Sentiment classification for Chinese News: A Comparison

李明安 National Taiwan University
資管四 B01705039 Group 16
b01705039@ntu.edu.tw

ABSTRACT

文章分類在資訊科學中是一個歷史悠久的研究領域。過去學者曾提出一些演算法嘗試解決這個問題，如 Naive Bayes classifier 和 Support Vector Machine，已能達到一定程度的準確率。然而，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。其中，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。本研究的目標是利用神經網路與其他方法的模型，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。

1. INTRODUCTION

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，基於現在語意摘要現在還不成熟，要透過機器學習摘要是件難事，但是語意分析分類相對來說比較容易，是可以普遍應用的技術，而現今常用的文章分類器不外乎 SVM，kNN 與 NB，特別是 SVM(Support Vector Machine)，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，是個用途廣泛、性能優異的模型，因此本篇研究將 SVM 納為比較的模型，而影響 SVM 的表現最主要在於 SVM 的 kernal function，因此會試驗許多不同的 kernel 做更詳細的表現比較。而作為傳統分類方法的 kNN，由於其運算量大、花費時間長而比較不被接受，然而實測得花費時間卻並沒有想像中來的多，而 Naïve Bayes 更是歷史悠久的方法，DNN(Deep Neural Network) 雖然是在近幾年崛起，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。

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

2. DATA PROCESSING

2.1 Data Set

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，把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 “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 Table 1中的總數為合併後各類別的文章總數，總詞數少於 6 個詞的文章則已經被排除。

2.2 Data Preprocessing

文章經過中文斷詞系統 Jeiba 處理，為避免 Jeiba 的中文詞量不足，加入教育部國語字典簡編版資料庫 (*http* : //140.111.34.182/*dict_concised_download.html*) 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，因此可以找出被誤切的詞彙，用人工方式篩選的方式，把出現頻率高但被錯誤斷詞的詞彙加入 Jeiba 的字典，然後再次斷詞。如此循

Table 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	=	23,972
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

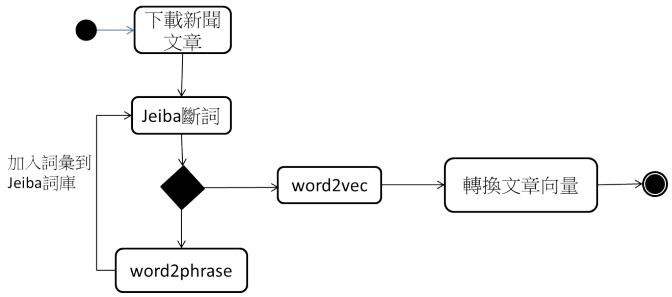


Figure 1: UML of data preprocessing

環地斷詞並利加入字典三輪後，共加入 1,966 個詞彙，參見 Figure 1的活動圖。經過斷詞處理後的文件，去除了非文字符號，並且將阿拉伯數字以 "#" 取代。斷詞處理後的文章，使用 Google 的 word2vec 訓練，用 word embedding 方法抽取詞的語意，將所有文章中出現的詞，一個詞轉換為 300 維的向量，稱為詞向量 (word vector)，接著我們以詞向量的字典 (一個詞對應一個向量)。接下來將一篇文章中的段落中出現的詞，作詞向量平均 (word vector averaging)[3]，得到段落向量 (pharagraph 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 a bag of word 的假設，在將 google 的 word vector 平均的同時，平均後的向量會落在平均所有語意關係的空間位置上。使用 word2vec，將文章語意轉換為低維度的向量，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。從 data set 總數 33 萬筆的文章中，每個類別各取 10,000 筆文章，共有 10 萬筆向量，作為訓練資料；其餘文章作為測試資料，共約 23 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。

Feed forwarding

Input：x (document vector)

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

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

...

