東海大學

資訊工程研究所

碩士論文

指導教授: 楊朝棟博士

以支持向量機及卷積神經網路在病理系統診斷結果應用之分析

An Analysis of Pathological Report Diagnosis Using Support Vector Machines and Convolutional Neural Networks

研究生: 王泳盛

中華民國一〇七年六月

東海大學碩士學位論文考試審定書

東海大學資訊工程學系 研究所

研究生 王 泳 盛 所提之論文

以支持向量機及卷積神經網路在病理系統診斷結果應用之分析

經本委員會審查，符合碩士學位論文標準。

學位考試委員會

召 集 人 許慶賢 簽章

委 員 劉榮春

黃頂晨

詹毓偉

指 導 教 授 楊朝棟 簽章

中華民國 107 年 6 月 8 日

摘要

本論文以機器學習應用於病理報告系統，為系統提供自動結果與診斷分類。對於臨床醫師而言，初步臨床臆測並不代表病人的最後診斷結果，還是必須透過病理醫師檢查病理切片，做出最終診斷。此外，病理醫師在檢查完畢後，必須登打報告給臨床醫師，其診斷類別甚多，則會因為討論而讓作業時間延長。因此，本論文透過機器學習來強化病理報告系統，以提升病理診斷及結果之成效。本論文基於機器學習，使用套件 scikit-learn 及深度學習框架 TensorFlow，訓練測試病理報告已達到診斷及結果分類。其中使用 SVM 及 Text-CNN 分類模型來做比較分析，其結果顯示應用於病理診斷中 SVM 分類較 Text-CNN 準確，原因為資料量多寡所影響，病理結果方面兩者並無差別，用於報告系統可即時回饋於病理醫師，並有效校正其判斷縮短臨床溝通。

關鍵字：病理報告、診斷分類、SVM(支持向量機)、Text-CNN(卷積神經網路)。

Abstract

This thesis use machine learning applied to the pathology reporting system to output result and diagnostic classification. In general, we can infer the clinical results of biopsy through the pathological report. However, the initial decision does not represent the final result for the clinicians, it must be professionally inspected by a pathologist, and hardly any progress can be made while waiting for the result. Furthermore, the pathologist needs to type the progress note to the clinicians when the result were checked; besides a number of diagnostic classifications will let the work time be extended cause of the discussion. So, this paper use machine learning to enhance the effectiveness of pathological diagnosis. This paper trains the test pathology reports to get the diagnosis and classifications with scikit-learn and TensorFlow, the deep learning framework. We use SVM and Text-CNN classification model for comparative analysis. The result shows SVM is more accurate than Text-CNN; the reason is the result affected by the amount of data. There is no difference in pathological results between the two. Our aims an to gives immediate feedback to the pathologist and effectively correct its judgment to shorten clinical communication.

Keyword : Pathology reporting, Diagnostic classification, SVM, Text-CNN

誌謝辭

在時隔 7 年後終於完成了當兵前的規劃，順利取得學位。學習方向一路從資訊管理至資訊工程，過程都非常順遂。這要非常感謝我的指導老師楊朝棟教授，楊老師在我求學過程當中，不管是在忙碌的平日及寒暑假都會撥空出來討論指引研究方向，給予我許多寶貴的意見，更教導我許多從未接觸過雲端計算及大數據技術，透過實驗及專案實作磨煉技術，讓我獲益良多，老師的教學態度真的讓我很欽佩，對於已經在工作我來說，能兼顧學業、工作本來就很不容易，謝謝楊老師教導知識、態度及經驗，讓我能學以致用，將這些學到的技術應用在工作上，讓我也能獲得贏在起跑點的機會。

在此也要感謝中華大學資訊工程學系許慶賢教授、東海大學資訊工程學系劉榮春教授、國立臺中教育大學資訊工程學系黃國展授教及靜宜大學資訊管理學系詹毓偉助理教授在口試過程中，給予學生諸多指導及許多修改建議，讓我的論文內容能夠更臻於完善。

在求學過程中，認識了許多好朋友及好同學及好老師，雖然年齡上有很大的距離，但彼此的經驗及心得分享卻是沒有距離的。在此特別感舜文學長，不論技術或態度上，從他身上學習到很多，也謝謝實驗室的同學們，希望相聚相識的緣分不會隨著畢業而消散。

Table of Contents

List of Figures

List of Tables

Chapter 1

簡介

1.1 研究動機

近年來，機器學習已應用在各領域上，隨著 AlphaGo [1] 在人機大戰的勝利，機器學習變成家喻戶曉的名詞，也在各領域遍地開花，醫療領域更是蓬勃發展，如 IBM 的 Watson for Oncology（人工智慧癌症治療輔助系統）、google 利用機器學習來檢測糖尿病性視網膜病變 [2]，都是極為成功的發展。其中，現今機器學習應用於醫學研究上皆聚焦於圖片分析來輔助醫師診斷，其一原因為機器學習其中分支項目深度學習 (deep learning) 此項研究技術的發展，在圖片分析有極為重大的發展所造成，但需完成類似醫學研究報告，如沒有較完整的設備基礎或資料量，則難以完成。然而能輔助醫師的並不只有圖片，隨著病歷電子化越來越普及，許多病歷、病摘、報告已妥善利用文字型態保存至資料庫中。機器學習除了圖型辨識，另一個領域則是自然語言處理 (Natural Language Processing)，是作用於文字讓電腦理解人類語言中的句子或詞語，對於了解病歷很有幫助。

然而，自然語言處理至 1950 年開始 [3] 發展至今已衍生許多任務，例如：機器翻譯、自動摘要、篇章分析、語意分析、本文分類…等，在現行生活中也有許多實際應用如：新聞分類 [4]、垃圾郵件過濾、google 翻譯、Siri 都是人們熟悉的應用。其中，新聞分類在過去的媒體或新聞廠商常常都以人工處理分類，

相當消耗人力資源及時間，透過自然語言任務中本文分類已有極大的效果改善此問題 [5]，基於上述，隨著醫院規模大小病理件數一定不同，病理醫師人數也會有所差異，如區域醫院每天約 50 件以上病理檢體，而醫師配置只有 1 名，而人在做事情是會疲憊的，有時會將對應診斷歸類至大項，這將對未來相關類型之臨床研究或治療造成無法快速判讀，本論文將透過本文分類簡化病理醫師完成報告後，須對報告判定正確的腫瘤診斷 (MCODE)、檢體部位 (TCODE) 及病理結果，改善登打報告之速度及正確率，讓病理醫師更專注於顯微鏡下的檢查，其流程如 Figure 1.1，其腫瘤診斷標準判定分類代碼標準為 ICD-O-3 多重原發性腫瘤分類標準 [6]，病理結果代碼為 Table 1.1。

Figure 1.1: 病理報告檢查至臨床流程

Table 1.1: 病理結果代碼檔

代碼	結果
O	其他
B	良性腫瘤
U	惡性不明
M	惡性腫瘤
P	癌前病變
I	感染
X	惡性腫瘤及感染

1.2 論文目標與貢獻

本論文將運用監督式學習建立本文分類模組，對病理報告系統文字部分進行訓練分析，依據病理醫師登打之報告提供相對正確之腫瘤診斷 (MCODE)、檢體部位 (TCODE) 及病理結果，減少因人工因素導致病理診斷不太理想之情況。模型建置將利用開源套件 scikit-learn 中貝氏分類器 (Naive Bayes Classifier)、支持向量機 (Support Vector Machine) 方法建立，並建置深度學習框架 TensorFlow 中 Text-CNN，比較傳統本文分類與深度學習方法應用於病理報告系統之差異，提出較適用之模型，可即時回饋於病理醫師，模型建立後除放入線上執行並將過去尚未至病理報告系統的病理文字報告給予對應診斷及結果並將其整理至資料庫中，改善臨床或病理醫師找不到過去資料遺漏之問題。

其模型建置完畢後與現行病理報告系統做串接，將建立 RESTful 架構來達到跨平台服務之功能，跳脫本機或程式語言之限制，並達到功能分工，透過此串接也可讓程式人員各分其職提高撰寫之效率，另外病理醫師可透過此串接回饋模型判定診斷之錯誤或正確，使模型更加精準。模型建置中原本設定為深度學習模型會比傳統模型來得好，其結果顯示應用於病理診斷中傳統分類較深度學習準確，其原因是資料量多寡所影響。

1.3 論文架構

本論文主要利用機器學習中本文分類技術來協助病理醫師診斷歸類，並透過模型回歸過去舊有未歸類之報告，改善病理醫師報告速度及臨床溝通。如 Figure 1.2、Figure 1.3第一章簡述機器學習相關研究過程與結果，再於本文分類應用伸出本論文的研究目標及方向；第二章介紹相關研究背景及其各項套件技術概念、本文中將使用之運用工具進行說明簡述；第三章節中將描述論文中使用的系統架構概觀與機器的資源配置規格；第四章實驗環境配置與實驗數據處理與結果呈現；最後於，第五章將對於本論文之實驗過程、結果，經彙整後作出最終結論，並且從中延伸出未來的研究方向。

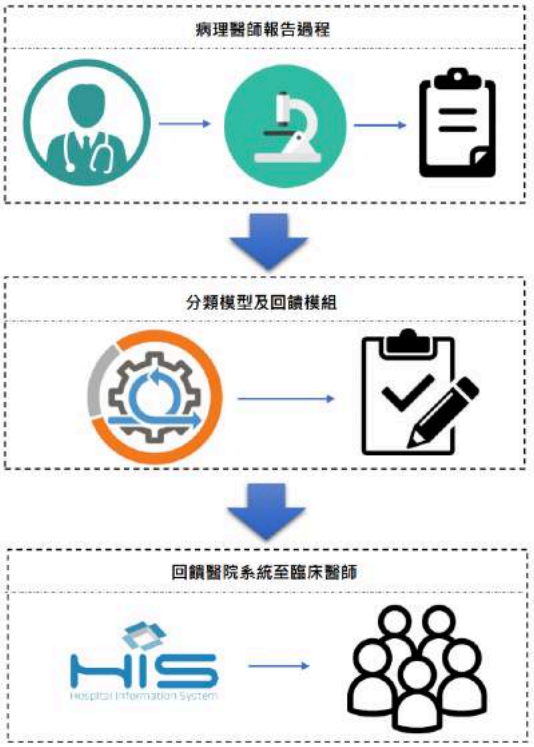


FIGURE 1.2: 病理與臨床流程

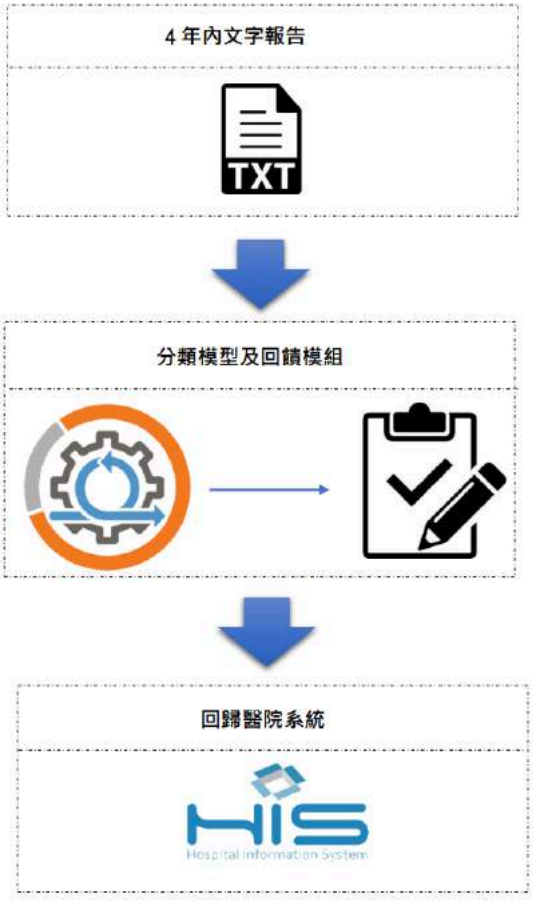


FIGURE 1.3: 回歸過去舊有未歸類之報告

Chapter 2

研究背景與相關研究

2.1 研究背景

2.1.1 自然語言處理

所謂的自然語言處理 (Natural Language Processing, NLP) 指的就是人類所使用的語言，如語音、文字等，其目的是希望透過資訊技術讓電腦能夠理解人類所使用的語言 [7]。自然語言處理原本是屬於人工智慧 (Artificial Intelligence) 領域中很重要的一項技術，隨著技術的成熟已逐漸應用到文字探勘、手寫輸入、語音辨識、文件分析、本文分類、知識管理等領域中。

2.1.2 機器學習

機器學習是人工智慧的一個分支。[8] 人工智慧的研究是從以「推理」為重點到以「知識」為重點，再到以「學習」為重點，一條自然、清晰的脈絡。顯然，機器學習是實現人工智慧的一個途徑，即以機器學習為手段解決人工智慧中的問題。機器學習在近 30 多年已發展為一門多領域交叉學科，涉及機率論、統計學、逼近論、凸分析、計算複雜性理論等多門學科。機器學習理論主要是設計和分析一些讓電腦可以自動「學習」的演算法。機器學習演算法是一類從

資料中自動分析獲得規律，並利用規律對未知資料進行預測的演算法。因為學習演算法中涉及了大量的統計學理論，機器學習與推斷統計學聯繫尤為密切，也被稱為統計學習理論。演算法設計方面，機器學習理論關注可以實現的，行之有效的學習演算法。很多推論問題屬於無程式可循難度，所以部分的機器學習研究是開發容易處理的近似演算法。機器學習的可分三種如 Figure 2.1學習方法定義如下

- 監督學習 (Supervised Learning)：
  給定的訓練資料集中學習出一個函式，當新的資料到來時，可以根據這個函式預測結果。監督學習的訓練集要求是包括輸入和輸出，也可以說是特徵和目標。訓練集中的目標是由人標註的。

- 無監督學習 (Unsupervised Learning)：
  與監督學習相比，訓練集沒有人為標註的結果。常見的無監督學習演算法有聚類。

- 增強學習 (Reinforcement Learning)：
  透過觀察來學習做成如何的動作。每個動作都會對環境有所影響，學習物件根據觀察到的周圍環境的反饋來做出判斷。

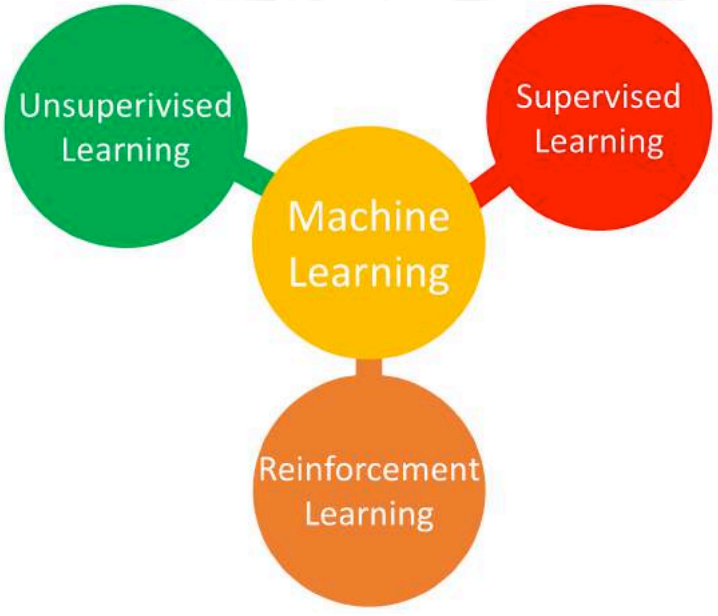


FIGURE 2.1: 機器學習方法

2.1.3 本文分類

文本分類問題算是自然語言處理領域中一個非常經典的問題，相關研究最早可以追溯到上世紀 50 年代。當時是通過專家規則（Pattern）進行分類，甚至在 80 年代初，一度發展到利用知識工程建立專家系統。這樣做的好處是短平快的解決 top 問題，但顯然天花板非常低，不僅費時費力，覆蓋的範圍和準確率都非常有限。後來伴隨著統計學習方法的發展，特別是 90 年代後大數據文本數量增長和機器學習學科的興起，逐漸形成了一套解決大規模文本分類問題的經典玩法，這個階段的主要方法為人工特徵工程和淺層分類模型 [9]。如 Figure 2.2

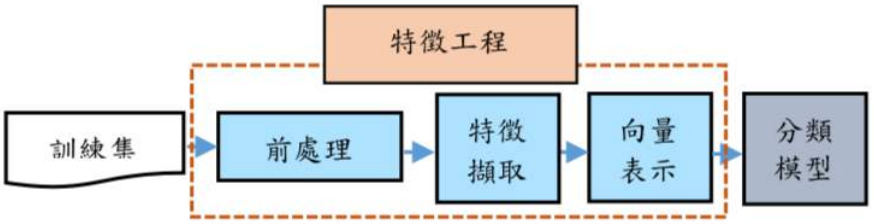


FIGURE 2.2: 本文分類技術過程

2.1.4 特徵工程

特徵工程 [10] 在機器學習中往往是最耗時耗力的，但卻極其的重要。抽象來講，機器學習問題是把數據轉換成訊息再提煉到知識的過程，特徵是“數據至訊息”的過程，決定了結果的上限，而分類器是“信息至知識”的過程，則是去逼近這個上限。然而特徵工程不同於分類器模型，不具備很強的通用性，往往需要結合對特徵任務的理解。

文本分類問題所在的自然語言領域自然也有其特有的特徵處理邏輯，傳統分本分類任務大部分工作也在此處。文本特徵工程分位文本預處理、本文表示、特徵提取三個部分，最終目的是把本文轉換成電腦可理解的格式，並封裝足夠用於分類的信息，即很強的特徵表達能力。

本文特徵工程三部分：

- 本文預處理：

  本文預處理是在本文中提取關鍵字表示的過程，也是數據挖掘的一種技術，實際的數據常常不完整、不一致甚至缺乏某些訊息，更可能夾雜著大量的錯誤或混亂訊息，本文預處理就是解決這些問題的有效法，對原始本文進行第一部的處理。

- 本文表示：

  本文表示的目的是把本文預處理後的訊息轉換成電腦能夠理解的方式，是決定分類質量最重要的部分，傳統的方法常用詞袋模型 (Bag Of Words,BOW) 或向量空間模型 (Vector Space Model)。

- 特徵提取：

  本文轉為向量後極為重要的課，將各本文適合的特徵項選出來，根據某個評估指標獨立的對詞項進行評分排序，從中選擇得分最高的特徵項，過濾掉其餘的特徵。常用評估指標有頻率、關係度、統計量等。其中特徵全中最為經典的算法為 TF-IDF 方法，主要思路是一個詞的重要度與在類別內的詞頻成正比，與所有類別出現的次數成反比。

2.1.5 Term Frequency-Inverse Document Frequency (TF-IDF)

TF-IDF 是一種用來評估詞彙與文章關聯程度的統計方法。詞彙的重要性隨著它在文件中出現的次數成正比增加，但同時會隨著它在語料庫中出現的頻率成反比下降。單一詞彙 t_i 的詞頻 (term frequency, TF)，可由 (2-1) 式計算得知，其中 $n_{i,j}$ ，$n_{k,j}$ 分別表示詞彙 t_i ，t_k 在文件 d_j 出現的字數，分母即為文件 d_j 中所有辭彙出現次數之總和。

$$TF_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}} \quad ，(2\text{-}1)$$

逆向文件頻率 (inverse document frequency,IDF) 是一個計算詞彙重要性的方法。某一特定詞彙的 IDF ，可以由 (2-2) 式得到，其中 |D| 是語料庫中的文件總數，$|\{j : t_i \in d_j\}|$ 表示包含詞彙 t_i 的文件數目。

$$IDF_i = log\frac{|D|}{1+|j:t_i \in d_j|}，(2\text{-}2)$$

由 (2-1) 式及 (2-2) 式，我們可以計算出一詞彙 t_i 的 TF-IDF 權重，見 (2-3) 式。

$$TF - IDF_{i,j} = TF_{i,j} \times IDF_i，(2\text{-}3)$$

2.1.6 N-Gram(N 連詞模型)

N-Gram 是一種基於統計語言模型的算法。它的基本思想是將文本裡面的內容按照字節進行大小為 N 的滑動窗口操作，形成了長度是 N 的字節片段序列。每一個字節片段稱為 gram，對所有 gram 的出現頻度進行統計，並且按照事先設定好的閾值進行過濾，形成關鍵 gram 列表，也就是這個文本的向量特徵空間，列表中的每一種 gram 就是一個特徵向量維度。以『我是男孩子』為例，若 2-gram 抽取結果為：我是 / 是男孩子。

2.1.7 Naive Bayes Classifier (單純貝氏分類器)

Naive Bayes Classifier 依據貝氏定理 (Bayes) 為基礎，主要的運作原理，是透過訓練樣本，學習與記憶分類根據所使用屬性的關係，產生這些訓練樣本的中心概念對未歸類的資料進行類別預測，已得到受測試資料物件的目標值。Naive Bayes 分類器做了一個簡化的假設，對於結果來說每個參數之間出現的機率是互相獨立的，此簡化的假設使 Naive Bayes 分類器有簡單快速的。Naive Bayes 公式如 (4) 所示，分別計算在一給定文件下，文件歸屬於每一類別的機率，並將文件的類別指定給機率高的類別。

$$p(Doc|C_k) = \prod_{j=1}^{m} P(W_j|C_k)，(2\text{-}4)$$

2.1.8 支持向量機法 (Support Vector Machine, SVM)

支援向量機法 (Support Vector Machine, SVM) 與類神經網路相似，是 1995 年由貝爾實驗室的 Vapnik 所提出，SVM 是以統計理論與核心函數 (Kernel Function) 為基礎所衍生出來的學習方法，起初 SVM 是被開發用來建構線性可分割的超平面，但現在則主要被拿來應用在非線性迴歸的分類上，由於在分類的實際應用上有不錯的表現，目前已被成功的應用在文字分類 [11]、人臉辨識 [12]、生物資訊 [13] 等等，且成為現在機器學習 (machine learning) 與資料探勘 (data mining) 主要方法之一。

支持向量機是基於定義決策邊界的決策平面的概念。決策平面是一組在具有不同類別成員的對象之間分開的平面，如 Figure 2.3。在圖中可看出分為橘色和藍色兩類，而任意出現新物體（白色圓圈），在右側出現則會歸類為藍色。[14]

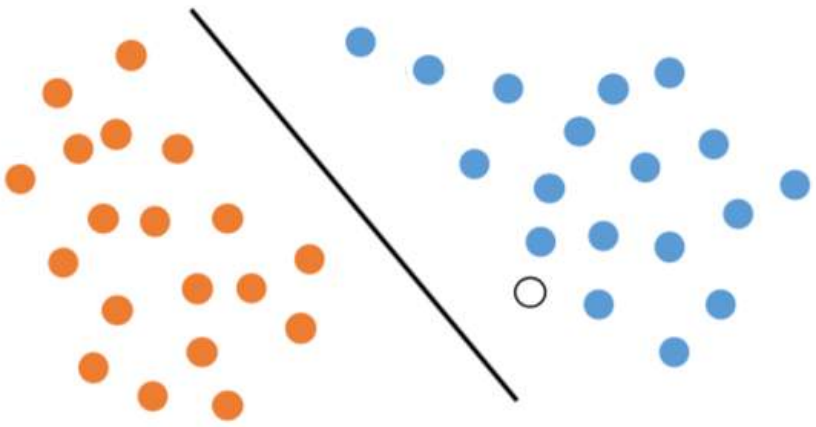


FIGURE 2.3: 決策平面

上述是一個線性分類的經典例子，即用一條線將一組對象分成它們各自的組別（在這種情況下為橘色和藍色）。然而，大多數分類任務並沒那麼簡單，並且通常需要更複雜的結構來實現最佳分離，如與 Figure 2.4與 Figure 2.3相比很明顯發現橘色和藍色物體要完全分離需要一條曲線 (比線條更複雜)。當遇到此類無法以簡單線性分割的問題時，必須藉由物件空間 (Instance space) 中將資料轉換成特徵空間 (Feature space)，如 Figure 2.5，這就是支持向量機的基本思維。

