文章分類演算法的比較研究—以中文新聞為例

Text Categorization for Chinese News: A Comparative Study

李明安(Ming-An Li), 蔡卓忻(Cho-Hsin Tsai)
臺灣大學資訊管理學系, 臺灣大學電機工程學系
b01705039@ntu.edu.tw, b01901101@ntu.edu.tw

摘要

文章分類在資訊科學中是一個歷史悠久的研究領域,目標為將任何長度的文章,根據其文意放入適當的類別中。過去學者曾提出一些演算法嘗試解決這個問題,如貝氏分類器(Naïve Bayes classifier) 和支持向量機(Support Vector Machine),已能達到一定程度的準確率。近年來興起的機器學習演算法,為全世界各領域的技術發展帶來重大改變。近五年來,在語音辨識領域獲得重大成功的分類模型—深層神經網路(Deep Neural Network),可以透過訓練找出資料的潛在規則,有效改善語音辨識的準確率。而在影像辨識、語意分析等其他領域中,許多學者逐漸開始應用此方法去突破研究上的瓶頸。本論文利用神經網路與其他演算法,在不同的條件下對中文新聞文章做分類,因而得出彼此的差異以及優劣勢。實驗結果顯示 Deep Neural Network 的表現最為突出,在足夠訓練資料量下擊敗所有其他的分類器,F1 成功達到 78%,並且以快速運算的優勢勝過其他演算法。本論文成果顯示 DNN 有望應用於中文資訊處理的研究。

關鍵字:資訊處理, 文章分類, 支持向量機, 最近鄰居法, 深層神經網路

Abstract

Text categorization (or classification) is aimed at placing text documents, regardless of their lengths, in their appropriate category according to their content or metadata. An extensive study has existed for a long time, such as Naïve Bayes Classifiers and the Supporting Vector Machines (SVMs). Their performances are acceptable. Recently, many machine learning algorithms have shown significant improvements on several tasks. For the past five years, a classifier model—the Deep Neural Network (DNN), has been very successful in the field of speech recognition. A trained DNN model has a potential of finding the specific pattern from the database and improves recognition accuracy. In some other fields, such as image recognition and semantic analysis, many researchers have started to apply DNN models to make a breakthrough. In this paper, we delve into the subject of Chinese text categorization. A detailed comparison of the performance between the DNN and other 8 algorithms under different conditions is presented. The experimental results illustrate the high micro-averaged F1 score provided by the DNN, especially when a sufficiently large training dataset is used. It defeats all the other classifiers and achieves an F1 score of 78%. Furthermore, it outperforms others due to its high computational speed. Our work shows a potential for deep learning techniques on Chinese text analysis.

Keywords: Information Processing, Text Categorization, SVM, kNN, DNN

1. 背景介紹

文章分類為人類獲取新知的過程中提供了許多貢獻,除了幫助人類篩選出所個人需要的文章,也大幅節省人類自行分類所花費的時間,而透過機器學習去解決這個問題,讓知識學習組織化、快速化,正是一項長久以來學者一直在探討的問題。自然語言處理(NLP) 正是在解決此一問題的相關領域,基於現在語意摘要技術還不成熟,要透過機器學習來摘要文章是件難事,但語意分類的任務卻是相對容易,是現今普遍應用的技術。現今常用的文章分類器有 SVM(Support Vector Machine) 、 K-Nearest Neighbors (kNN)。特別是 SVM,最早是在 1995 年提出,在這二十年間成為成熟並普遍使用的分類模型,至今仍不斷有新型的 SVM 產生,因此本篇研究將 SVM 納為比較的標準之一。而影響 SVM 的表現最主要在於 SVM 的 kernel function,因此本文試驗了許多不同的 kernel 做更詳細的比較。而作為傳統分類方法的 kNN 雖然簡單易懂,由於其運算量大、花費時間長而比較不被接受,然而我們實測得花費時間卻並沒有想像中來的多。DNN(Deep Neural Network)雖然是在近年崛起的模型,但其實是在很早期就被發明的方法,只是在近年來在架構上有所改善,性能已經有大幅的提升。因此本研究就挑選上述的模型,作為後續比為比較的基礎,並且從各方面測試去衡量這些模型的表現差異。本論文成果顯示深度學習(Deep Learning)有望應用於資訊處理的研究

