後透過 SVM 訓練產生分類模型。

測試部分分為四個步驟：第一步，我們隨機抽取 Classic4 資料來當作測試文件樣本；第二步，透過 NSA 處理測試文件樣本，得出所需的 N-gram 特徵值；第三步，將所有選

出的 N-gram 特徵值，以統計和加權的方法轉換成 SVM 能接受的格式；第四步，使用已訓練好的分類模型來做預測分類。

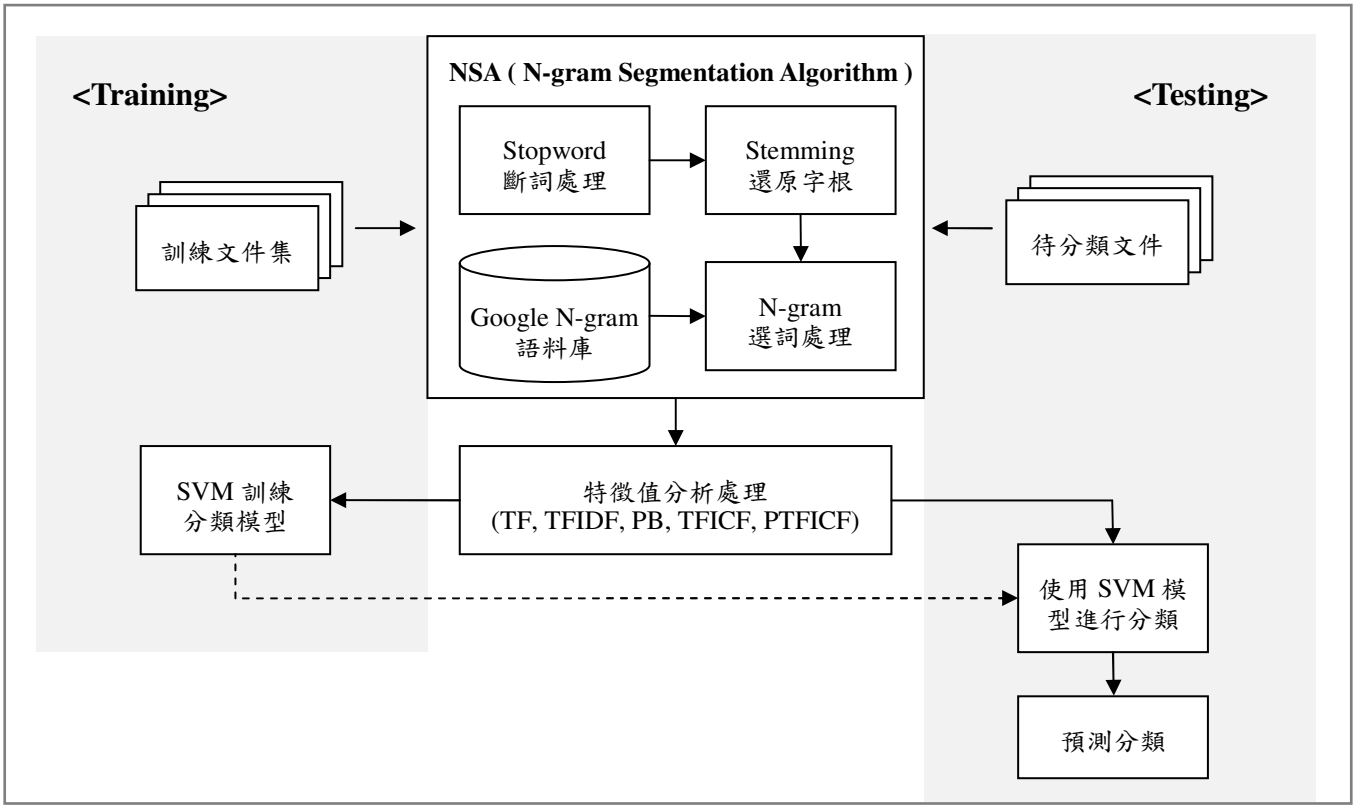

圖 2 文件分類系統架構圖

3.2 系統環境

本研究採用 Google 於 2006 年所提供的 Google N-gram Dataset[1]來作為本研究裡 N-gram 的比對資料庫，由於原始資料過於龐大（約 70GB），因此我們必需寫程式去過濾 Google N-gram 資料庫中不必要的字詞。過濾處理分為兩個部分：第一部分為去除詞彙中含有非英文字元，只留下純英文的詞彙；第二部份為過濾詞彙中含有 Stopwords 的單字，因為 NSA 方法裡也會過濾文章內的 Stopwords[2]單字，因此相對的，在 Google N-gram 資料庫中的詞彙，也可以去除含有 Stopwords 的單字以減少資料庫大小，整個處理完後的 Google N-gram 資料庫約 10GB。