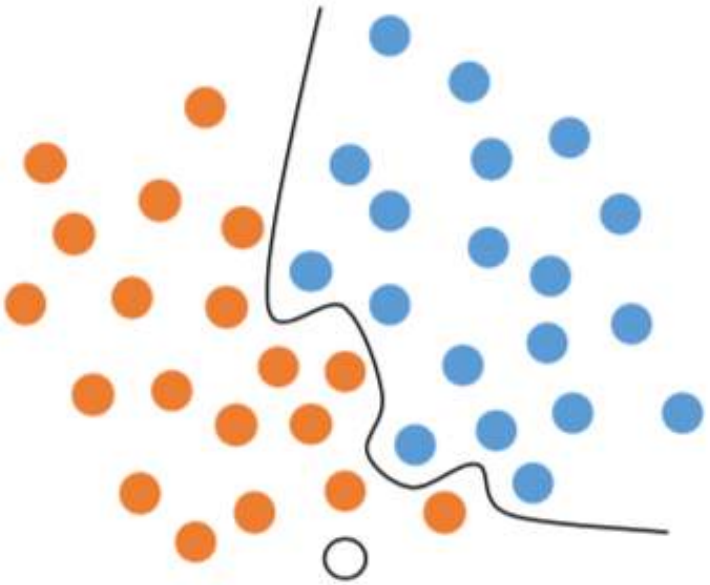


FIGURE 2.4: 真實分類情況

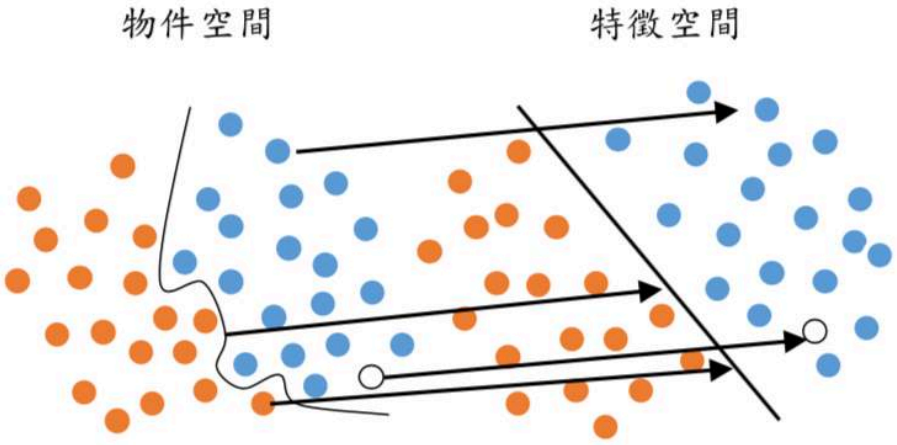


FIGURE 2.5: 物件空間轉特徵空間

2.1.9 深度學習

機器學習所延伸出其中一項技術，稱為深度學習。深度學習主要是嘗試利用抽象的演算方式建構出複雜結構或多重非線性結構的網路架構，而每層網路的架構個別有其應用的意義，並在多個處理層網路對資料進行深層抽象的演算

法。[15] 基於神經網路發展歷史深度學習框架追朔到 1980 年福島邦彥所提出的認知機。由第一章節提到現今較多醫學發展都是圖型分析，因近幾年高效能圖形處理器 (GPU) 的出現，加速了數值和矩陣運算的速度，使得人工神經網路的學習演算執行時間得到了顯著的縮短，促使深度學習重新獲得各界的關注。

深度學習的進步並不是因為 GPU 的出現而才那麼多人關注的，主要原因是可透過演算法訓練資料進行主動特徵學習，從訓練資料中進行案例學習，並能夠自動化去判定未知的圖片，取代人工方式來辨別。對於自然處理語言，可經由句子中詞與詞之間關係來表示分類，而句子中基本的單位是單詞，透過文字轉向量 (word vector) 則可看作一種強度向量值如 Figure 2.6，而通過抽象性的特徵表示則更能夠容易達到學習之成效。然而，深度學習網路主要是利用監督式學習方式與分層式特徵提取的演算法來替代傳統抽樣不佳的特徵及人為主觀性的選擇特徵之方式。

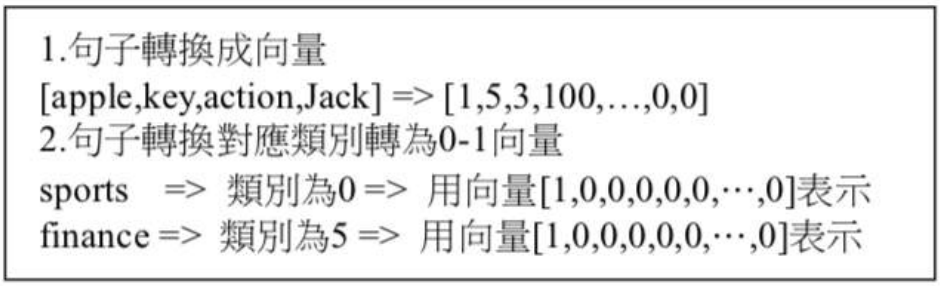


FIGURE 2.6: 轉向量例子

2.1.10 Text-CNN

大多數自然語言任務的輸入不是像圖像素，而是句子或文字。每一行對應一個標記，通常是一個單詞，有可能是一個符號，也就是說每一行都表示單詞或向量，則可套用至ＣＮＮ中的矩陣呈現，再透過類似 N-Gram 方式掃描，即可做到類似時ＣＮＮ的卷積。針對自然語言的深度學習本論文使用 Yoon Kim 發表的 Convolutional Naural Networks for Sentence Classification [16]，架構如 Figure 2.7。架構說明如下

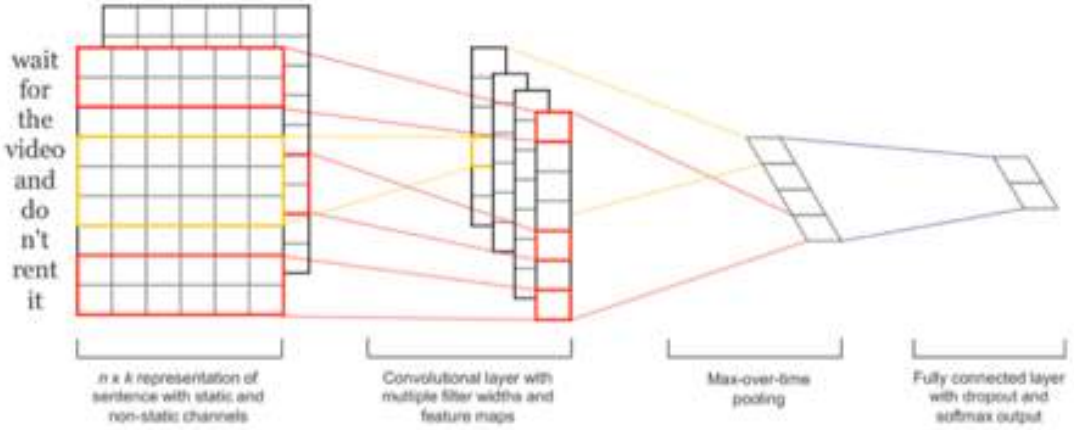


FIGURE 2.7: Text-CNN 過程

- 輸入層：

  如 Figure 2.7 所示，輸入層式句子中的詞語對應的 word vector 依序 (由上到下) 排列的矩陣，假設句子有 n 個詞，vector 的維度為 k，這個矩陣就是 n*k。

- 第一層卷積層：

  輸入層通過卷積操作得到若干個 Feature Map，卷積窗口的大小為 h*k，其中 h 表示縱向詞語的個數，而 k 表示 word vector 的維度。通過這樣一個大型的卷積窗口，將得到若干個列數為 1 的 Feature Map。

- 池化層：

  如 Figure 2.7 中使用 Max-over-time Pooling 的方法。這種方法就是簡單的以之前一維的 Feature Map 中提出最大的值，最大的值代表著最大的訊息。可以看出，這種 Pooling 方式可以解決可變長度的句子輸入問題 (因為不管 Feature Map 中有多少值，只需要提取其中的最大值)。

- 全連接 +Softmax 層：

  池化層的一維向量的輸出通過全連接的方式，連接一個 Softmax 層，Softmax 層可根據任務的需要設置 (通常反映著最終類別上的機率分佈)。

2.1.11 RESTful

一個架構符合 REST 原則，就稱它為 RESTful 架構，而 REST，即 Resource Representational State Transfer (具象狀態傳輸) 的簡寫 [17]，是一種分散式超媒體系統的軟體架構風格如 (WWW)，目的是便於不同軟體/程式在網路中互相傳遞資訊如 Figure 2.8。

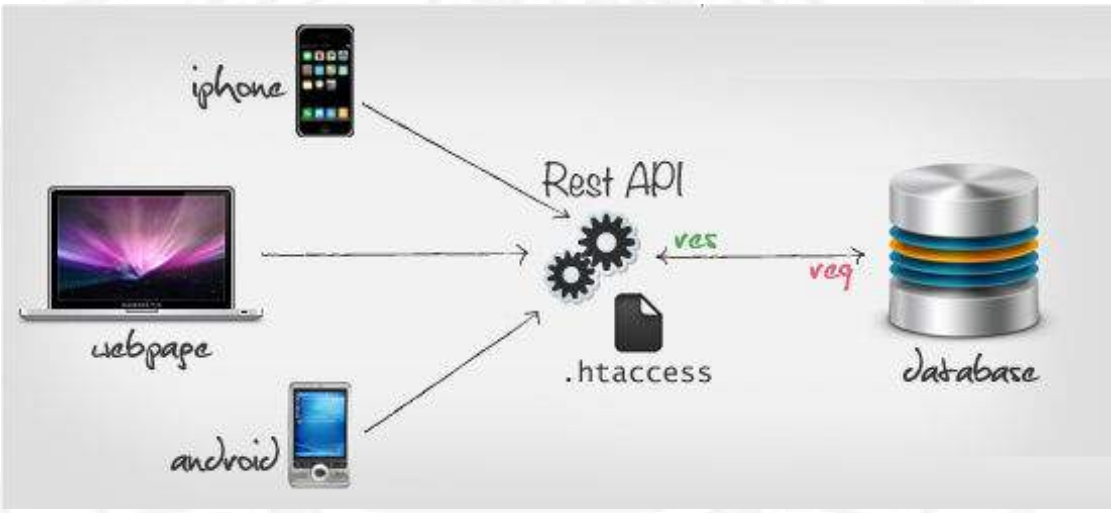


FIGURE 2.8: 不同軟體互相傳遞

簡單解釋 REST 就是資源在網路上以某種形式進行狀態轉移，將名詞分開介紹如下列項目

- 資源 (Resource) ：
  資源就是網路上一個實體，或者說是網路上一個具體的訊息，它可以是一段本文、一張圖片、一首歌曲、一種服務，總之就是一個具體的存在。而我們可以用一個 URL 指向它，每種資源對應一個特定的 URL，要得到資源只要訪問 URL 就可取得，因此 URL 就成了每一個資源獨一無二的識別。

- 表現層 (Representational) ：
  資源是一個訊息實體，它可以有很多表現形式，例如可以是 TXT、HTML、XML、JSON 等格式，甚至可以採用二進制格式或圖片格式來表現。

- 狀態變化 (State Transfer)：

  訪問一個網站，就代表客戶端和服務器的一個互動過程。在這過程中，勢必涉及到數據和狀態的變化。具體來說通過 HTTP 協定裡面四個表示操作的動詞 GET、POST、PUT、DELETE。它們分別對應四種基本操作：GET 用來獲取資源，POST 用來新建資源（也可以用於更新資源），PUT 用來更新資源，DELETE 用來刪除資源。

2.2 相關研究

隨著網路發達及資訊科技的進步，醫療資訊也蓬勃發展，機器學習的過去發展至近年有許多的躍近醫療發使用又是很多人研究的主題，如利用神經網路診斷紅斑性鱗狀細胞疾病的組合 [18]、基於患者概況分析的預防保健模型使用大數據分析方法用於糖尿病患者 [19]、使用深度學習幫助病理學家檢測癌症 [20] 等。而機器學習取代人工醫師判定或瞭解病患情況好像已經不是夢想，但不難發現需要做到完美的機器學習需要大量的成本 [21]，本論文希望將機器學習應用於病理報告系統已協助病理醫師快速完成報告以利臨床使用，但成本考量無法以圖型來做機器學習方法，並開始研究診斷部分，以下為相關論文。

Meystre and Haug [22]，將自然語言應用在醫學臨床決策中，此論文可以藉由醫師輸入的摘要解析出，常見的疾病分類，也提供相對應的程式，以便『即時』而『準確』的提供病患有可能的診斷，準確率高達 94%，而又發現 GK Savova [23] 使用自然語言處理應用於放射筆記中發現動脈疾病及 LTE Cheng [24] 從非結構化 MRI 報告中辨別腫瘤狀態 - 現有報告中信息的完整性和自動化自然語言處理的實用性也是極具參考的論文，發現自然語言處理，可以現有數據及非高成本達到效率不錯的醫療協助。

Jose 發表以三種機器學習方法對腫瘤診斷自動分類的實證評估 [25]，論文中提到診斷碼不僅是必要的評估患者病情嚴重程度，也是獲得描述性數據的醫療評估標準，內容中使用了 SVM 分類器準確度高達 86 %，發現 SVM 分類器在其領域也有極大的發展 [26] [27] [28]，確定此分類法為其一比較分類法。

而近年來深度學習的成功 [29]，也造就自然語言深度學習的發展 Yoon Kim 由論文中可看到 Text-CNN 用於二分類情緒分析也有將近 81% 的準確度。其研究激發可與傳統自然語言分類應用於診斷分類來做比較。

資訊技術的進步造就了新的溝通架構，網際網路即是大家每天都在使用的其中一個發展，但所謂資訊發展很多都還是由單機為開始，有時單機的運算能力並不強，故需要多台電腦處理，而歷史的發展也使一些程式無法跨平台使用，而由 Fielding 所描述的 REST 架構即可提供一個跨平台的接口，讓不同平台或不同端點得以溝通，成本也只要網際網路支援即可，而也有以簡易方法即可架設之放式 [30]，實際 REST 運用上也已經有所實作 [31] 提供跨平台的服務以不是問題。

Chapter 3

系統設計與實作

3.1 系統架構

3.1.1 硬體配置架構

本論文硬體配置架構為三部分測試訓練端、模組放置端、客戶端如 Figure 3.1，以下為各端點解釋：

- 測試訓練端 (Test & Training) ：
  此端點為本文分類系統建構位置並建立傳統文本分類及深度學習文本分類。

- 模組放置端 (Module)：
  將訓練過後完成之模型放置雲端虛擬環境中提供模型服務，以便不同系統使用。

- 客戶端 (Client)：客戶端應用程式呼叫雲端模組服務。

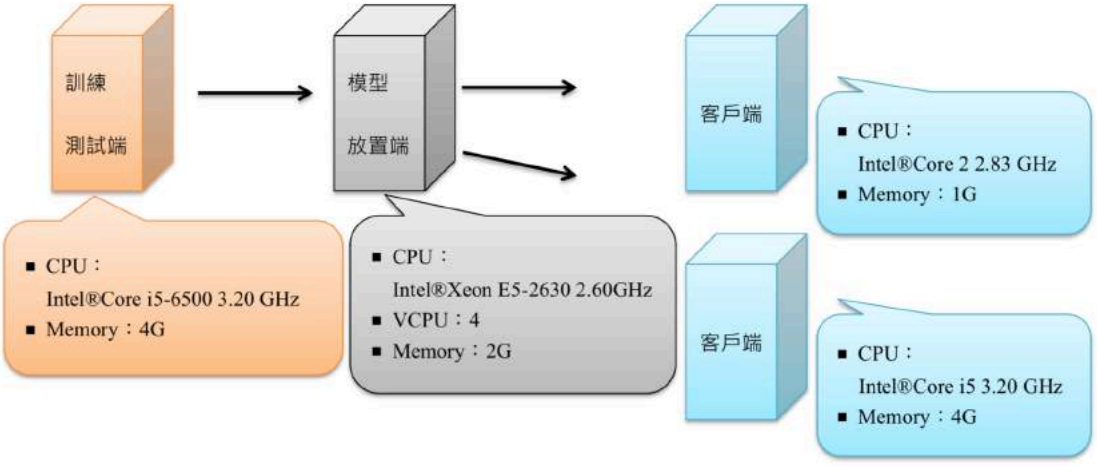


FIGURE 3.1: 硬體配置架構圖

3.1.2 本文分類模型架構

本論文研究探討機器學習應用病理報告系統分析，機器學習部分使用其分支自然語言中本文分類任務，其架構可分四大模組做處理，如 Figure 3.2，以下概述四大模組的主要功能：

(1) 資料擷取模組 (DataGet Module)：

此模組為資料彙整的第一步驟，主要因為各病理文字報告散落於各院區資料庫中，固此模組須先做資料彙整。

(2) 標籤模組 (Tag Module)：

此模型必須將各院區所彙整病理報告，分別於前方標記 ICD-O-3 所規定之 MCDOE、TCODE 和病理結果，以幫助後續模組分類使用。

(3) 自然語言處理模組 (NLP Module)：

主要目的及讓病理報告內容只保留有效字，並將整篇病理報告做切割 (Segmentaion)，以便後續作轉換或計算權重之用，因此該模組包括：篩選 (Filter Process)、切割 (Segmentaion Process)

(4) 訓練與測試模組 (Training and Testing Module)：

主要目的是利用傳統本文分類方法及深度學習方法來互相比較用於病理報

告各分類之準確度，本論文使用的分類器為 Naive Bayes Classifler、SVM、Text-CNN。

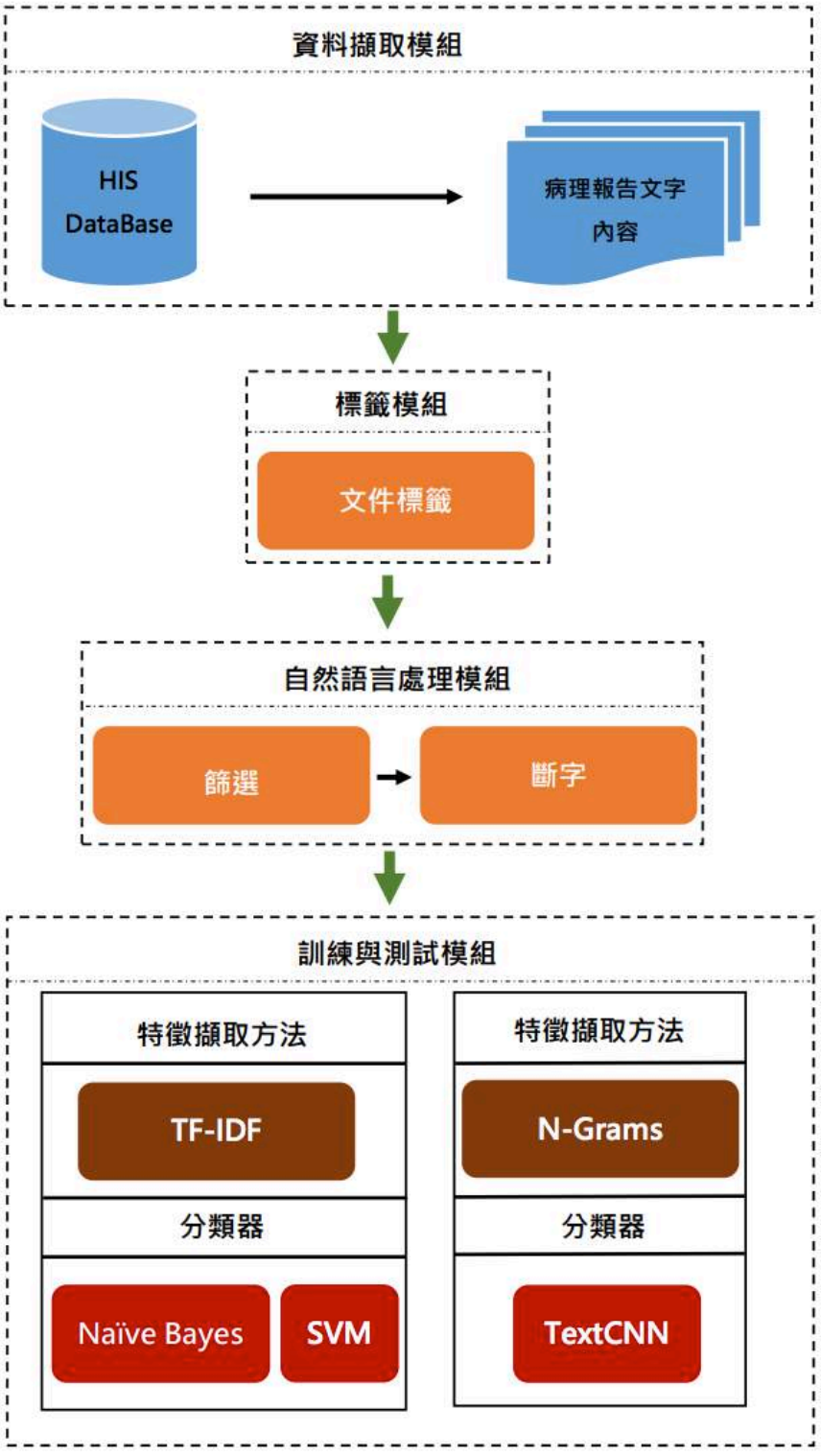


FIGURE 3.2: 本文分類模型架構

3.1.3 資料擷取模組

資料擷取模組如 Figure 3.3，主要進行資料收集散落在院所病理報告。

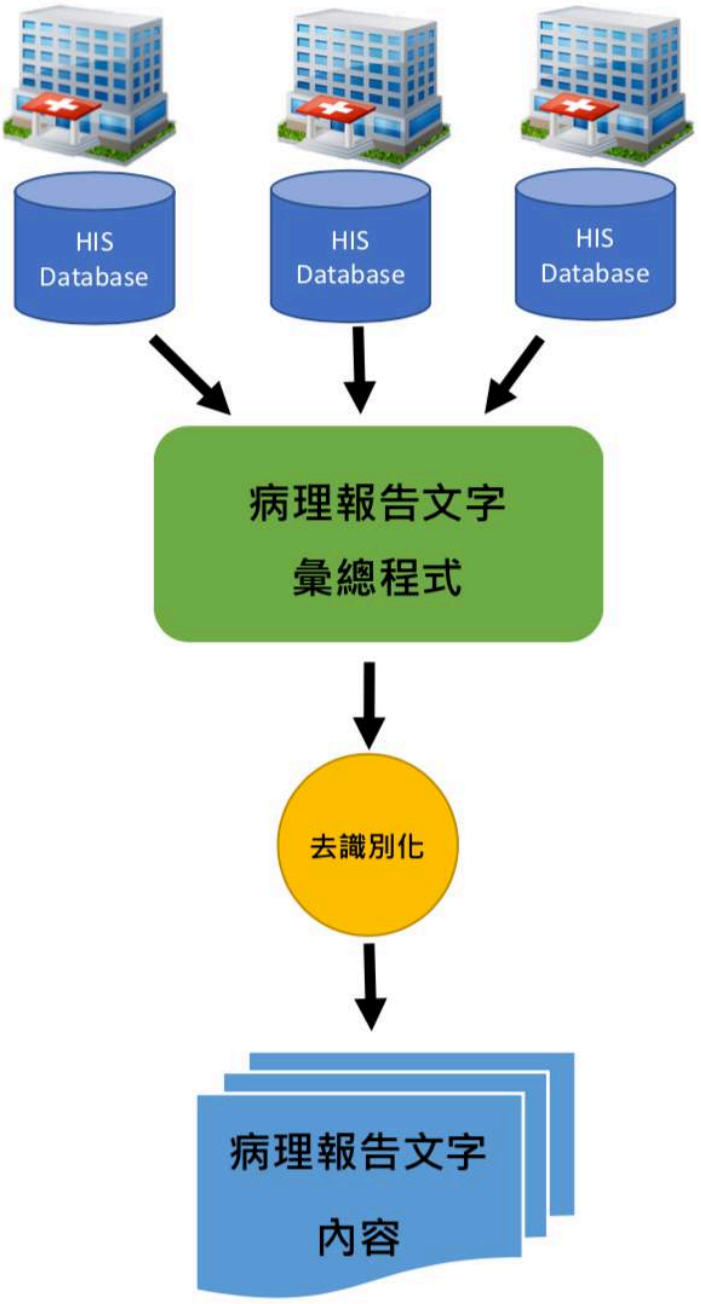


FIGURE 3.3: 資料擷取模組

首先可看到利用病理報告文字彙總程式由各院區 HIS(Health Information System) 資料庫取得多筆病理報告文字並匯總。彙總後將文字資料去識別化，已確定病理報告無法看見特定患者訊息。