2. 資料處理

1.1 資料集

資料集合為從 UDN 新聞網所蒐集的 339,060 篇網路新聞文章 (自 2015/02/10 至 2015/12/02),依照 UDN 新聞網原本的分類作為答案,每一篇文章都屬於一個類別,其中的「即時」類為包含許多主題的類別,因此將 「即時」 類底下的子類別,併入其他類別,合併後總共 10 個類別,包含運動、娛樂、生活、社會、地方、要聞、兩岸、全球、產經以及股市,故 表 1 中的總數為合併後各類別的文章總數,而總詞數少於 6 個詞的文章則已經被排除。

1.2 資料前處理

文章經過中文斷詞系統 Jeiba 處理。為避免 Jeiba 的中文詞彙不足,加入教育國字典簡編版資料庫[1] 其中的 38,270 個字詞,以改善斷詞準確率,將斷詞後的結果利用 Google 的 word2phrase 將可能可以合併的斷詞連接,找出可能被誤斷的詞彙,用人工方式篩選的方式,將出現頻率高但被錯誤斷詞的詞彙加入 Jeiba 的字典,然後重複上述動作再次斷詞。如此循環地斷詞並利加入字典三輪後,共加入 1,966 個詞彙,流程參見圖 1。經過斷詞處理後的文件,將阿拉伯數字以 "#" 取代,並去除非文字符號。斷詞處理後的文章,使用 Google 的 word2vec 訓練,以 word embedding 方法抽取詞的語意,將所有文章中出現的詞,每個詞轉換為一個 300 維的向量,稱為詞向量 (word vector),接著我們以詞向量字典(一個詞對應一個向量)查找文章中的段落中出現的詞,作詞向量平均 (word vector averaging)[1],得到段落向量 (paragraph vector),再將一個文章中出現的段落向量平均,得到文章向量 (document vector),因此每篇文章都會轉換為一個 300 維的向量,而詞向量平均的作法相當於使用了 bag of

[1] 教育國字典簡編版資料庫: http://140.111.34.182/dict_concised_download.html

words 的假設。使用 word2vec 將文章語意轉換為低維度的向量，同時可以有效解決資料稀疏 (data sparsity) 以及維數災難 (curse of dimensionality) 的問題，讓模型有更好的表現與更快速的運算。

在資料集的部分，我們從總數 33 萬筆的文章中，10 個類別各取 10,000 筆文章，共有 10 萬筆文章，作為訓練資料；其餘文章作為測試資料，共約 24 萬篇。而每個類別取一萬篇也是確保某些數量較少的類別，例如「股市」，有足夠多的資料作為測試資料，避免不同類別間測試資料比例差異過大。

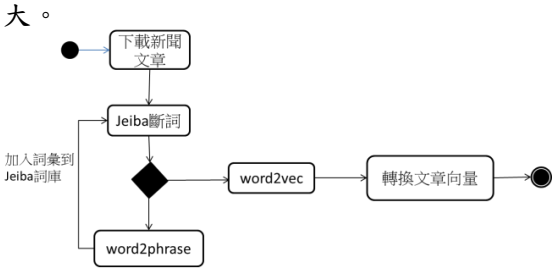


圖 1: 資料前處理流程圖

Feed forwarding

Input：x (document vector)

$\sigma(W_1 x + b_1) = z_1$

$\sigma(W_2 z_1 + b_2) = z_2$

$\cdots$

$\sigma(W_6 z_5 + b_6) = y$

Output：y (distribution of class probability)

Back propagation

Cost：$-\sum_{i=1}^{N}(\hat{y}_i \times \log(y_i))$, categorical cross entropy

Objective：min(Cost)