$\sigma(W_6 z_5 + b_6) = \mathrm{y}$

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

Figure 2: DNN architecture

3. CLASSIFIER MODELS

3.1 Deep Neural Network(DNN)

Deep Neural Network，深層神經網路，簡稱 DNN，是深度學習 (Deep Learning) 的一種架構，透過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 Figure 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 back propagation，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。

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，每一層神經元數量依序是，$300 \times 1500 \times 3000 \times 1000 \times 500 \times 100 \times 10$，如 Table 3所示。DNN 的架構以 Keras 套件撰寫，使用的 Activation function 為 PReLU 及 softmax，Cost functin 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)

k-Nearest Neighbors，最近鄰居法，簡稱為 kNN，尋找在

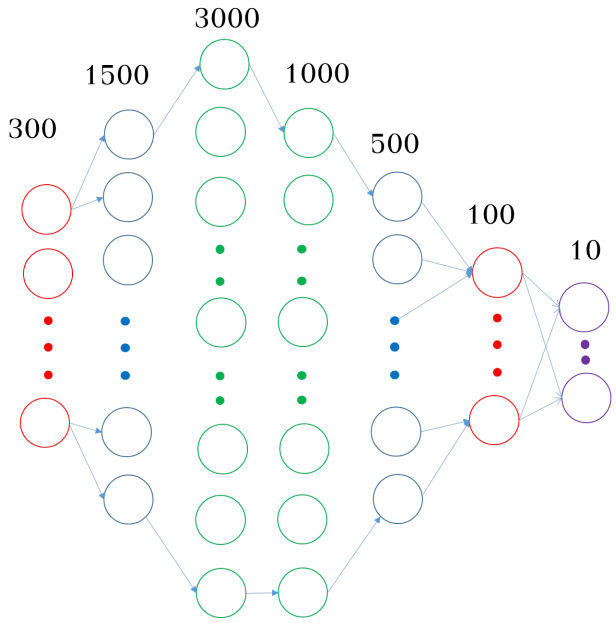


Figure 3: DNN algorithm

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 Figure 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。

3.3 Support Vector Machine(SVM)

Support Vector Machine，支持向量機，簡稱為 SVM。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 Natural language processing 等其他領域。

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 會判斷向量屬於這一類，或是其他剩餘的類別，比較每個分類器出來的分數最高者，向量屬於該類別。