3.1.4 標籤模組

標籤模組如 Figure 3.4，主要進行每份去識別話病理報告文字各 MCODE、TCODE、病理報告結果標籤。

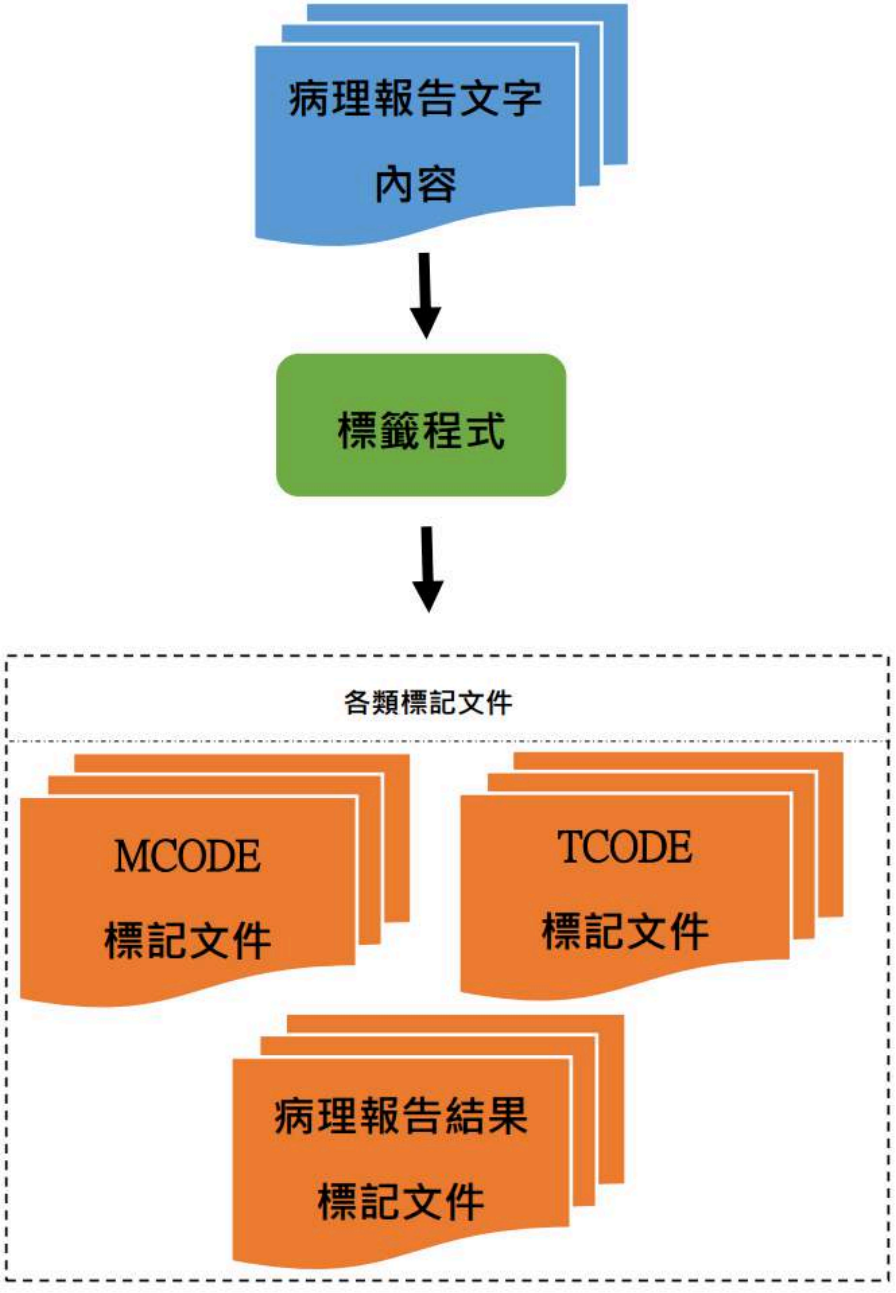


FIGURE 3.4: 標籤模組

此模組將匯總病理報告文件複製成三份，於各病理報告文字內容前方標記 MCODE、TCODE、病理報告結果，以便下一模組將其類別各自分類。

3.1.5 自然語言處理模組

自然語言處理模組如 Figure 3.5主要的目的就是去除在病理報告文件中的固定字或多餘字，並將文字切割以便特徵選取 (Feature Selectionu)、分類個項目以及訓練與測試之用。模組包括兩大處理：篩選程式 (Filter Processing)，另一為斷字切割程式 (Segmentation Processing)。

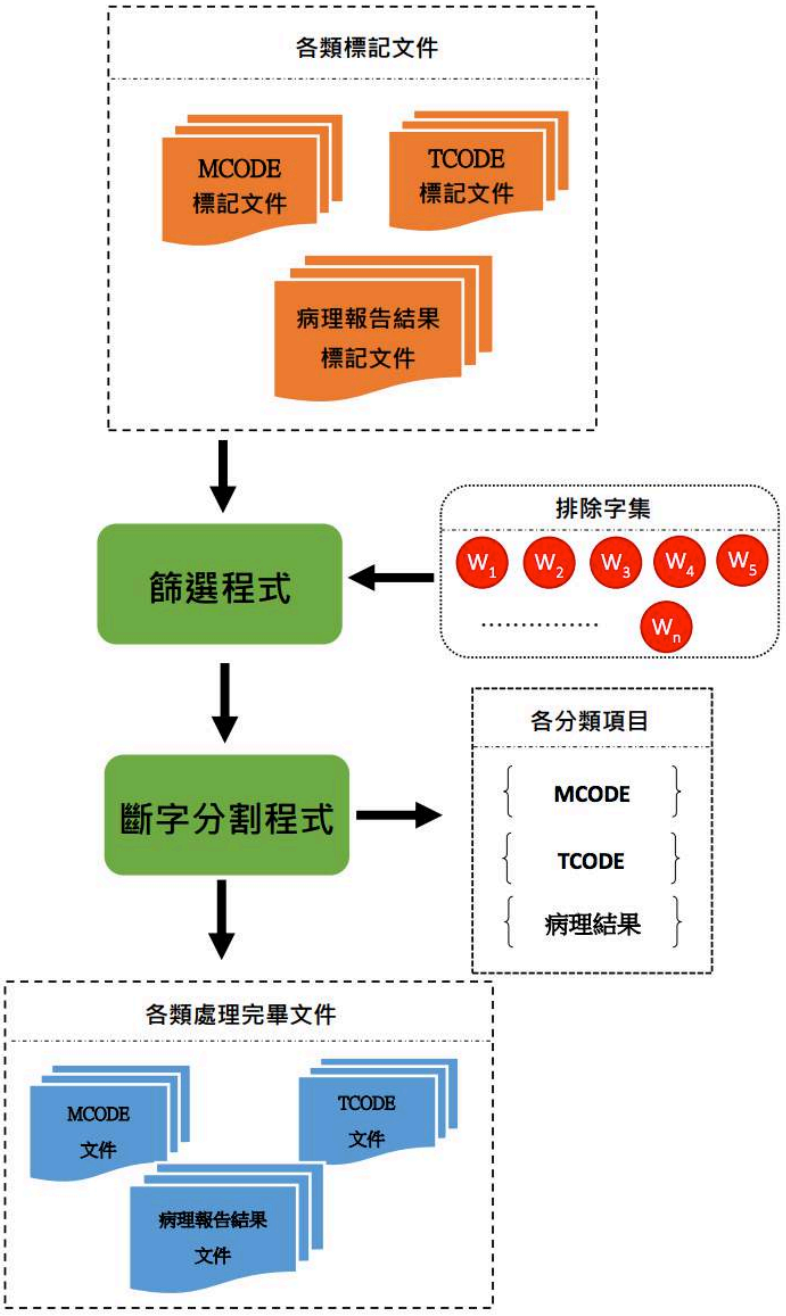


FIGURE 3.5: 自然語言處理模組

由於原本病理報告內容總是會含有一些既定格式或直接影響分類之準確率之文字，例如醫師簽名、病理序號、標題... 等如 Table 3.1，因此使用篩選程式 (Filter Processing) 將多餘或影響準確度的字去除，接著使用斷字分割程式 (Segmentation Processing) 將先前標籤模組之 MCODE、TCODE、病理結果分割使其成為各分類項之目標並將各文件的文字全部切成一個一個字，執行上續兩種程式後，文件中都拆解成有效的文字。

TABLE 3.1: 影響排除對照表

病理排除文字例表	
Pathologic diagnosis	Pathological diagnosis
Gross description	Microscopic description
Comment	Dr. xxxx (簽名)
Maligncancy	Infection
Pathological No.	....

經過標籤模型與自然語言處理模型後，每筆病理報告資料定義 (Doc)，於標籤模組複製出標記之文件集合 (C) 可表示為 $C = \{Doc_1, Doc_2, Doc_3, ..., Doc_n\}$。由於透過標籤每個 Doc 都含有兩個重要的要素：其一為各 MCODE、TCODE 診斷碼 $(mcode)(tcode)$ 或病理結果 (rpm)，其二為病理報告 $(report)$，因此 Doc 定義為 $Doc_n = \{mcode_n, report_n\}$，而病理報告 $(report)$ 透過自然語言模組處理後形成許多有效文字 $(Word)$，則 $(report)$ 可以再次定義為 $report_n = \{w_{n1}, w_{n2}, w_{n3}, ..., w_{nm}\}$，其中 w_{nm} 表示第 n 篇病理報告中的第 m 個字。

3.1.6 訓練與測試模組

訓練與預測模組，主要將病理報告中取得特徵加以使用分類器來訓練測試如 Figure 3.6。

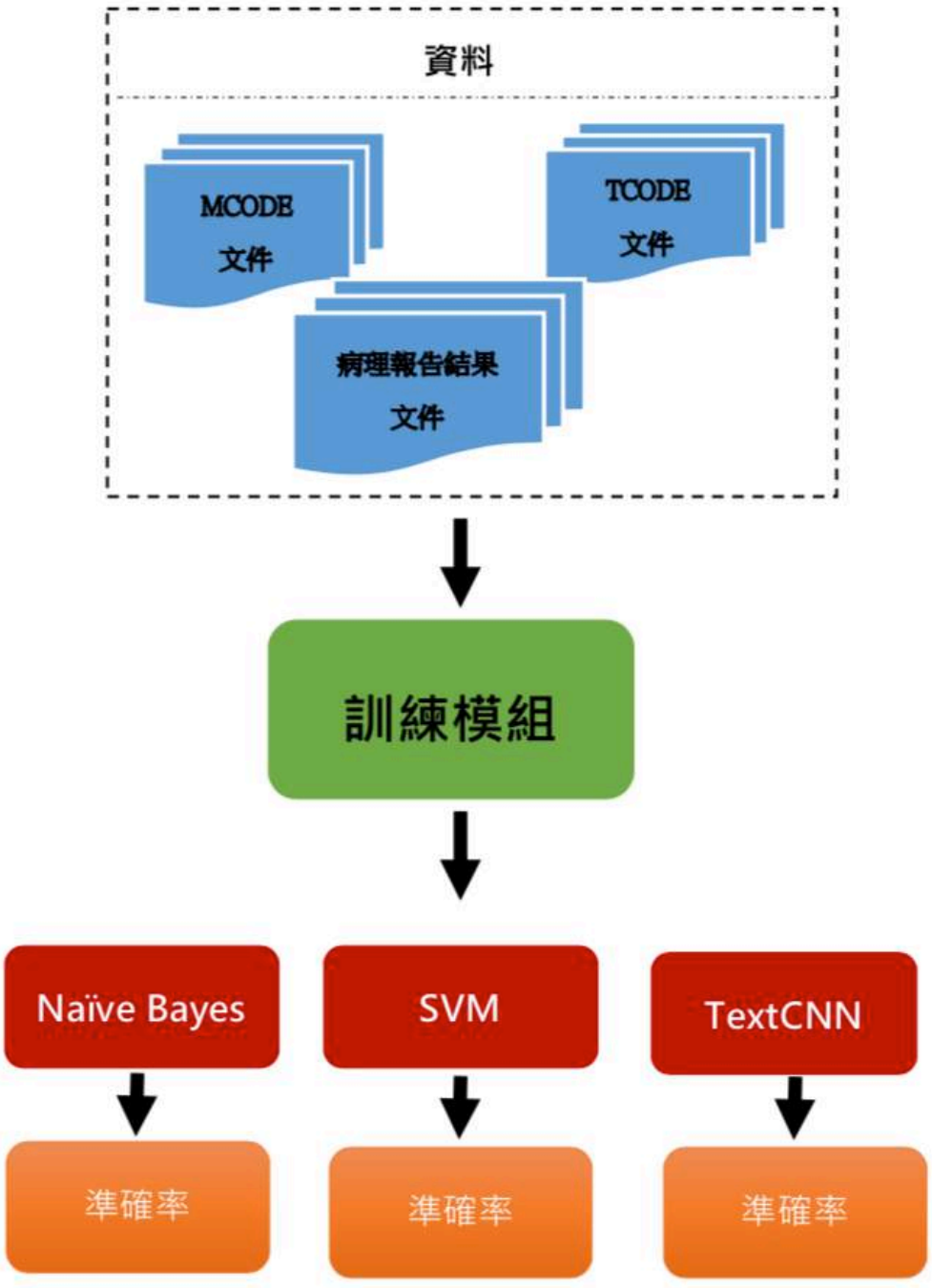


FIGURE 3.6: 訓練與測試模組

將以兩種形式來執行傳統式及深度學習方式，傳統擷取特徵方式使用 TF-IDF 方式並選用兩種分類器 Naive Bayes 及 SVM，深度學習方式使用擷取特徵 N-Grams 分類器使用 Text-CNN，下一段落將會細說其處理過程。

傳統方式以 TF-IDF 方式取得特徵權重如 Figure 3.7其公式如下所示。

$$TF-IDF_{i,j} = TF_{i,j} \times IDF_i \text{，(3-1)} \quad IDF_i = log\frac{|D|}{1+|j:t_i \in d_j|} \text{，(3-2)}$$

其中 $TF_{i,j} \times IDF_i$ 代表每一個字計算後的 TF-IDF 結果值，$|D|$ 代表病理報告總件數，$|j : t_i \in d_j|$ 代表該 word 在病理報告出現篇數。假設目前共有兩篇病理報告，其中內容分別是 $report_1 = \{w_a, w_b, w_c, w_d, w_c, w_b, w_e, w_a\}$，$report_2 = \{w_d, w_c, w_b, w_b\}$，若要計算 w_a 的 TF-IDF 值，首先必須算出 IDF_i，由於一共兩篇病理報告，因此 $|D| = 2$，再者因 w_b 共出現在兩篇文中 $(report_1)$ 中，因此 $|j : t_i \in d_j| = 1$，所以 $IDF_i = log(\frac{2}{1}) = 0.301$，則 $TF_{i,j} \times IDF_i = 0.25 \times 0.301$，及 w_a 的 TF-IDF 值為 0.0775。

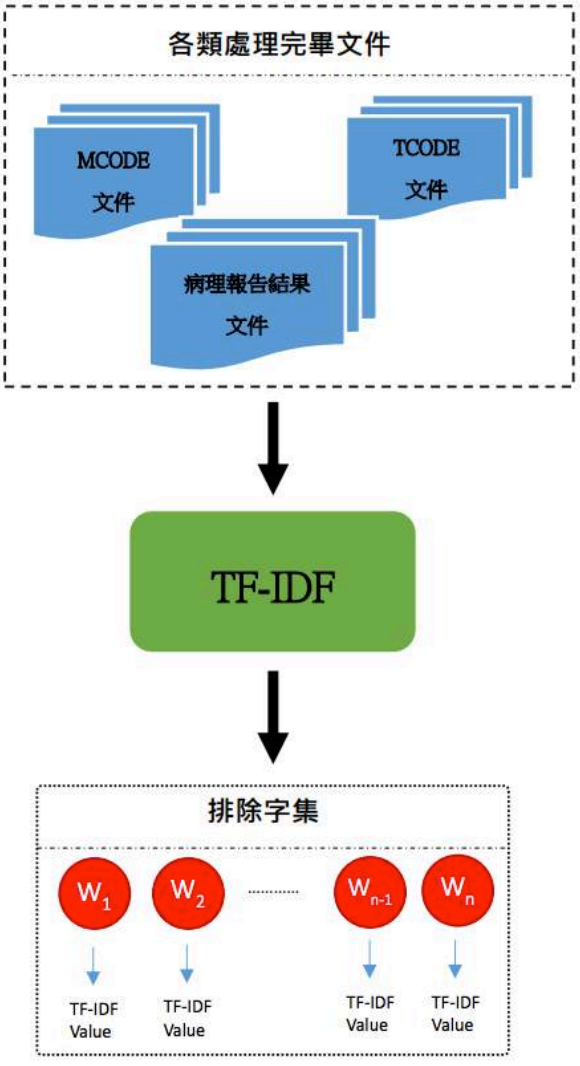


FIGURE 3.7: TF-IDF 取得特徵權重

以 IF-IDF 確定病理報告文字是否需要後，將其放入 Naive Bayes 及 SVM 中訓練並預測其 MCODE、TCODE、病理報告結果，之準確率。

深度學習部分使用 Text-CNN 來做分類，以下將分 2 部分解釋運行過程：

1. 將病理報告轉向量過程：

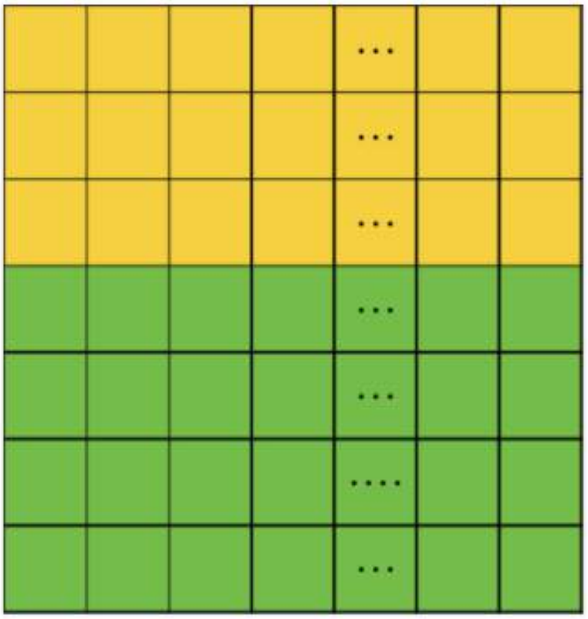
   假設句子向量最大長度 (maxSentenceLength) = 56，不足長度補 0，其原因為 CNN 的輸入需要一個統一維度，maxSentenceLength 可以看成 image 的高。病理報告文件 (X_text) 中有很多病理報告 (doc) 其中有許多句子，舉例可表示 $X\_text = [\{'W_a \;\; W_b \;\; W_c'\}, \{'W_d \;\; W_e'\}, ..., \{'W_f \;\; W_g \;\; W_c'\}]$，將其轉換成向量字典 (dic)，呈現為 $dic = \{W_a : 1, W_b : 1, W_c : 3, ...\}$，則 X_text 可轉換成下列句子向量 $[[1 \;\; 2 \;\; 3 \;\; 0 \;\; .. 0], [4 \;\; 5 \;\; 0 \;\; 0 \;\; .. 0],$ $[m-1 \;\; m \;\; 3 \;\; 0 \;\; .. 0]]$，而有新的 doc 例如 $'W_a \;\; W_d \;\; W_e'$ 則轉換成 [1 4 5 0 ..0]，句子轉換向量後講其中單字轉向量建構一個 $M \times embeddingSize$ 大小的隨機矩陣。其中 M 是字典 dic 的大小 $embeddingSize$ 是詞向量的位數。若設定 128 則轉換如下 Figure 3.8。

index	vocab	W_{m-128}
0	W_a	[[-0.68702912 0.77074695 ..., -0.58884251 0.56960434]
1	W_b	[-0.35301754 -0.74718404 ..., 0.8568539 -0.97576588]
2	W_c	[-0.67536247 0.76219553 ..., -0.5886898 0.56818742]
3	W_d	[0.41945928 -0.25878668 ..., 0.26236984 0.52249086]
4	W_e	[-0.10533118 0.50658941 ..., 0.87267232 0.16990113]
5	W_c	[0.41945928 -0.25878668 ..., 0.26236984 0.52249086]
....	....	....
M-1	W_g	[-0.45372915 0.4625392 ..., -0.5167408 0.13936591]
M	W_h	[-0.0376482 0.22046733 ..., 0.84804225 -0.57162309]]

FIGURE 3.8: 向量轉換隨機矩陣

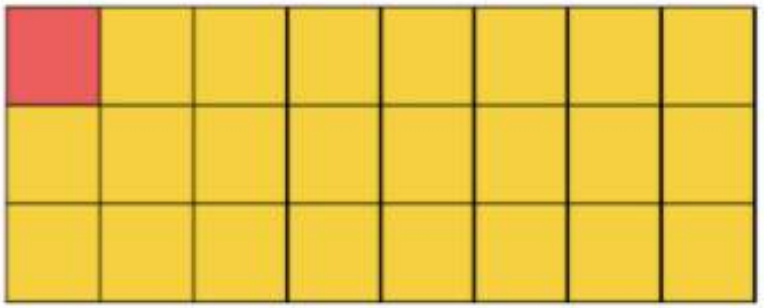
2. 卷積過程：

   先將 Figure 3.8 簡化表示為下圖 Figure 3.9，並設定卷積需要之卷積核 Figure 3.10 其中可看到 filter_size 設定為 3，而文字的 filter 和 N-Grams 極為相似，設定完畢後開始執行卷積其操作如 Figure 以下將詳加說明：



$$\text{input} = (\text{batchsize} \times \text{width} \times \text{height})$$
$$= (1 \times 128 \times 56)$$