Update：$W = W - \eta \frac{\partial Cost}{\partial W}$, $b = b - \eta \frac{\partial Cost}{\partial b}$, for $\forall$ W and b

圖 2: DNN 演算法

表 1: 新聞文章總數

	類別	文章數		即時類的子類別	文章數		總數
	即時	193,259					
1	運動	11,207	+	運動	9,717	=	20,926
2	娛樂	10,762	+	娛樂	10,389	=	21,151
3	生活	14,298	+	生活	26,034	=	40,332
4	社會	9,849	+	社會	25,231	=	35,080
5	地方	18,466	+	地方	21,156	=	39,622
6	要聞	17,371	+	要聞	24,808	=	42,179
7	兩岸	12,539	+	兩岸	11,433	=	23,979
8	全球	15,802	+	國際	28,947	=	44,749
9	產經	15,361	+	財經	35,542	=	50,903
10	股市	20,146	+			=	20,146
						總計	339,060

3. 分類器模型

3.1 Deep Neural Network (DNN)

深層神經網路(Deep Neural Network，DNN)是深度學習 (Deep Learning) 的一種架構，使用隨機梯度下降法(Stochastic gradient decent)不斷迭代的輸入訓練資料並更新模型中的參數，讓模型的預測愈來愈準確，最後收斂在 local optimum。每個迭代 (iteration) 的運算分為兩個部分，參見 圖 2。第一個部分是 feed forwarding，輸入一個文章向量到 input layer W_1 後，經過模型中間多層的 hidden layers W_2 to W_5 運算，最後由 output layer W_6 輸出預測的類別機率分布 y。第二個部分是 backpropagation，將預測的機率分布 y 與正確的機率分布 $\hat{y}$ 作 categorical cross entropy，得到 cost(或稱作 loss)，以 cost 對模型中所有參數，即為每一層的 weights 與 biases，計算 gradients 值，並更新模型中所有參數，完成了一個迭代。

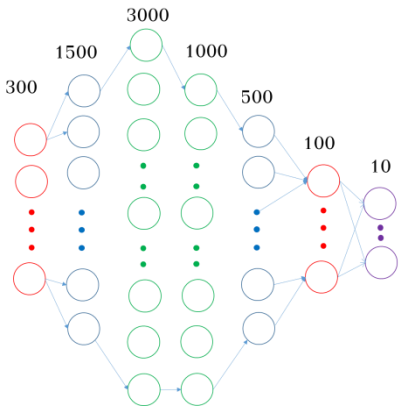


圖 3: DNN 架構

本次試驗用的 DNN 模型架構，包含 5 層 hidden layer，第一層為 input layer，最後一層為 output layer，每一層神經元數如圖 3 所示。DNN 的架構以 Keras 套件[2]撰寫，使用的 Activation function 為 PReLU 及 softmax，Cost function 為 Categorical cross entropy，Optimizer 為 Adagrad 與 Dropout，並且對訓練資料使用

minibatch，以上所使用的參數都可能使 DNN 加速收斂並達到更好的表現。

3.2 K-Nearest Neighbors (kNN)

最近鄰居法(k-Nearest Neighbors，kNN)尋找在特徵空間中 k 個最近的訓練樣本，讓每個樣本對文章向量投票，每個樣本的權重都相同，票數最高的類別即為該文章的類別。本次測試中，kNN 模型所使用的為文章相似度分別為 Euclidean Distance 與 Cosine Similarity 兩種，使用前者的在之後的試驗中簡稱為 kNN-dist，後者簡稱為 kNN-cos。Euclidean Distance 愈短以及 Cosine Similarity 愈大則兩向量相似度愈高。為了取到最佳的 k 值，我們以較小的訓練量進行實驗，以 Euclidean Distance 作為相似度，參見 圖 4。在不同訓練量為 20,000 及 2,000 下取 k 為 15 附近時，可以有最好 F1 score，因此取 k = 15 作為模型的參數。