本研究使用 libsvm-3.1 作為訓練和測試的分類器，SVM 是利用統計以及回歸理論所建立的機器學習程式，目前在許多分類應用上，例如文件和影像，SVM 大多有很好的效果，但是缺點是訓練的時間很長。圖 3 為 SVM 訓練的格式，特徵值越多，則訓練的時間越長，本研究在建立預測模型訓練的時間不一，模型訓練最快能在 3 小時內訓練完，最慢則需要 2~3 天。

類別索引　特徵索引 1：特徵值 1　特徵索引 2：特徵值 2…

圖 3.3.1 訓練格式

3.3 N-gram Segmentation Algorithm

NSA 斷詞演算處理使用 C#程式開發，NSA 共分成三個步驟：斷詞處理(Stopwords)、還原字根(Stemming)和 N-gram 選詞處理。

3.3.1 Stopwords 斷詞處理

本研究使用 Chris Buckley 和 Gerard Salton[2]所提供的 Stopwords List 清單共 571 個單字，圖 4 為 Stopwords 斷詞處理示意圖，該功能會根據您建置的 Stopwords List，將您所輸入的英文句子做分段處理。

3.3.2 Stemming 還原字根

我們使用 Leif Azzopardi 所提供的 Porter Stemming 演算法(C#.Net 版本)[6]，透過 3.3.1 節上的 Stopwords 斷詞處理後，我們擷取 1-gram 的字詞來做還原字根的動作，例如“computers”→“computer”、“questions”→“quest”等，透過這個步驟減少 1-gram 不同單字的相異數，增加重複字詞出現的次數，改善特徵值加權的效果，圖 5 為還原字處理示意圖。