FIGURE 3.9: 簡化隨機矩陣呈現



$$W = [\text{filter\_size}, \text{embedding\_size}]$$
$$= [3, 128]$$

FIGURE 3.10: 卷積核

(1) 紅色方框內是一個 patch, 第一個 patch 和卷積核的乘積對應 conv 列向量的第一個元素，patch 以上倒下滑動，得到完整的 conv。

(2) 給卷積操作的輸出 conv 增加偏置 b，得到 con_b。

(3) con_b，池化處理後，得到最終結果 con_b_pool，其中 con_b_pool 的解釋為一個卷積操作，即對輸入樣本進行了一次特徵映射 (feature_map) 而一次 feature_map 得到關於輸入一個特徵，而舉例模型中，一共有 $128 \times 3 = 384$ 個卷積操作，所以將會得到輸入的 384 個特徵，即可用取得之特徵進行預測，最後將其特徵送入 softmax 進行分類公式如下。Figure 3.11

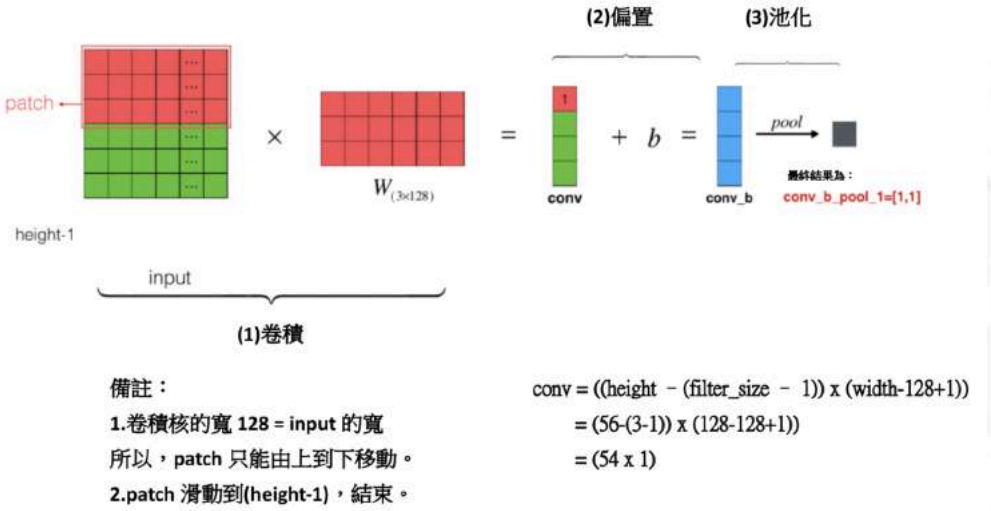


FIGURE 3.11: 卷積過程

3.2 實驗方法

3.2.1 本文分類測試

本論文利用機器學習來強化病理報告系統，以提升病理診斷及結果分類之成效，將實驗傳統機器學習與深度學習於病理報告系統診斷及結果分類，是否具有差異性，以下將說名其資料來源、實驗項目、及評估項目。

3.2.1.1 資料來源

本論文之樣本以某區域醫院之病理報告，資料樣本期間為：2014 年 12 月至 2017 年 1 月。由於病理診斷及結果是由病理醫師根據顯微鏡下所檢查之最終結

果，在登打報告描述並自行選則相對應的診斷及結果，本論文只取診斷及結果進行實驗，總報告筆數約 2 萬 7 千筆，其中M CODE 和 TCODE 筆數為 7370 筆如 Table 3.2、Table 3.3，其中O、I 病理結果不會標記病理報告之診斷。

Table 3.2: MCODE 資料來源

	O	B	U	M	P	I	總件數
MCODE	-	3799	182	2731	658	-	7370
無診斷	19697	-	-	-	-	39	19736
總件數	19697	3799	182	2731	658	39	27106

Table 3.3: TCODE 資料來源

	O	B	U	M	P	I	總件數
TCODE	-	3799	182	2731	658	-	7370
無診斷	19697	-	-	-	-	39	19736
總件數	19697	3799	182	2731	658	39	27106

3.2.1.2 實驗項目

本實驗將機器學習應用於病理報告系統之分析，已無人工介入篩選特徵方式，來分析診斷及結果之分類。以傳統較為成功機器學習的兩種分類器 Naive Bayes 及 SVM 與現今較流行深度學習方法 Text-CNN 作為實驗，而各分類方法所對應使用擷取特徵 (Feature) 方法 Naive Bayes 及 SVM 使用 TF-IDF，而 Text-CNN 使用 N-Grame 其表格如 Table 3.4

Table 3.4: 實驗項目

實驗變數	方法論		
	Naive Bayes	SVM	Text-CNN
TF-IDF	F-Measure	F-Measure	-
N-Grame	-	-	F-Measure

3.2.1.3 評估項目

此階段研究最終目的就是取得分類的準確率用以評估病理報告診斷及結果之成效，故採用 Precision、Recall 和 F-measure 方法作為評估指標，則下三項目解釋 Precision、Recall 和 F-measure 公式：

(1) 精確率 (Precision)：預測分類中預測正確的機率。公式如 Figure 3.12

$$\text{Precision} = \frac{TP}{TP+FP} \times 100\%$$

FIGURE 3.12: 精確率 (Precision)

(2) 召回率 (Recall)：實際分類中預測正確的機率。公式如 Figure 3.13

$$\text{Recall} = \frac{TP}{TP+FN} \times 100\%$$

FIGURE 3.13: 召回率 (Recall)

(3) F-measure：由於 Precision 與 Recall 通常會有一直偏高另一值偏低的情形，而 F-measure 就是取得兩者的平衡。公式如 Figure 3.14

$$\text{F-measure} = \frac{2 \times precision \times recall}{precision + recall}$$

FIGURE 3.14: F-measure

最高 F-measure 分類器將另外使用混淆矩陣 (Confusion Matrix)、ROC 曲線 (Receiver Operating Characteristic Curve) 作為不同面向之評估指標，下列為各項評估指標解釋：

(1) 混淆矩陣 (Confusion Matrix)：混淆矩陣又稱誤差矩陣，是表示精度評價的一種標準格式，用 n 行 n 列的矩陣形式來表示，如 Figure 3.15，其中定義有四種預測屬性：

		Prediction Class	
		YES	NO
Actual Class	YES	True Positive (TP)	False Negative (FN)
	NO	False Positive (FP)	True Negative (TN)

FIGURE 3.15: 混淆矩陣

- 真陽性 (TP,True Positive)：肯定正確，又稱：命中 (hit)
- 偽陽性 (FP,False Positive)：錯誤的肯定，又稱：假警報 (false alarm)
- 真陰性 (TN,True Negative)：正確的否定，又稱：正確拒絕 (correct rejection)
- 偽陰性 (FN,False Negative)：錯誤的否定，又稱：未命中 (miss)

(2) ROC 曲線 (Receiver Operating Characteristic Curve)：ROC 曲線經常被使用來顯示分類器的 TP-rate(True positive rate) 與 FP-rate(False positive rate) 的損益平衡狀況，X 軸表示 FP-rate、Y 軸表示 TP-rate，如 Figure 3.16顯示三個分類器單次分類的結果，(0,0) 代表分類器必定不會將資料分類到目標樣本，(1,1) 則表示分類器必定會將資料分類到目標樣本上，而 (0,1) 代表是一個完美的分類器。

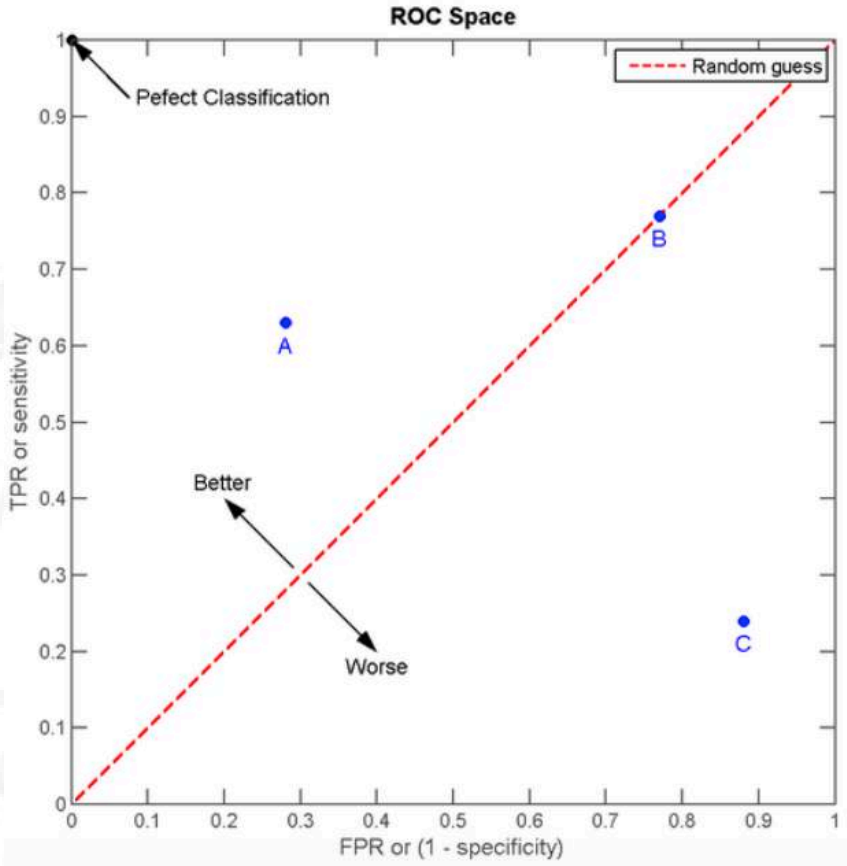


FIGURE 3.16: 三個獨立的分類器在 ROC 上

(3) RP 曲線 (Precision-Recall Curve)：精準率為 y 軸，召回率為 x 軸如 Figure 3.17，精確率越高，召回率越高，PR 曲線與 ROC 曲線的相同點是都採用了 TP-rate，而 PR 曲線使用了 Precision，因此 PR 曲線的兩個指標都聚焦於正例。類別不平衡問題中由於主要關心正例，所以在此情況下 PR 曲線被廣泛認為優於 ROC 曲線。

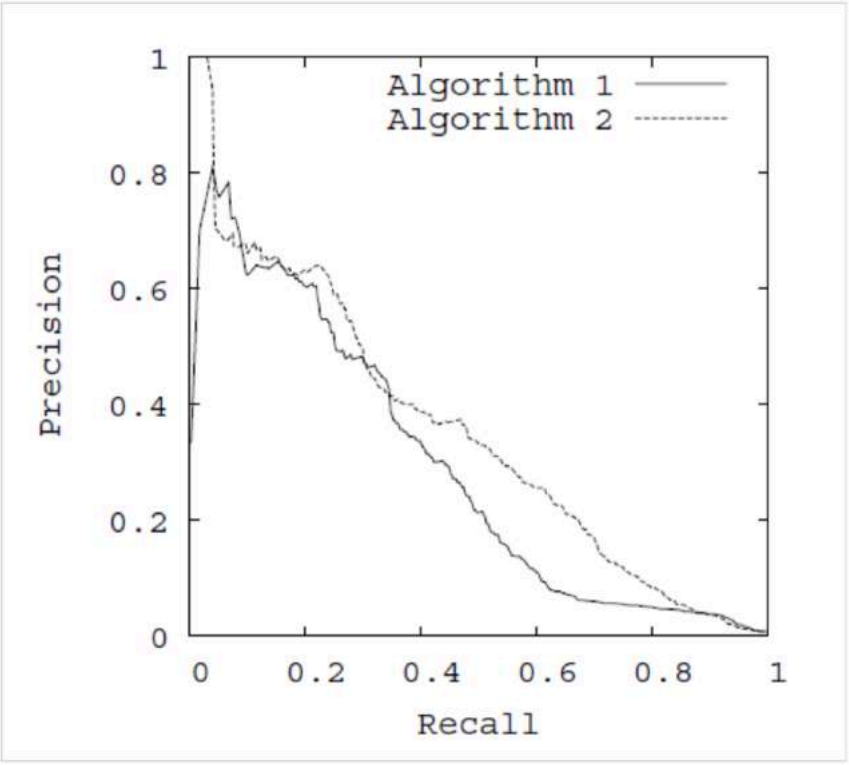


FIGURE 3.17: RP 曲線示意圖

3.2.2 模型與客戶端測試

本論文另一實驗取得準確率較高之訓練模型後，將模型放置雲端建立服務供各客戶端呼叫使用，客戶端將會有其作業系統 Window XP 32 位元、Window 7 32 位元，以固定樣本數執行，模型呼叫之速度檢測如 Figure 3.18

FIGURE 3.18: 模組與客戶端測試

Chapter 4

實驗結果

4.1 實驗環境

第一個實驗環境主要建立於 Figure 3.1所示測試訓練端，以單機電腦建置實驗環境，主要功能為各院區資料擷取整理、模型訓練並比較不同之模型之準確率，其軟硬體系統及套件版本如 Table 4.1所示，第二個實驗比較完成較為優秀之模型後放置於雲端虛擬機環境提供雲端服務，供各客戶端使用並評估使用之效率。其模型放置端與客戶端軟硬體系統及套件版本如 Table 4.2、Table 4.3、Table 4.4所示。

Table 4.1: 測試訓練端軟體配置

測試訓練端 (實體機)	
作業系統	Window10 64x
使用程式語言	Python v3.5
	C# 6.0
訓練測試框架	Scikit-learn 0.19.1
	Tensorflow v1.4
Framework	Microsoft .NET Framework 4.5

Table 4.2: 模組放置軟體配置

模型放置端 (虛擬機)	
作業系統	Ubuntu 16.04.3 64x
使用程式語言	Python v3.5
訓練測試框架	Scikit-learn 0.19.1 Tensorflow v1.4
API 建構套件	Flask-RESTful 0.3.1

Table 4.3: 客戶端配置 (clientA)

客戶端 (實體機)	
作業系統	Window XP 32x
使用程式語言	VB.NET
使用軟體	病理報告系統
Framework	Microsof .NET Framework 4.0

Table 4.4: 客戶端配置 (clientB)

客戶端 (實體機)	
作業系統	Window W7 32x
使用程式語言	VB.NET
使用軟體	病理報告系統
Framework	Microsof .NET Framework 4.5

而此架構中作業系統大不相同由測試訓練端為W indow10，模型放置端Ubuntu 16.04.3，客戶端W indow XP，訓練完後利用資料夾共享放置對應雲端位置，再到雲端放置端點建立 RESTful API 客戶端可執行此服務，然而撰寫C#、VB.NET 程式需使用 API 功能則需引用 Framework4.0 以上，所以可看到客戶端都必須提升 Framework 版本。

4.2 實驗結果

4.2.1 Naive Bayes 實驗結果

透過 TF-IDF 取得特徵數為 9,618 個，各分類總數 MOCDE 為 328、TCODE 為 186、病理結果為 6，則以列項目為 Naive Bayes 之 MOCDE、TCODE、病理結果之實驗結果：

(1) 診斷碼 MCODE：其實驗結果發現之最高準確度為 65% 如 Figure 4.1所示，精準度為 54%、召回率為 63%、F1-Measure 為 55%

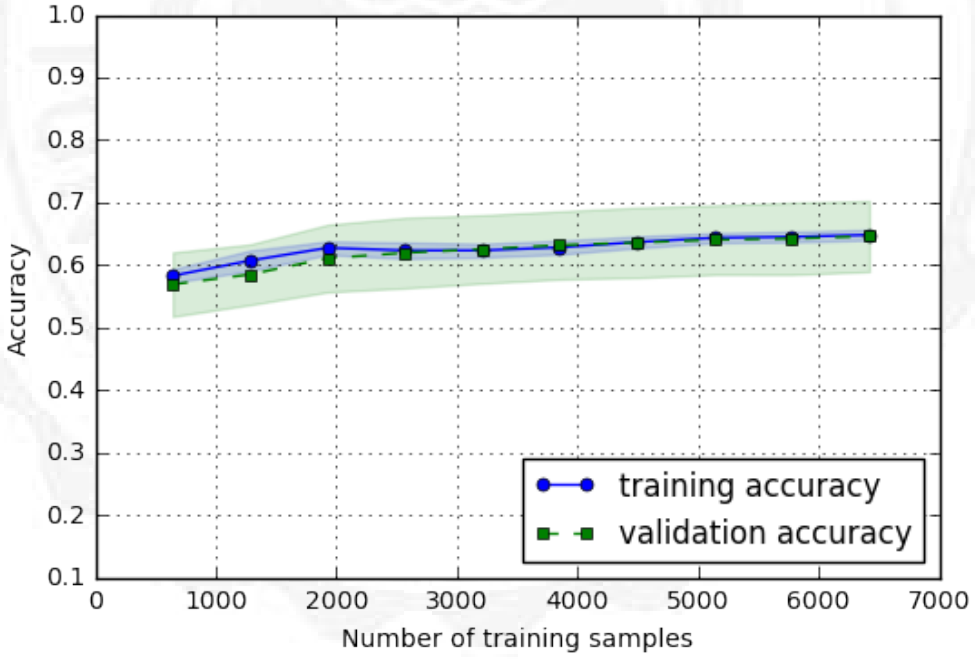


FIGURE 4.1: Naive Bayes MCODE 準確度

(2) 診斷碼 TCODE：其實驗結果發現之最高準確度為 58% 如 Figure 4.2所示，精準度為 53%、召回率為 58%、F1-Measure 為 51%

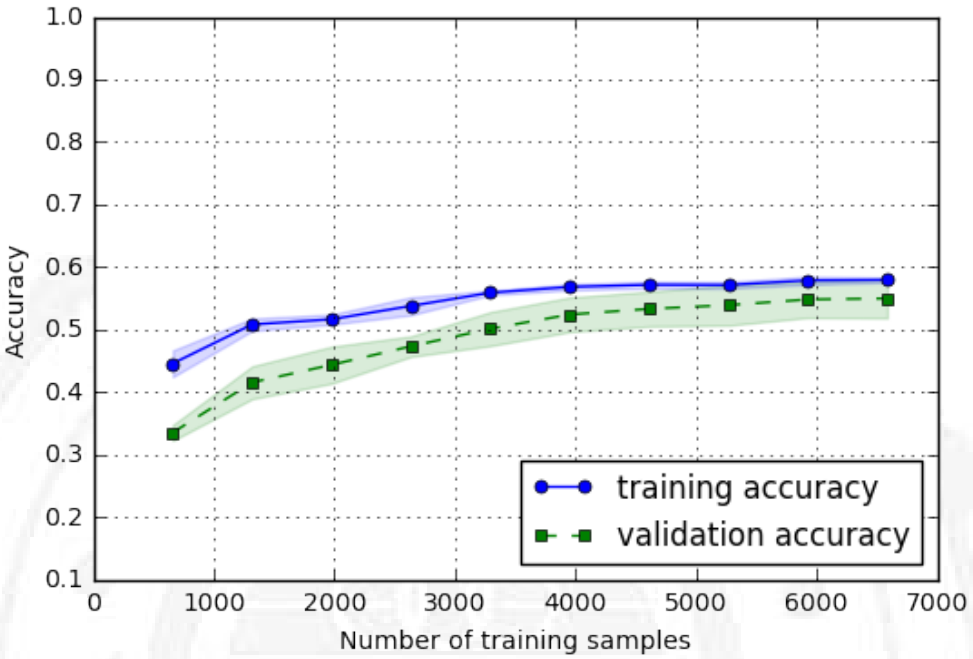


FIGURE 4.2: Naive Bayes TCODE 準確度

(3) 病理結果：其實驗結果發現之最高準確度為 95% 如 Figure 4.3所示，精準度為 94%、召回率為 95%、F1-Measure 為 94%

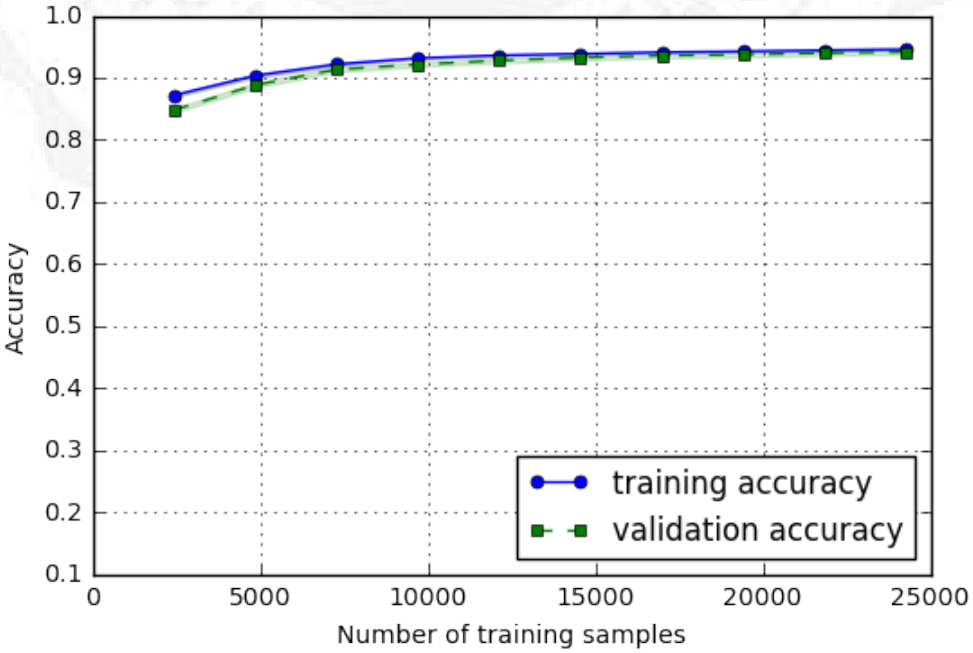


FIGURE 4.3: Naive Bayes 病理結果準確度

由上列實驗可看出 Naive Bayes 應用於病理報告系統於 MCODE、TCODE 的 F1-Measure 都不超過 55%，可見此方法可能不適合應用於此病理分類較多之場景，而病理結果卻高達 94% 其結果令人滿意。

4.2.2 SVM 實驗結果

透過 TF-IDF 取得特徵數為 9,618 個，各分類總數 MOCDE 為 328、TCODE 為 186、病理結果為 6，則以列項目為 SVM 之 MOCDE、TCODE、病理結果之實驗結果：

(1) 診斷碼 MCODE：其實驗結果發現之最高準確度為 95% 如 Figure 4.4所示，精準度為 94%、召回率為 95%、F1-Measure 為 94%