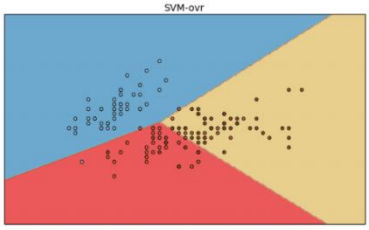


圖 5: SVM-ovr 的 decision boundary

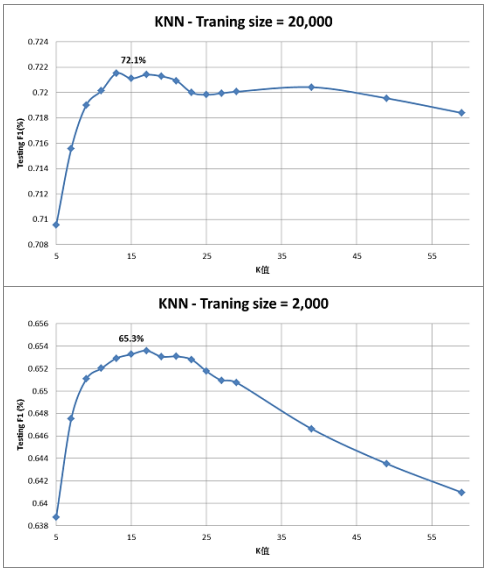


圖 4: 兩種不同的訓練資料量下， K 值與 F1 score 的關係

3.3 Support Vector Machine (SVM)

支持向量機(Support Vector Machine，SVM) 是一種線性分類器，藉由超平面(hyperplane) 分割特徵空間，而 non-linear kernel SVM 使用函式把資料點投射到更高維度的特徵空間，再對其作線性分割，由於 SVM 可以求得向量空間中唯一的 global optimum，加上運算快速，所以廣泛用於自然語言處理等領域。

3.3.1 SVM one-versus-rest (SVM-ovr)

SVM one-versus-rest，簡稱作 SVM-ovr，也有人稱作 SVM one-against-all。10 個類別中，產生 10 個 binary SVM，每個 binary SVM 會判斷某向量屬於此類，或是屬於其它剩餘的類別。比較每個分類器所產生的分數，我們定義某向量屬於分數最高的該類別。我們使用 library LIBLINEAR 撰寫 [3]，因此只支援 linear kernel，圖 5 為 SVM-ovr decision boundary 的示意圖。

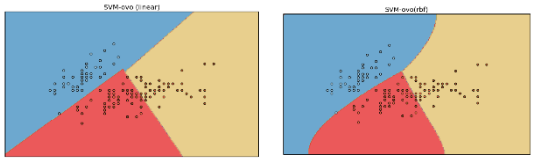
3.3.2 SVM one-versus-one (SVM-ovo)

SVM one-versus-one，簡稱作 SVM-ovo，也有人稱作 SVM one-against-one，10 個類別中，兩兩產生一個分類器，共有 45 個分類器，每個分類器會判斷向量比較類似兩類中的哪一類，最後計算得票數最高的類別，即為該向量的類別。我們使用 library LIBSVM 撰寫，支援多種 kernel，包含 linear, radial basis function(rbf), polynomial(poly), sigmoid，在之後的試驗依序簡稱為 SVM-ovo linear, SVM-ovo rbf, SVM-ovo poly, SVM-ovo sigmoid。由於 rbf, poly, sigmoid 是非線性 kernel，我們把 SVM-ovo linear 以外的統一稱作 SVM-ovo nonlinear，使用不同的 kernel，decision boundary 也會有所差異(圖 6)。SVM-ovo linear 與 圖 5 中的 SVM-ovr 的 decision boundary 非常相近，SVM-ovo rbf 呈輻射

狀，而 SVM-ovo poly 表現會因為 degree 的差異而有顯著差異，degree 愈小 decision boundary 愈接近線性，而在測試過後將 degree 設成 2 的表現比更大的 degree 值還來的好，因此將 degree 設為 2，同時 degree = 2 也是實務上常用的設定值。