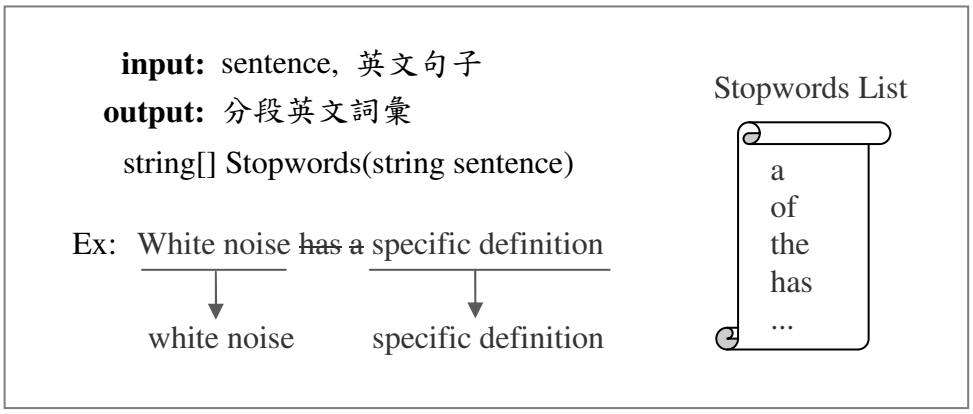

圖 4 Stopwords 斷詞處理示意圖

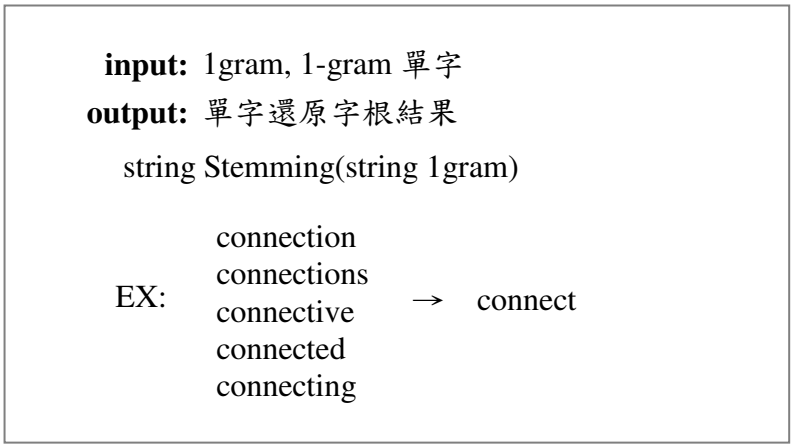

圖 5 Stemming 還原字根處理示意圖

3.3.3 N-gram 選詞處理

由 3.2 節系統環境中所提到的 Google N-gram 資料庫，經過處理完後仍約有 10GB，為了在這龐大的資料庫中搜尋到正確的 N-gram 字詞，我們建立一個 N-gram 索引檔，每 10,000,000 字詞為一個檔案。當程式在搜尋 N-gram 字詞前，會先查看 N-gram 索引檔，找到該 N-gram 詞彙落在哪個檔案後，才進入該檔案繼續搜尋 N-gram。

英文句子經過斷詞處理和還原字根後，我們會得到許多每個英文詞彙大小不一的片段，最小片段單位為 1，最大可達到 9 以上。圖 6 為 N-gram 選詞處理的程式邏輯架構，當詞彙片段大於 2 時，我們則需考慮詞與詞之間的組成是否有具意義，例如"topic deserves greater emphasis"經過 N-gram 選詞處理後，拆解成"topic"與" deserves greater emphasis "，因此當英文辭彙片段單位越大時，詞與詞之間的組成模式也越多樣，而我們則需要 Goolge N-grma 資料庫來判斷所擷取的 N-gram 辭彙是否有具意義。

```
input: phrase, 輸入的 n-gram 詞彙
       strar_index, 搜尋起始索引
       ngram, 欲拆解 ngram 長度

#Define SerachGoogleNgram( ), 比對輸入的 N-gram 與資料庫的 N-gram 是否相符
#Define NgramRecord( ), 紀錄 N-gram 詞彙

    void FetchNgram(string phrase, int start_index, int ngram){
        if (start_index+ngram <= phase.Length){
            if (SerachGoogleNgram(phrase)){        //找到該 N-gram 詞彙
                //記錄此 N-gram
                NgramRecord (phrase)
                //左拆解，右拆解
                FetchNgram(Left_phrase, start_index, ngram)
                FetchNgram(Right_phrase, start_index, ngram)
            }else{        //沒找到該 N-gram 詞彙
                //起始位置往前挪一位
                FetchNgram(phrase, start_index+1, ngram)
            }
        }else{
            //ngram 長度衰減 1，最小為 1
            if(ngram!=1) FetchNgram(phrase, 0, ngram-1)
        }
    }
```

圖 6 N-gram 選詞處理

3.4 特徵值轉換

經過 3.3 節的 NSA 處理後，可以得到每個文件所代表的 N-gram 特徵值， 而我們把所有文件中出現的 N-gram 建立成一份索引清單，並統計出現次數。詞彙權重計算我們使用過傳統的 TF、TFIDF、TFICF、PB 等詞彙加權方法，但分類成效都不太理想。最後本研究選用由林延璉[13]所提出的 PTFICF 的詞彙加權方法，不僅改善 SVM 訓練的時間，也提升預測分類的準確度。

許多研究進行文件內分析特徵值的基本元素為詞頻(Term Frequency, TF)，計算方法為公式(1)。TF 為文件內所有詞出現的頻率，然而許多研究透過加權的方式來強化文件特徵，而目前最被廣泛使用的詞頻特徵值為詞頻反轉文件頻(Term Frequency – Inverse Document Frequency, TFIDF)，計算方法為公式(2)、(3)。

$$tf_{ij} = \frac{t_{ij}}{\sum_{i=1}^{n} t_{ij}} \tag{1}$$

$$idf_i = \log\left(\frac{|D|}{\{d_j | t_i \in d_j, d_j \in D\}}\right) \tag{2}$$