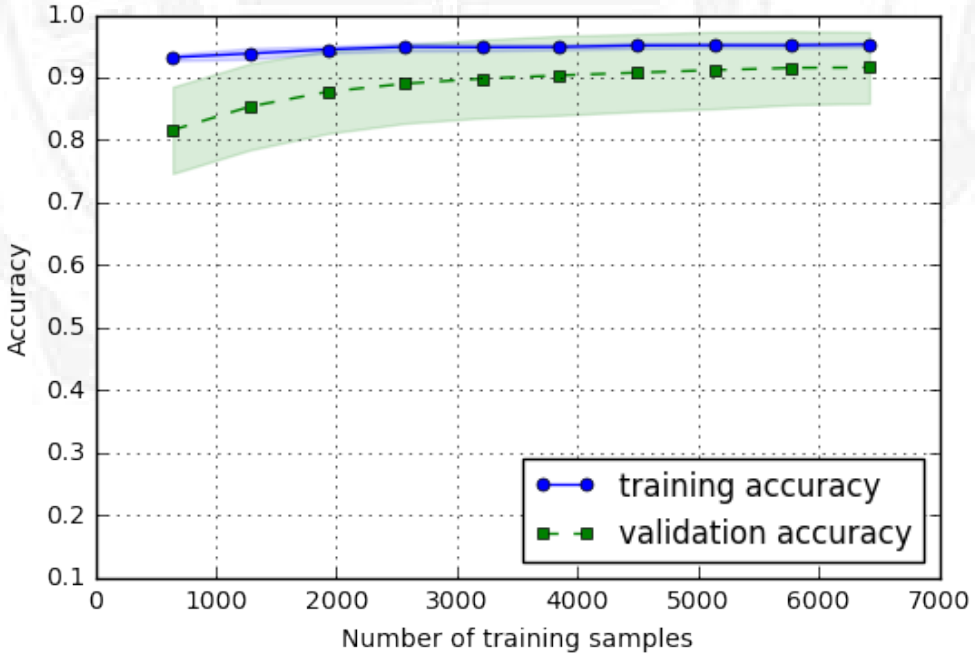


Figure 4.4: SVM MCODE 準確度

(2) 診斷碼 TCODE：其實驗結果發現之最高準確度為 94% 如 Figure 4.5所示，精準度為 93%、召回率為 94%、F1-Measure 為 94%

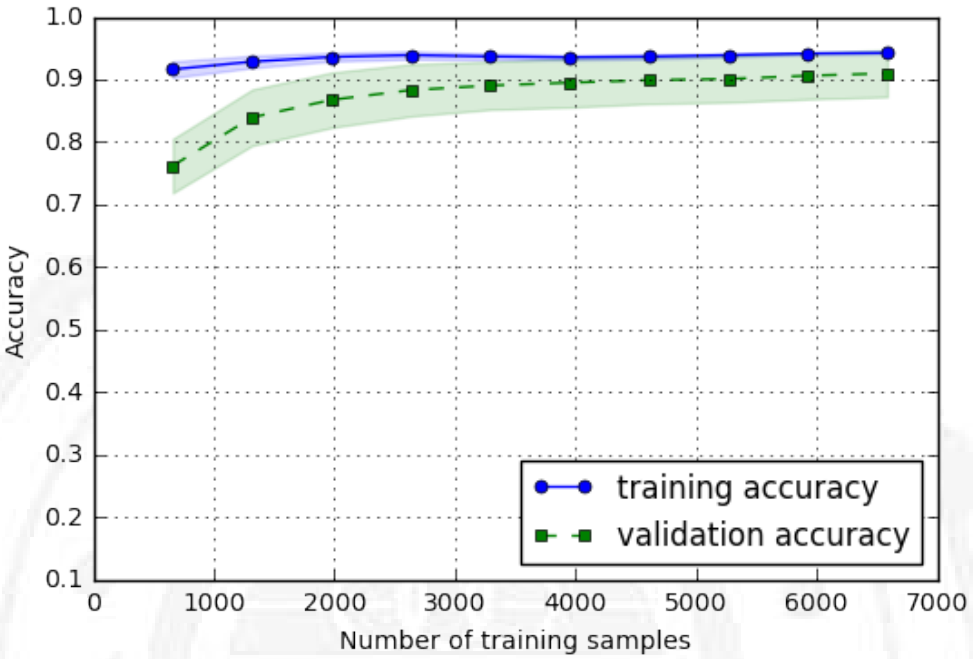


FIGURE 4.5: SVM TCODE 準確度

(3) 病理結果：其實驗結果發現之最高準確度為 98% 如 Figure 4.6所示，精準度為 98%、召回率為 98%、F1-Measure 為 98%

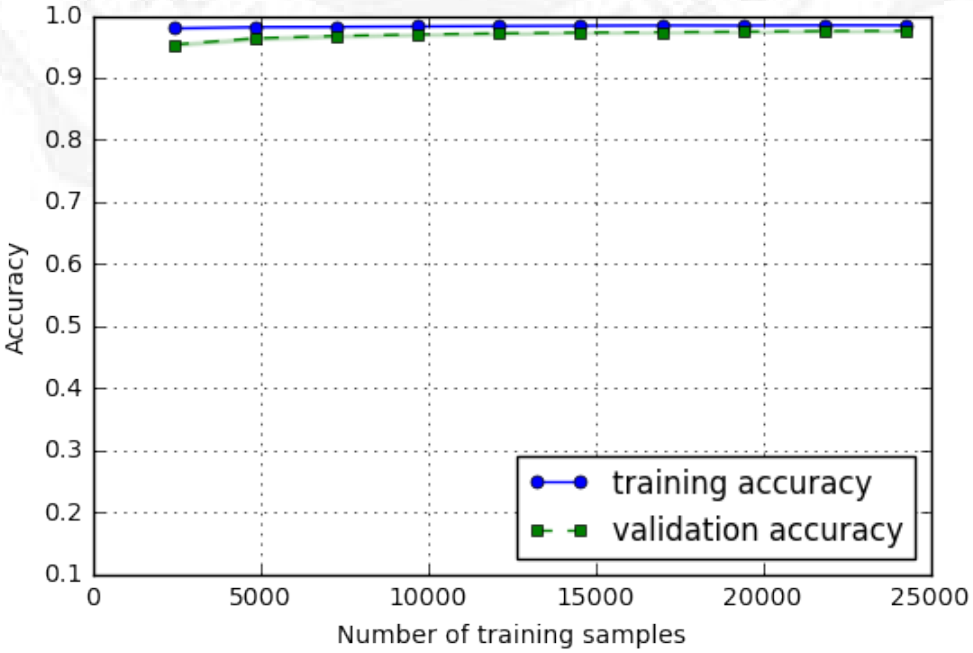


FIGURE 4.6: SVM 病理結果準確度

由上列實驗可看出 SVM 應用於病理報告系統於 MCODE、TCODE、病理結果的 F1-Measure 精準度都高於 90% 顯然比起 Naive Bayes 更為適合此場景。

4.2.3 Text-CNN 實驗結果

下列項目為 Text-CNN 之 MOCDE、TCODE、病理結果之實驗結果：

(1) 診斷碼 MCODE：其實驗結果發現之最高準確度為 91%，最低 loss 為 0.3 如 Figure 4.7所示，精準度為 89%、召回率為 91%、F1-Measure 為 90%

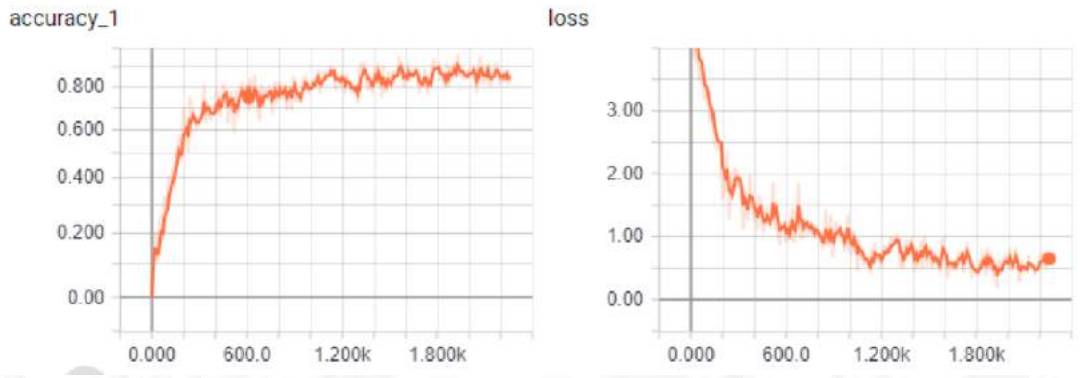


FIGURE 4.7: Text-CNN MCODE 準確度及 loss

(2) 診斷碼 TCODE：其實驗結果發現之最高準確度為 85%，最低 loss 為 0.5 如 Figure 4.8所示，精準度為 85%、召回率為 86%、F1-Measure 為 85%

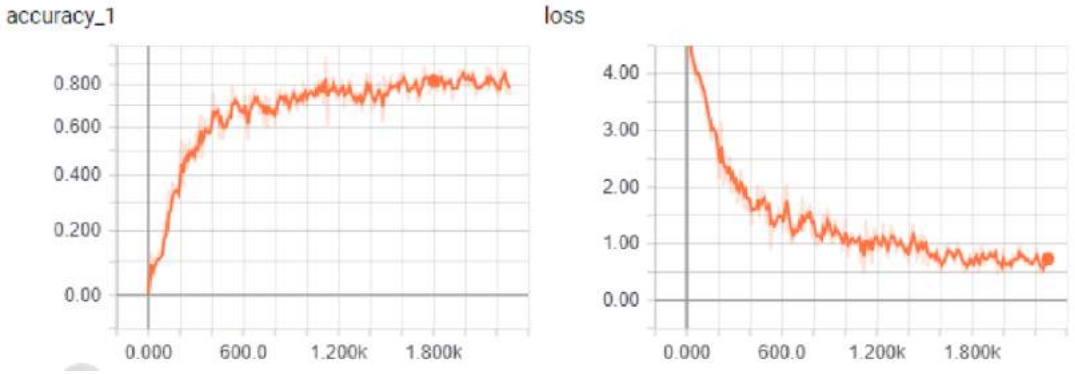


FIGURE 4.8: Text-CNN TCODE 準確度及 loss

(3) 病理結果：其實驗結果發現之最高確度為 98%，最低 loss 為 0.06 如 Figure 4.9所示，精準度為 98%、召回率為 98%、F1-Measure 為 98%

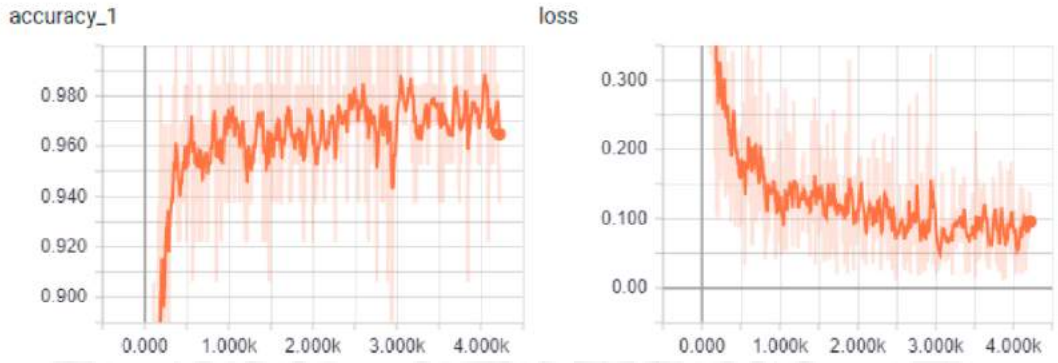


FIGURE 4.9: Text-CNN 病理結果準確度及 loss

4.2.4 分類器比較結果

下列總 Figure 4.10可看出各分類器對於病理結果之準確率並無太大的差別其原因為分類項目少且取得項特徵數眾多，而 MCODE、TCODE 很明顯看到其比較順序為 SVM>Text-CNN>Naive Bayes，而原預定 Text-CNN 將會比 SVM 分類器優秀。

	Naïve Bayes			SVM			TextCNN		
項目	精準度	召回率	F1-Measure	精準度	召回率	F1-Measure	精準度	召回率	F1-Measure
MCODE	0.54	0.63	0.55	0.94	0.95	0.94	0.89	0.91	0.9
TCODE	0.53	0.58	0.51	0.93	0.94	0.94	0.85	0.86	0.85
病理結果	0.94	0.95	0.94	0.98	0.98	0.98	0.98	0.98	0.98

FIGURE 4.10: 分類器結果總表

4.2.5 SVM 分類器探討

以下為 SVM 分類利用器混淆矩陣、ROC 曲線、RP 曲線實驗分析:

(1) 混淆矩陣 (Confusion Matrix)：以下使用病理結果來做實驗之呈現分兩總數值內容，樣本量及百分比 Figure 4.11、Figure 4.12，以混淆矩陣圖像化可看出一個錯誤分類就是病理結果X，發現其分類在M之結果，X 之類別為感染 & 惡性腫瘤則M為惡性腫瘤，其樣本數為 1:2731，由混淆矩陣圖示化呈現很明顯看出其錯誤，以百分比來做呈現可看出真實判定結果解釋較為直覺的如B 良性腫瘤、M 惡性腫瘤、P 癌前病變、O其他，判定比率都來到 92% 以上。

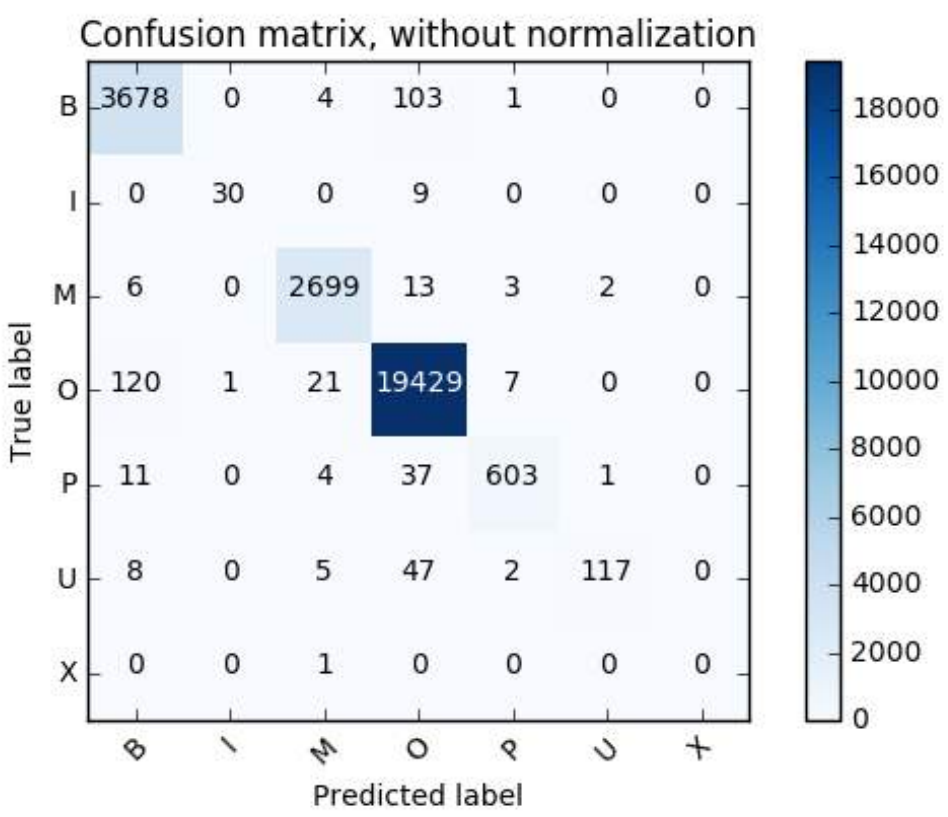


FIGURE 4.11: 混淆矩陣 (樣本數)

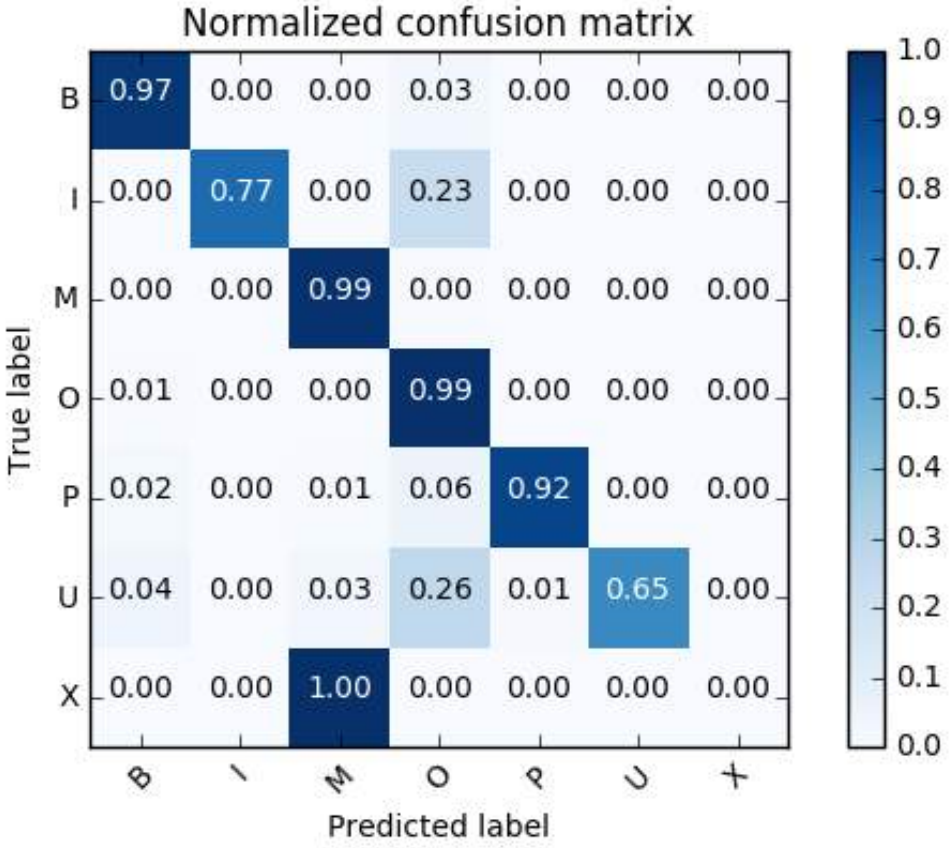


FIGURE 4.12: 混淆矩陣 (百分比)

(2) ROC 曲線 (Receiver Operating Characteristic Curve)：以下將病理結果分類使用 ROC 曲線 Figure 4.13之呈現，可看出此分類器用於病理結果判定上是相當不錯的，而 ROC 曲線越接近左上方代表越好，但是 ROC 曲線無法看出混淆矩陣所看出的錯誤，ROC 曲線只能辨定分類器好或不好。

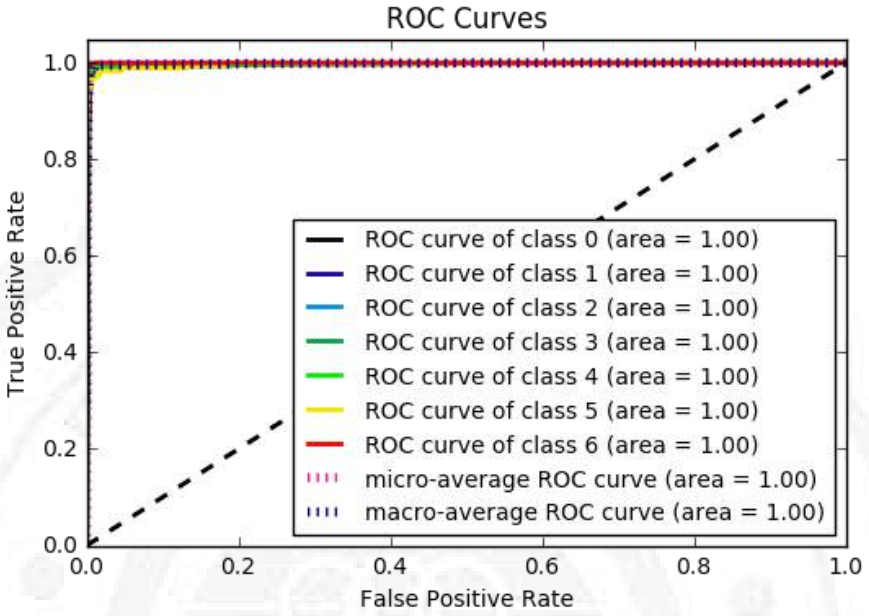


FIGURE 4.13: ROC 曲線

(3) RP 曲線 (Precision-Recall Curve)：以下將病理結果分類使用 RP 曲線 Figure 4.14之呈現其結果表現出較為貼近現實的分類狀況，另可看出此分類器用於病理結果判定上是相當不錯的，而 RP 曲線越接近右上方代表越好，但是還是不能透過此呈現發現類別 X 之錯誤，但其實很少結果會有等於 1 之結果答案，所以可判定其類別結果可能是有問題的。

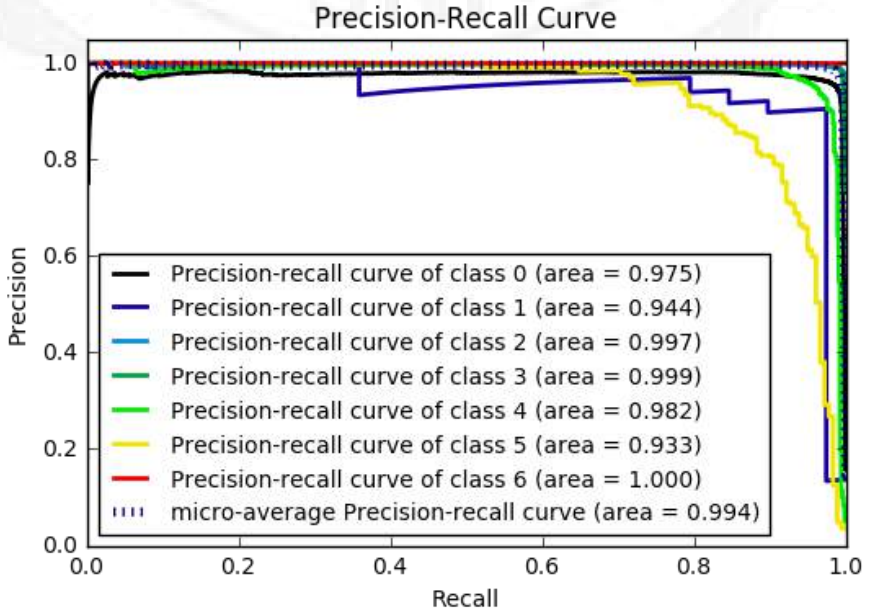


FIGURE 4.14: PR 曲線

4.2.6 模型與客戶端測試

將上一章節評估較為優良之分類模型放置雲端伺服器並建立服務，供不同的客戶端呼叫使用，而此實驗所測試之兩台客戶端其硬體規格如 Figure 3.1，作業系統分別為 WindowXP(client A) 及 Window7(client B)，以平均測試叫用 500 筆病理報告來觀察是否有速度之差異結果如 Figure 4.15、Figure 4.16，可看到 client A 客戶端叫用服務最低回應為 52 毫秒最高則為 78.13 毫秒，而 client B 客戶端叫用服務最低回應為 46 毫秒而最高回應為 59 毫秒，其中可看到 client A 平均 500 筆叫用平均速度為 55.296 毫秒而 clientB 則為 49.124 毫秒，可見硬體設備較用雲端服務是乎還是有些許差距，但用戶端感受不出來。