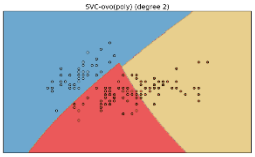
4. 各分類器實驗結果

本次試驗以 Micro-averaged F1 score 作為衡量不同模型的表現的依據。經 10 萬筆資料訓練，約 23 萬筆資料測試，在足夠多的訓練量對所有模型進行測試，因此測試的 F1 表現視為該模型最好的表現。圖 7 為測試結果，所有 Model 表現以 DNN 最為出色，但與 SVM-ovr 以及 SVM-ovo linear 差距不明顯，而 SVM-ovo nonlinear，表現卻明顯遜於 SVM-ovo linear 以及 SVM-ovr，也就說明了使用 linear kernel 是 SVM 中表現最好的 kernel。至於 kNN 有 kNN-dist 與 kNN-cos 不同相似程度的兩種模型，其中以 kNN-dist 明顯較好。F1 score 由最高到最低的排序依序是 DNN > SVM-ovo linear > SVM-ovr > kNN-dist >SVM-ovo rbf > kNN-cos，而 SVM-ovo poly 以及 SVM-ovo sigmoid 的 F1 score 過低，大部分的類別無法辨識，不足以作為可行的分類器。

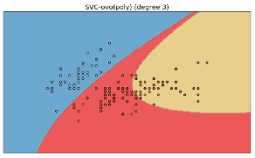


(a) SVM-ovo linear

(b) SVM-ovo rbf



(c) SVM-ovo poly(degree=2)



(d) SVM-ovo poly(degree=3)

圖 6: decision boundary of SVM-ovo

5. 小量訓練資料下實驗結果

在上一段我們已呈現 10 萬筆訓練資料所得到各個模型的表現，現在以更小訓練資料，與原本相同的測試資料作測試，試驗各個模型對於訓練資料不足時的表現如何。但由於 SVM-ovo sigmoid 表現始終落在 4%，明顯不適合用於此次試驗，故不放入比較。首先我們將原本的訓練資料 10 萬筆中，保持每個類別等量的取出 5 萬筆，然後每次都從上一個訓練資料集合中，取出約一半的訓練資料，因此有 6 組更小的訓練資料集合，分別是 50,000、20,000、10,000，5,000、2,000 以及 1,000。

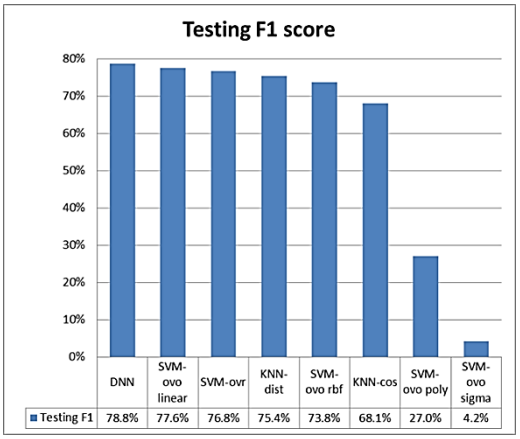


圖 7: 所有模型的Testing F1 scores (訓練資料量為100,000)

圖 8 展現了所有模型在不同訓練量下的表現。由於模型過多，我們將表現較好的模型移出至 圖 9 觀察。我們發現在訓練量較大時，所有模型的表現排序與 10 萬筆訓練資料相同，但是在更小的訓練資料集合下，DNN 輸給了 SVM-ovo linear 以及 SVM-ovr，在 2 萬筆訓練量時與 SVM-ovo linear 產生交叉，參見 圖 10，在一次的訓練紀錄中，訓練量在 2 萬筆時，DNN 的 Testing F1 已經收斂在 0.75 附近，已經接近 0.9 的仍持續上升，這樣的差距代表著訓練資料量的明顯不足，而且會隨著訓練量愈小，Training F1 與 Testing F1 差距會逐漸拉大，也因此推斷訓練量在 2 萬筆時，