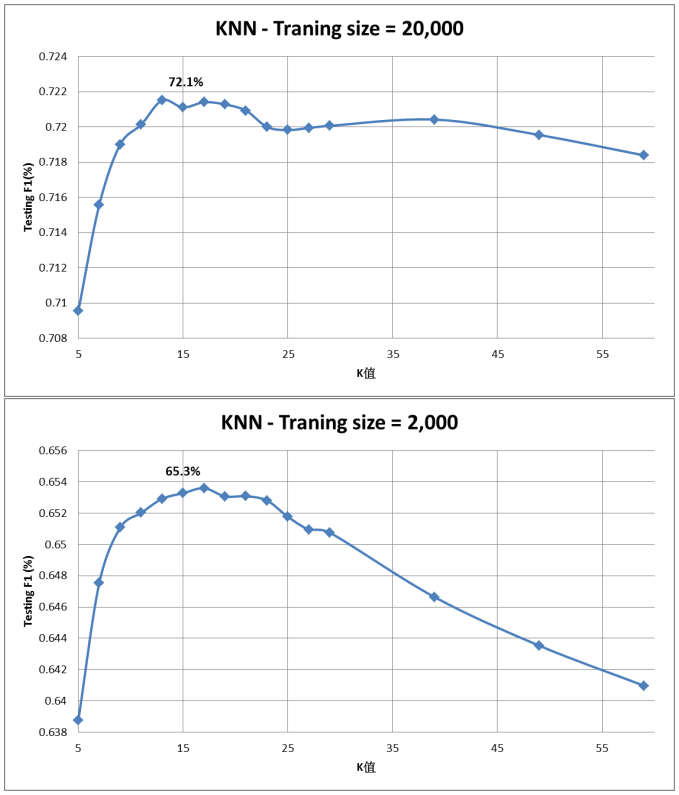


Figure 4: kNN-set k value

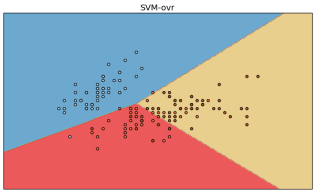
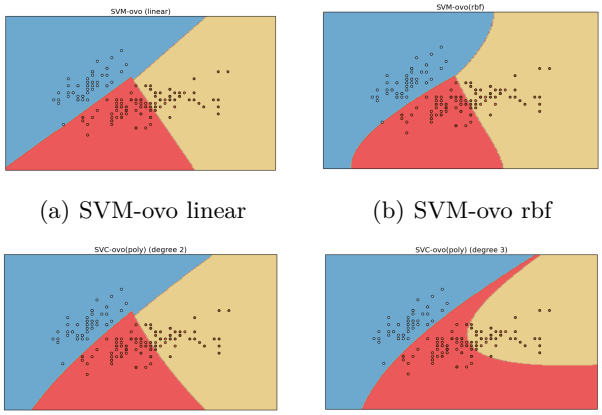


Figure 5: decision boundary of SVM-ovr

使用 library LIBLINEAR 撰寫，因此只支援 linear kernel，Figure 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。使用 library LIBSVM 撰寫，支援多種 kernel，包含 linear, Radial basis function(rbf), polynominal(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-



(a) SVM-ovo linear (b) SVM-ovo rbf

(c) SVM-ovo poly(degree=2) (d) SVM-ovo poly(degree=3)

Figure 6: decision boundary of SVM-ovo

ovo nonlinear，使用不同的 kernel，decision boundary 也會有所差異，見 Figure 6，SVM-ovo linear 與 Figure 5中的 SVM-ovr 的 decision boundary 非常相近，SVM-ovo rbf 呈輻射狀，而 SVM-ovo poly 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 值還好，因此將 degree 設為 2，同時 degree=2 也是常見的設定值。

3.4 Naïve Bayes(NB)

Naïve Bayes，貝是分類器，簡稱為 NB。本試驗使用 Multinomial naïve bayes 訓練資料，藉由最大化事後機率 (maximum a posteriori probability) 找出可能性最高的類別。本次試驗中 NB 所用的輸入與上述其他分類器不同，由於 NB 不能直接使用文章向量，因此改用轉換成文章向量前的文章。以 Log likelihood ratio 對經過斷詞處理後的文章挑選特徵 (feature selection)，假設 H_0:P($t_{ij}|C_i$)=P(t_{ij}|not C_i)，H_1:P($t_{ij}|C_i$)≠P(t_{ij}|not C_i)，對於所有的 term t_{ij} ∈class C_i，只要符合 P(H_1)> P(H_0)，t_{ij} 都會被選為 features，並且在訓練時使用 add-one smoothing。

4. MODEL PERFORMANCE

本次試驗以 Micro-averaged F1 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。Figure 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

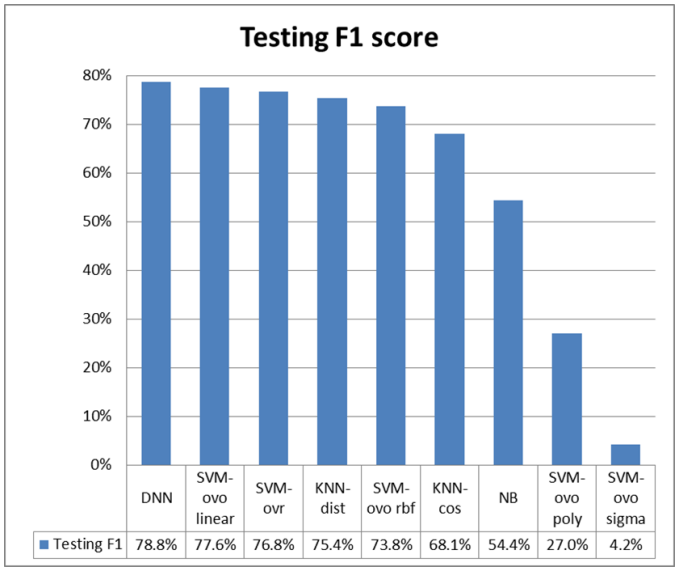


Figure 7: F1 score for all models when training size=100,000

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。所以 F1 score 由最高的排序，依序是 DNN > SVM-ovo linear > SVM-ovr > kNN-dist >SVM-ovo rbf > kNN-cos > NB，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。