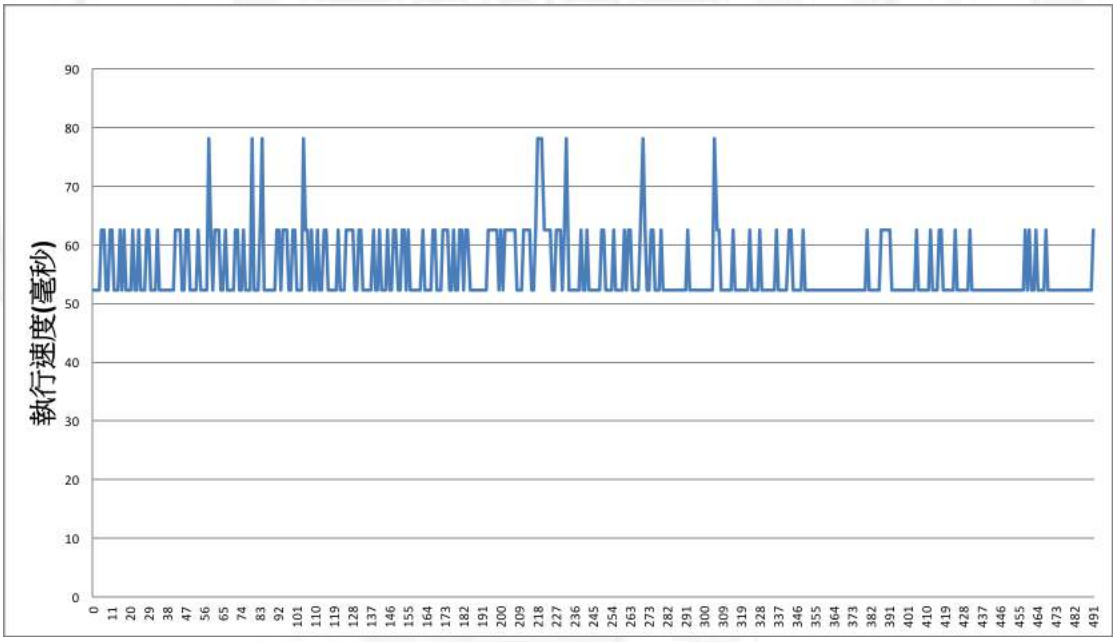


FIGURE 4.15: client A 執行速率

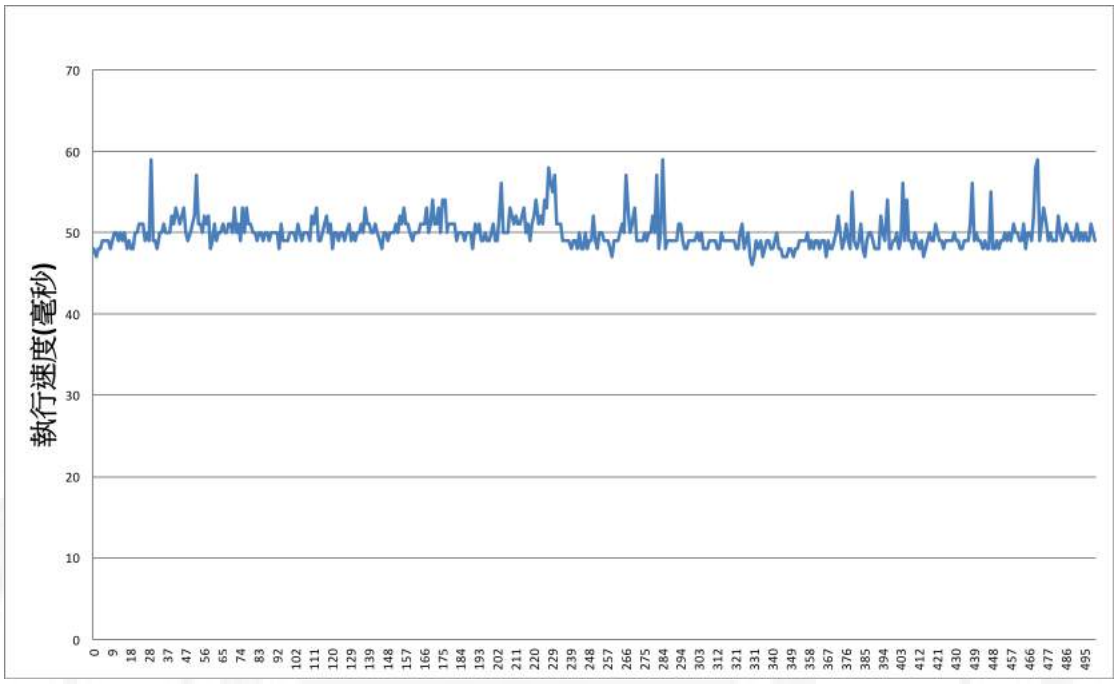


FIGURE 4.16: client B執行速率

Chapter 5

結論與未來方向

5.1 結論

機器學習的應用已經是大家耳熟能詳的技術，透過機器學習的應用，可以協助整理過去資料的規則性或發現平常沒有的規則。本論文主要以機器學習應用於病理報告系統，為系統提供自動結果與診斷分類，簡化病理醫師於登打報告後須決定相對應的診斷及結果並讓病理醫師可較快回應於臨床。第一部分實驗運用至病理報告系統至 E 化以來完整記錄之報告結果及診斷作為訓練來源，並以兩種傳統式機器學習及一種深度學習方式來比較分類器的優劣，發現傳統或深度學習之分類器對於病理結果部分差別並不大，而診斷分類可看出傳統比深度分類器優秀，其中診斷分類深度學習無法優於傳統機器學習有二項原因：多分類特定項目樣本數過少例如 MCODE 項目 8140/3 名稱為 Adencarcinoma, NOS，樣本數為 565 筆，8210/3 名稱為 Adencarcinoma in adenomatous polyp，樣本數為 1 筆，其相類似判斷時模型只能一直判斷為 8140/3 項目。特定分類項目樣本內容過度相似導致判斷失真如 Figure 5.1所示兩報告診斷由左上至下分別為 8330/3 Follicular carcinoma，樣本數為 10 筆、8341/3/3 Papillary microcarcinoma，樣本數為 12 筆，但報告格式太過類似導致分類趨近樣本數多的診斷，文字特徵也沒特別處理所導致，另外可將樣本再次改良將分類之樣本數量較少之分類排除可再次提高 Text-CNN 的準確度。第二部份因醫院環境較

為複雜須銜接較舊之作業系統及程式，故測試將訓練完成之模組放置雲端製作成雲端服務使用，並測試此方式使否有極大的效能差距，發現較舊之作業系統較用此服務還是有所差別，此差別在於啟動請求時有幾毫秒的差距，但使用者無明顯之感受。

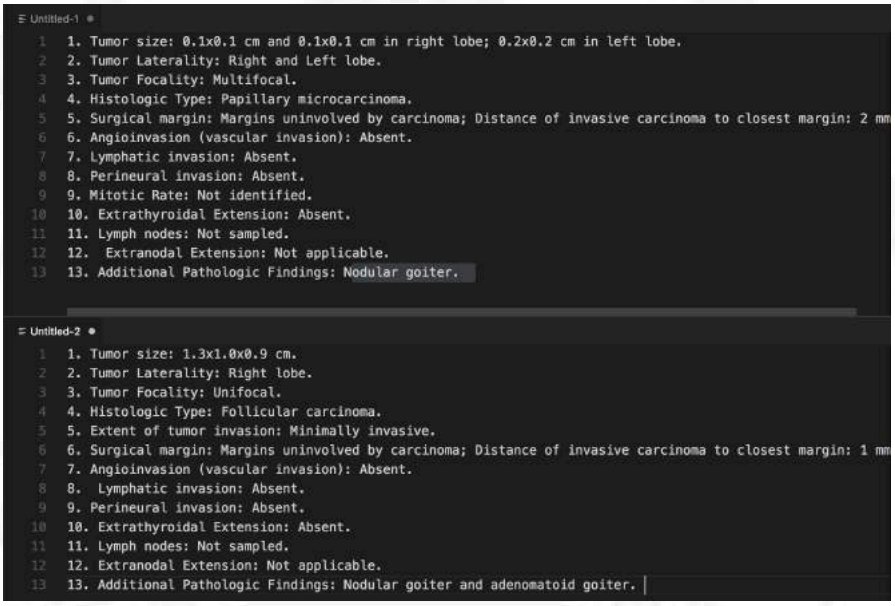


FIGURE 5.1: 報告內容過於類似

5.2 未來方向

本論文應用於醫院之病理系統是可行的，之後將會把過去 4 年尚未完整記錄於資料庫的病理報告透過此分類器回歸病理報告系統，並增設一新模組為回饋模組，供病理醫師可回饋其判定是否正確並給予意見，舉例說明若發現分類錯誤透過回饋機制輸入特定格式告知關鍵字及分類正確類別，另外如需移植推廣於其他醫院就必須重新學習病理報告，未來希望能提供此框架應用於醫院其他應用，此外此次分析並無使用人工選取或訓練特徵，故之後需改善其特徵取得模式，已讓分類更為準確，使深度學習更為完善。

參考文獻

[1] Fei-Yue Wang, Jun Jason Zhang, Xinhu Zheng, Xiao Wang, Yong Yuan, Xiaoxiao Dai, Jie Zhang, and Liuqing Yang. Where does alphago go: From church-turing thesis to alphago thesis and beyond. *IEEE/CAA Journal of Automatica Sinica*, 3(2):113–120, 2016.

[2] Varun Gulshan, Lily Peng, Marc Coram, Martin C Stumpe, Derek Wu, Arunachalam Narayanaswamy, Subhashini Venugopalan, Kasumi Widner, Tom Madams, Jorge Cuadros, et al. Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. *Jama*, 316(22):2402–2410, 2016.

[3] Madeleine Bates. Models of natural language understanding. *Proceedings of the National Academy of Sciences*, 92(22):9977–9982, 1995.

[4] Adhy Rizaldy and Heru Agus Santoso. Performance improvement of support vector machine (svm) with information gain on categorization of indonesian news documents. In *Application for Technology of Information and Communication (iSemantic), 2017 International Seminar on*, pages 227–232. IEEE, 2017.

[5] Eric Curea. Document retrieval and question answering in medical documents. In *Biomedical NLP Workshop*, pages 1–7, 2017.

[6] World Health Organization. Icd-o-3 多重原發性腫瘤分類標準, 2011.

[7] Robert Dale, Diego Mollá-Aliod, and Rolf Schwitter. Natural language processing in the undergraduate curriculum. In *Proceedings of the fifth Australasian conference on Computing education-Volume 20*, pages 9–13. Australian Computer Society, Inc., 2003.

[8] Ethem Alpaydin. *Introduction to Machine Learning. London: The MIT Press.* 2010.

[9] M Ikonomakis, Sotiris Kotsiantis, and V Tampakas. Text classification using machine learning techniques. *WSEAS transactions on computers*, 4(8):966–974, 2005.

[10] Gábor Berend and Richárd Farkas. Sztergak: Feature engineering for keyphrase extraction. In *Proceedings of the 5th international workshop on semantic evaluation*, pages 186–189. Association for Computational Linguistics, 2010.

[11] Wei-Hung Weng, Kavishwar B Wagholikar, Alexa T McCray, Peter Szolovits, and Henry C Chueh. Medical subdomain classification of clinical notes using a machine learning-based natural language processing approach. *BMC medical informatics and decision making*, 17(1):155, 2017.

[12] Aniruddha Dey, Shiladitya Chowdhury, and Manas Ghosh. Face recognition using ensemble support vector machine. In *Research in Computational Intelligence and Communication Networks (ICRCICN), 2017 Third International Conference on*, pages 45–50. IEEE, 2017.

[13] G-X Yu, George Ostrouchov, Al Geist, and Nagiza F Samatova. An svm-based algorithm for identification of photosynthesis-specific genome features. In *Bioinformatics Conference, 2003. CSB 2003. Proceedings of the 2003 IEEE*, pages 235–243. IEEE, 2003.

[14] StatSoft. Support vector machines ,http:// www.statsoft.com/ Textbook/ Support-Vector-Machines, 2013.

[15] Li Deng, Dong Yu, et al. Deep learning: methods and applications. *Foundations and Trends® in Signal Processing*, 7(3–4):197–387, 2014.

[16] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[17] Roy T Fielding and Richard N Taylor. *Architectural styles and the design of network-based software architectures*, volume 7. University of California, Irvine Doctoral dissertation, 2000.

[18] Elif Derya Übeyli. Combined neural networks for diagnosis of erythematosquamous diseases. *Expert Systems with Applications*, 36(3):5107–5112, 2009.

[19] Duggal Reena. Preventive care model based on patient profiling using big data analytics for diabetic patients. 2017.

[20] Yun Liu, Krishna Gadepalli, Mohammad Norouzi, George E Dahl, Timo Kohlberger, Aleksey Boyko, Subhashini Venugopalan, Aleksei Timofeev, Philip Q Nelson, Greg S Corrado, et al. Detecting cancer metastases on gigapixel pathology images. *arXiv preprint arXiv:1703.02442*, 2017.

[21] TechNews. 專注癌症診斷及決策，人工智慧醫療公司「paige.ai」完成 2,500 萬美元 a 輪融資, 2018.

[22] Stéphane Meystre and Peter J Haug. Natural language processing to extract medical problems from electronic clinical documents: performance evaluation. *Journal of biomedical informatics*, 39(6):589–599, 2006.

[23] Guergana K Savova, Jin Fan, Zi Ye, Sean P Murphy, Jiaping Zheng, Christopher G Chute, and Iftikhar J Kullo. Discovering peripheral arterial disease cases from radiology notes using natural language processing. In *AMIA Annual Symposium Proceedings*, volume 2010, page 722. American Medical Informatics Association, 2010.

[24] Lionel TE Cheng, Jiaping Zheng, Guergana K Savova, and Bradley J Erickson. Discerning tumor status from unstructured mri reports—completeness of information in existing reports and utility of automated natural language processing. *Journal of digital imaging*, 23(2):119–132, 2010.

[25] José Luis Jara, Max Chacón, and Gonzalo Zelaya. Empirical evaluation of three machine learning method for automatic classification of neoplastic diagnoses. *Ingeniare. Revista chilena de ingeniería*, 19(3), 2011.

[26] Umer Khan, Lars Schmidt-Thieme, and Alexandros Nanopoulos. Collaborative svm classification in scale-free peer-to-peer networks. *Expert Systems with Applications*, 69:74–86, 2017.

[27] Wei Zhou, Junhao Wen, Qingyu Xiong, Min Gao, and Jun Zeng. Svm-tia a shilling attack detection method based on svm and target item analysis in recommender systems. *Neurocomputing*, 210:197–205, 2016.

[28] Sebastián Maldonado and Julio López. Dealing with high-dimensional class-imbalanced datasets: embedded feature selection for svm classification. *Applied Soft Computing*, 67:94–105, 2018.

[29] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

[30] Gastón C Hillar. *Django RESTful Web Services: The easiest way to build Python RESTful APIs and web services with Django*. Packt Publishing Ltd, 2018.

[31] Ching-Hsiang Su, Wei-Chih Huang, Van-Dai Ta, Chuan-Ming Liu, and Sheng-Lung Peng. Exploiting a cloud framework for automatically and effectively providing data analyzers. In *Cloud and Service Computing (SC2), 2017 IEEE 7th International Symposium on*, pages 231–236. IEEE, 2017.

附錄 A

貝葉式分類器代碼

A.1 讀取資料

步驟 1. 引入相關套件以便程式運行。

```
%matplotlib inline
from sklearn import datasets,metrics
import pickle
import matplotlib.pyplot as plt
from sklearn.manifold import Isomap
```

步驟 2. 讀取資料並顯示。

```
#讀取資料
#訓練資料
twenty_train = datasets.load_files(訓練資料路徑)
#測試資料
twenty_test = datasets.load_files(測試資料路徑)

# 查看數據集的大小 訓練類別，訓練資料數量  測試類別，測試資料數量
print(len(twenty_train.target_names), len(twenty_train.data), len(twenty_test.
    target_names), len(twenty_test.data))
```

A.2 取得特徵

步驟 1. 計算詞頻

```
from sklearn.feature_extraction.text import CountVectorizer
count_vect = CountVectorizer(min_df=1, stop_words="english", decode_error='ignore')
X_train_counts = count_vect.fit_transform(twenty_train.data)
num_samples, num_features = X_train_counts.shape
print("#sample: %d, #feature: %d" % (num_samples, num_features))
```

步驟 2. 使用 TF-IDF 進行特徵提取。

```
from sklearn.feature_extraction.text import TfidfTransformer
tfidf_transformer = TfidfTransformer()
X_train_tfidf = tfidf_transformer.fit_transform(X_train_counts)
num_samples, num_features = X_train_tfidf.shape
print("#sample: %d, #feature: %d" % (num_samples, num_features))
tfidf_transformer.get_feature_names()
```

A.3 貝葉斯分類器及圖表繪製

步驟 1. 引入 from sklearn.pipeline 套件使用 Pipeline 執行緒。

```
from sklearn.pipeline import Pipeline
#建立
text_clf_NB = Pipeline([('vect', CountVectorizer(stop_words="english", decode_error='
    ignore')),
('tfidf', TfidfTransformer()),
('clf', MultinomialNB()),
])

t = time()
#執行
text_clf_NB = text_clf_NB.fit(twenty_train.data, twenty_train.target)
print("training time: %f" % round(time()-t, 3))
```

步驟 2. 圖表繪製呈現。

```
from sklearn.cross_validation import cross_val_score
from sklearn.learning_curve import learning_curve
```

```
scores = cross_val_score(estimator=text_clf_NB, X=twenty_train.data, y=twenty_train.
    target, cv=10, n_jobs=1)
print(scores)

train_sizes, train_scores, test_scores = scores = learning_curve(estimator=text_clf_NB
    , X=twenty_train.data, y=twenty_train.target,train_sizes=np.linspace(0.1, 1.0, 10)
    , cv=10, n_jobs=1)

train_mean = np.mean(train_scores, axis=1)
train_std = np.std(train_scores, axis=1)
test_mean = np.mean(test_scores, axis=1)
test_std = np.std(test_scores, axis=1)

plt.plot(train_sizes, train_mean, color='blue', marker='o', markersize=5, label='
    training accuracy')
plt.fill_between(train_sizes, train_mean + train_std, train_mean - train_std, alpha
    =0.15, color='blue')
plt.plot(train_sizes, test_mean, color='green', linestyle='--', marker='s', markersize
    =5, label='validation accuracy')
plt.fill_between(train_sizes, test_mean + test_std, test_mean - test_std, alpha=0.15,
    color='green')
plt.grid()
plt.xlabel('Number of training samples')
plt.ylabel('Accuracy')
plt.legend(loc='lower right')
plt.ylim([0.1, 1.0])
plt.show()
```

A.4 結果表格呈現

引入 numpy 並訓練模型放置 accuracy_score 函數中

```
import numpy as np
from sklearn.metrics import accuracy_score
docs_test = twenty_test.data
t = time()
pred_NB = text_clf_NB.predict(docs_test)
print("testing time: %f" % round(time()-t, 3))
accuracy = accuracy_score(twenty_test.target, pred_NB)
print("Accuracy:", accuracy)
print(metrics.classification_report(twenty_test.target, pred_NB))
```

A.5 儲存模型

使用.pickle 即可將訓練模型儲存

```
f = open('./檔名.pickle', 'wb')
obj ={"twenty_train":twenty_train,"text_clf_SVM":text_clf_SVM}
pickle.dump(obj,f)


f.close()
```

附錄 B

SVM 分類器代碼

B.1 讀取資料

步驟 1. 引入相關套件以便程式運行。

```
%matplotlib inline
from sklearn import datasets,metrics
import pickle
import matplotlib.pyplot as plt
from sklearn.manifold import Isomap
```

步驟 2. 讀取資料並顯示。

```
#讀取資料
#訓練資料
twenty_train = datasets.load_files(訓練資料路徑)
#測試資料
twenty_test = datasets.load_files(測試資料路徑)

# 查看數據集的大小 訓練類別，訓練資料數量  測試類別，測試資料數量
print(len(twenty_train.target_names), len(twenty_train.data), len(twenty_test.
    target_names), len(twenty_test.data))
```

B.2 取得特徵

步驟 1. 計算詞頻

```
from sklearn.feature_extraction.text import CountVectorizer
count_vect = CountVectorizer(min_df=1, stop_words="english", decode_error='ignore')
X_train_counts = count_vect.fit_transform(twenty_train.data)
num_samples, num_features = X_train_counts.shape
print("#sample: %d, #feature: %d" % (num_samples, num_features))
```

步驟 2. 使用 TF-IDF 進行特徵提取。

```
from sklearn.feature_extraction.text import TfidfTransformer
tfidf_transformer = TfidfTransformer()
X_train_tfidf = tfidf_transformer.fit_transform(X_train_counts)
num_samples, num_features = X_train_tfidf.shape
print("#sample: %d, #feature: %d" % (num_samples, num_features))
tfidf_transformer.get_feature_names()
```

B.3 SVM 分類器及圖表繪製

步驟 1. 引入 sklearn.svm import SVC 套件使用 Pipeline 執行緒。

```
from sklearn.pipeline import Pipeline
#建立
from sklearn.svm import SVC
text_clf_SVM = Pipeline([('vect', CountVectorizer(stop_words='english', decode_error='
    ignore')),
('tfidf', TfidfTransformer()),
('clf', SVC(kernel='linear', C=1.0, random_state=2)),
])

docs_test = twenty_test.data

t0 = time()
text_clf_SVM = text_clf_SVM.fit(twenty_train.data, twenty_train.target)
print("training time: %f" % round(time()-t0, 3))
```

步驟 2. 圖表繪製呈現。

```
from sklearn.cross_validation import cross_val_score
from sklearn.learning_curve import learning_curve

scores = cross_val_score(estimator=text_clf_SVM, X=twenty_train.data, y=twenty_train.
   target, cv=10, n_jobs=1)
print(scores)

train_sizes, train_scores, test_scores = scores = learning_curve(estimator=
   text_clf_SVM, X=twenty_train.data, y=twenty_train.target,train_sizes=np.linspace
   (0.1, 1.0, 10), cv=10, n_jobs=1)

train_mean = np.mean(train_scores, axis=1)
train_std = np.std(train_scores, axis=1)
test_mean = np.mean(test_scores, axis=1)
test_std = np.std(test_scores, axis=1)

plt.plot(train_sizes, train_mean, color='blue', marker='o', markersize=5, label='
   training accuracy')
plt.fill_between(train_sizes, train_mean + train_std, train_mean - train_std, alpha
   =0.15, color='blue')
plt.plot(train_sizes, test_mean, color='green', linestyle='--', marker='s',
   markersize=5, label='validation accuracy')
plt.fill_between(train_sizes, test_mean + test_std, test_mean - test_std, alpha=0.15,
    color='green')
plt.grid()
plt.xlabel('Number of training samples')
plt.ylabel('Accuracy')
plt.legend(loc='lower right')
plt.ylim([0.1, 1.0])
plt.show()
```

B.4 結果表格呈現

引入 numpy 並訓練模型放置 accuracy_score 函數中

```
import numpy as np
from sklearn.metrics import accuracy_score
docs_test = twenty_test.data
t1 = time()
```

```
pred_SVM = text_clf_SVM.predict(docs_test)
print("testing time: %f" % round(time()-t1, 3))
accuracy = accuracy_score(twenty_test.target, pred_SVM)
print("Accuracy:", accuracy)
print(metrics.classification_report(twenty_test.target, pred_SVM))
```

B.5 儲存模型

使用.pickle 即可將訓練模型儲存

```
f = open('./檔名.pickle', 'wb')
obj ={"twenty_train":twenty_train,"text_clf_SVM":text_clf_SVM}
pickle.dump(obj,f)

f.close()
```