$$tfidf_{ij} = tf_{ij} \times idf_i \tag{3}$$

t_{ij} 為詞 t_i 出現在文件 d_j 中的次數，n 為出現在文件 d 中的所有字詞數。

由 Debole et al. (2003) [7]研究所提出的機率式(Probability based, PB)特徵值加權法，其效果優於傳統 TF、TF-IDF 等方法，計算方法為公式(4)，表 1 為公式中所使用到詞彙機率計算的基本元素。

$$prob-based = tf_{ij} \times \log(1 + (a/b)(a/c)) \tag{4}$$

表 1 詞彙機率的基本元素

	C_k	C_k^-
t_i	a	b
t_i^-	c	d

C：文件集中的類別(Category)代號；k：類別在文件集中所屬的索引值；a：在類別 C_k 中，t_i 出現至少一次的文件數；b：非類別 C_k 中，t_i 出現至少一次的文件數；c：在類別 C_k 中，t_i 未出現的文件數；d：非類別 C_k 中，t_i 未出現的文件數。

由於監督式分類方法必須預先知道文件所屬的類別範圍，因此類別資訊必須為已知資訊，因此 Guan et al. (2009) [8]的研究提出詞頻反轉類別頻率(Term Frequency - Inverse Category Frequency)之概念，計算方法為公式(5)、(6)。

$$icf_i = \log\left(\frac{|C|}{\{C_k | t_i \in C_k, C_k \in D\}}\right) \tag{5}$$

$$tficf_{ij} = tf_{ij} \times icf_i \tag{6}$$

$|C|$：文件集中所有類別總數；$\{C_k | t_i \in C_k, C_k \in D\}$：文件集中詞 t_i 出現至少一次的類別總數

由林延璉所提出的機率式詞頻(Probability based Term Frequency, PTF)方法，將原本特徵值的詞頻進行延伸，每個 N-gram 的特徵值會依據所屬類別中出現的機率而自動調整，如公式(7)所示。

除了使用機率式詞頻外，同時還結合反類別頻率(ICF)，若該 N-gram 在較多類別出現，則代表對類別的辨識度較低；但若 N-gram 只出現在某一類別中，則代表對類別的辨識度較高，如公式(8)所示。

$$\begin{aligned} &\text{IF} \quad a >= b \quad \text{THEN} \quad ptf = tf * (1 + (a/b)) \\ &\text{IF} \quad a < b \quad \text{THEN} \quad ptf = tf * (1 + (a/b)) \end{aligned} \tag{7}$$

$$ptficf = ptf * icf \tag{8}$$

肆、實驗結果

我們使用 Classic4 資料集來進行測試分類的準確度，Classic4 有 4 個類別，分別為 CACM 有 3204 篇；CISI 有 1460 篇；CRAN 有 1398 篇和 MED 有 1033 篇，文件內容包含標題(Title)和內文(Text)兩種資訊。由於原始資料裡有些資料內文為 Null、None 或空值，我們過濾這些不符合條件的檔案後，Classic4 四類別分別為 CACM 有 1587 篇；CISI 有 1460 篇；CRAN 有 1398 篇和 MED 有 1033 篇，我們將每一分文件依照類別並擷取內文部分來做測試。我們每個類別取出 500 篇文章總計 2000 篇，其中每類 450 篇共計 1800 篇作為訓練樣本，訓練結束後便可以得到分類模型。剩餘每類 50 篇共計 200 篇作為測試樣本，配合輸入已訓練好的模型進行分類，就分類的結果比對原始分類，如果一致即為正確，反之則會分類錯誤。

表 2 Classic4 的查準率、查全率與 F_1 Value

類別	Precision	Recall	F_1 value
CACM	0.941	0.96	0.95
CISI	1.0	0.94	0.969
CRAN	0.979	0.96	0.969
MED	0.943	1.0	0.97
Average	0.965	0.965	0.964