對於 DNN 模型已經訓練量已經不足了，而在更小的訓練量中，其訓練資料量與測試資料量懸殊的比例，更是讓 SVM 線性分類器表現優勢。 DNN 在訓練量小時容易 overfitting，但 SVM 可以在小樣本、高維度特徵的訓練量下，有良好的表現 [4]，因此與 DNN 與 SVM-ovo linear 在摺線圖上產生交叉。特別的是 SVM-ovr 與 SVM-ovo linear 兩個同樣是 linear kernel 的 SVM 在所有的 Training size 上的 F1 都保持著一定的差距，因此可以判定兩者的 F1 差距是這兩種模型架構上的差異所造成的，也就是在 text classification 的領域，SVM-ovo 架構能夠比 SVM-ovr 有稍微更高的精確度。而 SVM-ovo linear 較 SVM-ovo nonlinear 佔優勢，乃因通常在 feature 數量夠多以及訓練量夠多時，nonlinear SVM 將資料點投射到高維度空間，並不能提升表現 [5]，因此 SVM-ovo linear kernel 能勝過 rbf kernel。至於 SVM-ovo poly 在訓練量愈小時，表現反而較好，推斷可能是在訓練大時產生的 overfitting 的情形。我們以斜率作為判斷模型在抵抗少量資料下的表現，最平緩的折線表現愈加，因此在抵抗資料量少的表現下，SVM-ovo linear = SVM-ovr > DNN >kNN-dist > SVM-ovo rbf。

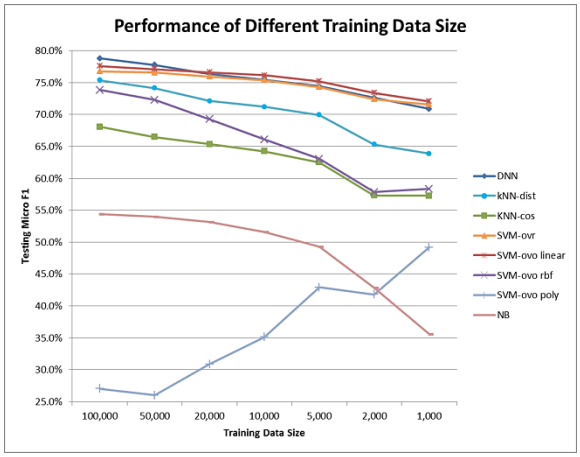


圖 8: 不同訓練資料量下所有模型的 Testing F1 scores

6. 運算時間

一個好的分類器要能應用在實務上，除了 F1 score 要高，最重要的是分類器執行任務的運算時間。運算時間又分為訓練時間與測試時間，我們以 7 組不同的訓練資料量來檢視不同模型的花費時間，而測試資料同樣是 24 萬(讀檔的時間不計入運算時間)。並且將 DNN 與 kNN 的算法透過 GPU 平行運算，大幅加速其矩陣運算的時間，測試使用的電腦設備為 CPU: Intel i5-3350P，GPU: Nvidia GTX650，使用 Python 的 library Theano 以及 Nvidia toolkit CUDA 7.5 支援平行運算的程式。由表 2 可觀察出，除了 kNN 不需訓練時間，其他的模型訓練時間隨著訓練量的增加而上升。特別可以觀察到 SVM 趨於指數增長: 資料量 2 倍成長時，訓練時間大約以 4 倍成長；資料量 2.5 倍成長時，則訓練時間約以 6 倍成長。但 DNN 與訓練時間無明確關係，乃因 DNN 是 iterative algorithm ，會因參數的不同會有不同的收斂時間，但是大致上愈少的訓練資料則愈快收斂，當收斂到 cost 沒有明顯下降時就會提前停止 (early stop)。我們同時也使用 minibatch，將多筆文章向量組成一矩陣作為一筆輸入資料，讓每一個 iteration 能有更好的更新，也使得更大量的訓練資料不會增加太多訓練時間。假設