附錄 C

Text-CNN 分類器代碼

C.1 Text-CNN 參數設定

cnn_model.py

```
# coding: utf-8

import tensorflow as tf



class TCNNConfig(object):
"""CNN配置參數"""

embedding_dim = 64  # 詞向量維度
seq_length = 600  # 序列長度
num_classes = 10  # 類別數
num_filters = 256  # 卷積核數目
kernel_size = 5  # 卷積核尺寸
vocab_size = 5000  #詞彙表達大小

hidden_dim = 128  # 全連接層神經元

dropout_keep_prob = 0.5  # dropout保留比例
learning_rate = 1e-3  # 學習率

batch_size = 64  # 每批訓練大小
num_epochs = 10  # 總迭代輪次
```

```
print_per_batch = 100  # 每多少輪輸出一次結果
save_per_batch = 10  # 每多少輪存入tensorboard


class TextCNN(object):
"""文本分類，CNN模型"""

def __init__(self, config):
self.config = config

# 三個待輸入的數據
self.input_x = tf.placeholder(tf.int32, [None, self.config.seq_length], name='input_x
    ')
self.input_y = tf.placeholder(tf.float32, [None, self.config.num_classes], name='
    input_y')
self.keep_prob = tf.placeholder(tf.float32, name='keep_prob')

self.cnn()

def cnn(self):
"""CNN模型"""
# 詞向量映射
with tf.device('/cpu:0'):
embedding = tf.get_variable('embedding', [self.config.vocab_size, self.config.
    embedding_dim])
embedding_inputs = tf.nn.embedding_lookup(embedding, self.input_x)

with tf.name_scope("cnn"):
# CNN layer
conv = tf.layers.conv1d(embedding_inputs, self.config.num_filters, self.config.
    kernel_size, name='conv')
# global max pooling layer
gmp = tf.reduce_max(conv, reduction_indices=[1], name='gmp')

with tf.name_scope("score"):
# 全連接層，後面接dropout以及relu激活
fc = tf.layers.dense(gmp, self.config.hidden_dim, name='fc1')
fc = tf.contrib.layers.dropout(fc, self.keep_prob)
fc = tf.nn.relu(fc)

# 分類器
self.logits = tf.layers.dense(fc, self.config.num_classes, name='fc2')
self.y_pred_cls = tf.argmax(tf.nn.softmax(self.logits), 1)  # 預測類別

with tf.name_scope("optimize"):
```

```
# 損失函□，交叉
cross_entropy = tf.nn.softmax_cross_entropy_with_logits(logits=self.logits, labels=
    self.input_y)
self.loss = tf.reduce_mean(cross_entropy)
# 優化器
self.optim = tf.train.AdamOptimizer(learning_rate=self.config.learning_rate).minimize(
    self.loss)

with tf.name_scope("accuracy"):
# 準確率
correct_pred = tf.equal(tf.argmax(self.input_y, 1), self.y_pred_cls)
self.acc = tf.reduce_mean(tf.cast(correct_pred, tf.float32))
```

C.2 Text-CNN 執行程式

run_cnn.py

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

from __future__ import print_function

import os
import sys
import time
from datetime import timedelta

import numpy as np
import tensorflow as tf
from sklearn import metrics

from cnn_model import TCNNConfig, TextCNN
from data.cnews_loader import read_vocab, read_category, batch_iter, process_file,
    build_vocab

base_dir = 'data/cnews'
train_dir = os.path.join(base_dir, 'cnews.train.txt')
test_dir = os.path.join(base_dir, 'cnews.test.txt')
val_dir = os.path.join(base_dir, 'cnews.val.txt')
vocab_dir = os.path.join(base_dir, 'cnews.vocab.txt')

save_dir = 'checkpoints/textcnn'
```

```
save_path = os.path.join(save_dir, 'best_validation')  # 最佳驗證□果保存路徑


def get_time_dif(start_time):
"""獲取已使用時間"""
end_time = time.time()
time_dif = end_time - start_time
return timedelta(seconds=int(round(time_dif)))


def feed_data(x_batch, y_batch, keep_prob):
feed_dict = {
model.input_x: x_batch,
model.input_y: y_batch,
model.keep_prob: keep_prob
}
return feed_dict


def evaluate(sess, x_, y_):
"""評估在某一數據上的準確率與損失"""
data_len = len(x_)
batch_eval = batch_iter(x_, y_, 128)
total_loss = 0.0
total_acc = 0.0
for x_batch, y_batch in batch_eval:
batch_len = len(x_batch)
feed_dict = feed_data(x_batch, y_batch, 1.0)
loss, acc = sess.run([model.loss, model.acc], feed_dict=feed_dict)
total_loss += loss * batch_len
total_acc += acc * batch_len

return total_loss / data_len, total_acc / data_len


def train():
print("Configuring TensorBoard and Saver...")
# 配置 Tensorboard，重新訓練時，將tensorboard資料夾餐廚，不然圖會覆蓋

tensorboard_dir = 'tensorboard/textcnn'
if not os.path.exists(tensorboard_dir):
os.makedirs(tensorboard_dir)

tf.summary.scalar("loss", model.loss)
tf.summary.scalar("accuracy", model.acc)
```

```
merged_summary = tf.summary.merge_all()
writer = tf.summary.FileWriter(tensorboard_dir)

# 配置 Saver
saver = tf.train.Saver()
if not os.path.exists(save_dir):
os.makedirs(save_dir)

print("Loading training and validation data...")
# 載入訓練與訓練集
start_time = time.time()
x_train, y_train = process_file(train_dir, word_to_id, cat_to_id, config.seq_length)
x_val, y_val = process_file(val_dir, word_to_id, cat_to_id, config.seq_length)
time_dif = get_time_dif(start_time)
print("Time usage:", time_dif)

# 建立session
session = tf.Session()
session.run(tf.global_variables_initializer())
writer.add_graph(session.graph)

print('Training and evaluating...')
start_time = time.time()
total_batch = 0  # 總批次
best_acc_val = 0.0  # 最佳驗證集準確率
last_improved = 0  # 紀錄上一次提升批次
require_improvement = 1000  # 如果超過1000輪未提升，提前□束訓練

flag = False
for epoch in range(config.num_epochs):
print('Epoch:', epoch + 1)
batch_train = batch_iter(x_train, y_train, config.batch_size)
for x_batch, y_batch in batch_train:
feed_dict = feed_data(x_batch, y_batch, config.dropout_keep_prob)

if total_batch % config.save_per_batch == 0:
# 每多少輪次將訓練結果寫入tensorboard scalar
s = session.run(merged_summary, feed_dict=feed_dict)
writer.add_summary(s, total_batch)

if total_batch % config.print_per_batch == 0:
# 每多少輪次輸出在訓練集和驗證集上的性能
feed_dict[model.keep_prob] = 1.0
loss_train, acc_train = session.run([model.loss, model.acc], feed_dict=feed_dict)
loss_val, acc_val = evaluate(session, x_val, y_val)  # todo
```

```
if acc_val > best_acc_val:
# 保存最好結果
best_acc_val = acc_val
last_improved = total_batch
saver.save(sess=session, save_path=save_path)
improved_str = '*'
else:
improved_str = ''

time_dif = get_time_dif(start_time)
msg = 'Iter: {0:>6}, Train Loss: {1:>6.2}, Train Acc: {2:>7.2%},' \
+ ' Val Loss: {3:>6.2}, Val Acc: {4:>7.2%}, Time: {5} {6}'
print(msg.format(total_batch, loss_train, acc_train, loss_val, acc_val, time_dif,
    improved_str))

session.run(model.optim, feed_dict=feed_dict)  # 運行優化
total_batch += 1

if total_batch - last_improved > require_improvement:
# 驗證集正確率長期不提升，提前結束訓練
print("No optimization for a long time, auto-stopping...")
flag = True
break  # 跳出循環
if flag:  # 同上
break


def test():
print("Loading test data...")
start_time = time.time()
x_test, y_test = process_file(test_dir, word_to_id, cat_to_id, config.seq_length)

session = tf.Session()
session.run(tf.global_variables_initializer())
saver = tf.train.Saver()
saver.restore(sess=session, save_path=save_path)  # 讀取保存的模型

print('Testing...')
loss_test, acc_test = evaluate(session, x_test, y_test)
msg = 'Test Loss: {0:>6.2}, Test Acc: {1:>7.2%}'
print(msg.format(loss_test, acc_test))

batch_size = 128
data_len = len(x_test)
```

```
num_batch = int((data_len - 1) / batch_size) + 1

y_test_cls = np.argmax(y_test, 1)
y_pred_cls = np.zeros(shape=len(x_test), dtype=np.int32)  # 保存預測□果
for i in range(num_batch):  # 逐批次處理
start_id = i * batch_size
end_id = min((i + 1) * batch_size, data_len)
feed_dict = {
model.input_x: x_test[start_id:end_id],
model.keep_prob: 1.0
}
y_pred_cls[start_id:end_id] = session.run(model.y_pred_cls, feed_dict=feed_dict)

# 評估
print("Precision, Recall and F1-Score...")
print(metrics.classification_report(y_test_cls, y_pred_cls, target_names=categories))

# 混淆矩陣
print("Confusion Matrix...")
cm = metrics.confusion_matrix(y_test_cls, y_pred_cls)
print(cm)

time_dif = get_time_dif(start_time)
print("Time usage:", time_dif)


if __name__ == '__main__':
if len(sys.argv) != 2 or sys.argv[1] not in ['train', 'test']:
raise ValueError("""usage: python run_cnn.py [train / test]""")

print('Configuring CNN model...')
config = TCNNConfig()
if not os.path.exists(vocab_dir):  # 如果不存在詞彙表，重建
build_vocab(train_dir, vocab_dir, config.vocab_size)
categories, cat_to_id = read_category()
words, word_to_id = read_vocab(vocab_dir)
config.vocab_size = len(words)
model = TextCNN(config)