5. SMALLER TRAINING DATA SIZE

在上一段已經以 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，故不放入圖表。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。

實驗結果 Figure 8，展現了所有模型在不同訓練量下的表現，由於模型過多，將表現較好的模型移出至 Figure 9觀察。在 Figure 9中，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 Figure 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 Traning F1 卻

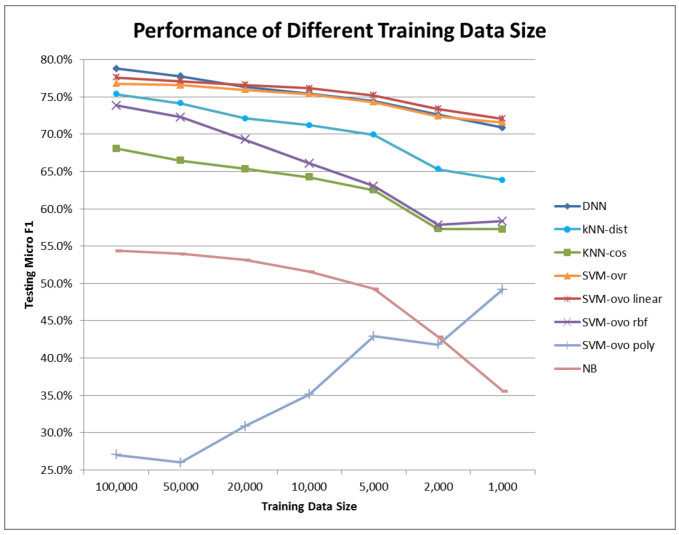


Figure 8: F1 score for all models in different training size

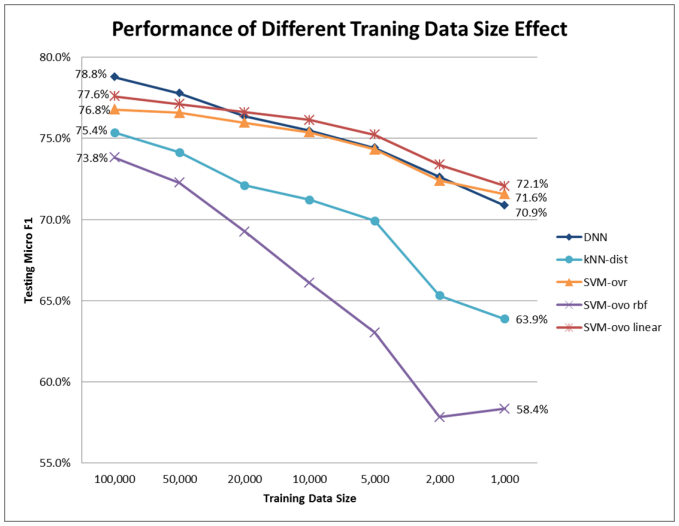


Figure 9: F1 score for some models in different training size

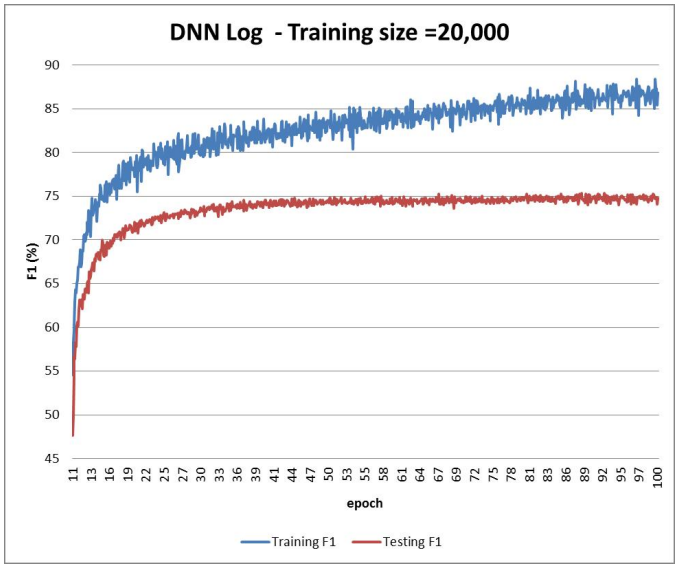


Figure 10: DNN Training and Testing F1 when training size=20,000

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，因為 DNN 在訓練量小時容易 overfitting，但在 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。而在 SVM-ovo linear 較 SVM-ovo nonlinear 優勢的原因則是，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 [1]，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，因此在抵抗資料見少的表現下，SVM-ovo linear = SVM-ovr > DNN > kNN-dist > SVM-ovo rbf。

6. COMPUTATION TIME

一個好的分類器能不能應用在實務上，除了表現以外，最