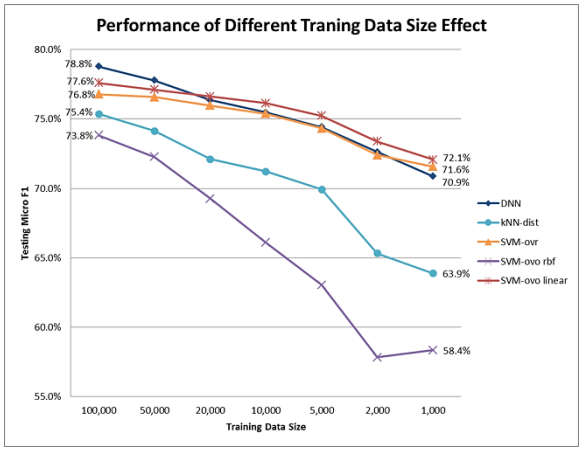


圖 9: 不同訓練資料量下部分演算法的 Testing F1 scores

訓練量為 n 時，使用 libsvm 的 SVM-ovo 時間複雜度落在 $O(n^2)$至 $O(n^3)$ 之間，使用 liblinear 的的 SVM-ovr 大約在 $O(n)$。

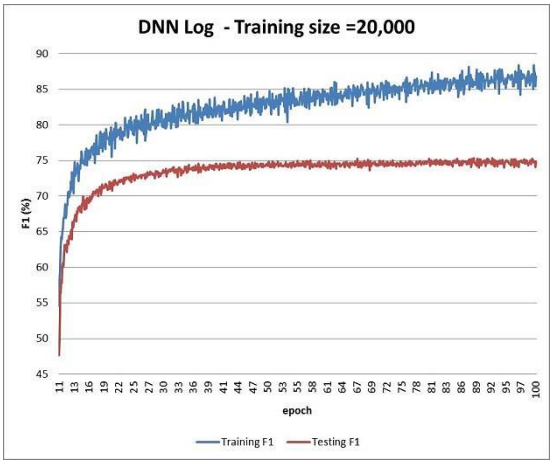


圖 10: DNN Training and Testing F1 scores (訓練資料量為20,000)

在測試時間的部分，從 表 3 可看出，在訓練資料量較大時，SVM 與 kNN 大致上都與訓練資料正比增長，而 DNN 與 SVM-ovr 則是不受訓練量影響。因為 DNN 與 SVM-ovr 都是 parameterized model，也就是 SVM-ovr 與 DNN 都已經有固定數量的 weights 與 biases 對測試資料運算，因此花費時間是固定的。從總花費時間來看，在資料量較大時，DNN 與 SVM-ovr 展現了他們的優勢，使用少許的時間獲得良好的成效，同時這也要歸因於現代硬體運算的進步，讓原本缺乏運算效率而難以實際應用的模型，如 DNN 和 kNN ，能夠讓人在可接受的時間內運算完畢，甚至 kNN 在 GPU 加速下會比非線性 kernel 的 SVM-ovo 還快。也因此若期盼選擇具有良好的 F1 score 並能在最短時間內完成訓練及測試的模型，其首選莫過於 SVM-ovr。SVM-ovr 是利用 Liblinear 實作經過特殊設計過的 SVM，專門用以處理大量的線性多類別分類任務，不但所花費的訓練時間遠小於其他的 SVM，F1 score 的表現上也與 SVM-ovo linear 相差不遠。然而 DNN 也不惶多讓，在層數達到6 層神經元架構下，其運算時間卻能少於 SVM-ovr 以外的其他模型。若分析測試時間的複雜度，假設訓練資料量為 n，測試資料量為 m，DNN 與 SVM-ovr 不受訓練資料量影響，時間複雜度都為 $O(m)$，SVM-ovo 與 kNN 為 $O(n \times m)$。在大量訓練資料量下，以總運算時間從最短至最長排序依序是 SVM-ovr < DNN < kNN-cos < SVM-ovo linear < kNN-dist < SVM-ovo nonlinear。若不使用 GPU 加速而使用 CPU 運算， DNN 的訓練時間增為 4 倍；測試時間部分，DNN 會增為 100 倍，約為 1408 秒。kNN 會則增為 2 倍。