表 3 提出方法與傳統方法結果之比較表

	F^2IHC[10]	FIHC[11]	Bi. k-means[12]	PTFICF[13]	**Proposed**

F_1 value	0.54	0.51	0.44	0.926	**0.964**

本實驗輸入 Classic4 200 篇，N-gram 擷取維度為 1~3-gram，分類正確總計 193 篇，平均正確率為 96.5%，表 2 為各類別的詳細正確率，分別使用查準率(Precision)、查全率(Recall)以及 F_1 Value 的詳細分數，三者最高皆為 1，越接近 1 表示越佳，F_1 Value 公式如 (9) 式，是使用 Precision 和 Recall 來計算總體效能表現。

$$F_1 = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recal}} \tag{9}$$

我們使用 Classic4 的資料集，平均準確度達到 96.5%，而把各類別的正確率分開來看，正確率落在 94%-100%之間，正確率已經相當高。我們在使用 PTFICF 之前，有使用 TF、TFIDF、PB、TFICF 等方法，他們各自的訓練資料在 SVM 訓練模型之下，TF 與 TFIDF 兩者的訓練準確率幾乎差不多，介於 84%~87%之間，與 Leopold et al. (2002)[9] 的研究說法一致；PB 的訓練準確度高達 90~95%；TFICF 的訓練準確度最低約 48%~59%。而在測試資料在 SVM 模型的預測分類上，均顯示分類在同一類，如原有 0、1、2、3 類，但預測結果都預測為 3 類，整體正確率為 25%~40%左右。我們研究發現，由於 NSA 所擷取的文件特徵值過多，一般傳統特徵值加權法無法有效將這些 N-gram 特徵值上形成一個完整的常態分配數值，故導致 SVM 在預測分類上出錯。反觀使用林延璉所提出的 PTFICF 特徵值加權法，不僅在訓練資料上達到 90~94%左右的效果〔圖 7〕，在測試資料上也達到 90~96%左右的預測分類效果〔表 2〕。

另外在 SVM 訓練時間上的改善，原本訓練一個分類模型需要耗時 2~3 天(約 40~50 小時)，而本研究的方法只需要 2~3 小時就能完成訓練及分類，改善約 20 倍左右的時間。我們針對大幅改善時間這個問題去作探討，發現 PTFICF 格式裡的特徵值，有部分的值為 0，因此減少 SVM 整體訓練的維度，進而讓訓練時間減少。PTFICF 的方法加強了每個文件中 N-gram 特徵值的權重，也刪去了文件中較不具意義的 N-gram 特徵值，這個方法對於文件裡有許多 N-gram 特徵值來說，是非常重要的改善。

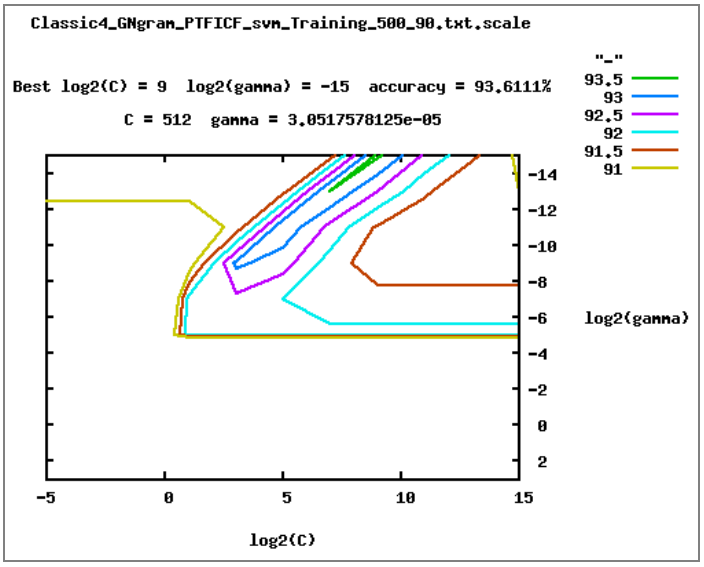

圖 7 PTFICF 訓練資料在 SVM 訓練模型上之結果

伍、結論與未來研究

隨著網路越來越發達，電子的文章以及文件呈現爆炸式的增長，而現今許多文章是沒有分類或是必需人工選擇分類或標籤，過去以有部份文件分類相關的研究，然而近年來隨著電子文章的增加而逐漸受到重視。

我們提出 NSA 斷詞架構，搭配 Google N-gram Dataset 以驗證 NSA 所擷取出的 N-gram 正確性，在 N-gram 維度擷取部分我們採用 3-gram 為最大維度，而非 5-gram，我們實驗出 1~3-gram 在文件內的辨識程度優於 1~5-gram，其中有一些原因是擷取出 1~3-gram 的相異性低於 1~5-gram，提升重覆詞在每個類別出現的次數，使某些 N-gram 的加權效果提升。另外同樣的概念也套用在還原字根的步驟，我們將 1-gram 還原成最基本的單字單元，降低 1-gram 每個詞的相異程度，提升 1-gram 重覆詞在每個類別出現的次數，進而整體提升 1-gram 的加權效果。