重要的就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，我們以不同的訓練量來檢視不同模型的花費時間，我們以 7 組不同的訓練資料量，而測試資料同樣是 23 萬，其中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 GPU 加速度的程式。由 Table 2觀察到，訓練時間的部分，除了 kNN 沒有訓練時間外，其他的模型訓練時間隨著訓練量的增加，特別可以觀察到 SVM 都是指數增長，大致上隨著資料量 2 倍成長，訓練時間也大約以 4 倍成長，資料量 2.5 倍成長，則訓練時間 6 倍成長，但 DNN 與訓練時間無明確關係，由於 DNN 是 iterative algorithm 會因為參數的不同會有不同的收斂時間，但是基本上愈少的訓練資料愈快收斂，當收斂到 cost 沒有明顯下降時就會提前停止 (early stop)，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時間。因此訓練時間複雜度方面，假設訓練量為 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 SVM-ovr 大約在 O(n)。

在測試時間的部分，參見 Table 3，在訓練資料量較大時，SVM,kNN 與 NB 大致上都是與訓練資料的增長成正比關係，DNN 與 SVM-ovr 則是不受訓練量影響，因為 DNN 與 SVM 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 能夠在讓人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。也因此若以能夠有良好的 F1 score 並且短時間內完成訓練及測試的模型，其首選莫過於 SVM-ovr，而 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，使用 6 層神經元架構，其實層數算相當多，其運算時間卻能少於其他 SVM-ovr 以外的模型。在測試時間複雜度的部分，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×m)。以總運算時間來看，大量訓練量時，總時間最短的排序：SVM-ovr > DNN > NB > kNN-cos > SVM-ovo linear > kNN-dist > SVM-ovo nonlinear

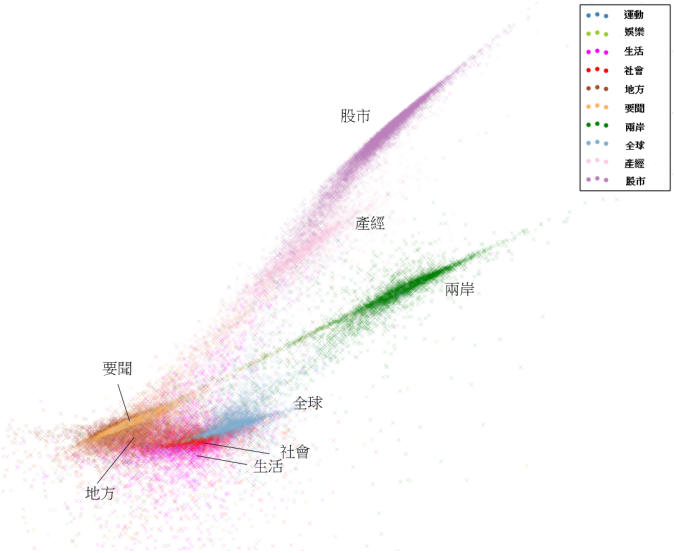
備註：在不使用 GPU 加速的情況下，也就是使用 CPU 運算，在訓練時間的部分 DNN 會增為 4 倍；測試時間部分，kNN 會增為 2 倍，DNN 會增為 100 倍，約為 1408 秒。