if sys.argv[1] == 'train':
train()
else:
test()
```

附錄 D

MCODE 分類報表

D.1 貝葉式分類法

	precision	recall	f1-score	support
0	0.00	0.00	0.00	3
1	0.00	0.00	0.00	6
2	0.00	0.00	0.00	14
3	0.00	0.00	0.00	1
4	0.00	0.00	0.00	1
5	0.00	0.00	0.00	1
6	0.00	0.00	0.00	82
7	0.00	0.00	0.00	43
8	0.00	0.00	0.00	6
9	0.00	0.00	0.00	3
10	0.00	0.00	0.00	4
11	0.00	0.00	0.00	1
12	0.00	0.00	0.00	1
13	0.00	0.00	0.00	1
14	0.00	0.00	0.00	16
15	0.00	0.00	0.00	1
16	0.00	0.00	0.00	5
17	0.00	0.00	0.00	1
18	0.00	0.00	0.00	29
19	0.00	0.00	0.00	1
20	0.00	0.00	0.00	1
21	0.98	0.95	0.96	208
22	0.00	0.00	0.00	1

23	0.00	0.00	0.00	2
24	0.00	0.00	0.00	19
25	0.00	0.00	0.00	1
26	0.00	0.00	0.00	49
27	0.58	0.98	0.73	544
28	0.00	0.00	0.00	24
29	0.00	0.00	0.00	1
30	0.00	0.00	0.00	5
31	0.00	0.00	0.00	1
32	0.00	0.00	0.00	2
33	1.00	0.22	0.36	45
34	0.00	0.00	0.00	1
35	0.00	0.00	0.00	1
36	0.00	0.00	0.00	1
37	0.00	0.00	0.00	1
38	0.88	0.94	0.91	87
39	0.00	0.00	0.00	22
40	0.00	0.00	0.00	2
41	0.00	0.00	0.00	6
42	0.00	0.00	0.00	2
43	1.00	0.04	0.08	46
44	0.00	0.00	0.00	1
45	0.00	0.00	0.00	1
46	0.00	0.00	0.00	7
47	0.00	0.00	0.00	1
48	0.00	0.00	0.00	3
49	1.00	0.62	0.76	65
50	0.00	0.00	0.00	3
51	0.00	0.00	0.00	29
52	0.00	0.00	0.00	2
53	0.00	0.00	0.00	2
54	0.00	0.00	0.00	12
55	0.00	0.00	0.00	38
56	0.00	0.00	0.00	6
57	0.00	0.00	0.00	41
58	0.49	0.97	0.65	849
59	0.00	0.00	0.00	129
60	0.00	0.00	0.00	7
61	0.00	0.00	0.00	3
62	0.00	0.00	0.00	6
63	1.00	0.02	0.04	44
64	0.00	0.00	0.00	3
65	0.00	0.00	0.00	1
66	0.00	0.00	0.00	5
67	0.00	0.00	0.00	1

68	0.00	0.00	0.00	1
69	0.00	0.00	0.00	18
70	0.72	0.99	0.84	1107
71	0.00	0.00	0.00	63
72	0.00	0.00	0.00	1
73	0.00	0.00	0.00	10
74	0.00	0.00	0.00	3
75	0.00	0.00	0.00	1
76	0.00	0.00	0.00	1
77	0.00	0.00	0.00	3
78	0.00	0.00	0.00	10
79	0.95	0.14	0.25	366
80	0.00	0.00	0.00	7
81	0.00	0.00	0.00	4
82	0.00	0.00	0.00	4
83	0.00	0.00	0.00	3
84	0.00	0.00	0.00	2
85	0.00	0.00	0.00	3
86	1.00	0.09	0.17	33
87	0.00	0.00	0.00	1
88	0.00	0.00	0.00	7
89	0.00	0.00	0.00	1
90	0.00	0.00	0.00	2
91	0.00	0.00	0.00	33
92	0.00	0.00	0.00	2
93	0.00	0.00	0.00	3
94	0.00	0.00	0.00	1
95	0.00	0.00	0.00	4
96	0.00	0.00	0.00	21
97	0.00	0.00	0.00	1
98	0.00	0.00	0.00	5
99	0.00	0.00	0.00	1
100	0.00	0.00	0.00	7
101	0.00	0.00	0.00	6
102	0.00	0.00	0.00	3
103	0.00	0.00	0.00	4
104	0.00	0.00	0.00	3
105	0.00	0.00	0.00	4
106	0.00	0.00	0.00	1
107	0.00	0.00	0.00	1
108	0.00	0.00	0.00	24
109	0.00	0.00	0.00	3
110	0.00	0.00	0.00	7
111	0.00	0.00	0.00	30
112	0.00	0.00	0.00	1

113	0.00	0.00	0.00	2
114	0.00	0.00	0.00	9
115	0.00	0.00	0.00	36
116	0.69	0.98	0.81	265
117	0.00	0.00	0.00	13
118	0.00	0.00	0.00	1
119	0.00	0.00	0.00	14
120	0.00	0.00	0.00	2
121	0.00	0.00	0.00	1
122	0.00	0.00	0.00	5
123	0.00	0.00	0.00	3
124	0.00	0.00	0.00	5
125	0.00	0.00	0.00	8
126	0.00	0.00	0.00	1
127	0.00	0.00	0.00	4
128	0.00	0.00	0.00	1
129	0.00	0.00	0.00	3
130	0.00	0.00	0.00	16
131	0.00	0.00	0.00	1
132	0.00	0.00	0.00	1
133	0.00	0.00	0.00	1
134	0.00	0.00	0.00	1
135	0.00	0.00	0.00	1
136	0.00	0.00	0.00	1
137	0.00	0.00	0.00	2
138	0.00	0.00	0.00	2
139	0.00	0.00	0.00	1
140	0.00	0.00	0.00	3
141	0.00	0.00	0.00	1
142	0.00	0.00	0.00	1
143	0.00	0.00	0.00	1
144	0.00	0.00	0.00	1
145	0.00	0.00	0.00	8
146	0.00	0.00	0.00	3
147	0.00	0.00	0.00	3
148	0.00	0.00	0.00	2
149	0.00	0.00	0.00	11
150	0.00	0.00	0.00	2
151	0.65	0.99	0.78	391
152	0.00	0.00	0.00	64
153	0.00	0.00	0.00	1
154	0.00	0.00	0.00	19
155	0.00	0.00	0.00	1
156	0.00	0.00	0.00	3
157	0.00	0.00	0.00	1

158	0.00	0.00	0.00	1
159	0.00	0.00	0.00	21
160	0.00	0.00	0.00	1
161	0.00	0.00	0.00	2
162	0.00	0.00	0.00	2
163	0.00	0.00	0.00	1
164	0.00	0.00	0.00	1
165	0.00	0.00	0.00	4
166	0.00	0.00	0.00	3
167	0.00	0.00	0.00	1
168	0.00	0.00	0.00	1
169	0.00	0.00	0.00	1
170	0.00	0.00	0.00	1
171	0.97	0.89	0.93	154
172	0.00	0.00	0.00	1
173	0.00	0.00	0.00	1
174	0.00	0.00	0.00	4
175	0.00	0.00	0.00	8
176	0.00	0.00	0.00	1
177	0.60	0.99	0.75	552
178	0.00	0.00	0.00	6
179	0.00	0.00	0.00	28
180	0.00	0.00	0.00	20
181	0.00	0.00	0.00	2
182	0.00	0.00	0.00	2
183	0.00	0.00	0.00	105
184	0.73	0.97	0.83	296
185	0.00	0.00	0.00	2
186	0.00	0.00	0.00	4
187	0.00	0.00	0.00	1
188	0.00	0.00	0.00	1
189	0.00	0.00	0.00	13
190	0.00	0.00	0.00	4
191	0.00	0.00	0.00	6
192	0.00	0.00	0.00	2
193	0.00	0.00	0.00	36
194	0.00	0.00	0.00	2
195	0.00	0.00	0.00	5
196	0.00	0.00	0.00	1
197	0.00	0.00	0.00	1
198	0.00	0.00	0.00	5
199	0.00	0.00	0.00	1
200	0.90	0.98	0.94	177
201	0.00	0.00	0.00	2
202	0.00	0.00	0.00	1

203	0.00	0.00	0.00	13
204	0.00	0.00	0.00	1
205	0.00	0.00	0.00	1
206	0.00	0.00	0.00	1
207	0.00	0.00	0.00	5
208	0.00	0.00	0.00	1
209	0.00	0.00	0.00	20
210	0.00	0.00	0.00	11
211	0.00	0.00	0.00	2
212	0.00	0.00	0.00	1
213	0.00	0.00	0.00	3
214	0.00	0.00	0.00	1
215	0.71	0.47	0.57	114
216	0.00	0.00	0.00	18
217	0.00	0.00	0.00	3
218	0.00	0.00	0.00	3
219	0.00	0.00	0.00	1
220	1.00	0.71	0.83	41
221	0.00	0.00	0.00	2
222	0.00	0.00	0.00	6
223	0.00	0.00	0.00	2
224	0.00	0.00	0.00	3
225	0.00	0.00	0.00	7
226	0.00	0.00	0.00	1
227	0.00	0.00	0.00	1
228	0.00	0.00	0.00	10
229	0.00	0.00	0.00	11
230	0.00	0.00	0.00	3
231	0.00	0.00	0.00	1
232	0.00	0.00	0.00	3
233	1.00	0.62	0.77	29
234	0.00	0.00	0.00	1
235	0.00	0.00	0.00	1
236	0.00	0.00	0.00	1
237	0.00	0.00	0.00	1
238	0.00	0.00	0.00	2
239	0.00	0.00	0.00	3
240	0.00	0.00	0.00	1
241	0.00	0.00	0.00	18
242	0.00	0.00	0.00	1
243	0.00	0.00	0.00	2
244	0.98	0.44	0.60	94
245	0.00	0.00	0.00	1
246	0.00	0.00	0.00	2
247	1.00	0.23	0.38	39

248	0.00	0.00	0.00	1
249	0.00	0.00	0.00	2
250	0.00	0.00	0.00	1
251	0.00	0.00	0.00	1
252	0.00	0.00	0.00	1
253	0.00	0.00	0.00	1
254	0.00	0.00	0.00	1
255	0.00	0.00	0.00	2
256	0.00	0.00	0.00	1
257	0.00	0.00	0.00	2
258	0.00	0.00	0.00	2
259	1.00	0.46	0.63	41
260	0.00	0.00	0.00	2
261	0.00	0.00	0.00	1
262	0.00	0.00	0.00	1
263	0.00	0.00	0.00	3
264	0.00	0.00	0.00	1
265	0.00	0.00	0.00	2
266	0.00	0.00	0.00	1
267	0.00	0.00	0.00	1
268	0.00	0.00	0.00	12
269	0.00	0.00	0.00	3
270	0.00	0.00	0.00	1
271	0.00	0.00	0.00	1
272	0.00	0.00	0.00	1
273	0.00	0.00	0.00	1
274	0.00	0.00	0.00	1
275	0.00	0.00	0.00	1
276	0.00	0.00	0.00	4
277	0.00	0.00	0.00	1
278	0.00	0.00	0.00	1
279	0.00	0.00	0.00	1
avg/total	0.54	0.65	0.55	7349

D.2 SVM

	precision	recall	f1-score	support
0	1.00	1.00	1.00	3
1	1.00	1.00	1.00	6
2	0.82	1.00	0.90	14

3	0.00	0.00	0.00	1
4	0.00	0.00	0.00	1
5	0.00	0.00	0.00	1
6	0.94	0.95	0.95	82
7	1.00	0.93	0.96	43
8	1.00	0.83	0.91	6
9	1.00	0.33	0.50	3
10	1.00	0.50	0.67	4
11	0.00	0.00	0.00	1
12	0.00	0.00	0.00	1
13	0.00	0.00	0.00	1
14	0.89	1.00	0.94	16
15	0.00	0.00	0.00	1
16	1.00	0.80	0.89	5
17	0.00	0.00	0.00	1
18	0.84	0.93	0.89	29
19	0.00	0.00	0.00	1
20	0.00	0.00	0.00	1
21	0.98	1.00	0.99	208
22	0.00	0.00	0.00	1
23	0.00	0.00	0.00	2
24	0.90	1.00	0.95	19
25	0.00	0.00	0.00	1
26	0.89	0.98	0.93	49
27	0.95	0.99	0.97	544
28	0.91	0.88	0.89	24
29	0.00	0.00	0.00	1
30	1.00	1.00	1.00	5
31	0.00	0.00	0.00	1
32	0.00	0.00	0.00	2
33	0.91	0.96	0.93	45
34	0.00	0.00	0.00	1
35	0.00	0.00	0.00	1
36	0.00	0.00	0.00	1
37	0.00	0.00	0.00	1
38	1.00	1.00	1.00	87
39	1.00	1.00	1.00	22
40	0.00	0.00	0.00	2
41	1.00	1.00	1.00	6
42	0.00	0.00	0.00	2
43	1.00	1.00	1.00	46
44	0.00	0.00	0.00	1
45	0.00	0.00	0.00	1
46	1.00	1.00	1.00	7
47	0.00	0.00	0.00	1

48	1.00	1.00	1.00	3
49	0.98	0.98	0.98	65
50	1.00	0.33	0.50	3
51	0.88	0.97	0.92	29
52	1.00	0.50	0.67	2
53	0.00	0.00	0.00	2
54	0.89	0.67	0.76	12
55	0.90	1.00	0.95	38
56	1.00	0.67	0.80	6
57	0.78	0.88	0.83	41
58	0.92	0.99	0.96	849
59	0.95	0.94	0.95	129
60	1.00	0.57	0.73	7
61	1.00	1.00	1.00	3
62	1.00	1.00	1.00	6
63	0.94	1.00	0.97	44
64	1.00	0.33	0.50	3
65	0.00	0.00	0.00	1
66	0.83	1.00	0.91	5
67	0.00	0.00	0.00	1
68	1.00	1.00	1.00	1
69	1.00	1.00	1.00	18
70	0.99	0.99	0.99	1107
71	0.91	0.97	0.94	63
72	0.00	0.00	0.00	1
73	1.00	0.90	0.95	10
74	1.00	1.00	1.00	3
75	0.00	0.00	0.00	1
76	0.00	0.00	0.00	1
77	1.00	1.00	1.00	3
78	1.00	1.00	1.00	10
79	0.99	0.96	0.97	366
80	0.00	0.00	0.00	7
81	0.80	1.00	0.89	4
82	0.00	0.00	0.00	4
83	1.00	0.33	0.50	3
84	1.00	1.00	1.00	2
85	1.00	1.00	1.00	3
86	0.79	1.00	0.88	33
87	0.00	0.00	0.00	1
88	0.83	0.71	0.77	7
89	0.00	0.00	0.00	1
90	0.00	0.00	0.00	2
91	1.00	1.00	1.00	33
92	0.00	0.00	0.00	2

93	1.00	1.00	1.00	3
94	0.00	0.00	0.00	1
95	1.00	0.25	0.40	4
96	0.84	1.00	0.91	21
97	0.00	0.00	0.00	1
98	1.00	1.00	1.00	5
99	0.00	0.00	0.00	1
100	1.00	0.86	0.92	7
101	1.00	1.00	1.00	6
102	1.00	1.00	1.00	3
103	1.00	1.00	1.00	4
104	1.00	1.00	1.00	3
105	1.00	1.00	1.00	4
106	0.00	0.00	0.00	1
107	0.00	0.00	0.00	1
108	0.87	0.83	0.85	24
109	1.00	1.00	1.00	3
110	1.00	1.00	1.00	7
111	0.94	1.00	0.97	30
112	0.00	0.00	0.00	1
113	1.00	0.50	0.67	2
114	1.00	0.44	0.62	9
115	0.81	0.94	0.87	36
116	0.89	0.98	0.93	265
117	1.00	0.23	0.38	13
118	0.00	0.00	0.00	1
119	1.00	0.93	0.96	14
120	1.00	0.50	0.67	2
121	0.00	0.00	0.00	1
122	1.00	0.60	0.75	5
123	1.00	1.00	1.00	3
124	1.00	0.60	0.75	5
125	1.00	0.50	0.67	8
126	0.00	0.00	0.00	1
127	0.67	1.00	0.80	4
128	0.00	0.00	0.00	1
129	1.00	0.67	0.80	3
130	1.00	1.00	1.00	16
131	0.00	0.00	0.00	1
132	0.00	0.00	0.00	1
133	0.00	0.00	0.00	1
134	0.00	0.00	0.00	1
135	0.00	0.00	0.00	1
136	0.00	0.00	0.00	1
137	1.00	1.00	1.00	2

138	1.00	0.50	0.67	2
139	0.00	0.00	0.00	1
140	1.00	1.00	1.00	3
141	0.00	0.00	0.00	1
142	0.00	0.00	0.00	1
143	0.00	0.00	0.00	1
144	0.00	0.00	0.00	1
145	0.57	1.00	0.73	8
146	1.00	1.00	1.00	3
147	0.00	0.00	0.00	3
148	1.00	1.00	1.00	2
149	1.00	1.00	1.00	11
150	0.00	0.00	0.00	2
151	0.95	0.98	0.97	391
152	1.00	0.92	0.96	64
153	0.00	0.00	0.00	1
154	0.86	0.95	0.90	19
155	0.00	0.00	0.00	1
156	1.00	1.00	1.00	3
157	0.00	0.00	0.00	1
158	0.00	0.00	0.00	1
159	0.91	0.95	0.93	21
160	0.00	0.00	0.00	1
161	1.00	1.00	1.00	2
162	1.00	1.00	1.00	2
163	0.00	0.00	0.00	1
164	0.00	0.00	0.00	1
165	1.00	1.00	1.00	4
166	1.00	1.00	1.00	3
167	1.00	1.00	1.00	1
168	0.00	0.00	0.00	1
169	0.00	0.00	0.00	1
170	0.00	0.00	0.00	1
171	0.99	0.98	0.99	154
172	0.00	0.00	0.00	1
173	1.00	1.00	1.00	1
174	0.80	1.00	0.89	4
175	0.80	1.00	0.89	8
176	0.00	0.00	0.00	1
177	0.99	0.99	0.99	552
178	0.86	1.00	0.92	6
179	1.00	0.96	0.98	28
180	0.95	0.90	0.92	20
181	1.00	1.00	1.00	2
182	0.00	0.00	0.00	2

183	0.98	0.99	0.99	105
184	0.95	0.99	0.97	296
185	0.00	0.00	0.00	2
186	0.67	1.00	0.80	4
187	0.00	0.00	0.00	1
188	0.00	0.00	0.00	1
189	1.00	1.00	1.00	13
190	0.67	0.50	0.57	4
191	0.67	1.00	0.80	6
192	0.00	0.00	0.00	2
193	1.00	1.00	1.00	36
194	0.00	0.00	0.00	2
195	1.00	0.60	0.75	5
196	0.00	0.00	0.00	1
197	0.00	0.00	0.00	1
198	1.00	1.00	1.00	5
199	0.00	0.00	0.00	1
200	0.98	0.98	0.98	177
201	1.00	0.50	0.67	2
202	0.00	0.00	0.00	1
203	0.87	1.00	0.93	13
204	0.00	0.00	0.00	1
205	0.00	0.00	0.00	1
206	0.00	0.00	0.00	1
207	0.00	0.00	0.00	5
208	0.00	0.00	0.00	1
209	0.91	1.00	0.95	20
210	0.83	0.91	0.87	11
211	0.00	0.00	0.00	2
212	0.00	0.00	0.00	1
213	1.00	0.67	0.80	3
214	0.00	0.00	0.00	1
215	0.96	0.96	0.96	114
216	1.00	1.00	1.00	18
217	1.00	1.00	1.00	3
218	1.00	1.00	1.00	3
219	0.00	0.00	0.00	1
220	1.00	1.00	1.00	41
221	1.00	1.00	1.00	2
222	1.00	1.00	1.00	6
223	1.00	1.00	1.00	2
224	1.00	1.00	1.00	3
225	0.88	1.00	0.93	7
226	0.00	0.00	0.00	1
227	0.00	0.00	0.00	1

228	1.00	1.00	1.00	10
229	1.00	1.00	1.00	11
230	1.00	1.00	1.00	3
231	0.00	0.00	0.00	1
232	1.00	1.00	1.00	3
233	0.97	1.00	0.98	29
234	0.00	0.00	0.00	1
235	0.00	0.00	0.00	1
236	1.00	1.00	1.00	1
237	0.00	0.00	0.00	1
238	1.00	1.00	1.00	2
239	1.00	1.00	1.00	3
240	0.00	0.00	0.00	1
241	0.86	1.00	0.92	18
242	0.00	0.00	0.00	1
243	0.00	0.00	0.00	2
244	0.93	0.94	0.93	94
245	0.00	0.00	0.00	1
246	1.00	0.50	0.67	2
247	1.00	0.95	0.97	39
248	0.00	0.00	0.00	1
249	1.00	1.00	1.00	2
250	0.00	0.00	0.00	1
251	0.00	0.00	0.00	1
252	0.00	0.00	0.00	1
253	0.00	0.00	0.00	1
254	0.00	0.00	0.00	1
255	1.00	1.00	1.00	2
256	0.00	0.00	0.00	1
257	1.00	1.00	1.00	2
258	1.00	1.00	1.00	2
259	0.89	1.00	0.94	41
260	1.00	1.00	1.00	2
261	0.00	0.00	0.00	1
262	0.00	0.00	0.00	1
263	1.00	0.67	0.80	3
264	0.00	0.00	0.00	1
265	1.00	1.00	1.00	2
266	0.00	0.00	0.00	1
267	0.00	0.00	0.00	1
268	1.00	1.00	1.00	12
269	1.00	1.00	1.00	3
270	0.00	0.00	0.00	1
271	0.00	0.00	0.00	1
272	1.00	1.00	1.00	1

273	1.00	1.00	1.00	1
274	1.00	1.00	1.00	1
275	0.00	0.00	0.00	1
276	0.80	1.00	0.89	4
277	0.00	0.00	0.00	1
278	0.00	0.00	0.00	1
279	1.00	1.00	1.00	1
avg/total	0.94	0.95	0.94	7349

附錄 E

TCODE 分類報表

E.1 貝葉式分類法

	precision	recall	f1-score	support
0	0.00	0.00	0.00	7
1	0.00	0.00	0.00	2
2	0.00	0.00	0.00	3
3	0.00	0.00	0.00	19
4	0.00	0.00	0.00	17
5	0.00	0.00	0.00	7
6	0.98	0.56	0.71	72
7	0.00	0.00	0.00	7
8	0.00	0.00	0.00	1
9	0.00	0.00	0.00	3
10	0.00	0.00	0.00	3
11	0.00	0.00	0.00	3
12	0.00	0.00	0.00	17
13	1.00	0.33	0.49	40
14	0.00	0.00	0.00	4
15	0.00	0.00	0.00	7
16	0.00	0.00	0.00	7
17	0.00	0.00	0.00	1
18	1.00	0.32	0.48	38
19	0.00	0.00	0.00	11
20	0.00	0.00	0.00	4
21	0.00	0.00	0.00	4
22	0.00	0.00	0.00	1

23	0.00	0.00	0.00	29
24	0.00	0.00	0.00	1
25	0.00	0.00	0.00	6
26	0.95	0.45	0.61	42
27	0.00	0.00	0.00	15
28	0.00	0.00	0.00	2
29	0.00	0.00	0.00	10
30	0.00	0.00	0.00	2
31	0.80	0.91	0.85	112
32	0.00	0.00	0.00	1
33	0.00	0.00	0.00	8
34	0.00	0.00	0.00	12
35	0.00	0.00	0.00	7
36	0.67	0.83	0.74	115
37	0.00	0.00	0.00	6
38	0.00	0.00	0.00	2
39	0.00	0.00	0.00	5
40	0.00	0.00	0.00	1
41	0.89	0.81	0.85	133
42	0.00	0.00	0.00	10
43	0.00	0.00	0.00	3
44	0.00	0.00	0.00	4
45	0.00	0.00	0.00	1
46	0.00	0.00	0.00	78
47	0.00	0.00	0.00	3
48	0.00	0.00	0.00	212
49	0.00	0.00	0.00	65
50	0.00	0.00	0.00	124
51	0.00	0.00	0.00	23
52	0.00	0.00	0.00	127
53	0.63	0.40	0.49	531
54	0.00	0.00	0.00	2
55	0.35	0.95	0.51	658
56	0.00	0.00	0.00	53
57	0.54	0.59	0.56	307
58	0.00	0.00	0.00	13
59	0.00	0.00	0.00	3
60	0.00	0.00	0.00	2
61	0.99	0.95	0.97	95
62	0.00	0.00	0.00	1
63	0.00	0.00	0.00	7
64	0.00	0.00	0.00	8
65	0.00	0.00	0.00	3
66	0.00	0.00	0.00	1
67	0.00	0.00	0.00	2

68	0.00	0.00	0.00	5
69	0.00	0.00	0.00	20
70	0.00	0.00	0.00	1
71	0.00	0.00	0.00	7
72	0.00	0.00	0.00	1
73	0.00	0.00	0.00	1
74	0.00	0.00	0.00	5
75	0.00	0.00	0.00	8
76	0.00	0.00	0.00	11
77	0.00	0.00	0.00	2
78	0.00	0.00	0.00	1
79	0.00	0.00	0.00	1
80	0.00	0.00	0.00	25
81	0.00	0.00	0.00	20
82	0.00	0.00	0.00	14
83	0.00	0.00	0.00	2
84	0.00	0.00	0.00	13
85	0.72	0.82	0.77	113
86	0.00	0.00	0.00	4
87	0.00	0.00	0.00	2
88	0.79	0.95	0.86	95
89	0.00	0.00	0.00	4
90	0.00	0.00	0.00	1
91	0.00	0.00	0.00	1
92	0.00	0.00	0.00	2
93	0.00	0.00	0.00	6
94	0.00	0.00	0.00	2
95	0.00	0.00	0.00	2
96	0.00	0.00	0.00	3
97	0.00	0.00	0.00	3
98	0.00	0.00	0.00	17
99	0.00	0.00	0.00	2
100	0.00	0.00	0.00	3
101	0.00	0.00	0.00	14
102	0.95	0.63	0.76	30
103	0.00	0.00	0.00	1
104	0.00	0.00	0.00	22
105	0.00	0.00	0.00	16
106	0.00	0.00	0.00	15
107	0.42	1.00	0.59	332
108	0.98	0.57	0.72	213
109	0.74	0.41	0.53	218
110	0.89	0.22	0.35	151
111	0.74	0.43	0.54	149
112	0.00	0.00	0.00	9

113	0.00	0.00	0.00	1
114	0.00	0.00	0.00	1
115	0.00	0.00	0.00	1
116	0.00	0.00	0.00	6
117	0.00	0.00	0.00	26
118	0.00	0.00	0.00	2
119	0.90	0.72	0.80	187
120	0.54	0.98	0.69	304
121	0.88	0.47	0.61	192
122	0.00	0.00	0.00	24
123	0.00	0.00	0.00	40
124	0.00	0.00	0.00	15
125	0.70	0.88	0.78	260
126	0.00	0.00	0.00	5
127	0.00	0.00	0.00	7
128	0.00	0.00	0.00	2
129	0.00	0.00	0.00	1
130	0.50	0.99	0.66	525
131	0.00	0.00	0.00	1
132	0.00	0.00	0.00	20
133	0.00	0.00	0.00	7
134	0.00	0.00	0.00	7
135	0.00	0.00	0.00	1
136	0.92	0.97	0.94	157
137	0.00	0.00	0.00	16
138	0.00	0.00	0.00	1
139	0.83	1.00	0.91	295
140	1.00	0.71	0.83	107
141	0.00	0.00	0.00	3
142	0.00	0.00	0.00	2
143	0.00	0.00	0.00	3
144	1.00	0.93	0.96	69
145	0.00	0.00	0.00	4
146	0.00	0.00	0.00	3
147	0.00	0.00	0.00	2
148	0.00	0.00	0.00	12
149	0.00	0.00	0.00	11
150	0.00	0.00	0.00	9
151	0.00	0.00	0.00	14
152	0.00	0.00	0.00	1
153	0.00	0.00	0.00	1
154	0.00	0.00	0.00	1
155	0.00	0.00	0.00	1
156	0.92	0.76	0.83	29
157	0.00	0.00	0.00	1

158	0.00	0.00	0.00	11
159	0.00	0.00	0.00	24
160	0.00	0.00	0.00	1
161	0.00	0.00	0.00	1
162	0.00	0.00	0.00	4
163	0.00	0.00	0.00	5
164	0.00	0.00	0.00	7
165	0.00	0.00	0.00	2
166	0.00	0.00	0.00	5
167	0.00	0.00	0.00	30
168	0.00	0.00	0.00	4
169	0.99	1.00	1.00	110
170	0.00	0.00	0.00	4
171	0.00	0.00	0.00	2
172	0.00	0.00	0.00	5
173	0.00	0.00	0.00	6
174	0.00	0.00	0.00	22
175	0.00	0.00	0.00	9
176	0.00	0.00	0.00	3
177	0.00	0.00	0.00	5
178	1.00	0.07	0.12	46
179	0.00	0.00	0.00	1
180	0.00	0.00	0.00	1
181	1.00	0.04	0.09	45
182	0.00	0.00	0.00	6
183	0.00	0.00	0.00	1
184	0.00	0.00	0.00	8
avg/total	0.53	0.58	0.51	7435

E.2 SVM

	precision	recall	f1-score	support
0	1.00	0.29	0.44	7
1	0.00	0.00	0.00	2
2	0.00	0.00	0.00	3
3	0.64	0.84	0.73	19
4	0.94	1.00	0.97	17
5	1.00	0.86	0.92	7
6	0.95	1.00	0.97	72
7	1.00	0.86	0.92	7

8	0.00	0.00	0.00	1
9	0.75	1.00	0.86	3
10	1.00	0.33	0.50	3
11	1.00	1.00	1.00	3
12	0.94	0.94	0.94	17
13	0.81	0.97	0.89	40
14	1.00	1.00	1.00	4
15	0.00	0.00	0.00	7
16	1.00	1.00	1.00	7
17	0.00	0.00	0.00	1
18	0.97	1.00	0.99	38
19	1.00	0.91	0.95	11
20	1.00	1.00	1.00	4
21	1.00	1.00	1.00	4
22	0.00	0.00	0.00	1
23	0.94	1.00	0.97	29
24	0.00	0.00	0.00	1
25	1.00	1.00	1.00	6
26	1.00	0.98	0.99	42
27	0.94	1.00	0.97	15
28	0.00	0.00	0.00	2
29	0.91	1.00	0.95	10
30	0.00	0.00	0.00	2
31	0.93	0.99	0.96	112
32	0.00	0.00	0.00	1
33	0.00	0.00	0.00	8
34	1.00	0.58	0.74	12
35	1.00	0.29	0.44	7
36	0.84	0.98	0.91	115
37	1.00	0.83	0.91	6
38	0.00	0.00	0.00	2
39	0.00	0.00	0.00	5
40	0.00	0.00	0.00	1
41	0.92	0.99	0.96	133
42	1.00	0.90	0.95	10
43	1.00	1.00	1.00	3
44	1.00	0.50	0.67	4
45	0.00	0.00	0.00	1
46	0.87	0.83	0.85	78
47	1.00	0.67	0.80	3
48	0.84	0.88	0.86	212
49	0.78	0.83	0.81	65
50	0.85	0.85	0.85	124
51	0.86	0.78	0.82	23
52	0.81	0.83	0.82	127

53	0.90	0.93	0.92	531
54	0.00	0.00	0.00	2
55	0.99	0.98	0.98	658
56	1.00	1.00	1.00	53
57	0.99	0.95	0.97	307
58	1.00	1.00	1.00	13
59	1.00	1.00	1.00	3
60	1.00	1.00	1.00	2
61	0.98	1.00	0.99	95
62	0.00	0.00	0.00	1
63	1.00	1.00	1.00	7
64	1.00	1.00	1.00	8
65	1.00	0.67	0.80	3
66	0.00	0.00	0.00	1
67	1.00	0.50	0.67	2
68	0.71	1.00	0.83	5
69	1.00	1.00	1.00	20
70	0.00	0.00	0.00	1
71	1.00	0.86	0.92	7
72	0.00	0.00	0.00	1
73	0.00	0.00	0.00	1
74	0.83	1.00	0.91	5
75	0.78	0.88	0.82	8
76	0.79	1.00	0.88	11
77	0.00	0.00	0.00	2
78	0.00	0.00	0.00	1
79	0.00	0.00	0.00	1
80	0.86	0.96	0.91	25
81	1.00	1.00	1.00	20
82	1.00	1.00	1.00	14
83	0.00	0.00	0.00	2
84	1.00	1.00	1.00	13
85	0.97	0.99	0.98	113
86	1.00	0.75	0.86	4
87	1.00	1.00	1.00	2
88	0.97	1.00	0.98	95
89	1.00	0.75	0.86	4
90	0.00	0.00	0.00	1
91	0.00	0.00	0.00	1
92	1.00	0.50	0.67	2
93	0.86	1.00	0.92	6
94	1.00	1.00	1.00	2
95	1.00	0.50	0.67	2
96	1.00	0.33	0.50	3
97	1.00	0.33	0.50	3

98	1.00	0.94	0.97	17
99	1.00	1.00	1.00	2
100	1.00	1.00	1.00	3
101	0.88	1.00	0.93	14
102	0.97	1.00	0.98	30
103	0.00	0.00	0.00	1
104	0.96	1.00	0.98	22
105	1.00	0.94	0.97	16
106	1.00	1.00	1.00	15
107	0.96	0.98	0.97	332
108	0.96	0.95	0.96	213
109	0.89	0.95	0.92	218
110	0.96	0.95	0.95	151
111	0.97	0.97	0.97	149
112	0.00	0.00	0.00	9
113	0.00	0.00	0.00	1
114	0.00	0.00	0.00	1
115	0.00	0.00	0.00	1
116	0.71	0.83	0.77	6
117	1.00	0.96	0.98	26
118	1.00	0.50	0.67	2
119	0.99	0.96	0.98	187
120	0.98	0.98	0.98	304
121	0.95	0.98	0.96	192
122	1.00	0.96	0.98	24
123	0.95	0.97	0.96	40
124	1.00	1.00	1.00	15
125	0.94	0.98	0.96	260
126	0.00	0.00	0.00	5
127	1.00	0.86	0.92	7
128	0.00	0.00	0.00	2
129	0.00	0.00	0.00	1
130	0.98	0.99	0.99	525
131	0.00	0.00	0.00	1
132	1.00	1.00	1.00	20
133	1.00	0.86	0.92	7
134	0.00	0.00	0.00	7
135	0.00	0.00	0.00	1
136	0.95	1.00	0.97	157
137	1.00	1.00	1.00	16
138	0.00	0.00	0.00	1
139	0.98	1.00	0.99	295
140	0.98	0.96	0.97	107
141	0.00	0.00	0.00	3
142	1.00	1.00	1.00	2

143	1.00	0.67	0.80	3
144	1.00	1.00	1.00	69
145	0.00	0.00	0.00	4
146	1.00	1.00	1.00	3
147	0.00	0.00	0.00	2
148	1.00	0.92	0.96	12
149	1.00	0.91	0.95	11
150	0.69	1.00	0.82	9
151	1.00	0.79	0.88	14
152	0.00	0.00	0.00	1
153	0.00	0.00	0.00	1
154	0.00	0.00	0.00	1
155	0.00	0.00	0.00	1
156	0.90	0.97	0.93	29
157	0.00	0.00	0.00	1
158	1.00	0.64	0.78	11
159	0.80	1.00	0.89	24
160	0.00	0.00	0.00	1
161	0.00	0.00	0.00	1
162	1.00	0.75	0.86	4
163	1.00	0.80	0.89	5
164	0.88	1.00	0.93	7
165	0.00	0.00	0.00	2
166	1.00	1.00	1.00	5
167	0.88	1.00	0.94	30
168	1.00	0.50	0.67	4
169	1.00	1.00	1.00	110
170	1.00	1.00	1.00	4
171	1.00	0.50	0.67	2
172	1.00	1.00	1.00	5
173	1.00	1.00	1.00	6
174	1.00	1.00	1.00	22
175	0.90	1.00	0.95	9
176	0.00	0.00	0.00	3
177	1.00	0.80	0.89	5
178	0.88	1.00	0.94	46
179	0.00	0.00	0.00	1
180	0.00	0.00	0.00	1
181	0.93	0.96	0.95	45
182	1.00	0.67	0.80	6
183	0.00	0.00	0.00	1
184	1.00	0.62	0.77	8
avg/total	0.93	0.94	0.94	7435

附錄 F

病理結果分類報表

F.1 貝葉式分類法

	precision	recall	f1-score	support
0	0.93	0.83	0.88	3786
1	0.00	0.00	0.00	39
2	0.89	0.96	0.92	2723
3	0.96	0.98	0.97	19578
4	0.99	0.84	0.91	656
5	0.00	0.00	0.00	179
6	0.00	0.00	0.00	1
avg/total	0.94	0.95	0.94	26962

F.2 SVM

	precision	recall	f1-score	support
0	0.96	0.97	0.97	3786
1	0.97	0.77	0.86	39
2	0.99	0.99	0.99	2723
3	0.99	0.99	0.99	19578
4	0.98	0.92	0.95	656
5	0.97	0.65	0.78	179

6	0.00	0.00	0.00	1
avg/total	0.98	0.98	0.98	26962