本研究實作以 N-gram 方式為架構的文件分類系統，使用 Classic4 資料集進行測試，而在傳統特徵值 TF、TFIDF、PB、TFICF 的加權測試下，整體正確率為 25~40%左右，其效果不彰的原因主要是 N-gram 特徵值過多，傳統加權法無法將這些特徵值轉換成常態分布。而在 PTFICF 的特徵值加權下，整體正確率達到 96.5%，各類別正確率在 94%~100%。

本研究使用 NSA 處理訓練文件集需耗時 1~2 天左右的時間，N-gram 斷詞處理時間依照文件大小而有所增減，NSA 處理系統是在單機作業系統上運作，未來可以將它改成雲端處理(MapReduce)模式，將大量欲訓練的文件集，分成好幾個部分送至雲端伺服器處理，以改善單機處理 NSA 過久的問題。

參考文獻

[1] Google Ngram Dataset, http://books.google.com/ngrams/datasets

[2] Stopwords List generated by Chris Buckley and Gerard Salton, http://www.lextek.com/manuals/onix/stopwords2.htm

[3] M. F. Porter, “An Algorithm for Suffix Stripping,” Program 14, pp. 130-137 1980.

[4] Yiming Yang and Xin Liu, “A Re-Examinati on of Text Categorization Methods,” Proceedings of the 22nd Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval, pp. 42-49 1999.

[5] Steinbach, Karypis, and Kumar, “A comparis on of document clustering techniques,” Work shop on Text Mining, pp. 109-111 2000.

[6] Leif Azzopardi code Porter Stemming by C#.Net, Nov 2002, http://tartarus.org/~martin/PorterStemmer/csharp2.txt

[7] Debole, F., & Sebastiani, F., 2003, “Supervised term weighting for automated text categorization”, Proceedings of the 2003 ACM symposium on applied computing, pp. 784-788.

[8] Guan, H., & Zhou, J., & Guo, M., 2009, “A class-feature-centroid classifier for text categorization”, Proceedings of the 18th international conference on World wide web, pp. 201-210.

[9] Leopold, E., & Kindermann, J., 2002, “Text categorization with support, vector machines – How to represent texts in input space”, Machine Learning, vol. 46, pp. 423–444.

[10] Chen, C L., & Tseng, S.C., & Liang, T., 2010, “Mining fuzzy frequent itemsets for hierarchical document clustering” Information Processing and Management, vol.46, pp. 193-211.

[11] Fung, B. C. M., & Wang, K., & Ester, M., 2003, “Hierarchical document clustering using frequent itemsets”,Proceedings of the 3th SIAM, pp. 59-70.

[12] Steinbach, M., & Karypis, G., & Kumar, V, 2000, “A comparison of document clustering techniques”, Workshop on Text Mining, pp. 109-111.

[13] 林延璉，2011 年，“使用句構分析模型與向量支持機的自動文件分類架構”，國立高雄第一科技大學，碩士論文。

A New Auto Document Category System by Using Google N-gram and Probability based Term Frequency and Inverse Category Frequency

Jun-Liang Lin
Department of Information Management
National Kaohsiung First University of Science and Technology
u9924812@nkfust.edu.tw

Wen-Chen Huang
Department of Information Management
National Kaohsiung First University of Science and Technology
wenh@nkfust.edu.tw

Abstract

Auto Document Category puts relative documents in the same or closer position. It helps users to find the desired data more efficiently. Our research proposes to use N-gram to implement an Auto Document Category System. We use Stopwords, Stemming, and Google N-gram Dataset to create N-gram Segmentation Algorithm (NSA). We not only use NSA to fetch keywords in the documents of Classic4 dataset, but also use Probability based Term Frequency and Inverse Category Frequency (PTFICF) to test the performance of term weight. Finally, these terms are trained by Support Vector Machine (SVM) and are used to predict classification results. The average accuracy for auto categorizing is 96.5%, and the F1 value is 96.4%, which is higher than other related methods.

Keywords: Google N-gram, SVM, Text Classification