7. PCA VISUALIZATION ANALYSIS

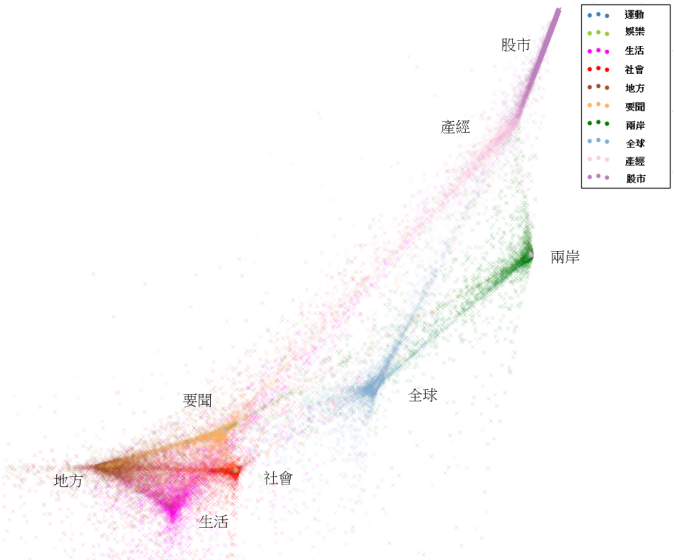
Principle Component Analysis(PCA)，常用於縮減向量維度與資料視覺化，而仍能保持原始資料間的相關性。參見 Figure 11，將 DNN, SVM-ovo linear 以及 SVM-ovo rbf 在 10 萬筆訓練量下訓練出的模型，所預測的 10 維文章類別機率，將每個類別取 5000 個資料點，投射到 2 維平面來觀察，其中運動類與娛樂類的由於投射到平面後與其他類別的的距離過遠，所以不顯示在圖內。可看到不同分類器間的 F1 score(分別是 78.8%, 76.8%, 73.8%) 表現相近，類別間的相對位置也類似，然而其資料點分散程度卻有些差異。由於 PCA 能原始資料點間的變異數有一定的解釋力，所以投射在 2 維上的資料點發散程度，可以表示原始資料的變異程度，因此非線性分類器 DNN 與 SVM-ovo rbf 發散程度比 SVM-ovo linear 來的高，顯現非線性分類器 high variance and low bias 的特性，在訓練小量資料以及高雜訊的向量會特別容易 overfitting 而表現較差，也可能是非線性分類器在訓練量小時，F1 表現不佳的原因，而 SVM-ovo linear 則是相對 low variance and high bias，兩者各有其優劣勢，雖然在 NLP 領域，線性分類器比非線性分類器還常見 [2]，然而在大量資料量而且能夠好抽取 feature 的情況下，如使用 word2vec 工具，孰優孰劣就不見得有一個定論了。