7. 結論

綜合以上資訊，SVM-ovo linear 與 SVM-ovr 會在小型的訓練資料下勝出，兩者表現相近，若是追求快速的運算時間，SVM-ovr 明顯勝出，若是不考慮時間，而追求稍微較好的表現，則 SVM-ovo linear 是最佳的選擇。但若是大量資料以及同時追求最佳表現的條件下，DNN 是最好的選擇，在充足的訓練資料下 DNN 在所有模型中表現最好，然而參數的調整勢必會影響結果，由於缺乏一個固定的數學模型，DNN 理論上不是每次都能訓練到一個固定且最佳的結果，幸運的是，在我多次訓練 DNN 的過程中，DNN 能有穩定且一致的表現，幾乎每次都能達到相近的 F1 值，所以 DNN 在文章分類會是個可行的方法，並且由於 DNN 彈性的架構，使得 DNN 具有很大的潛力有更好的表現，去適應不同的任務，當資料量減少時，DNN 在縮減層數以及神經元數目後，可能可以避免 overfitting 的問題，因此適度的調整 DNN 模型理論上可以達到更好的表現。本論文成果顯示 DNN 在資訊處理的研究中不但是個可行的模型，同

時也是個有潛力超越其他模型的架構，而在深度學習(Deep Learning)的技術中，還有更多複雜的模型能應用在相關領域的研究，有望在將來突破資訊處理的限制，突破現今的技術瓶頸。

表 2: 不同訓練資料下各演算法所需要的訓練時間(單位:秒)

Training Time / Training size	DNN	SVM-ovr	SVM-ovo linear	SVM-ovo poly	SVM-ovo rbf	SVM-ovo sigmoid
100,000	504.97	95.94	3545.58	6715.68	2219.80	6156.36
50,000	438.78	38.49	954.42	1883.22	698.04	1541.93
20,000	450.46	14.83	175.44	283.90	145.05	247.50
10,000	276.06	5.31	50.43	60.39	42.20	60.67
5,000	263.23	2.12	14.33	14.44	13.03	14.61
2,000	254.97	0.61	3.13	2.20	2.35	2.19
1,000	463.25	0.26	1.03	0.56	0.58	0.56

表 3: 不同訓練資料下各演算法所需要的測試時間(單位:秒)

Testing Time / Training size	DNN	SVM-ovr	SVM-ovo linear	SVM-ovo poly	SVM-ovo rbf	SVM-ovo sigmoid	kNN-dist	kNN-cos
100,000	13.26	0.95	2969.43	7315.55	5820.49	7334.53	6954.23	3099.78
50,000	13.26	0.96	1505.81	4121.61	3269.73	4224.32	3517.64	1596.15
20,000	13.22	0.92	657.68	1710.59	1507.10	1694.37	1490.40	724.24
10,000	13.49	0.93	364.31	826.53	799.33	835.73	843.08	446.57
5,000	13.26	0.93	202.40	416.82	422.82	407.68	516.03	295.71
2,000	13.30	0.94	90.89	163.58	172.56	161.25	284.28	156.70
1,000	13.34	0.94	50.58	77.19	79.85	73.43	259.60	125.99

參考資料

[1] Maas, A. L. & Ng, A. Y. (2010). A probabilistic model for semantic word vectors. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*.

[2] Chollet, F. (2015). Keras. *GitHub repository*. Software available at https://github.com/fchollet/keras

[3] Chang, C. C. & Lin, C. J. (2011). LIBSVM : a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1--27:27. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm

[4] Severyn, A., Nicosia, M., Barlacchi, G., & Moschitti, A. (2015). Distributional neural networks for automatic resolution of crossword puzzles. In *Association for Computational Linguistics (ACL)*.

[5] Hsu, C. W., Chang, C. C., & Lin, C. J. (2003). A practical guide to support vector classification.

[6] Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features (pp. 137-142). Springer Berlin Heidelberg.