8. CONCLUSION

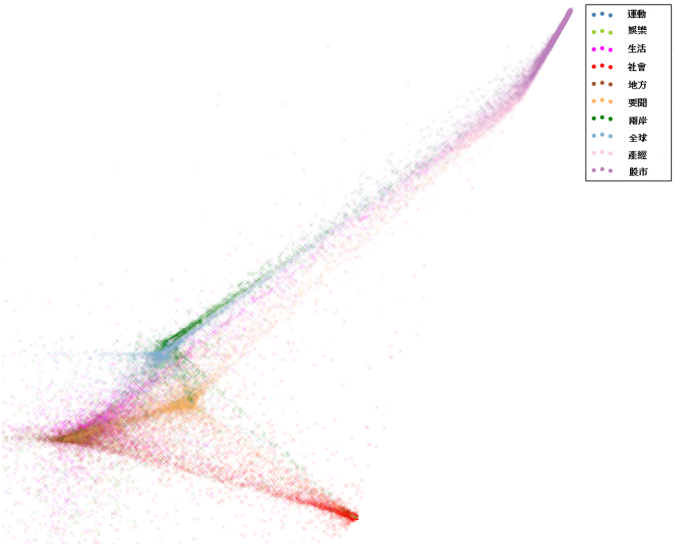
綜合以上資訊，若是同時追求運算速度與表現的條件下，DNN 是最好的選擇，在充足的訓練資料下 DNN 在所有模型中表現最好，然而參數的調整勢必會影響結果，由於缺乏一個固定的數學模型，DNN 不是每次都能訓練到一個固定且最佳的結果，幸運的是，在我多次訓練 DNN 的過程中，DNN 能有穩定且一致的表現，幾乎每次都能達到相近的 F1 值，所以 DNN 在文章分類會是個可行的方法，並且由於 DNN 彈性的架構，使得 DNN 具有很大的潛力有更好的表現，去適應不同的任務，並且當資料量減少時，DNN 在縮減層數以及神經元數目後，可能可以避免 overfitting 的問題，讓 DNN 模型有更好的表現。然而在小型的訓練資料下，SVM-ovo linear 與 SVM-ovr 會勝出，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



(a) DNN probability prediction



(b) SVM-ovo linear probability prediction



(c) SVM-ovo rbf probability prediction

Figure 11: PCA Visualizing Prediction

間，而追求稍微較好的表現，則 SVM-ovo linear 是最佳的選擇。

Table 2: Training Computation Time

Training Time / Training size	DNN	SVM-ovr	SVM-ovo linear	SVM-ovo poly	SVM-ovo rbf	SVM-ovo sigmoid	NB
100,000	504.97	95.94	3545.58	6715.68	2219.80	6156.36	52.48
50,000	438.78	38.49	954.42	1883.22	698.04	1541.93	4.12
20,000	450.46	14.83	175.44	283.90	145.05	247.50	1.62
10,000	276.06	5.31	50.43	60.39	42.20	60.67	0.80
5,000	263.23	2.12	14.33	14.44	13.03	14.61	0.39
2,000	254.97	0.61	3.13	2.20	2.35	2.19	0.16
1,000	463.25	0.26	1.03	0.56	0.58	0.56	0.08

Table 3: Testing Computation Time

Testing Time / Training size	DNN	SVM-ovr	SVM-ovo linear	SVM-ovo poly	SVM-ovo rbf	SVM-ovo sigmoid	kNN-dist	kNN-cos	NB
100,000	13.26	0.95	2969.43	7315.55	5820.49	7334.53	6954.23	3099.78	918.26
50,000	13.26	0.96	1505.81	4121.61	3269.73	4224.32	3517.64	1596.15	456.47
20,000	13.22	0.92	657.68	1710.59	1507.10	1694.37	1490.40	724.24	284.62
10,000	13.49	0.93	364.31	826.53	799.33	835.73	843.08	446.57	193.50
5,000	13.26	0.93	202.40	416.82	422.82	407.68	516.03	295.71	135.89
2,000	13.30	0.94	90.89	163.58	172.56	161.25	284.28	156.70	80.44
1,000	13.34	0.94	50.58	77.19	79.85	73.43	259.60	125.99	57.01

References

[1] Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin, et al. *A practical guide to support vector classification.* 2003.

[2] Thorsten Joachims. *Text categorization with support vector machines: Learning with many relevant features.* Springer, 1998.

[3] Andrew L Maas and Andrew Y Ng. "A probabilistic model for semantic word vectors". In: *NIPS Workshop on Deep Learning and Unsupervised Feature Learning.* 2010.

[4] Aliaksei Severyn et al. "Distributional neural networks for automatic resolution of crossword puzzles". In: *Association for Computational Linguistics (ACL).* 2015.